


Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (32th cycle)

Interfaces for human-centered production and use of computer graphics assets

Alberto Cannavò
* * * * * *

Supervisor
Prof. Fabrizio Lamberti

Doctoral Examination Committee:
Prof. Bill Kapralos, Referee, University of Ontario Institute of Technology
Prof. Adalberto Simeone, Referee, Katholieke Universiteit Leuven
Prof. Giuseppe Serra, Università di Udine
Prof. Riccardo Sisto, Politecnico di Torino
Prof. Filippo Stanco, Università di Catania

Politecnico di Torino
June 05, 2020

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Alberto Cannavò

Turin, June 05, 2020

Summary

Today, computer graphics can be regarded as a core enabling technology, supporting the development of an incredible number of services and applications. As a matter of example, computer graphics can be leveraged as a powerful representation instrument with information visualization, it can enable the creation of non-existent scenarios with virtual reality, and it can even enhance the real world by adding synthetic assets to it with augmented and mixed reality.

Domains that could benefit from this technology are expanding every day, covering traditional areas like video-game and movie production, but also getting ever more commonplace in other scenarios, including manufacturing, education and training, medical practice, cultural heritage and even sports, to name a few.

Despite its positive effects, the growing diffusion of computer graphics is also posing significant challenges to researchers and developers. Challenges concern the whole computer graphics pipeline, from content creation to utilization. The rise of challenges is also due to the fact that the set of users involved in the above processes is getting ever larger, and their skill level is becoming ever more heterogeneous. To make an example, the production of a blockbuster movie could involve hundreds to thousands of actors, designers, programmers, directors, etc., each with his or her own attitudes to technology, in general, and to computer graphics and human-machine interaction, in particular.

Based on the above considerations, the main objective of the research activities carried out during the Ph.D. and presented in this document was to improve the effectiveness of existing methods, tools and paradigms for the production and use of computer graphics contents, by leveraging, among others, recent advances in the fields of multi-modal interaction and intelligent computing.

Acknowledgements

I wish to thank Prof. Fabrizio Lamberti for believing in this research project. I thank him above all for his constructive criticisms and his invaluable suggestions, which guided me during the Ph.D. period.

Thanks also go to all those who took part in the experimental studies discussed in this document for their kind availability and the time they dedicated to me.

I would like to gratefully acknowledge all the members of the research lab, who shared with me fatigues and a lot of working hours.

My sincere thanks also go to all my colleagues and friends at HKU who I knew during my experience in Hong Kong. The months that I spent with them represented for me an opportunity to grow not only from the professional, but especially from the personal point of view.

I wish to express my gratitude to Angela Chiocca and Giorgia Brocca for being my family in Turin.

I would like to gratefully acknowledge all my friends (both in Turin and in Sicily) for providing me with unfailing support, even at distance, especially in the last months characterized by the lockdown restrictions imposed by the serious health crisis.

A final thought also goes to my family, who despite the distance, supported me and, with great confidence in my abilities, allowed me to achieve this important goal.

I would like to dedicate this thesis to my loving family

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Today, computer graphics (CG) can be considered as a core enabling technology, since it is able to support an incredible number of services and applications in different domains [134]. Due also to its widespread usage, it is difficult to find a unique and clear definition for it [124]. Generally, with CG one refers to that set of methodologies and tools meant to support the generation of pictures using a computer. The term "picture" could be interpreted in different ways, depending on the context CG is applied to. As a matter of example, a picture could represent a realistic rendering of the real world, as well as graphics, like plots, histograms, pie-charts, etc. Alternative interpretations of the term picture in the context of CG include graphics user interfaces (GUIs) and dedicated software suites [163].

Since its birth, which can unanimously be dated back to the early 1960's when the system called "Sketchpad" was presented in the doctoral thesis of I. Sutherland [296] at MIT, the interest in CG quickly grew both in academia and industry. This aspect is confirmed, for example, by the number of members attending the annual SIGGRAPH meeting[1], companies worldwide for which CG represents the main source of revenue (e.g., NVIDIA[2], Intel Corporation[3], etc.), as well as courses and degree programs (not only in the Computer Science field) where CG is taught [124].

Domains that could benefit from the considered technology are expanding every day. For instance, a first application of CG is represented by the design of GUIs for software belonging to heterogeneous domains. Although this application of CG is very common nowadays, it does not represent anymore its core usage scenario since,

[1] ACM SIGGRAPH: https://www.siggraph.org/

[2] NVIDIA: https://www.nvidia.com/en-us/

[3] Intel Corporation: https://www.intel.com/content/www/us/en/company-overview/company-overview.html

in this case, the emphasis is more devoted on the design of more intuitive human-computer interfaces (HCIs) rather than on producing complex/realistic graphic content [44, 132, 242]. Computer graphics techniques can serve also to enhance the quality of real photos or to add/remove components in pictures, especially in fields like advertising and visual art [145, 288]. Another example of CG usage is represented by the possibility to explore/analyze high-dimensional data through two- (2D) or three- (3D) dimensional representations or interactive animations that usually refer to the broad domain of information visualization [291].

Besides abstract representations, CG techniques are also employed to generate realistic images or animations, for example, in the context of product design [219]. Pioneers of this area are the Computer-Aided Design (CAD) tools employed for the realization of industrial products, like cars, airplanes, buildings and everyday objects. The availability of accurate, reconstructed 3D models are more and more leveraged in novel application domains ranging from medical practice (e.g., for digitally reconstructing bones and internal organs from X-ray images), to movies and video-game production (e.g., for generating realistic scenarios that can be observed from any point of view or adding visual effects to scenes recorded with traditional live action shooting), education and training (e.g., for simulating procedures or assessing the performance of trainees), cultural heritage (e.g., for representing artworks or ancient sites), etc. Finally, new technologies, such as Virtual Reality (VR) and Augmented Reality (AR) started to take advantage of CG techniques to create non-existing scenarios [129] or to enhance the real world by adding synthetic objects to it [303].

Despite the positive effects and benefits related to the introduction of CG in various application domains, its growing diffusion is also posing new and significant challenges. The main issues, faced by researchers and developers worldwide, concern not only a restricted/specific set of operations carried out with/through CG; rather, the whole pipeline is involved, from content creation [111, 299] to utilization [25, 255]. This condition is also related to another important aspect of CG diffusion: in fact, analyzing the set of users involved in the creation and use of graphics assets, it is possible to note that this set is getting ever larger and heterogeneous [72]. Nowadays, more and more users with different skills and backgrounds are approaching this field. For example, the production of common blockbuster movies involves hundreds to thousands of actors, designers, programmers, directors, etc., each with his or her own attitude to technology.

Based on the above considerations, the main objective of this Ph.D. thesis is to present researches that I carried out in the last three years with the aim of improving the effectiveness of existing methods, tools and paradigms for the production and use of CG contents.

As highlighted by the title of this document, attention was posed in particular on aspects related to human-centered design. As reported in the introduction of the ISO 9241-210:2019 standard on "Ergonomics of human-system interaction — Part

210: Human-centered design for interactive systems" [140], human-centered design can be defined as follows: "*Human-centred design is an approach to interactive systems development that aims to make systems usable and useful by focusing on the users, their needs and requirements, and by applying human factors/ergonomics, and usability knowledge and techniques. This approach enhances effectiveness and efficiency, improves human well-being, user satisfaction, accessibility and sustainability; and counteracts possible adverse effects of use on human health, safety and performance*".

Summarizing, the goal of the human-centered design is making systems usable and effective by considering users' needs and requirements. For this reason, many of the results discussed hereinafter are focused on human-machine interaction (HMI) and its impact on the usability of the developed systems.

1.1 Thesis organization

In order to present the activities carried out considering the above goal, discussion has been organized in several chapters according to the following structure.

- Computer animation: How to bring a virtual character to life.
- Is immersive virtual reality the ultimate interface for 3D animators?
- Interfaces and methods supporting the generation of graphics assets.
- Leveraging graphics assets: Interactive applications development.
- Augmented, mixed and virtual reality: A new perspective for human-machine interaction.

With respect to aspects concerning the generation of graphics assets, research activities carried out during the Ph.D. were focused on the computer animation field (Chapter 2 and Chapter 3), since computer-generated animations are massively used in a growing number of application domains ranging from movie and video-game production, architecture, industrial design, product advertising [76, 200], to name a few. Specific stages constituting the pipeline generally adopted for virtual character animation were considered, by studying the impact that alternative interfaces and technologies can have on the execution of common animation tasks.

The broad domain related to the generation of 3D content was studied in depth (Chapter 4), by also considering the possibility of taking advantage of new interactive tools based on VR/AR technologies as well as of intelligent computing-based approaches for developing such contents.

Proceeding forward on the computer graphics pipeline, new tools of making generated 3D content accessible through so-called interactive applications are needed. Hence, the latter one was established as the goal of research activities developed in the domain of computer assets usage (Chapter 5).

Lastly, Chapter 6 focuses on the last stage of graphics pipeline by addressing the usage of created assets into specific application domains. In particular, aspects related to HMI were investigated in order to provide users with improved haptic and visual feedback on aspects concerning the operations being performed. Use cases considered virtual environment-based control, sport training activities and human-robot interaction (HRI).

Chapter 2

Computer animation: How to bring a virtual character to life

Work described in this chapter was originally presented in [173, 172].

2.1 Introduction

The generation of virtual character animations is becoming fundamental for a wide range of applications, from the production of movies and video-games to the creation of virtual environments used, e.g., in education, cultural heritage, product design, social networking scenarios, etc. [76, 185]. The process leading to the generation of 3D animated characters is usually characterized by very labor-intensive tasks and by the involvement of users with significant expertise. These facts are mainly related to the need of using sophisticated interfaces [59].

Considering the wide range of users who are approaching the computer animation field, the difficulties introduced by use of the traditional interfaces could represent an important limitation. This aspect could be particularly critical for unskilled users, who, for example, might want to leverage animations in rapid prototyping tasks that are typical of the early production stages of both digital and non-digital content [79, 330]. Therefore, new approaches that try to trade-off the severe requirements imposed by the existing animation systems with an higher productivity/intuitiveness (possibly paid in terms of control accuracy) are becoming of great interest [179].

Making it easier the creation of character animations was actually the main goal behind two research activities that will be discussed in the following, which led to the creation of two animation systems. Both the systems leverage non-traditional interfaces, e.g., based on reconfigurable tangible props and/or voice commands, as well alternative interaction methods, e.g., based on animator's performance or natural language processing (NLP), able to make it easier the interaction with

created contents.

2.2 Motion capture and reconfigurable tangible interfaces

According to [170], the complexity associated with the generation of character animations in traditional animation systems based on keyframing lays on the posing step. In fact, these systems still rely on the use of 2D traditional interfaces like mouse and keyboard (M&K). Hence, animators are requested to select and manipulate, through an interface that is natively 2D, a potentially large number of on-screen "handles" for articulating a character's virtual skeleton. This set of handles is often referred to as an "armature" or a "rig", constituted by rigid segments named "bones". Handles can be used by the animators to directly or indirectly manipulate the degrees of freedom (DOFs) of all the individual character's parts/joints [142].

New interaction paradigms have been proposed by the research community to cope with these issues. One of the solutions becoming widely used is performance-driven animation. In this approach, the physical performance of an actor is captured and interactively transferred to the virtual character to be animated. In this way, the movements of the character can be manipulated in real-time, thus giving the animator a direct feedback about the animation being created [280].

A form of performance-driven animation that has become very common in many movie and video-game productions is motion capture [225]. However, the costs for configuring the setup and the skills required to use it generally make this technique suitable especially for professional animation studios [175]. Furthermore, setting the mapping between the performer's and character's movements when the topology of the character's armature presents many differences with the human skeleton could represent a complex task, which requires sophisticated configuration steps [322]. Automatic retargeting systems exist. However, the techniques proposed so far do not allow the animators to obtain a fine control on the resulting poses [257].

Another interaction paradigm is represented by the so-called tangible interfaces, or tangible user interfaces (TUIs) [139]. According to [305], TUIs can be defined as systems that give physical form to digital information, using physical artifacts both as representations and controls for computational media. Work in the literature ([120, 330]) confirmed the improved control that TUIs can provide on the articulation of virtual characters, especially for novice users, compared to techniques based, e.g., on tactile feedback and intuitive 3D perspectives. Although TUIs have been successfully used both for keyframing and performance-driven animation, they were never considered in a single animation pipeline that combines these two approaches or which is able to benefit from the use in the same system of different interaction paradigms. Finally, TUIs were often based on specially shaped

hardware, since they were developed to animate only a given character. When designed for re-configurability, they could become particularly costly and/or hard to assemble. Moreover, if the structure of the TUI is not fixed, creating the mapping between physical modifications of the interface and movements of the character could require complex setup mechanisms.

Considering the above observations, an animation system was designed and developed, in which information about the pose of an instrumented TUI based on reconfigurable components can be combined with data captured by tracking the animator's body to create 3D character animations with both keyframing and performance-driven techniques. While manipulating the character's pose with the TUI, the animator can manage its 3D position through his or her body. An automatic configuration mechanism is implemented to support animators in the assembly and configuration of the interface that best fits the DOFs of the character to be animated. The system has been integrated in the Blender[1] open-source animation tool, making it suited, also in terms of costs, to both professional and non-professional users.

Animation functionalities commonly used in Blender, as well as configuration mechanisms needed by the proposed interaction method are accessible through customized voice commands, thus letting the animator focus only on the tangible prop held in his or her hands and on the movements to be performed. Experiments, which have been carried out with the aim to compare the devised system with the traditional M&K interface, confirmed its suitability for both skilled and unskilled users.

2.2.1 Related work

In this section, relevant work pertaining to techniques and interfaces for computer animation are reviewed.

Body and hand motion tracking

Motion tracking systems were devised to cope with the limitations of 2D interfaces (e.g., based on pen and touch/multitouch input [57]). Early motion tracking systems were characterized by a low control over the virtual characters being animated, and allowed animators to use only their hands (with or without gloves) to define keyframes or produce performance-driven animations [165, 237]. The control of more complex virtual characters characterized by a large number of DOFs is generally obtained more effectively by using high-dimensional full-body motion capture systems [138]. However, the high quality of the animations generated is often balanced with the need for expensive setups, large spaces for capturing the

[1]Blender: https://www.blender.org/

performance, skilled performers, and strenuous post-processing steps [142]. More recent technological advancements limited, for some application contexts, the impact of these requirements, by allowing also novice users to record full-body motion data leveraging, e.g., consumer-level vision-based marker-less tracking systems or wearable sensors [11, 180]. Besides the technology used, a consideration that must be taken into account both for keyframing and performance-driven animation is the methodology adopted for mapping the DOFs captured by the tracking system onto the DOFs of the character to be animated [294]. The above mapping mechanism may be either direct or indirect. The first approach is generally adopted for controlling human-shaped (anthropomorphic) characters characterized by a number of DOFs similar to that of the capture technology [138]. Conversely, indirect mapping is often applied to animate non-anthropomorphic characters or to retarget motion to a different number of DOFs [174]. Several methodologies have been developed with the aim of enabling the automatic retargeting of a performer's movements into the motions of arbitrarily shaped characters [18]. An approach focused on rigged characters proposed in [265] considered the similarity between controlling and controlled DOFs. Alternative methodologies not needing the character's mesh rig are also available [59, 257]. However, the approaches seen so far force the animator to act as the virtual character, by assuming poses that could be non-natural. This drawback could make it difficult or even unfeasible to create all the desired motions. Various techniques have been designed to address this limitation, e.g., by reconstructing a realistic character's motion from a predefined set of already-existing action samples that are blended together according to the actual performance of the animator [276]. These techniques generally do not fit with the requirements of professional productions, since animators usually require fine-grained control over the configuration of each character's joint. Therefore, manual solutions have been adopted, which often ask animators to work with sophisticated/complex interfaces and assume that mapping will be a task quite hard to accomplish [20, 135]. It is worth observing that both retargeting approaches suffer from the fact that the control interface (i.e., the animators' hand or body) has a fixed topology, which can be considerably different from that of the virtual character. Thus, it is not possible to ignore the mental effort required during the animation process.

Generically and specially shaped tangible devices

TUIs were designed to make general-purpose HMI more intuitive, by leveraging the affordances offered by physical objects used for the interaction [139]. Their usage for computer animation is not new, and it can be dated back to the late 1980's, when the mechanical devices called "Waldos" were leveraged for manipulating computer-generated puppets performing on TV shows [110, 259]. Devices and materials considered to develop such interfaces are rather heterogeneous. In some cases, paper tags have been used [15, 176]. Sometimes, physical and haptic props

recreating commonly used interfaces were leveraged, like those used by puppeteers to pull the strings of a real mannequin [229]. More frequently, special-purpose mechanical devices have been chosen, designed to imitate as much as possible the shape of the character to be animated. This has been done under the hypothesis that intuitiveness could be enhanced by decreasing the distinction between the tangible and virtual worlds [84, 179]. For example, in [231], an instrumented human-shaped doll is proposed to reconstruct poses by retrieving them from matching motion capture data available on a repository. Another solution is presented in [92], where the system is able to track the joints of the physical device used by the animator through stereo camera. In [330], passive joints of the TUI are replaced by servo motors. According to the observations presented in [330], besides the advantage related to the possibility of reconfiguring the device to previous poses, active joints could be also used to provide animators with physical feedback, to compensate gravity or to recreate natural behaviors. Analogous approaches have been leveraged also for animating non-anthropomorphic characters. For example, in [149], a chicken-shaped plush toy is used to control the actions performed by a similar character into an interactive game. In [279], a teddy bear-shaped robotic interface was proposed both as an input device (using sensors embedded in the robot's arms) and output device (through vibrotactile feedback). The effectiveness of TUIs for character animation was proven in several scenarios ranging from home entertainment to professional productions. For instance, in [306], a low-cost arm-shaped device with embedded sensors based on an Arduino board is introduced; the device allows users to animate an articulated arm presenting the same DOFs. In [164], the authors describe the design of the sophisticated dinosaur-shaped instrumented armatures that were used to produce many of the stop motion shots of Jurassic Park.

Reconfigurable tangible interfaces

Despite the benefits connected to the use of TUIs for animating virtual characters, the fixed structure of solutions seen so far could represent a critical constraint to their applicability in general-purpose animation scenarios [330]. Therefore, various methodologies have been proposed whereby components (both instrumented and non-instrumented) can be assembled in many ways to make the interface assume the desired configuration. Reconfigurable TUIs were initially employed for 3D modeling [316] and, more recently, began to be adopted also for 3D animation [254]. Interfaces described in the previous paragraph present the limitation of being unaware of the exact topology built by the user. However, this information could represent an important piece of knowledge capable of making it easier the construction of the mapping [142]. A first example of an hub-and-strut construction kit, able to reconstruct both the topology and geometry of the model being

assembled by the user based on information gathered by elementary bricks is presented in [318]. The interface was designed as a general-purpose modeling tool for studying physical chemistry, mechanical robotics, etc. Although this interface was not capable of offering the fine-grained control required by complex characters, results demonstrated its applicability for controlling simple digital puppets. This idea was further investigated in [142], where the effectiveness of a modular and reconfigurable topology-aware TUI targeted to character posing was assessed through a user study with unskilled subjects. The interface is built by assembling hot-pluggable instrumented components provided with accurate sensing capabilities, thus making the interface suitable for both rapid prototyping and precise posing. The whole structure of the character to be animated can be reconstructed since the size of the components is very compact. This makes it easier to retarget the deformations of the physical prop to the virtual character. Although schematics of the components are available as open hardware, compactness of the design actually makes the interface really costly and hard to recreate, especially for users with limited skills. Moreover, animators could animate complex armatures, by subdividing them into several parts and controlling each part by reusing the same tangible prop. Although this feature was provided by the system, activating the control of each part, as well as creating the mapping between parts and the TUI still requires users to work with a M&K interface. Furthermore, retargeting results might not be as intuitive as in the case of a complete armature. In [104], the solution proposed in [142] was extended to tailor the control of virtual characters characterized by tens or hundreds of DOFs to be controlled with a TUI composed of a small set of elements through a retargeting strategy based on rig simplification. To deal with this challenge, authors proposed to find a simple geometric skeleton (meant as a subset of the original armature) which can be matched with the given number of tangible elements. Based on available elements, the number of DOFs handled at the same time can be adjusted, thus allowing animators to control the posing accuracy. However, this methodology relies on a preparatory step, in which a number of sample poses have to be specified in order to allow the system to create a priority rank of DOFs to be controlled. Hence, results could be affected by the quality and size of the sample set. Another limitation of systems described so far is that they collect only relative measurements. Thus, to control, for example, the character's position, other interfaces should be used in separate animation steps. Some efforts to address the above issue were made under simplified conditions. For example, in [120] and [216], rigid objects were controlled and tracked to animate corresponding 3D models in a virtual scene, but only unarticulated tangible devices were considered.

Moving from the advantages and drawbacks mentioned above, a new solution was designed. It combines general-purpose consumer-level hardware with an automatic mapping process meant to support the construction of the TUI, with the aim of helping the animator assemble, based on available bricks, the interface that

best fits the particular character to be animated. Differently than in [104] where a rig simplification is used, the proposed solution is meant to produce an intuitive configuration composed of a limited set of tangible components that let the animator control, in separate steps, all the DOFs of a rigged virtual character. The tangible prop, as well as the animator handing the interface, are immersed into an affordable motion capture system; this way, the animator can control the position of 3D virtual characters in a natural way by making use of both keyframing and performance-driven techniques.

2.2.2 Proposed system

The architecture of the proposed system, which was originally presented in [173] is depicted in Fig. 2.1. The architecture includes an *Animation software*, which is controlled by the animator through a *Interaction agent*, which is responsible for managing data gathered by multiple *Input devices*. The overall architecture of the system was designed in order to support further extensions to its blocks. Thus, for example, the blocks in Input devices group together interfaces based on different sensing technologies, e.g., cameras, rotation sensors, microphones, that can be used to control virtual characters and create animations. In the following, greater details regarding the individual blocks are provided.

Input devices

The system presented in [173] supports three types of input interfaces: the *Tangible interface*, the *Body tracking interface* and the *Speech interface*. However, the set of supported input devices can be extended by considering different interfaces, like 3D mouses (SpaceMouse Compact[2]), hand tracking solutions (e.g., the Leap Motion controller[3], and the Myo's armband[4]), etc.

In order to implement the Tangible interface block, a collection of off-the-shelf Lego Mindstorm EV3 bricks of the core set[5] and the expansion set[6] is chosen. Besides a number of bricks and components that can be used to assemble the interface of interest, the two sets contain also a number of devices ranging from medium and large servo motors to different types of sensors, e.g., ultrasonic, infrared, touch, and

[2]SpaceMouse Compact: https://www.3dconnexion.com/spacemouse_compact/en/

[3]Leap Motion: https://developer.leapmotion.com/

[4]Myo: https://support.getmyo.com/hc/en-us

[5]Lego Mindstorm Education EV3 Core Set (#5003400): https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-core-set/5003400

[6]Lego Mindstorm Education EV3 Expansion Set (#45560): https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-expansion-set-by-lego-education/45560

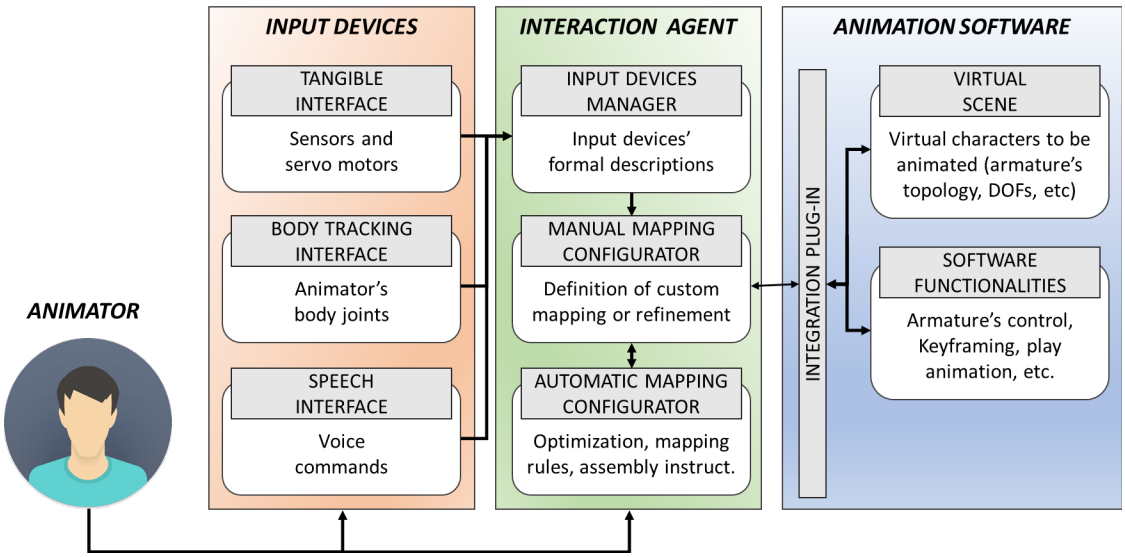


Figure 2.1: Architecture of the proposed animation system [173].

gyro sensors. Data gathered by sensors and servo motors are collected by means of a component named Intelligent brick. Numerical readings, included in a JSON file, are sent by the Intelligent brick to the Interaction agent block through a Wi-Fi/USB connections. The connections are managed through a third-party API[7] that ensures a sampling rate of 10 Hz and a latency around 370 ms when the Wi-Fi connection is used (worst results were registered with the USB connection). The Microsoft Kinect sensor device is adopted as the Body tracking interface since it provides position and orientation information for a 20-joint representation of the animator's skeleton. Data collected by the Microsoft Kinect are accessed through a dedicated SDK[8]. The Speech interface is based on the Microsoft Speech Platform library[9] .

Due to the heterogeneous nature of the data provided by the considered interfaces, each block was used to control different elements of the animation creation process. In particular, from the Tangible interface and Body tracking interface blocks it is possible to collect position and orientation data that can be used to control specific transformations of characters and 3D objects to be animated. Voice commands recognized through the Speech interface are used to activate specific *Software functionalities* of the Animation software.

[7]Lego Mindstorm EV3 API for .NET: https://github.com/BrianPeek/legoev3

[8]Kinect for Windows SDK v1.8: https://www.microsoft.com/en-us/download/details.aspx?id=40278

[9]Microsoft Speech Platform: https://msdn.microsoft.com/en-us/library/jj127858.aspx

Interaction agent

The Interaction agent includes the *Input device manager*, the block in charge of handling the data gathered by the Input devices in order to make them usable by the Animation software for creating animations. The mapping between the available sensors of the input devices, i.e., rotation sensors, servo motors and body-tracked joints (later referred to as "interface elements"), and the translational and rotational DOFs of bones in the virtual characters to be animated (in the following named "target armature") can be configured following two different approaches. The former is based on the *Manual mapping configurator* block, which allows animators to manually define the assignment of each interface element to specific DOFs through a dedicated graphics interface. However, if the target armature is made up of a vast amount of bones, the manual configuration of each DOF could represent a tedious task. Moreover, when the number of the available interface elements is not sufficient to manipulate all the desired DOFs at once, the set up of a proper mapping could represent a complex task. The complexity is related to the need to find a mapping able to optimize the affordance of using the available interface elements to control all the DOFs, taking into account factors like the similarity between the tracked body parts or the alternative assemblies of available tangible bricks and the topology of the target armature, the typology and the corresponding range of DOFs and the relative interface elements to control with. For these reasons, an *Automatic mapping configurator* was developed with the aim to propose a solution to the animator which makes it possible to control with a possible assembly of the interface elements all the DOFs of the target armature, minimizing the cost of assigning to each DOF a specific element. The output of the Automatic mapping configurator can present solutions with make use only of one type of interface (Tangible interfaces or Body tracking interface) or consider the two interfaces in a combined way. If the animator chooses the combined way, he or she can benefit in the same configuration from advantages brought by TUI, i.e., for articulating armature's joints, and the possibility to capture absolute positioning information to be used to move elements and/or animating them with performance-driven animation. When the animators chooses the Tangible interface, the rules which define the mapping are also used to build step-by-step instructions required to assemble servo motors and sensors considered in the proposed solution, with the aim to make it the assembly of the tangible prop easier for both skilled and unskilled users. The Automatic mapping configurator supports both the forward kinematics (FK) and inverse kinematics (IK), hence it is able to produce solutions in which animators can control directly the rotational DOFs of a bones chain (FK) or its end-effectors (IK).

Fig. 2.2 shows the overall workflow that can be used to create a possible mapping for a character (lamp character) with a small number of DOFs. The Automatic mapping configurator, considering the available interface elements and the target

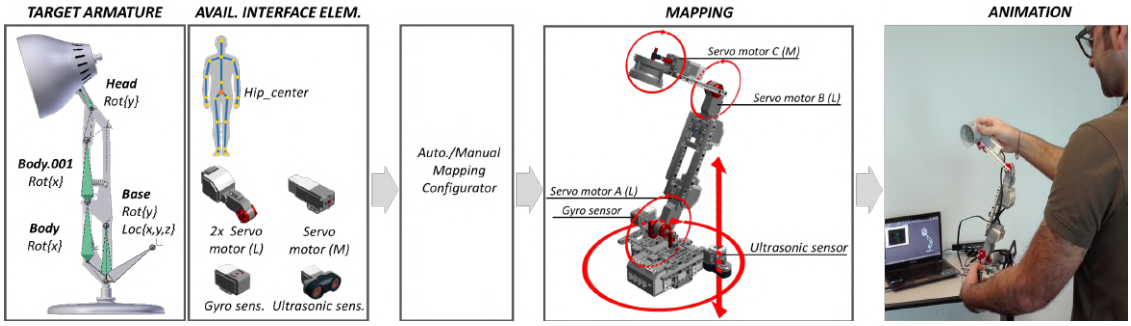


Figure 2.2: Automatic/Manual definition of mapping rules to manipulate the target armature of a lamp character with a given set of tangible bricks and body-tracked joints.

armature (on the left of Fig. 2.2), proposes a set of mapping rules and an assembly for the TUI (represented in the middle) that can be successively refined by the user through the Manual mapping configurator. At the end of the process (on the right), the generated mapping is adopted by the animator to control the lamp character with an interface that, in this case, mimics the topology of the character. When the number of DOFs to be manipulated is higher than the available interface elements, it is not possible to control all the parts of the character at once. Hence, the Automatic mapping configurator considers all the possible decompositions of the target armature composed by the so-called "partitions" and finds the best decomposition that minimizes the assignment cost of each DOFs– interface element pair. When the target armature is decomposed, animators control only one partition at a time with its specific mapping rules. In [173], the Interaction agent was developed as a Windows Presentation Foundation (WPF) application using the C# language.

Animation software

This block aims to host and make available to animators information concerning the *Virtual scene*, i.e., the scene which contains elements to be animated. The 3D graphics suite named Blender is selected for this purpose. The system presented in [173] supports the manipulation and animation of two types of transformations: rigid transformations applied to generic 3D objects, and armature-based deformations applied to rigged virtual characters. The Software functionalities mentioned above include the set of functionalities of the animation software needed for the creation of animations. Functionalities considered in [173] range from moving the current frame of the timeline to enabling/disabling continuous keyframing (in order to switch between performance-driven and keyframing animation), inserting/deleting/copying/pasting keyframes, starting the playback of the animation created, etc. Moreover, when the configurations created by the two configurators present multiple partitions to be controlled, a different voice command is associated

to each partition in order to activate its manipulation through the corresponding interface elements. Future developments of the system could make available the possibility to manage more complex commands, e.g., to retrieve in real-time prerecorded animations from a user-defined library.

In order to ensure a high degree of flexibility, a block named *Integration plug-in* was developed targeted to a specific animation software in order to convert the above information into a new representation that can be managed/used directly by the selected animation software. This solution makes it possible to easily extend the proposed architecture to another animation software by changing only the implementation of the plug-in. In [173], the Integration plug-in was developed as a Python script for Blender.

2.2.3 Automatic mapping

In the following, more details will be provided on the approach adopted by the Automatic mapping configurator to generate the configurations.

Optimization procedure

The overall procedure includes the following steps:

1. *Selection of the mapping mode*: in this step, the animator is asked to choose the set of interface elements to be considered. The available modalities are TUI-, NUI- or TUI+NUI-based mapping mode. The TUI-based mapping mode will take into account only data coming from the Tangible interface block, i.e., from sensors and servo motors available, whereas the NUI- (i.e., Natural User Interface)-based mapping will make use of data coming from the Body tracking block. In [173], the two mapping modes are considered as alternatives because the manipulation of the interface elements could bring the animator to occlude several joints tracked by the Microsoft Kinect depth sensor. Moreover, it is generally uncomfortable for the user to hold the tangible prop while manipulating the pose of a virtual character with his or her arms. For this reason, the TUI+NUI-based mapping mode combines only the set of available interface elements of the Tangible interface with the Kinect-tracked hip joint, since its usage does not affect the operations with the tangible prop. Finally, further options can be configured by the animator, i.e., the presence/exclusion of sensors in the generated configurations (in the TUI+NUI-based mapping mode, the hip joint is assimilated to a sensor); the possibility to leverage/ignore the presence of possible symmetries for the creation of the mapping; and whether to manage position and rotation DOFs combined in the same configuration (P&O control) or separately (P|O control).

2. *Representation of the target armature*: in order to define a mapping, the topological information of the target armature, hosted in the Animation software, must be collected by the mapping process. In particular, for each bone the following information is retrieved:

   - bone's name;
   - its DOFs;
   - its position in the kinematics chain;
   - the number and the name of bones connected to it;
   - the presence of possible bone's constraints, i.e., copy-location/rotation constraints.

   The above information is organized into a graph-based representation. Each node of the graph represents a bone of the target armature, whereas edges are associated with relations among bones. Fig. 2.3 shows an example of a target armatures composed of five bones, each characterized by different names, position and orientation DOFs and bones relations.

   As said, when the complexity of the target armature is high, it could happen that the number of interface elements is lower than the DOFs to be controlled. In this case, a partitioning step is performed, with the aim of determining all the possible alternatives in which nodes, belonging to the target armature's graph, can be grouped. The constraint to be considered in the partitioning step is that the total number of DOFs that can be controlled at once with the available interface elements needs to be greater or equal to the sum of the DOFs in each partition. In order to determine the number of DOFs provided by the interface elements, a distinction is made depending on the control modality (P|O or P&O) selected. In particular, for the P|O control modality, position and orientation DOFs of the interface elements are considered separately in order to account for the fact that the Kinect-tracked hip joint and the ultrasonic sensors natively provide position DOFs, whereas gyro sensor and servo motors yield orientation DOFs. Moreover, in this control modality, the partitioning is performed by considering only orientation DOFs. When the P&O control modality is selected, each interface element is assumed to provide generic DOFs, hence no distinction is made. The partitioning process is handled with two parameters: *MaxBonesInPartition* and *MaxDofsInPartition*. The first parameter, that ranges from 1 to the partition size, refers to the maximum number of bones that a single partition can contain. The second parameter indicates the maximum number of DOFs that a partition can contain. The lower bound of the second parameter is given by the bone with the highest number of DOFs to be controlled, whereas, the higher bound is determined by counting the total number of available interface elements.

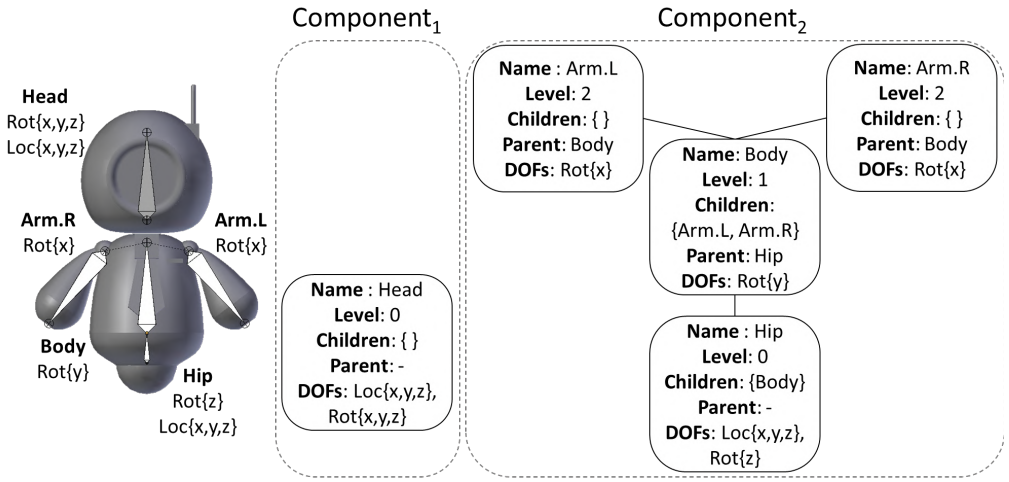


Figure 2.3: Visual representation of graph generated by a target armature composed of 5 bones and 13 DOFs.

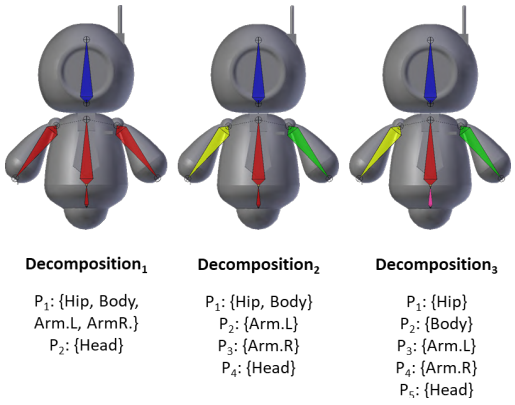
As said, the number of DOFs to be considered is determined by the control modality selected. For the P|O control modality, it is considered the bone with the highest number of position or orientation DOFs, depending on the type of partitioning of interest. For the P&O control modality, the value is determined by considering the bone in which the sum of position and orientation DOFs has the highest value. It is worth noticing that if the total number of available interface elements is lower than the minimum value of the second parameter, then there will be at least one bone that will present its DOFs split among different partitions, making the resulting mapping not intuitive. During the partitioning process, the two parameters are varied producing alternative ways of decomposing the target armature (in the latter referred to as "partial decompositions").

To perform the partitioning, each of the weakly-connected components of the graph (for example, the target armature in Fig. 2.3 presents two components identified with different shades), is explored adopting a depth-first search approach on all the nodes in the graph. During the graph exploration, a new partition is created when the number of bones/DOFs exceeds the current value of the parameters mentioned above, and when the node, i.e. the bone, visited has no relation with the last bone inserted in the current partition. Once the values of the two parameters have reached the respective lower bound (each adjustment of the values corresponds to a visit of the graph), the next component is considered and iteratively explored. Possible duplicates are removed. The Cartesian product is used to combine the partial decomposition generated by all the graph components to produce all the sets of partitions (called "graph decompositions") to be considered for

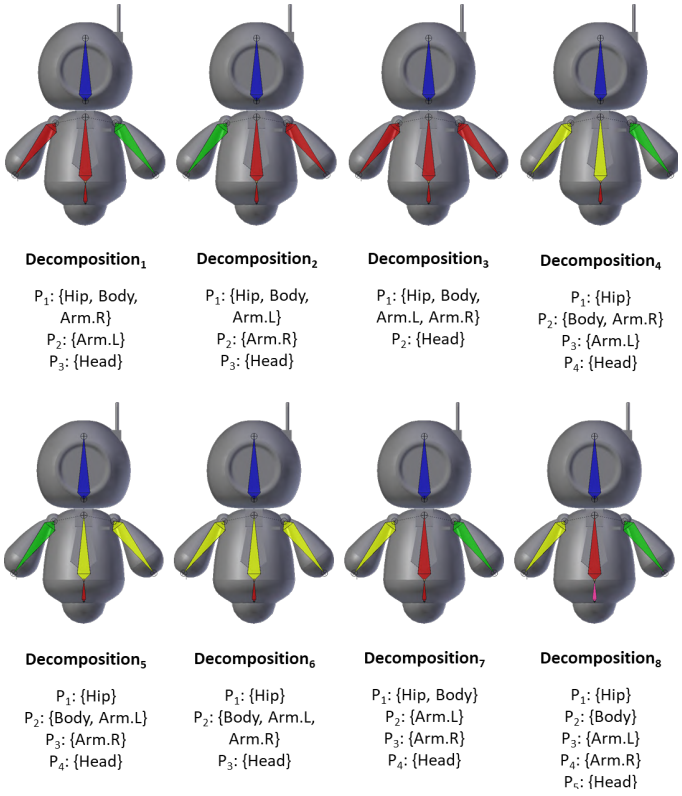
further processing. If during the exploration of the graph a split (i.e., a bone presenting more than one child) is identified, it is possible to include in the elaboration of the mapping the presence of symmetries (i.e., bones marked with the suffix .L/.R in the Blender's notation and with a parent bone). In particular, if the animator wants to consider symmetries, only partitions containing both the parent and the child bones, or with the parent and child bones all separated, will be created. Otherwise, bones with .L and .R are ignored. All the bones which present copy location/rotation constraints or belong to inverse kinematics chains are neglected, since positions and orientations are not controlled directly by the animator (for the inverse kinematics chains, only the end-effectors are considered for the remaining part of the processing). For the sake of clarity, an example of the partitioning process is provided in Fig. 2.4. In particular, it is assumed that the TUI+NUI-based mapping mode has been selected by the animator, the tangible prop includes one medium servo motor, three large servo motor and one ultrasonic sensor (i.e., 5 DOFs plus 3DOFs coming from the Kinect-tracked hip joint), and symmetries have to be maintained. The partitioning process would generate the three decompositions presented in Fig. 2.4a. Otherwise, should the animator decide to ignore the symmetries, the decompositions created would be eight, as reported in Fig. 2.4b.

3. *Problem formulation*: in [173], finding the automatic mapping is formulated as an assignment problem, in which the goal is to minimize a function that considers the cost of assigning the available interface elements to all the bones contained in each partition created from the target armature. By defining the "source armature" as one the possible configurations of the interface elements and representing it as a collection of bones like the target armature, it is possible to define the assignment cost through a matrix C where each element c_{ij} represents the cost of assigning the i-th bone in a specific partition of the target armature to the j-th bones of a given source armature.

4. *Source armatures generation*: if the animator has selected the NUI-based mapping mode, it is possible to assemble just one source armature, which is composed by the bones of the skeleton tracked by the Kinect. On the contrary for the TUI or the TUI+NUI- based mapping modes, the set containing all the possible source armatures has to be generated. This process relies on the creation of so-called "source armature templates", i.e., armatures comprised of bones that do not have any interface elements assigned (yet), that can be used to control the DOFs of the target armature. Source armature templates are generated by computing first all the permutations with repetition of length n (with n equals to the number of available interface elements) of the set composed by the three axes (x, y, z). The 3^n permutations generated represent all the possible DOF sequences. After the generation of the DOF sequences, the



(a) Considering symmetries



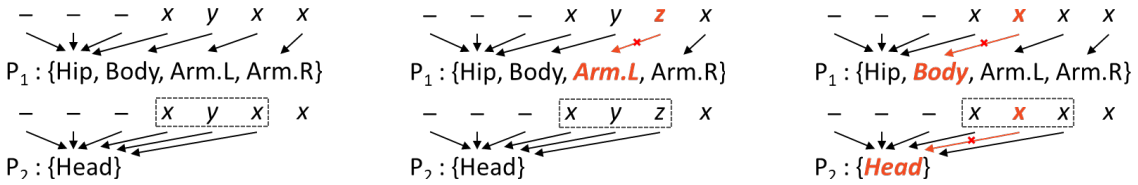
(b) Without considering symmetries.

Figure 2.4: Decompositions and partitions generated by the Automatic mapping configurator for the target armature shown in Fig. 2.3.

number of occurrences (i.e., the presence of the same DOFs) found between the target armature and the above DOF sequences is computed differently for the P|O or the P&O control modality. For the first modality, occurrences are calculated only considering the orientation decompositions. When P&O

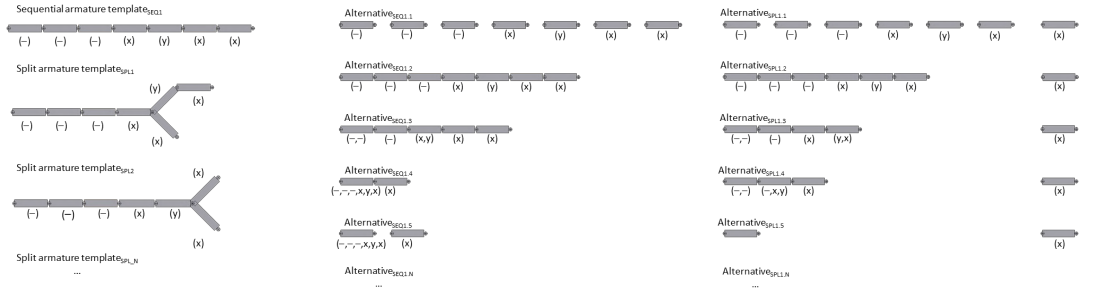
modality is selected, although partitions could contain bones with both position and orientation DOFs, position DOFs are disregarded since, in this case, mapping an interface element that controls a specific axis on a given DOF is not necessary (for example, a position DOF along the x axis could be easily controlled by a servo motor mounted along the y axis). In the computation of the occurrences for the bones with three rotation DOFs, all the alternative representations based on Proper Euler (i.e., z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y) and Tait-Bryan (i.e., x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z) angles are considered. DOF sequences that present the highest number of occurrences are selected. DOFs which resulted to be unused are removed when P|O control modality is chosen, otherwise (case of P&O control modality), they are interpreted as generic DOFs or "don't care" (-). For example, for the first decomposition of the target armature shown in Fig. 2.4a, the DOF sequence that appears multiple times in the two partitions is (-, -, -, x, y, x, x). Fig. 2.5 summarizes the process. Then, the DOF sequences are combined sequentially or adding splits to create the source armature templates. Due to the fact that DOF sequences can be combined in various ways to create new bones characterized by a different number of DOFs, several "alternative" source armature templates are generated. Fig. 2.6a presents a subset of source armatures templates generated by the same DOF sequence (-, -, -, x, y, x, x). Alternatives that can be generated considering different associations of the bones in the same source armature template are illustrated for a sequential and split source armature template in Fig. 2.6b and Fig. 2.6c. It can be observed that split source armature templates in Fig. 2.6a would have not been created in case the animator decided to ignore symmetries or partitions do not contain splits. Finally, all the source armatures are created by associating to each bone of the alternative source armature templates an interface element.

5. *Solution to the assignment problem*: the resulting assignment problem is solved by applying the Hungarian algorithm [168] to the cost matrix C, as already proposed in previous work [142, 265]. The output of the Hungarian algorithm (later referred to as the "partition cost") represents the minimum cost of assigning a specific partition of the target armature to a particular source armature. A so-called "decomposition cost" is then computed by summing all the partition costs for all the partitions belonging to the same decomposition. For the P|O control modality, the decomposition cost is calculated considering all the (source armature, orientation decomposition) pairs. The pair characterized by the minimum cost is used to control orientations. If the target armature contains also position DOFs and animator has selected the TUI- or the TUI+NUI-based mapping modes, the resulting source armature is enriched with additional interface elements that currently are not part



(a) (b) (c)

Figure 2.5: Process for determining the DOF sequences with the highest number of occurrences in the two partitions included in the first decomposition of Fig. 2.4a; (a) sequence with the highest number of occurrences, (b) sequence whose DOFs do not match all the DOFs of P_1, (c) sequence whose DOFs do not match all the DOFs of neither P_1 nor P_2. Non-matching DOFs are marked in bold and red. Dashed boxes indicate the particular Proper Euler/Tait-Bryan representation considered.

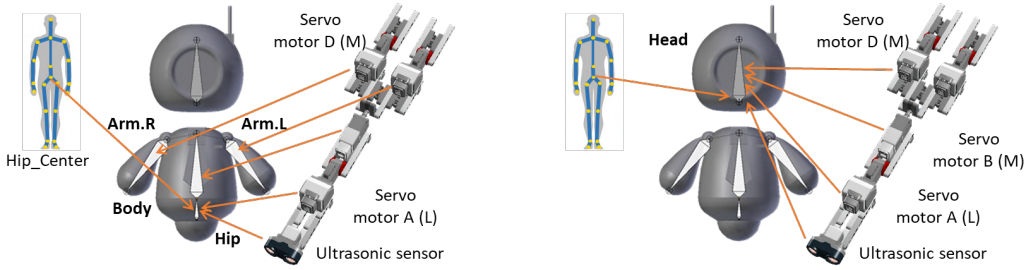


(a) (b) (c)

Figure 2.6: Subset of (a) sequential and split source armature templates that would be generated for the DOF sequence (-, -, -, *x*, *y*, *x*, *x*), (b) alternatives for the *Sequential armature template*$_{SEQ1}$, and (c) alternatives for the *Split armature template*$_{SPL1}$.

of it and the Hungarian algorithm is ran on the partitions of the position decompositions. It can be observed that the last step is not performed for the P&O control modality since the decomposition cost already includes both the position and orientation DOFs. At this point, the mapping configurations that allow the animator to control all the DOFs in the target armature are finally generated, considering possible bones with .L/.R previously neglected. Fig. 2.7 shows the configurations created for the target armature presented in Fig. 2.3 by selecting the TUI+NUI-based mapping mode.

Fig. 2.8 shows further sample configurations generated by the Automatic mapping configurator for other virtual characters. Lists of configurations, as well as bones involved are represented with different colors. Arrows indicate the element-to-bone mapping. In Fig. 2.8c, underlined number refers to configurations belonging to the hidden side of the character.



(a) Body's configuration (b) Head's configuration

Figure 2.7: The two configurations created by the automatic procedure to control the (a) body and (b) head of the armature in Fig. 2.3 using the TUI+NUI-based mapping mode. Arrows indicate the mapping between bone and interface elements for each configuration.

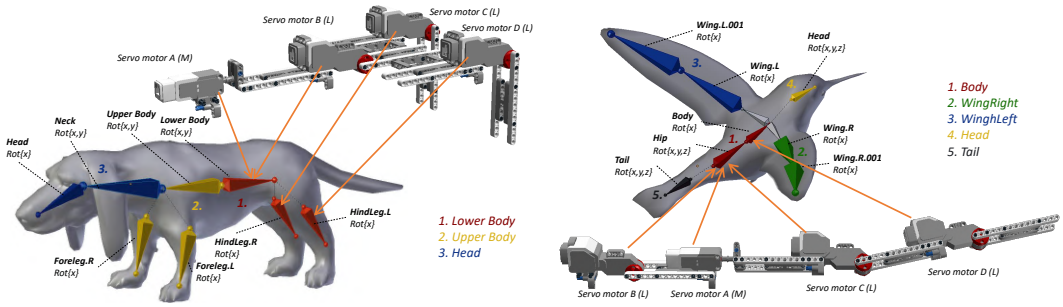
The first three sets of configurations (Fig. 2.8a, Fig. 2.8b and Fig. 2.8c) have been generated by selecting the TUI+NUI based mapping mode and considering three large and one medium servo motors (plus the Kinect-tracked hip joint) available. Fig. 2.8d illustrates the results of the NUI-based mapping mode on a man character.

6. *Assembly generation and instructions* Besides assigning each interface element to a DOF of the target armature, the mapping configuration also defines how the tangible prop has to be assembled. The animator can choose to assemble its own tangible props (respecting the rules defined by the mapping configuration) or alternatively can follow the building instructions generated automatically at the end of the above process. To generate the instructions, each sensor and motor is considered as a part of an "elementary" block that can be connected to other blocks through a set of standard mounting points. Splits can also be generated through this approach. A subset of possible elementary blocks designed for both servo motors and sensors are reported in Fig. 2.9.

   In order to identify the sequence of elementary blocks to be assembled, a breadth-first search on the template armature graph is first performed. Then, the position in the final assembly of each block is determined by translating it in the 3D space (as shown in Fig. 2.10) in order to match the corresponding mounting points. The 3D translation is computed as follows.

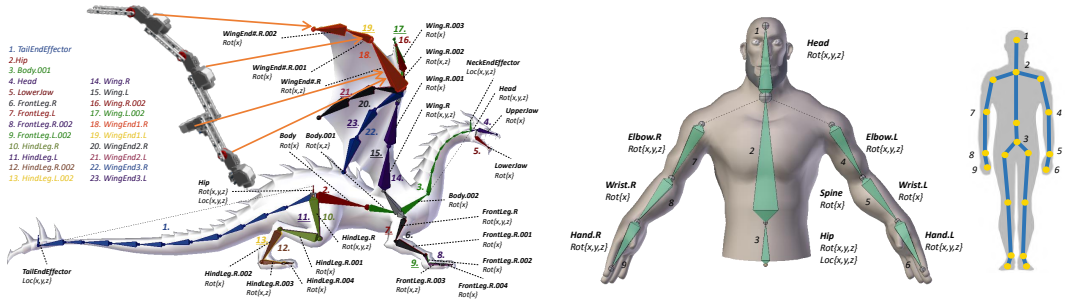
$$POS(i) = POS(par(i)) + O(par(i)) + D(sib(i)) \tag{2.1}$$

   where *par(i)* specify the parent of given block *i*, *POS(par(i))* is a function that returns the 3D coordinates of the parent block, *O(par(i))* is the offset, introduced by the parent block, that has to be considered for reaching the new



(a) Lower Body configuration for a dog character

(b) Body configuration for a bird character



(c) WingEnd1.R configuration for a dragon character

(d) Configuration for a man character

Figure 2.8: Configurations created for further characters using different configurations of the Automatic Mapping Configurator.

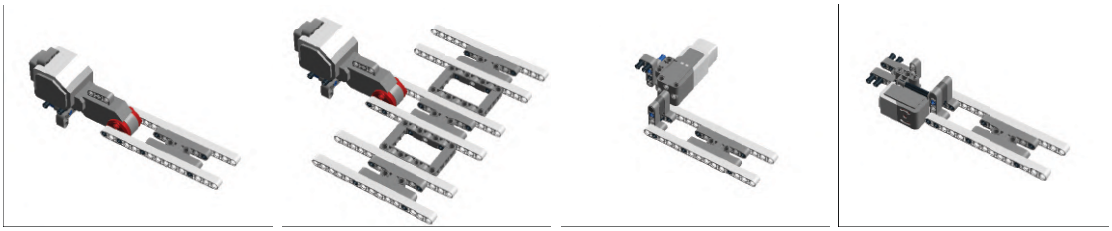
mounting location, *sib(i)* indicates the number of siblings of the ith block, and *D(sib(i))* is the lateral displacement that needs to be added after the parent block for considering split assembly.

The output generated by the above process is a LEGO Digital Designer XML (.LXFML) file. This file contains the 3D model of the resulting tangible prop, which can be visualized through the Lego Digital Designer Tool[10]. The above tool can also be used to automatically generate step-by-step (animated) assembly instructions (a screenshot showing one of the instruction steps is illustrated in Fig. 2.11).

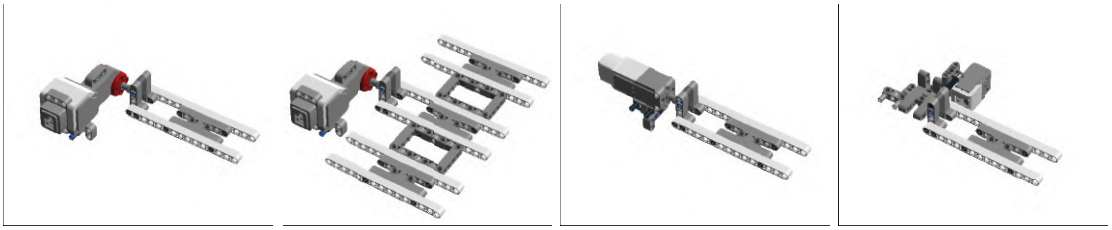
Metrics

Each entry c_{ij} of the cost matrix C is calculated as the sum of several contributions computed with the metrics presented in the following. Each metric provides a value in the range [0,1], which evaluates, both in objective and subjective terms,

[10]Lego Digital Designer Tool: https://www.lego.com/en-us/ldd



(a) Large motor (b) Split L. motor (c) Medium motor (d) Gyro sens.



(e) Large motor (f) Split L. motor (g) Medium motor (h) Gyro sens.

(i) Large motor (j) Split L. motor (k) Medium motor (l) Gyro sens.

Figure 2.9: A subset of the elementary blocks considering the large and medium servo motors as well as the gyro sensor mounted along (a-d) the x axis, (e-h) the y axis, (i-l) for z axis.

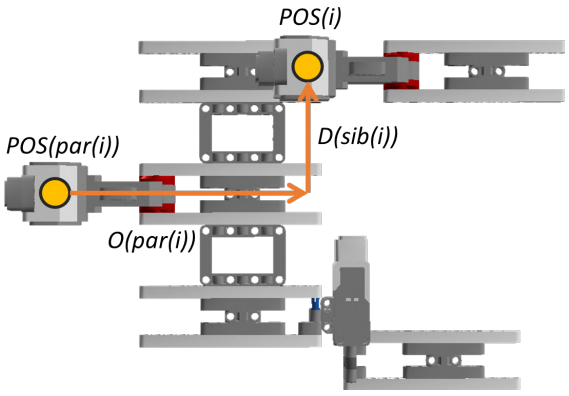


Figure 2.10: A visual representation of the formula for computing the 3D position of an elementary block.

the cost of mapping a bone belonging to a partition of the target armature to a bone of the source armature.

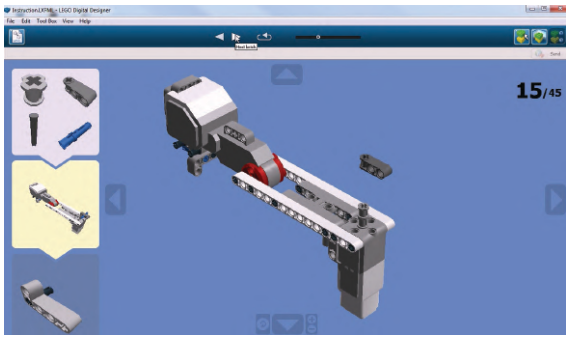


Figure 2.11: Screenshot of the Lego Digital Designer Tool showing the animated step-by-step instructions to build a tangible prop.

1. *Node Similarity*

   This metric measures the topological similarity between the graph-based representation of a source armature (G_S) and the partition of the target armature (G_T). This measure is based on a methodology proposed in [332] that evaluates the structural similarity of neighborhoods. A score vector x_k is computed by iterating equation below

$$x_k \leftarrow (T \otimes S + T^T \otimes S^T + D_{T_I} \otimes D_{S_I} + D_{T_O} \otimes D_{S_O}) \cdot x_{k-1} \tag{2.2}$$

   where $\leftarrow$ represents a vector normalization in the [0,1] range performed with the Frobenius norm in [173], T and S are the adjacency matrices (i.e., square matrices representing a graph) of G_T and G_S, the symbol $\otimes$ indicates the Kronecker's matrix product, D_{S_I}, D_{T_I}, D_{S_O} and D_{T_O} are the diagonal matrices containing the out-degree and the in-degree values for every node in G_T and G_S. As in [265], the number of iterations has been set to 11. Since the initial condition x_0 can be chosen arbitrarily because no prior information about node similarity is available, it has been set conventionally to be the all-ones vector as in [332]. The normalization performed at each iteration sets a value equals to 1 for the entries in the score vector that correspond to nodes that have the same position in the graphs structures. At the end of the computations, a score matrix (N) is obtained by concatenating the score vector resulting from the last iteration. Finally, the cost matrix C is obtained by converting the n_{ij} entries of N into c_{ij} values applying the formula $c_{ij} = 1 - n_{i,j}$. The contribution of this metric is increased when the NUI-based mapping mode is selected, since in this case the topology of the source armature (the Kinect-tracked skeleton) is given and, hence, higher intuitiveness is expected to be achieved when the topology of the target armature matches the Kinect-tracked skeleton.

2. *DOF Coverage*

   The DOF Coverage metric aims to penalize the mapping between bones that do not share DOFs. The contribution of this metric is computed as

$$DOC(b_T, b_S) = 1 - \frac{controllable\_dofs(b_T, b_S)}{dofs} \tag{2.3}$$

   where b_T and b_S are bones of the target and the source armature, respectively, *controllable_dofs()* is a function that returns the number of DOFs in b_T that can be manipulated with b_S, and *dofs* is the sum of DOFs in b_T. For the computation of this metric, all the alternative representations of a bone with three rotation DOFs based on Proper Euler and Tait-Bryan angles are considered for b_T.

3. *Component Range*

   This metric penalizes mapping that proposes the use of components with a small operating range to control DOFs with greater ranges. It is calculated as

$$COR(b_T, b_S) = \sum_{i=0}^{dofs} \frac{range\_cost(dof_i, c_i)}{dofs} \tag{2.4}$$

   where dof_i is the i-th DOF of b_T, c_i is the component of the interface element associated with dof_i, and $range\_cost()$ is a function defined as

$$range\_cost(dof_i, c_i) = 1 - \frac{range(dof_i, c_i)}{max\_range} \tag{2.5}$$

   where $range()$ is a function that, given a specific DOF (dof_i) and a component (c_i), returns the operating range of that component along the DOF, whereas max_range represents the maximum range for the components used. Values returned from the $range()$ function are reported in Tables 2.1 – 2.3. In particular, the range for the tangible components were derived from the user manual of Lego Mindstorm EV3[11] (Table 2.1). For obtaining the Kinect joints, a distinction was made between position and orientation DOFs. Starting from position DOFs (Table 2.2), horizontal and vertical range was computed as $2 \times animator\_dist \times tan(\alpha/2)$ and $2 \times animator\_dist \times tan(\beta/2)$, respectively, where α is the Kinect horizontal viewing angle (57°), β is the Kinect vertical viewing angle (43°), $animator\_dist$ is the distance between

[11] Lego Mindstorm EV3 Education User Guide: https://education.lego.com/en-us/support/mindstorms-ev3/user-guides

the user and the Kinect that is recommended by Microsoft for a stable tracking (3.5m). The depth range is set to the value provided by the Kinect official documentation. For the joints belonging to the lower part and to the center of the Kinect-tracked skeleton (i.e., Hip, Knee, Ankle, Foot, Hip Center, Spine, Shoulder Center, Head), vertical range was reduced to take into account that certain positions would be particularly uncomfortable for the animator (like keeping the hip above the head, or bringing the head to the same height of the feet, etc.). The orientation range of Kinect-tracked joints reported in Table 2.3 has been measured experimentally for each joint. The *max_range* value was set to 4m for position DOFs, which is the nominal range for the Microsoft Kinect working in the Default Range Mode, whereas for orientation DOFs was set to 360°.

Table 2.1: Values used in the computation of the Component Range (COR) metric for the Lego Mindstorm EV3 servo motors and sensors.

Component	Range
Gyro sensor	∞ (360°)
Ultrasonic sensor	2,55m
Large motor	∞ (360°)
Medium motor	∞ (360°)

Table 2.2: Values used in the computation of the Component Range (COR) metric for the position DOFs of the Kinect joints.

Part	Range
Upper joints	Horizontal, 3.8m
	Vertical, 2.79m
	Depth, 3.2m
Lower joints	Horizontal, 3.8m
	Vertical, 0.9m
	Depth, 3.2m
Center joints	Horizontal, 3.8m
	Vertical, 0.45m
	Depth, 3.2m

4. *Component Annoyance*

   This metric is designed to measure the counter-intuitiveness of using a component to manipulate a specific DOF. It is computed as

Table 2.3: Values used in the computation of the Component Range (COR) metric for the orientation DOFs of the Kinect joints.

Joint	Rot(x)	Rot(y)	Rot(z)
Head	85°	0°	60°
Shoulder_Left/Shoulder_Right	60°	0°	10°
Elbow_Left/Elbow_Right	170°	170°	120°
Wrist_Left/Wrist_Right	160°	0°	0°
Hand_Left/Right	50°	0°	50°
Hip_Center	90°	80°	60°
Knee_Left/Knee_Right	120°	0°	90°
Ankle_Left/Ankle_Right	50°	0°	0°

$$COA(b_T, b_S) = \sum_{i=0}^{dofs} \frac{annoyance\_cost(dof_i, c_i)}{dofs} \tag{2.6}$$

where the *annoyance_cost*() function returns a cost value that describes the annoyance for the user to control the considered DOF (dof_i) with a given component (c_i). Values reported in Table 2.4 were obtained empirically by considering several factors that could influence mapping intuitiveness. The first factor considered is referred to as Component Type (CT). Its cost is set to 0 if the component provides orientation (position) DOF(s) and the DOF to be controlled is an orientation (position) one. Otherwise, the cost is set to 1. The second factor considered is named Component Blocking (CB) and it helps to differentiate between active and non-active components. Its cost is set to 0 if the component can be blocked, 1 otherwise. The third factor is referred to as Assembly Easiness (AE), and is set to 0 if the assembly of the component does not require the animator to build complex mounting structures, 1 otherwise. Finally, the fourth factor is named Axis Similarity (AS), and considers the fact that it would be preferable to map a DOF of the Kinect-tracked joint on the same axis (cost set to 0). Servo motors and sensors are characterized by only one DOF: servo motors and gyro sensors are considered as providing a generic orientation DOF, ultrasonic sensors are assumed to offer a generic position DOF.

5. *Position in Chain*

   This metric aims to penalize the assignment of bones that take up different positions in the two kinematics chains. Hence, it is calculated as

$$PIC(b_T, b_S) = \frac{|level(b_T) - level(b_S)|}{max\_level} \tag{2.7}$$

Table 2.4: Annoyance costs estimated for the interface elements used in the computation of the Component Annoyance (COA) metric.

DOF	CT	CB	AE	AS	Cost	DOF	CT	CB	AE	AS	Cost
Large Motor						**Medium Motor**					
Rot(x)	0	0	0	0	0	Rot(x)	0	0	1	0	0.25
Rot(y)	0	0	1	0	0.25	Rot(y)	0	0	0	0	0
Rot(z)	0	0	1	0	0.25	Rot(z)	0	0	1	0	0.25
Loc(x)	1	0	1	1	0.75	Loc(x)	1	0	1	1	0.75
Loc(y)	1	0	1	1	0.75	Loc(y)	1	0	1	1	0.75
Loc(z)	1	0	1	1	0.75	Loc(z)	1	0	1	1	0.75
Gyro sensor						**Ultrasonic sensor**					
Rot(x)	0	1	1	0	0.5	Rot(x)	1	1	0	1	0.75
Rot(y)	0	1	1	0	0.5	Rot(y)	1	1	0	1	0.75
Rot(z)	0	1	1	0	0.5	Rot(z)	1	1	0	1	0.75
Loc(x)	1	1	1	1	1	Loc(x)	0	1	0	1	0.5
Loc(y)	1	1	1	1	1	Loc(y)	0	1	0	0	0.25
Loc(z)	1	1	1	1	1	Loc(z)	0	1	0	1	0.5
Kinect joint(x)						**Kinect joint(y)**					
Rot(x)	1	1	0	0	0.5	Rot(x)	1	1	0	1	0.75
Rot(y)	1	1	0	1	0.75	Rot(y)	1	1	0	0	0.5
Rot(z)	1	1	0	1	0.75	Rot(z)	1	1	0	1	0.75
Loc(x)	0	1	0	0	0.25	Loc(x)	0	1	0	1	0.5
Loc(y)	0	1	0	1	0.5	Loc(y)	0	1	0	0	0.25
Loc(z)	0	1	0	1	0.5	Loc(z)	0	1	0	1	0.5
Kinect joint(z)											
Rot(x)	1	1	0	1	0.75						
Rot(y)	1	1	0	1	0.75						
Rot(z)	1	1	0	0	0.5						
Loc(x)	0	1	0	1	0.5						
Loc(y)	0	1	0	1	0.5						
Loc(z)	0	1	0	0	0.25						

where *level*() is a function returning the total number of bones between the bone passed as input and its root, and *max_level* indicates the maximum depth that can be reached by exploring the graph of the target armature. In order to compute this metric for the Kinect-tracked skeleton, it is considered as made up of five different chains, as reported in Table 2.5, where the first bone represents the root.

6. *Symmetry*

Table 2.5: Chains of the Kinect skeleton considered in the computation of the Position in Chain (PIC) metric.

Chain	Bones
Upper right	Shoulder_Right
	Elbow_Right
	Wrist_Right
	Hand_Right
Upper left	Shoulder_Left
	Elbow_Left
	Wrist_Left
	Hand_Left
Center	Hip_Center
	Spine
	Shoulder_Center
	Head
Lower right	Hip_Right
	Knee_Right
	Ankle_Right
	Foot_Right
Lower left	Hip_Left
	Knee_Left
	Ankle_Left
	Foot_Left

The cost computed by this metric is $SYM(b_T, b_S) = 0$ if the two bones b_T and b_S present the same symmetry (for example, *BoneSrcArm1.R* and *BoneTrgArm1.R*), $SYM(b_T, b_S) = 1$ if they have associated opposite symmetry (for example, *BoneSrcArm1.L* and *BoneTrgArm1.R*), and $SYM(b_T, b_S) = 1$ if only one of the two bones has a symmetry assigned (for example, *BoneSrcArm1* and *BoneTrgArm1.R*). With respect to the bones of the Kinect-tracked skeleton belonging to the arms and legs chains, they have a symmetry depending on which side of the skeleton they are placed, whereas bones of the central chain, i.e., *Hip_Center*, *Spine*, *Shoulder_Center* and *Head*, have not symmetries associated.

7. *Partition Count*

   This metric is in charge of penalizing the creation of a large number of configurations, which would mean dealing with partitions characterized by a small number of bones. This condition should be avoided since it can generate a high cognitive load for the user, who would be required to remember many

voice commands. The contribution of this metric is computed as

$$PAC(b_T, b_S) = \frac{partitions(b_T)}{max\_partitions} \tag{2.8}$$

where *partitions()* is a function that returns the total number of partitions generated by the decomposition to which the bone specified as input belongs to, and *max_partitions* is the number of partitions contained in the decomposition with the largest number of partitions.

2.2.4 Experimental evaluation

In order to assess the effectiveness of the designed system, a user study was carried out in [173]. In this section, the experimental setup is first described. Then, the performance metrics will be presented.

Case studies

In order to investigate separately different aspects of the proposed solution, four case studies were designed, each characterized by different levels of complexity.

In the first case study, the devised task (in the following referred to as the $lamp_{ref}$ task) focused on the creation of a simple animation (composed of three poses/ keyframes) for a lamp character. The lamp is characterized by four moving bones with seven DOFs to be controlled, and it was selected in order to intentionally keep low the complexity of the armature considered. Fig. 2.12 shows the unique configuration created for the target armature by the Automatic mapping configurator selecting the TUI+NUI-based mapping mode. The poses to be created (illustrated in Fig. 2.13) request the animator to control both the articulation of a number of joints as well as the absolute position in the virtual space of the character. Voice commands had to be pronounced in order to advance the timeline and insert keyframes. A video showing the $lamp_{ref}$ task is available for download[12]

The second task, named $crocodile_{ref}$ task, aimed to evaluate the performance of the animators in the articulation of a medium complexity armature. The armature of the selected character (i.e., a crocodile) includes 16 moving bones with 24 DOFs to be controlled using both inverse and forward kinematics. The configurations to manipulate the crocodile character were created by selecting the TUI+NUI-based mapping mode and considering the availability of one ultrasonic sensor, two large servo motors, and the Kinect-tracked hip joint. The six configurations generated by the Automatic mapping configurator are reported in Fig. 2.14. Voice commands allow animators to change the set of manipulated bones by activating one of the

[12] $lamp_{ref}$ task: https://youtu.be/h-4GBxjtvgU

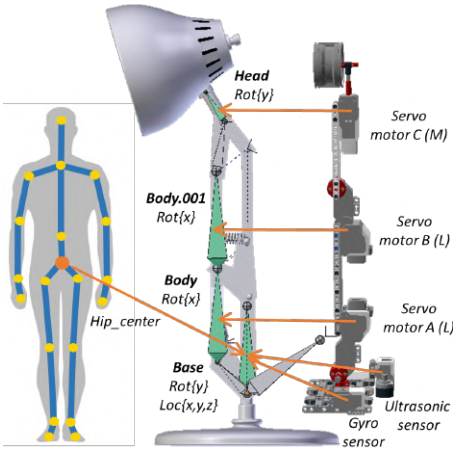


Figure 2.12: Configuration created for the target armature used in the $lamp_{ref}$ task, and mapping on the interface elements.

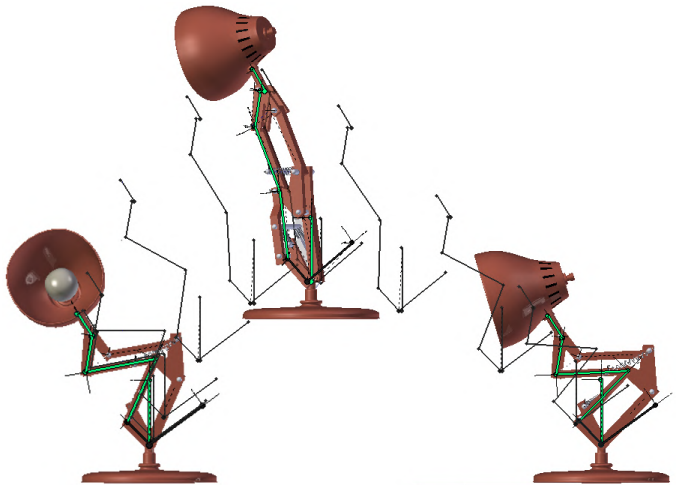
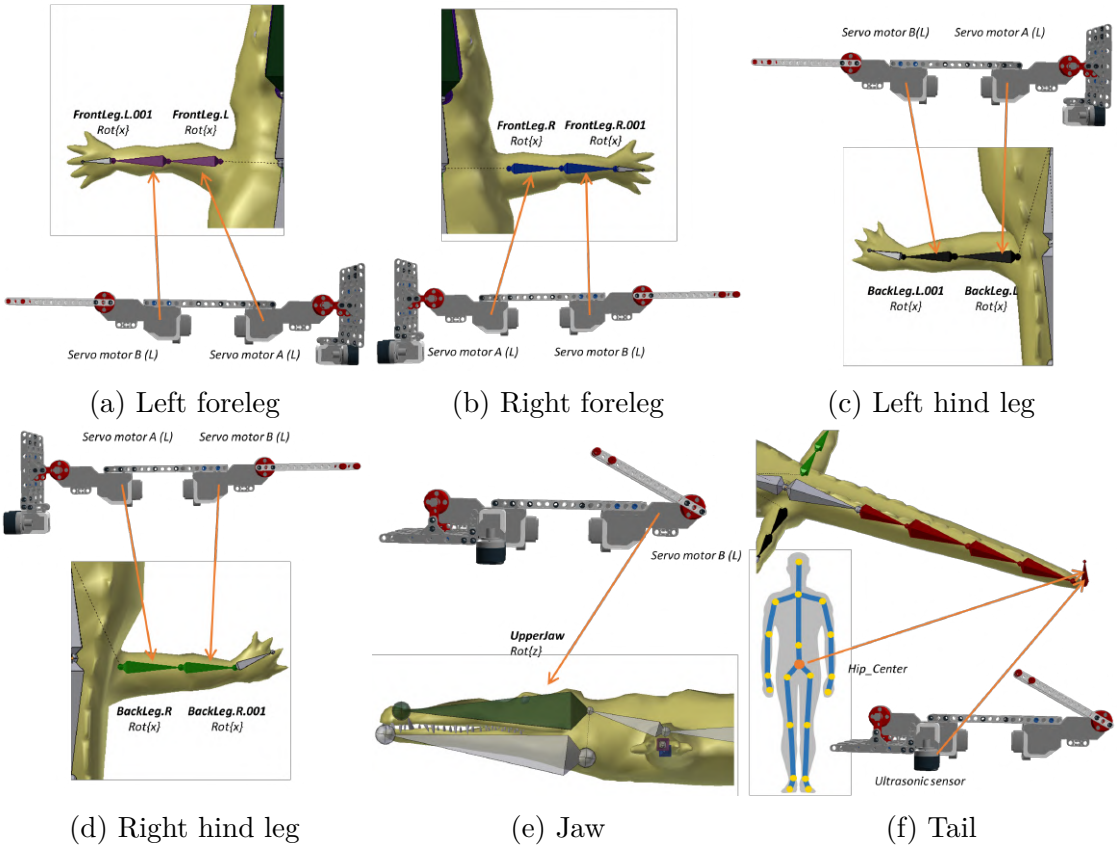


Figure 2.13: The three poses/keyframes to be created by participants during the $lamp_{ref}$ task.

available configurations. The target pose to be realized is shown in Fig. 2.15. A video showing the above task is available for download[13]

In the third use case ($dyno_{ref}$ task) a much more complex armature of a dyno character was considered, presenting 38 moving bones with 88 DOFs. Fig. 2.16 shows the 13 configurations generated automatically by considering a set of interface elements that includes one medium servo motor, three large motor plus the

[13] $crocodile_{ref}$ task: https://youtu.be/ODfcjeJAaiU



(a) Left foreleg (b) Right foreleg (c) Left hind leg

(d) Right hind leg (e) Jaw (f) Tail

Figure 2.14: Configurations created for the target armature used in the $crocodile_{ref}$ task, and mapping on the interface elements. Configurations were activated through the given voice commands.

Kinect-tracked hip joint and the TUI+NUI mapping mode. Colors are used to represent different configurations and the corresponding voice commands. Similarly to the first task ($lamp_{ref}$), also the third task required the animator to change the current frame in the timeline in order to match/recreate two different reference poses (illustrated in Fig. 2.17. A video is available for download[14]

Finally, the last case study aimed to investigate the creation of animations in a usage scenario closer to a real one. Thus, in this task ($dyno_{free}$ task) the constraint to reproduce provided reference poses was removed. Animators were requested to produce from scratch five different poses for the dyno character presented in the previous task. The only constraint set was to move (not necessarily in all the five keyframes) all the various parts of the character (torso, arms, legs, head, and tail). A video showing an animator creating a walk cycle animation is available for

[14] $dyno_{ref}$ task: https://youtu.be/Lfs0MUohxd8

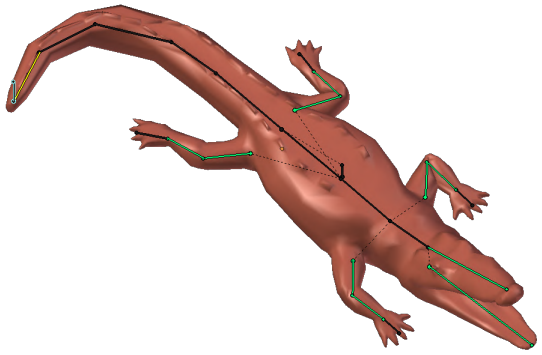


Figure 2.15: The character's target pose to be created by participants during the $crocodile_{ref}$ task.

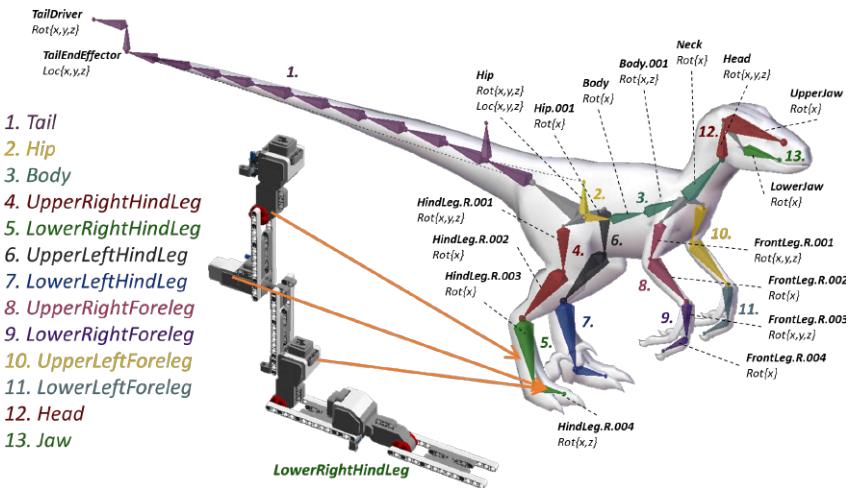


Figure 2.16: Configurations created for the target armature used in the $dyno_{ref}$ and $dyno_{free}$ tasks and an example of bone-to-element mapping for the LowerRightHindLeg configuration. Configurations were activated through voice commands.

download[15].

[15] $dyno_{free}$ task: https://youtu.be/DozMaDil4y4

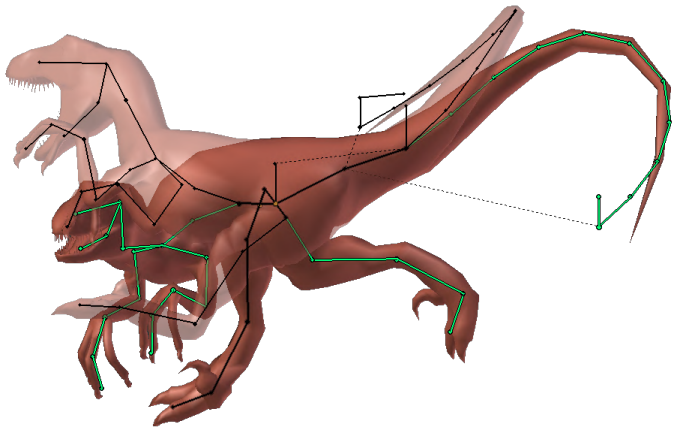


Figure 2.17: The two poses/keyframes to be created by participants during the $dyno_{ref}$ task.

Participants and procedure

The four experiments were carried out by involving 52 participants (32 for the first and second task, 20 for the third and fourth task), aged between 20 and 42 years. Participants were selected among the students and staff of Politecnico di Torino with different skills and backgrounds. In particular, half of the participants reported familiarity with one or more 3D graphics animation suites and they attended/taught at least a computer animation course. These participants were considered as skilled users (SKUs). The remaining participants were considered to be unskilled users (UNUs), since they had no experience in the computer animation field. The choice of involving two kinds of users made it possible to explore the effectiveness of the devised system for a wide spectrum of usage conditions. A within-subjects study design approach was used; hence, each participant was requested to complete the tasks by using both the proposed systems and the native Blender's interface based on M&K. In order to avoid learning effects, the order of the interface to start with was defined in a way that half of the participants started the experiment with the proposed system, whereas remaining participants used first the M&K interface. Before starting the tasks, participants were asked to undergo a preliminary training phase aimed to help them to get familiar with tangible and natural interface. In particular, in this phase, users were free to manipulate armatures similar to those used in the tasks. UNUs were also introduced to the basics of computer animation and to the use of the M&K interface with the aim of letting them get acquainted also with this interaction mean. In all the devised tasks, participants started to work with the character in the rest pose. Before starting to manipulate every character's parts, participants had to perform a reset of the configuration with the aim to register the current pose of the tangible and natural interface to the rest pose of the virtual character. This operation can be done by

pronouncing a dedicated voice command after having visually matched the virtual character's part to be controlled and the corresponding interface elements to be used. This made it possible to build a direct and straightforward perception of the current character's pose, since the animator had the feeling of manipulating bone chains with a device that presents the same articulated joints. At the end of this reset procedure, participants had to iteratively pass through the following steps to complete the task:

1. manipulate the position and/or orientation of the controlled armature parts until target armature perfectly overlaps the reference armature;
2. activate a different configuration to control another set of bones;
3. insert a keyframe;
4. move to the next keyframe and iterate the above steps until the animation is completed.

As done in previous work ([142, 330]), no minimum accuracy threshold nor time limit were set a priori. Participants were free to decide when they completed their task, either because they felt that the pose could not be further improved, or because they believed that the pose was properly replicated. However, during the execution of the $lamp_{ref}$ and $crocodile_{ref}$ task, an audio signal produced by the system informed participants that the difference between the controlled parts and the target pose went below a given threshold. For the fourth task, since there were no specific poses to be realized, it was left to the participants to decide when animation could be considered as completed.

Performance indicators

The evaluation was conducted considering both objective and subjective measures. Performance indicators presented below are those used in the objective evaluation of the first three tasks, since a measurable goal (the references poses to be reproduced) was defined. The set of indicators adopted in [173] were an extension of the set already used in [142], defined to consider the possibility offered by the devised system to also control the bones' position.

The first indicator considered is the *completion time (T)*, which measures the time needed by a participant to execute the assigned task, i.e., to reproduce with the given interface the requested poses in all the keyframes. The completion of the task was identified by the insertion of the last keyframe. The time spent to navigate the timeline was neglected.

The second indicator, named *animation accuracy (A)*, evaluates the proximity of the pose obtained by the manipulated armature with respect to the reference. The values of this indicator range from 0% (the two armatures perfectly overlap)

to 100% (the distance between the two armatures measured at the beginning of the interaction). For each keyframe, the animation accuracy indicator is calculated as the average of the normalized Euclidean and angular distances between each bone of the controlled armature and the corresponding bone of the reference armature. The metric is computed as

$$D = \frac{1}{2} \cdot \sum_{i=0}^{n} \left(\frac{\delta_i}{\Delta} + \frac{\theta_i}{\Theta} \right) \tag{2.9}$$

where n is the total number of bones, δ_i and θ_i represent respectively the Euclidean and the angular distance measured between the i-th bones of the two armatures; Δ and Θ are two normalization factors calculated as

$$\Delta S = \sum_{i=0}^{n} \delta_i^*, \qquad \Theta = \sum_{i=0}^{n} \theta_i^* \tag{2.10}$$

where δ_i^* and θ_i^* are the Euclidean distance and absolute angular displacement between the position/orientation of the i-th bone when set in rest pose and its target position/orientation. When the two terms Δ and Θ are equal to 0, i.e., no position or orientation changes to the bone are necessary, the contribution of the corresponding term is neglected.

The last indicator is the *amount of work (W)*, and it estimates the work necessary to complete the animation task providing a flavor of a how simple or complex it is reaching a given goal [142]. Small values denote a quick convergence to the target pose; conversely, large values are to be interpreted as an indication of a slow update which prevents animators from rapidly reaching the reference pose. It is calculated as

$$W = \frac{1}{2 \cdot T} \int_0^t \sum_{i=0}^{n} \left(\frac{\delta_i(t)}{\Delta} + \frac{\theta_i(t)}{\Theta} \right) dt \tag{2.11}$$

where $t = 0$ at the time in which the task starts, $t = T$ is the maximum between the two times spent by the user to reach the minimum animation accuracy. Fig. 2.18 shows the progressive decrease of the animation accuracy measured for a user carrying out the $dyno_{ref}$ task with both the proposed and the M&K interface. The amount of work can be visually interpreted as the area below the two curves.

With respect to subjective aspects, the usability of the proposed system was evaluated by asking participants to complete a post-test questionnaire based on the ISO 9241-400 standard (which supports the investigation of ergonomics and human factors for physical input devices used within interactive systems). Questions related to the perceived accuracy, operation speed, physical effort, mental effort and intuitiveness experienced with the proposed and the M&K interface were rated on a seven-points Likert scale. Finally, additional comments specific to the pros, and cons of the two interfaces were additionally collected. The same questionnaire was

administered to the 32 participants executing the $lamp_{ref}$ and $crocodile_{ref}$ task and the 20 participants involved in the $dyno_{ref}$ and $dyno_{free}$ task. For the $dyno_{free}$ task, it was decided to collect only the overall user preference for either the proposed or the M&K interface since animations generated by participants could not be directly compared.

2.2.5 Results

In this section, results obtained in the user study are discussed by first considering the objective measurements, then focusing on subjective evaluations collected through the post-test questionnaire.

Objective evaluation

Results obtained by UNUs in terms of completion time, animation accuracy and amount of work are reported for the execution of the $lamp_{ref}$, $crocodile_{ref}$ and $dyno_{ref}$ tasks in Fig. 2.19. Results shown for both the proposed and the M&K interface have been computed by averaging the value of the collected metrics on all the keyframes.

Statistical significance was studied through paired Student's t-tests ($p < 0.05$). In order to test the normality of data, in this thesis, it was decided to adopt the graphical interpretation and not statistical tests, such as the Kolmogorov–Smirnov and the Shapiro–Wilk test, since this approach is more suitable in the cases whereby numerical methods might be over- (at small sample size) or under-sensitive (at large sample size) [339, 217]. With the aim of fully reporting the results of Student's t-tests, it was chosen to indicate the degrees of freedom, t statistics, and the effect size (i.e., Cohen's d), in addition to the significance value (p-values are truncated to two decimal digits). The same statistics will be reported in the following of this thesis, except for the cases requiring the use of different scores (that will be first introduced).

On average, users were significantly faster with the proposed interface than with M&K. In particular, users were 35% faster (M&K: $M = 20$ s, $SD = 6$ s; Prop.: $M = 13$ s, $SD = 7$ s; $t(47) = 6.34$, $p < 0.01$, $d = 1.08$) for the $lamp_{ref}$ task; 20% faster (M&K: $M = 47$ s, $SD = 11$ s; Prop.: $M = 38$ s, $SD = 9$ s; $t(15) = 3.47$, $p < 0.01$, $d = 0.88$) for the $crocodile_{ref}$ task; 30% faster (M&K: $M = 7$ min 5 s, $SD = 2$ min 26 s; Prop.: $M = 5$ min 0 s, $SD = 1$ min 42 s; $t(19) = 6.34$, $p < 0.01$, $d = 0.99$) for the $dyno_{ref}$ task, when they used the proposed interface. Similar considerations were confirmed also for the amount of work, which was, on average, significantly lower with the proposed interface than with M&K in the $lamp_{ref}$ (M&K: $M = 61.31\%$, $SD = 6.73$; Prop.: $M = 38.23\%$, $SD = 9.76$; $t(47) = 14.92$, $p < 0.01$, $d = 2.75$), $crocodile_{ref}$ (M&K: $M = 64.91\%$, $SD = 6.01$; Prop.: $M = 54.95\%$, $SD = 8.38$; $t(15) = 3.85$, $p < 0.01$, $d = 2.75$), and $dyno_{ref}$ (M&K: $M = 51.48\%$,

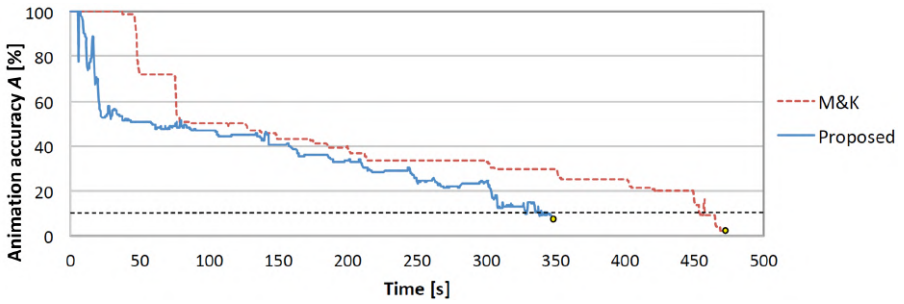


Figure 2.18: Example of amount of work for a user approaching the first pose/frame of the $dyno_{ref}$ task. Horizontal dashed line represents the 10% threshold on animation accuracy.

$SD = 13.86$; Prop.: $M = 43.53\%$, $SD = 4.93$; $t(19) = 3.25$, $p < 0.01$, $d = 2.75$) task. However, the advantages coming from the proposed solution were paid in term of animation accuracy, that resulted to be significantly lower compared to the M&K interface in the $lamp_{ref}$ (M&K: $M = 2.18\%$, $SD = 1.84$; Prop.: $M = 6.28\%$, $SD = 3.31$; $t(47) = -6.80$, $p < 0.01$, $d = -1.51$), $crocodile_{ref}$ (M&K: $M = 4.02\%$, $SD = 1.65$; Prop.: $M = 7.58\%$, $SD = 4.15$; $t(15) = -4.23$, $p < 0.01$, $d = -1.13$), and $dyno_{ref}$ (M&K: $M = 4.19\%$, $SD = 5.71$; Prop.: $M = 9.77\%$, $SD = 3.65$; $t(19) = -3.74$, $p < 0.01$, $d = -1.13$) task.

Regarding results achieved by SKUs that are depicted in Fig. 2.19, it was not possible to find statistically significant differences for the completion time in the $crocodile_{ref}$ and $dyno_{ref}$ tasks. However, a significant difference was found for the $lamp_{ref}$ task, in which participants performed 25% faster with the proposed interface than with M&K (M&K: $M = 17$ s, $SD = 6$ s; Prop.: $M = 13$ s, $SD = 6$ s; $t(47) = 3.69$, $p < 0.01$, $d = 0.71$). Similarly to UNUs, also for SKUs the amount of work was significantly lower with the proposed interface in the $lamp_{ref}$ (M&K: $M = 59.54\%$, $SD = 7.32$; Prop.: $M = 38.94\%$, $SD = 11.49$; $t(47) = 11.59$, $p < 0.01$, $d = 2.14$), $crocodile_{ref}$ (M&K: $M = 63.32\%$, $SD = 4.61$; Prop.: $M = 56.21\%$, $SD = 7.67$; $t(15) = 3.31$, $p < 0.01$, $d = 1.12$), and $dyno_{ref}$ (M&K: $M = 51.46\%$, $SD = 14.19$; Prop.: $M = 41.96\%$, $SD = 6.94$; $t(19) = 3.24$, $p = 0.03$, $d = 0.85$) task. Regarding the animation accuracy, it can be observed that the accuracy of the final pose was still lower in the $lamp_{ref}$ (M&K: $M = 1.71\%$, $SD = 1.32$; Prop.: $M = 6.05\%$, $SD = 3.89$; $t(47) = -8.10$, $p < 0.01$, $d = -1.50$), $crocodile_{ref}$ (M&K: $M = 2.72\%$, $SD = 1.60$; Prop.: $M = 7.38\%$, $SD = 3.17$; $t(15) = -5.19$, $p < 0.01$, $d = -1.86$), and $dyno_{ref}$ (M&K: $M = 2.99\%$, $SD = 1.80$; Prop.: $M = 8.08\%$, $SD = 3.60$; $t(19) = -6.29$, $p < 0.01$, $d = -1.79$) task.

By analyzing further the above results it is possible to notice that advantages of using the proposed interfaces were more evident in the $lamp_{ref}$ task, since it allowed the animators to simultaneously control both the articulation of the character and its position in the virtual space (possibility not considered in the other two tasks),

helping them to be more effective.

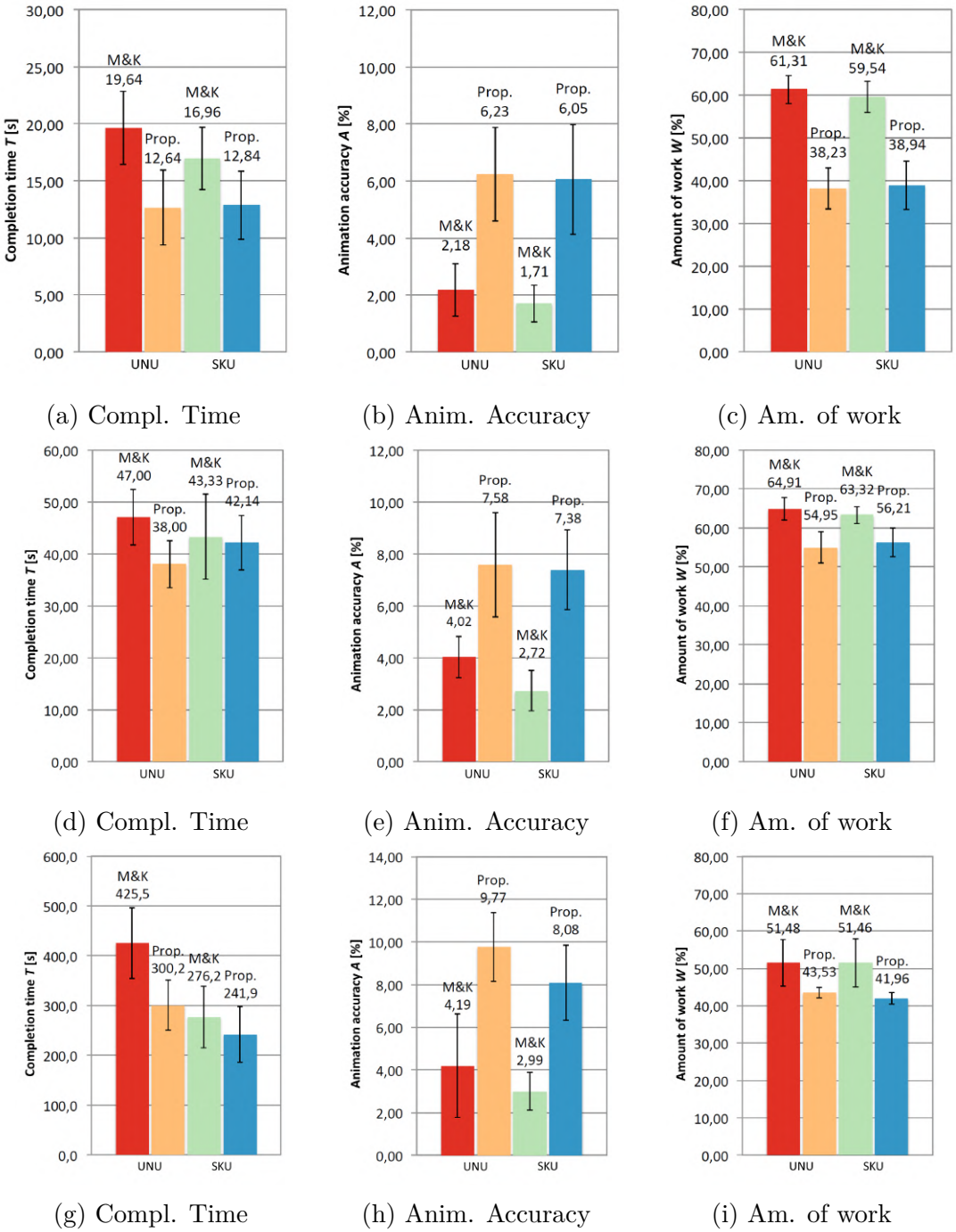
Comparing values of the operation speed indicator achieved by SKUs and UNUs it can be noticed that the gain was lower for the first category than the second one. This fact can be explained by two observations. The first observation is that, by focusing on the M&K interface, SKUs users were, on average, faster than UNUs in completing the tasks, probably because they were accustomed to work with the used software with M&K. On the other hand, observing the operation speed with the proposed interface, it is possible to notice that differences between SKUs and UNUs were lower than with M&K. These observations can explain why SKUs have benefited from the proposed system less than UNUs, and they can be regarded as a confirmation of the effectiveness of the training phase in clearing differences among the two groups. Finally, it is worth observing that small differences in terms of operation speed between the two interfaces obtained by SKUs represent a promising result for the proposed interface since they indicate that, although these users had previous experience only with M&K, they were able to execute the tasks in almost the same time with both the interfaces. With respect to the animation accuracy indicator, SKUs confirmed their abilities, since they were more accurate than UNUs in all the tasks. In general, animation accuracy was lower with the proposed interface than with the M&K interface. This result probably depends on the limited sensitivity of the tangible bricks used[16]. It is worth observing that the audio signal notification had no influence on the operation speed and animation accuracy, since participants decided to continue articulating the character by spending 13% and 5% more of the overall posing time in the $lamp_{ref}$ and $crocodile_{ref}$ (i.e., tasks with audio notification) and 4% for the $dyno_{ref}$ task (without audio notification).

Interesting outcomes can be obtained by comparing results in terms of amount of work for SKUs with the M&K interface with those obtained by UNUs with the proposed one. In fact, it can be noticed that UNUs were able to complete the tasks with a lower amount of work than SKUs. When comparable completion times were recorded, the fact that the amount of work was lower means that the proposed system allowed UNUs to quickly draft a rough pose of the character but made them spend more time to further refine it. In other words, UNUs with the proposed interface were faster than SKUs with M&K to approach the target pose, but this advantage was paid by a lower accuracy in the final pose.

Subjective evaluation

Fig. 2.20 reports subjective results obtained after the completion of the tasks; results were obtained by averaging scores among the participants and remapping

[16]Lego Mindstorm Education user guide: http://www.nr.edu/csc200/labs-ev3/ev3-user-guide-EN.pdf



(a) Compl. Time (b) Anim. Accuracy (c) Am. of work

(d) Compl. Time (e) Anim. Accuracy (f) Am. of work

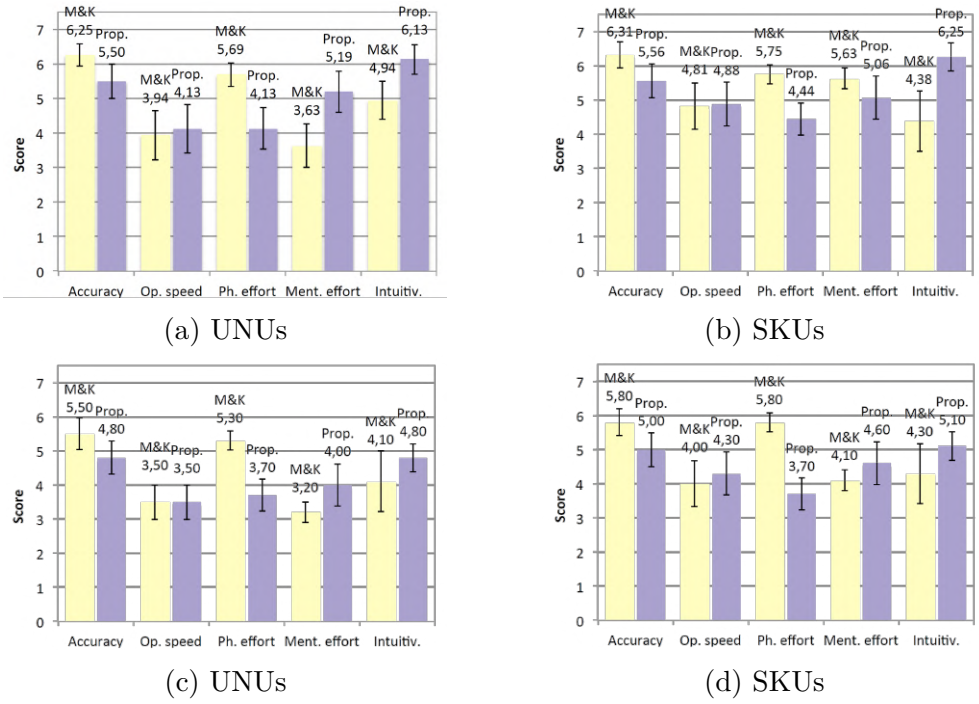
(g) Compl. Time (h) Anim. Accuracy (i) Am. of work

Figure 2.19: Results in terms of completion time, animation accuracy and amount of work for the (a–c) $lamp_{ref}$, (d–f) $crocodile_{ref}$ and (g–i) $dyno_{ref}$ obtained by both UNUs and SKU.

them on a better-to-worse (7-to-1) scale. The remapping of the averages on a different scale was done for the sake of clarity.

As with the objective evaluation, statistical significance was studied through paired Student's t-tests ($p < 0.05$). Although non-parametric tests, e.g., the Wilcoxon Signed Rank or the Kruskal-Wallis Test, are considered to be more suited for this type of data, in this thesis it was decided to make use of Student's t-tests as suggested by [270]. From the analysis of the results without distinguishing between the two user categories, it can be noticed that both UNUs and SKUs perceived the proposed interface as more intuitive than M&K in the $lamp_{ref}$ and $crocodile_{ref}$ task (UNUs – M&K: $M = 4.84$, $SD = 1.12$; Prop.: $M = 6.13$, $SD = 0.89$; $t(15) = -4.06$, $p < 0.01$, $d = -1.17$; SKUs – M&K: $M = 4.38$, $SD = 1.82$; Prop.: $M = 6.25$, $SD = 0.86$; $t(15) = -4.03$, $p < 0.01$, $d = -1.32$) and in the $dyno_{ref}$ task (UNUs – M&K: $M = 4.10$, $SD = 1.40$; Prop.: $M = 4.80$, $SD = 0.76$; $t(9) = -2.24$, $p = 0.03$, $d = -0.62$; SKUs – M&K: $M = 4.30$, $SD = 1.38$; Prop.: $M = 5.10$, $SD = 0.72$; $t(9) = -2.49$, $p = 0.02$, $d = -0.73$). Feedback collected through the open questions suggests that this result is related to the greater awareness of the virtual character to be controlled and to the affordance of the proposed interface. However, the proposed interface was also characterized by a higher physical effort as reported for the $lamp_{ref}$ and $crocodile_{ref}$ task (UNUs – M&K: $M = 5.69$, $SD = 0.70$; Prop.: $M = 4.13$, $SD = 1.26$; $t(15) = 4.28$, $p < 0.01$, $d = 1.53$; SKUs – M&K: $M = 5.75$, $SD = 0.58$; Prop.: $M = 4.44$, $SD = 0.96$; $t(15) = 4.20$, $p < 0.01$, $d = 1.65$) and the $dyno_{ref}$ task (UNUs – M&K: $M = 5.30$, $SD = 0.48$; Prop.: $M = 3.70$, $SD = 0.82$; $t(9) = 6.00$, $p < 0.01$, $d = 2.37$; SKUs – M&K: $M = 5.80$, $SD = 0.63$; Prop.: $M = 3.70$, $SD = 0.82$; $t(9) = 9.00$, $p < 0.01$, $d = 2.86$). This was probably due to the need to hand the tangible assembly (with its size and weight) and by the requirement to stand when the animator's skeleton has to be tracked. Finally, participants from both user categories perceived the proposed interface as less accurate than the M&K one, as already revealed by the objective evaluation. These results were observed only in the $lamp_{ref}$ and $crocodile_{ref}$ task (UNUs – M&K: $M = 6.25$, $SD = 0.68$; Prop.: $M = 5.50$, $SD = 1.03$; $t(15) = 2.53$, $p = 0.02$, $d = 0.86$; SKUs – M&K: $M = 6.31$, $SD = 0.79$; Prop.: $M = 5.56$, $SD = 1.03$; $t(15) = 2.81$, $p = 0.01$, $d = 0.82$).

Upon further analysis of the results of UNUs, it can be observed that they thought that the proposed interface performed significantly better in terms of mental effort in the $lamp_{ref}$ and $crocodile_{ref}$ (M&K: $M = 3.63$, $SD = 1.31$; Prop.: $M = 5.19$, $SD = 1.22$; $t(15) = -3.83$, $p < 0.01$, $d = -1.23$) and $dyno_{ref}$ (M&K: $M = 3.20$, $SD = 0.63$; Prop.: $M = 4.00$, $SD = 0.94$; $t(9) = -2.45$, $p = 0.01$, $d = -1.00$) task. No statistically significant results were obtained for the perceived operation speed, although objective results contrast with these findings. This outcome may be related to the additional time spent by the users to refine their results after reaching a rough pose, which was due to the limited accuracy of the tangible prop that forced them to further adjust the pose trying to get closer to the reference



(a) UNUs (b) SKUs

(c) UNUs (d) SKUs

Figure 2.20: Subjective results comparing the proposed and M&K interfaces for the execution of the (a, b) $lamp_{ref}$, $crocodile_{ref}$, and (c, d) $dyno_{ref}$ tasks. For mental and physical effort, scores have been inverted (hence higher scores indicate a lower effort i.e., a better result).

without actually improving in a noticeable way the animation accuracy. These corrections probably made them believe that the task was a longer process.

With respect to SKUs, no statistically significant differences were found for the perceived operation speed and mental effort, possibly due to the fact that the benefits of the proposed interface are limited by the fact that users were already accustomed to the M&K interface but not to the tangible and natural interfaces.

Preferences collected for the fourth task are aligned with the subjective results described above. In fact, starting from UNUs, only three out of the 10 participants expressed their preferences for the M&K interface. Preferences of SKUs are equally distributed between the two interfaces; notwithstanding, this can be interpreted as a promising result given their better knowledge of the M&K interface. In this thesis, it was decided to report only the percentages, although statistical tests, e.g., the Friedman test, could be applied for a more in-depth analysis.

2.2.6 Future developments

Considering the limitations of the proposed system described in the discussion above, future work may consider further improving the integration of body tracking

and TUIs. For example, the challenge of occlusions could be solved by adopting wearable technologies, whereas the limited accuracy of the affordable off-the-shelf input elements could be tackled by considering, for instance, voice commands (to provide the animator the possibility to fine-tune the mapping during the animation). Another aspect which was not fully considered in [173] concerns the availability of active components (i.e., servo motors), that could be used to inform the animator of rig constraints or to reload pre-recorded configurations. Finally, efforts could be focused on improving the process that allows the animator to register the interface elements with the character's part to be controlled. This could be done by making use of algorithms able to consider relations between sizes of physical and virtual components or by adopting technologies like, e.g., immersive virtual reality and/or augmented reality.

2.3 Multimodal interfaces based on live performance and natural language processing

The analysis of the literature concerning 3D content creation reveals that, so far, solutions have focused primarily on the production phase, which conventionally encompasses art design, modeling, shot layout, animation, lighting and rendering tasks [121, 170, 276, 324]. However, the creative development phase has attracted the interest of the research community, since this is the phase where the capacity to quickly test different options becomes fundamental for converging to the best result in the shortest time possible [281]. In this phase, users who lack animation skills (e.g., directors, screenwriters, etc.) could benefit from the availability of tools allowing them to quickly create the draft of the intended 3D scene [235]. Similar needs also exist in other contexts. For example, computer-animated storytelling techniques, which arose over the last years, could benefit from simplified interfaces letting stakeholders with limited computer animation skills including, like educators, children, etc. to develop animations with some loss of quality [20, 116, 120, 161, 162].

Several animation techniques that can be used both by animators and generic users have been proposed. An example is those techniques based on natural interaction mechanisms, such as touch or in-the-air gestures [170], as well as speech and voice commands [208]. Another example is provided by techniques that make use of NLP methods to parse a description of the scene to be animated and recreate it using a library of objects, characters and predefined actions [191, 235, 278]. Some of these systems were designed to produce an output that can be refined later, whereas others aimed to provide animators with tools to create the final animations through a live and interactive performance. Nonetheless, in all of these solutions, it is possible to find a number of limitations. For instance, natural interfaces proved to be extremely effective for posing characters, but they require the

user to manage the rig that controls the geometry at each keyframe [173]. Fully automatic NLP-based techniques could produce unrealistic animations and require the full description of the scene and additional information such as characters' and objects' position, action's duration, etc. to be made explicit in the text [235].

In the following, the character animation system originally presented in [172] is discussed. The system allows the user to have a direct control on character's position and orientation in space and time by leveraging the performance-based animation approach. However, in order to cope with issues regarding the animation of non-anthropomorphic characters and to produce realistic movements, it leverages an indirect method for activating a character's actions based on multimodal interactions. In particular, when the animator performs a body gesture or issues a voice command, the system identifies the best action to activate at that time depending on a text-based description of the scene (later referred to as screenplay) and a library of a character's actions provided as input for the given animation. Precisely, the NLP is used to semantically find matches between the meaning of the issued command, actions (verbs, direct objects, etc.) mentioned in the screenplay, the position of the character and other objects in the virtual scene, and previously performed actions. The use of NLP makes it possible to adopt the same command for dynamically activating different actions depending on the context. This feature reduces the number of commands to be remembered. In the proposed system, the role of the performer becomes of paramount importance. In this case, different from fully automatic NLP-based animation techniques [235] based on a precise description of the scene, the performer has the possibility to personalize character's movements. Moreover, he or she can introduce improvisations by creating slight variations of the animations described with less details in the screenplay. Experiments carried out through a user study with both unskilled and skilled users confirmed the benefits of the proposed system, showing also its effectiveness in terms of time required for animations creation.

2.3.1 Related work

Various methodologies based on different techniques and tools have been proposed with the aim of making the production of computer animation ever more intuitive and effective. A first category of work focused on the development of new interaction methods that can replace or complement the systems based on the traditional Windows-Icons-Menus-Pointer (WIMP) paradigm. This paradigm is the most common method to make users interact with desktop computers by using the aforementioned elements of the GUI, i.e., windows, icons, menus, and pointer. For instance, in [208], a technique is described that allows users to create virtual character animations based on audio clips, by manipulating character's expressions and movements depending on the acoustic and semantic properties of user's expressions. The work in [170] proposes a hand tracking interface based on gesture

recognition. The interface, which was built using the Leap Motion controller, can be leveraged to control the position and orientation of the virtual character's joints. According to the findings presented by the authors, such a spatial interface can be more effective than M&K especially for character posing.

A second category of related work regard the use of performance-based animation. Here, prior work has proposed approaches to automatically or semi-automatically retarget a performer's skeleton to a virtual object's or character's rig [59, 324]. Although these solutions proved to be effective in a number of scenarios, there are situations in which the animator can find it difficult to perform a specific character's movement, e.g., because unnatural or too complex. For this reason, solutions have been proposed (e.g., in [190]) to address these issues, by designing a methodology capable of identifying the performer's intentions (e.g., to make the virtual character backflip) and create the required animation by blending the performer's motion data (e.g., depicting a simple hop on the spot) with pre-recorded actions (e.g., the complex backflip action). According to the authors, methods such as the one defined in [190] can produce more natural and credible animations, while at the same time decreasing the mental effort demanded to the performer for activating the actions. The work in [276] combined motion retargeting techniques with predefined character actions, though the innovative aspect is represented by the possibility to introduce personalization/improvisation in the animation of non-anthropomorphic characters.

Lastly, a third category of related work investigates the possibility to automatically generate animations from text descriptions. Work belonging to this category generally takes advantage of the use of NLP techniques to recognize specific elements such the main characters of the story/animation, their relations with other objects, their actions, etc. For example, the work in [190] describes a technique that receives as input a user-defined screenplay; the screenplay is parsed in order to identify the most relevant characters and the actions they are expected to perform. A limitation of this work is flexibility, since users can leverage only a limited set of words in the screenplay (basically the names of the actions). This limitation is addressed in [68] and [327], which describe systems able to make use of semantic relations among terms in order to increase users' freedom in the choice of screenplay's terms. In [235] and [99], more sophisticated techniques which leverage semantic thesauri are presented to let the same verb in the screenplay activate different actions depending on the object/context the character is currently interacting with/in.

By considering the pros and cons of the related work introduced above, in [172] a virtual character animation system was proposed to combine the performance- and NLP- based techniques with a multimodal interface. The possibility to activate existing actions is one of the fundamental aspects of this system, since this approach is generally adopted in other animation pipelines [68, 99, 190, 235, 276, 327] as a possible alternative to the performance animation alone [59, 324]. For example, in [190, 276], movements of the user are combined with pre-recorded actions which

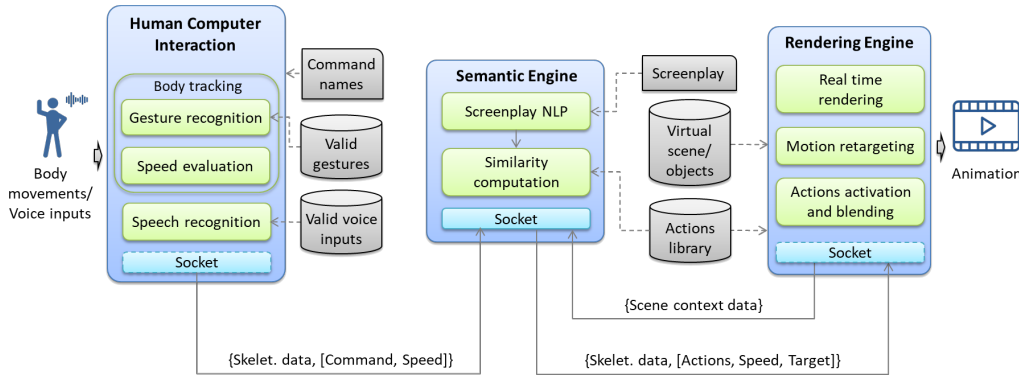


Figure 2.21: Architecture of the proposed system [172].

are played dynamically, based not only on the user's performance but also on the content of the screenplay provided as input, and on the context in which the virtual character is situated into. As in [68, 99, 235, 327], the system leverages NLP to determine the relevant actions that the virtual character is expected to perform, and uses a semantic thesaurus to free the user from the difficulty of using a limited set of terms for writing the screenplay. Moreover, the system makes the user assume an active role, since his or her performance is used to manage actions' timing and control the character's orientation in the virtual scene. This way, the user can actually provide information that would be difficult or laborious to include in the screenplay with the appropriate level of detail. Finally, as in [170, 173, 208], the system allows the user to control the activation of the actions through a multimodal interface (based on gesture and voice inputs), which could enhance system's usability, especially when animation skills are not available.

2.3.2 Proposed system

The architecture of the system originally presented in [172] is depicted in Fig. 2.21. It includes the following modules: Human-Computer Interaction (HCI), Semantic Engine (SE) and Rendering Engine (RE). In the following, detailed descriptions of each module will be provided.

Human-Computer Interaction module

This module is in charge of capturing voice input and the motion data issued/performed by the animator. The module relies on the Microsoft Kinect 2.0 device, though other devices can be potentially leveraged. Data gathered by the

Microsoft Kinect device are accessed through the dedicated Windows SDK[17]. In order to animate virtual charas in [172], it was chosen to rely on the approach based on commands that activate specific characters' actions, rather than on pure motion retargeting. This approach makes it possible to customize the voice and gesture inputs by choosing among those that guarantee the lowest mental or physical effort. In addition, even non-anthropomorphic characters (i.e., characters that have an armature presenting a topology different from the human skeleton), can be easily animated without asking animators to assume unnatural poses or perform difficult gestures. Although the animation is based on gesture and voice recognition, the positions and orientations of joints in the Kinect-tracked skeleton are collected and transferred to the next modules, for further processing. In particular, information can be combined with triggered actions in order to personalize the movements performed by the virtual character making actions more realistic. In the future, different retargeting techniques, e.g., based on facial expressions, could be adopted as alternative input means, in order to transfer to the virtual character other aspects of the animator's personality. The system proposed in [172] supports three types of commands:

- *Stateless*: this command type can be considered as the "default" one. When the HCI module recognizes one of these commands (e.g., jump), it directly forwards it to the next modules, making the virtual character perform the corresponding action once. For this reason, these commands are referred to as stateless.

- *Stateful*: once one of these commands is recognized, the HCI module continues to forward the corresponding message until the the stop command is issued by the animator. The behavior of stateful commands make them particularly useful for repetitive actions whereby the same animation has to be performed several times, e.g., to make the character run for a long time.

- *Parametric*: parametric commands allow the performer to control the speed of the virtual character's animation. This can be required, e.g., for a walk action. When the HCI module recognizes the associated gesture, it also starts to monitor the speed of movements performed by the animator. This information is sent to the next modules in order to parametrize the velocity of the triggered actions' playback.

Before starting to animate, the performer is asked to configure the system by identifying the set of commands needed for the animation to be created. Once the set of commands is defined, he or she has to associate to each command at least

[17]Kinect for Windows SDK 2.0: https://www.microsoft.com/en-us/download/details.aspx?id=44561

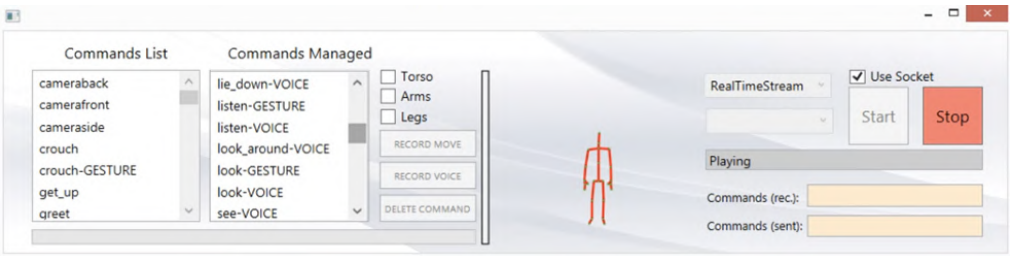


Figure 2.22: Interface for creating system configurations.

one corresponding voice or gesture input. Voice inputs, as well as gestures to be recognized, have to be recorded by the performer by using the interface shown in Fig. 2.22. The set of gestures to be recognized by the system are recorded in a repository as a sequence of positions assumed by the Kinect-tracked joints in the 3D space. Tracked joints are split in three sets: torso, arms and legs. The set of joints to be tracked while recording the gestures can be specified. The possibility to distinguish among the different sets of joints prevents the system from recording/recognizing movements performed with joints not involved in the gesture. Moreover, this approach lets the system simultaneously recognize gestures performed with different parts of the animator's body. The gestures made by the performer are recognized through the Dynamic Time Warping (DTW) algorithm proposed in [27]. With respect to voice commands, the performer is asked to specify the terms to be recognized before animating the character. Real-time recognition is then implemented with the Kinect speech recognition library. Once the performer has defined the configuration, the system can be used to create animations of the virtual character. Data of the performer's skeleton tracked by the Kinect are mapped onto the virtual character's armature to control its orientation through the Kinect-tracked spine joint data. Information about the skeleton data and the recognized gestures are displayed in the interface (right side of Fig. 2.22). When a parametric gesture is recognized, the speed is computed by comparing the information about the dominant joint and axis recorded in the repository of recognizable gestures with the real-time motion data tracked by the Kinect sensor. Command names selected for voice activation are checked in real-time to find a match with the voice inputs pronounced by the performer. For all the inputs that pass a given confidence threshold, the system issues the corresponding commands. The communication between the HCI module and the SE module takes place through the periodical exchange of a JSON string on a socket. This string includes: the name of the recognized command, the speed to be used for managing parametric commands and the performer's skeleton data.

Semantic Engine module

With the aim to make the system more flexible in terms of possible screenplays to be managed as well as to allows the performer to activate a large number of character's actions by using only a limited set of commands (easily to remember), a semantic-based matching was implemented in the SE module. Once the commands issued by the performer generated through the HCI module are received, this module finds the matches between commands and information extracted from the screenplay as well as from data describing the current virtual context (e.g., virtual character's position in the scene, its distances with respect to other objects, etc.). Matches are calculated by considering the meanings/synonyms of the terms used to refer to actions, commands, and virtual scene elements (i.e., objects handled by the RE module) and the behavior expected for the character (described in the screenplay). With this semantic module, scene creators, screenwriters and performers can use different terms for referring to objects, characters, commands, behaviors, etc. since relations among terms used and corresponding elements are automatically identified. The SE module processing includes two phases named *Screenplay NLP* and *Similarity computation*.

Screenplay NLP During the Screenplay NLP phase, the screenplay is preprocessed by using the Stanford Parser [274] with the aim of analyzing the grammatical structure of the sentences in the screenplay. This library was chosen because it represents one of the most common tools freely available and its high level of accuracy was previously proven in [58]. Moreover, the literature presents research work reporting promising results and systems which already took advantage of this library to automatically process texts like novels and movie scripts [12, 107, 221, 230, 337]. The parser is included in the Stanford CoreNLP integrated toolkit [204]. The toolkit also contains a number of (multi-language) instruments that can be used for sentiment analysis or as named entity recognizers. In the future, actual capabilities of the system developed in [172] could be improved, e.g., to make it work with screenplays written in different languages or to extract directly from the screenplay the character's emotional state to be transferred to the final animation. The main goal of the Screenplay NLP phase is to identify key elements in the text of the screenplay in order to extract a list of sub-sentences which represent the main actions that the virtual character should perform. The sub-sentences supported so far belong to the following categories or to a combination of them:

- *Subject – predicate*: information extracted from the sentence refer to a subject (noun or pronoun) and to a verb, e.g., "the man jumps";
- *Subject – predicate – prepositional adverb*: sentence contains a subject and a phrasal verb, e.g., "the man stands up";

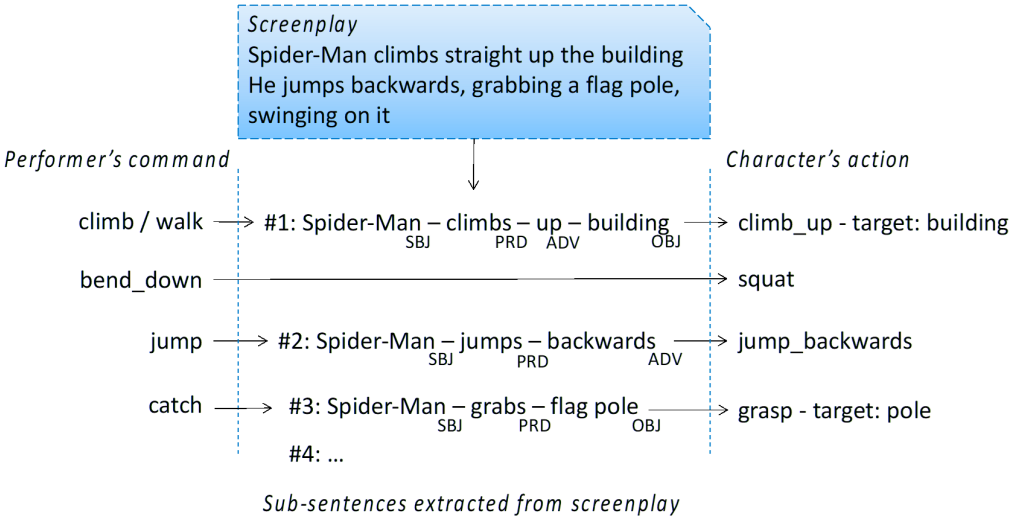


Figure 2.23: An example based on an excerpt of the Spider-Man screenplay showing the processing performed by the SE module.

- *Subject – predicate – object*: in addition to the subject and the verb, a direct object is contained in the sentence, e.g., "the man reads the book".

Predicate may be comprised by more than one verb, e.g., "singing while walking". Information related to the location in which actions take place, e.g., "inside the garden", "on the street" can also be extracted from each sub-sentence. Data extracted are added to a list, representing the ordered sequence of actions to be executed by the virtual character starting from the beginning of the animation to its end. When the performer starts to animate a character, the system tries to find a match between commands issued and the first element inserted in the list. When a match is found, the first/current item is removed from the list of actions, and the system starts again to find a match between the newly issued command and the following element.

The steps constituting this phase are illustrated by the simple example reported in Fig. 2.23. The example is based on the following screenplay:

"Spider-Man climbs straight up the building.
He jumps backwards, grabbing a flagpole, swinging on it".

The figure shows the set of sub-sentences extracted in the Screenplay NLP phase, e.g., *"Spider-Man – climbs – up – building"*; *"Spider-Man – jumps – backward"*, etc.

Similarity computation For each gesture or voice command issued by the performer, the system evaluates a similarity score. The computation of this score

considers three aspects: the actions that the character has to perform (expressed in the sub-sentences list generated in the Screenplay NLP phase); contextual information extracted from the virtual scene (e.g., character's location, name of the objects surrounding the character, distance between character and objects, etc.); and the list containing the actions available for the virtual characters (whose names are provided by the RE module). Fig. 2.23 shows the list of sub-sentences in the middle. Moreover, it also presents the names of the configured commands that can be issued by the performer (on the left) and the names of the available character's actions to be activated when the corresponding commands are issued (on the right). Two conditions may occur while calculating the similarity score. The first condition occurs when a match is found among all the issued command, sub-sentence elements, and virtual scene context information. When this situation occurs, the system first selects the actions (among those defined in the character's actions library) which exhibit the highest similarity score with the verbs and related elements of the sub-sentences; if the direct object involved in the action is found too, then it is set as the target for the given action. For the sake of clarity, from the first sub-sentence of the Spider-Man screenplay it can be observed that character (Spider-Man) is expected to "climb up" a "building". Should the animator issue the "climb" command, the system would recognize a correspondence between the issued command and the sub-sentence "climb_up" element (which contains a concatenation of a verb with a preposition). Should the performer issue a command corresponding to the "walk" action, the system would found anyway a similarity with the "climb_up" element, because a semantic relation between the terms "walk" and "climb_up" exists. When a match is found between the issued command and the sub-sentence element (only verbs or concatenation of verbs plus propositions), the system can create an association of such sub-sentence's elements with the actions of the character's library. If a "climb_up" action is already defined, the system will select it. Then, the SE module would check whether a target object is reported in the sub-sentence (the "building", in this example). If a target is found, the system triggers the "climb_up" action only if the character is close to an object with that or a similar name. It is worth observing that in this way, the system is able to automatically select target objects for actions without the need for the performer to explicitly define them. Similar conclusions are also valid for the third sub-sentence, in which the character was expected to "grab" a "flag pole". When the issued command is not related to any screenplay sub-sentence, the second condition occurs. To handle this case, the system finds an action in the library of character actions, whose name is more semantically similar to the command recognized. In order to study this behavior, it is possible to observe the second command ("bend_down") in the example reported in Fig. 2.23. When the performer issues this command, no explicit reference to the Spider-Man "bending down" action can be detected in the screenplay. For this reason, when this condition occurs, the system identifies "squat" as the action that is most similar to the verb "bend_down" among those

contained in the actions library. This mechanism allows the performer to add improvisations during the animation; in other words, the system leaves the performer free to decide if the generated animation has to strictly stick to the screenplay or it can be enriched by the performer with additional elements. The *ConceptNet* semantic thesaurus tool [188] and its similarity measure are used to let the system identify associations among sub-sentences' terms with scene objects' names and locations. The threshold for defining a correspondence can be manually configured by the user. Not all the relations found in ConceptNet are used, since the huge number of records produced also considers relations which may lead to wrong matches. For example, the antonym relation identified by ConceptNet between the terms "climb" and "descend" may lead to incorrect results. In [172], synonyms and relations like "derivedFrom" and "relatedTo" were considered. Similar to the Stanford CoreNLP toolkit, ConceptNet also supports multilingual terms; thus, future extensions of the devised system to screenplays written in languages different than English are still possible. As previously mentioned, sub-sentences are processed sequentially by the SE module, which automatically moves forward to the next sub-sentence when there is a perfect match with a command issued by the performer. In order to identify the current sub-sentence, rules were defined, which specify whether a sub-sentence should be considered for the system as accomplished or not. The set contains the rules reported below.

- When the performer triggers the action described in a sub-sentence, the system starts to elaborate the next one.
- When the sub-sentence contains two verbs corresponding to two actions to be performed simultaneously, (e.g. "speaking while walking") the system waits until the performer issues the commands corresponding to both actions. It is worth observing that, in order to satisfy this rule, the performer has to trigger different actions at the same time.
- When it is possible to perform an action multiple times (e.g., actions containing an undefined number of targets, like "kicks the balls"), the system only partially moves to the next sub-sentence, leaving the possibility to the performer to still issue commands related to the previous sub-sentence; this allows the animator to continue to interact with different target objects in the scene. This condition remains valid until the action contained in the next sub-sentence is triggered with a suitable command.

In addition, a new set of rules was introduced to specify when actions have to be activated, and when an interaction with objects in the virtual environment has to take place.

- The position of the character into the virtual scene is monitored especially when the current sub-sentence contains location details (e.g., Spider-Man is

assumed to be allowed to "climb" a wall only if he is "outside"). In this way, specific actions can be triggered only if the character is in the expected position

- Similarly, locations of possible target objects in sub-sentences and their distance from the virtual character are considered. In this way, the character can interact only with closer objects placed according to the screenplay. For example, if the screenplay specifies that the character should "grab a book that is on the table", the system would not make it grab a book that is "on the floor".

The SE module is also leveraged for parametric commands, with the aim to automatically identify the proper action to be triggered according to the recognized speed. This is the case, for instance, of the parametric "walk" command. Generally, this command is issued by means of a gesture. The speed of the corresponding actions could be parameterized according to the recognized speed. However, when the speed increases, the performer would expect the issued command to activate no more a walk action, but rather a run action. In order to implement this mechanism, three thresholds were used (slow, normal, and fast). When the command speed overcomes the slow/fast threshold, ConceptNet is used to identify terms that are semantically linked to the slow/fast version of the issued command (e.g., "run" could represent the "fast" version of the "walk" action, and vice versa). This mechanism allows the animator to trigger an even larger set of actions through the same set of commands, simply changing their speed. The animator can change the camera view in two ways. The first one is based on specific commands issued while creating the animation. The second one relies on specific changes inserted in the screenplay with ad-hoc uppercase tags, which are automatically activated when the SE module processes the sub-sentence.

The communication between the SE module and the RE module relies on JSON strings transmitted on a socket connection. In particular, the SE module communicates to the RE module the name, speed and possible target objects of actions that are considered suitable from the similarity computation. The communication is enriched with the skeleton data gathered by the HCI module, which are used for implementing motion retargeting and to orientate the character.

Rendering Engine module

The RE module is in charge of creating animations in real-time. Once it receives information about actions to be played and the tracked skeleton data, it takes care of combining them in order to create the character animation. Moreover, as said, it is also responsible for sending to the SE module data describing the current position of the character in the virtual scene and its distances from the surrounding objects. In [172], it was chosen to implement this module using the Blender Game Engine

(BGE), though, in the future, other game engines (e.g., Unity or Unreal Engine) could be possibly used. The BGE allows animators to record frames of the produced animations, which may be possibly refined at a later time. As pointed out above, the RE module provides the SE module with the library containing the character's actions, with the aim of determining the names of the available actions. Possible character's actions can be classified in the following categories:

- *simple*: actions reproduced with a constant speed only once when triggered (e.g. "squat", "jump", etc.);
- *parametric*: actions that can be parameterized according to the speed of commands issued by the animator (e.g. "walk", etc.):
- *target*: actions requiring a target to be executed (e.g. "grasp", etc.).

Depending on the way that the above actions are activated, it is possible to identify a further, orthogonal categorization:

- *loop*: actions which continue to be active until the stop triggering command is received (e.g. "walk");
- *play*: actions reproduced only once when triggered (e.g. "jump");
- *stateful*: actions that pose the character in a different way with respect to the pose assumed by the character when the action was triggered (e.g. "bending down").

A number of priorities and incompatibilities among actions were defined to prevent conflicts when the performer issues multiple commands affecting the same joints of the virtual character at the same time.

2.3.3 Experimental evaluation

The evaluation was carried out through a user study that involved several participants (both with and without animation skills). Participants were asked to create two short animations (between 1 and 2 min each) using the devised system. The object to be animated was a non-anthropomorphic character, namely, a dog. The dog had to act according to given screenplays, which asked it to perform a specific set of actions into three different environments, i.e., a garden, a walkway, and a river. To this aim, participants had to create animations through their performance for activating pre-defined actions of the dog.

Screenplays

Fig. 2.24 reports the two screenplays designed with the aim of testing the main functionalities of the devised system. In the following, the two animations to be created according to these screenplays are referred to as Animation 1 and Animation 2, respectively. Reading the screenplay, it can be noticed that only the main actions expected to be performed by the character are given; further details, e.g., on exact timing and location where actions have to be executed, were not provided. Timing, as well as locations, had to be decided by the performer, who had to interact with the animation system through his or her performance. Participants were left free to improvise, by adding new actions to those expected by the system, thus creating variations of the above animations.

System configuration

The basic configuration that allowed participants to complete the animations included 25 command names. This configuration, that simulates the work done by the performer before actually starting to animate, was designed taking into account the tradeoff between the robustness of command recognition and the mental and physical effort requested to issue those commands. The process that led to the definition of the basic configuration can be summarized as follow: first, all the action verbs mentioned in the screenplays were included as commands in the animation system by activating for them the voice recognition. Then a few gestures were recorded to activate actions of the non-anthropomorphic character that could be easily assimilated to anthropomorphic movements (e.g. walking, crouching, etc.). Movements to be performed as well as the joints involved in the gestures were defined with the goal of making the system able to properly distinguish them. All the recognized gestures had a corresponding voice command to activate those actions that could be particularly tiring for the participants (e.g., when the dog has to walk for a long time in the virtual scene, which is performed with a walk in place gesture). Finally, additional voice commands, e.g., “stand_up”, as an alternative to “get_up”, etc., were considered in order to further show the use of semantics and provide participants with greater flexibility in the choice of terms that can be pronounced for activating the same action. Table 2.6 reports names/types of commands included in the basic configuration. Before animating the character, participants were allowed to test the basic configuration of the system and, if needed, they were allowed to introduce new commands or adjust the existing ones by recording/choosing a new gesture/voice input for them. The character’s actions library included 17 actions, which are reported in Table. 2.7. It is worth observing that names did not always match perfectly verbs used in the screenplays. These differences simulate the working conditions of a reasonable usage scenario, where screenplays and character’s actions are generated by different subjects, possibly at different times.

```
The dog is in the garden (CAMERASIDE) and is wandering wagging its tail.
The dog exits from the garden (CAMERAFRONT) and walks looking around (CAMERASIDE).
The dog looks at the balls on the walkway and smells the foods.
Once the dog arrives close to the river, it lies down for a moment on the shore.
The dog gets up (CAMERABACK) and hops on the log that is floating on the water.
Finally, the dog jumps on the bridge.
```

(a) Animation 1

```
The dog is walking in the garden.
It sees a bird on a bench.
It jumps on the bench.
The bird flies to another bench and the dog looks around for it.
The dog hops off the bench.
It wanders around and sniffs the grass.
It sits (CAMERABACK).
Suddenly, it hears some twittering birds on the fence.
It walks towards them.
When the dog is close to the birds (CAMERASIDE), it yaps and hops happily.
```

(b) Animation 2

Figure 2.24: Screenplays adopted for the experimental evaluation.

Table 2.6: Valid gestures and voice commands used in the experimental setup. Entries with the $_{ST}$ subscript refer to the stateless version of the action, whereas those with the $_{PR}$ subscript refer to the parametric one.

Gestures		**Voice commands**		
crouch$_{ST}$	crouch	jump	sit	walk
greet$_{PR}$	get_up	lie_down	smell	wander
look$_{ST}$	greet	listen	sniff	wander_around
smell$_{ST}$	hear	look	speak	yap
walk$_{PR}$	hop	look_around	stand_up	camera_left/front/side
listen$_{ST}$	hop_off	see	wag	

Table 2.7: Actions defined for the character used in the experimental setup.

Actions		
get_up (simple)	look_around (simple)	sniff (simple)
hop_off (simple)	rest (simple)	stand_up (simple)
jump (target, simple)	run (param.)	wag (param.)
listen (simple)	sit (simple)	walk (param.)
look (target)	smell (target, simple)	yap (simple)

Animations

During the experiments, participants, i.e., performers, were able to visualize the application interface through a dedicated display. The Microsoft Kinect sensor was placed in front of them to track their skeleton and capture voice commands.

Fig. 2.25 reports a number of screenshots captured from the system interface while a participant was creating Animation 1. It is possible to split the interface into three parts. The upper part displays the triggered actions, their speed, and target objects, if any (on the left side). Moreover, on the right side, the performer's skeleton was added to the visualization for the sake of clarity, even though it is only shown in the configuration panel previously illustrated in Fig. 2.22. The central part shows the virtual scene, the character and the animation being created in real-time. The interface automatically outlines with colored arcs (blue or red lines), relations between the character and closer target objects. In Blender, target objects have been labeled with the prefix "Actor_" in order to distinguish them from other objects in the virtual scene. Finally, on the bottom, the interface shows the screenplay sentences/sub-sentences that are relevant to the performer. According to the command issued by the performer, sentences are classified as:

- *Previous sentence* (P): a sentence that contains actions that have already been triggered;
- *Current sentence* (C): a sentences that contains action verbs for which a match is expected with issued commands;
- *Next sentence* (N): a sentence waiting to become active, according to criteria illustrated above.

Sentence are shown using different fonts and colors: an italic red font indicates actions that have already been performed; an underlined green font specifies the current action that the performer is expected to activate at that time; finally, normal white font represents actions to be activated later.

A video showing a participant creating Animation 1 is available for download[18]. The creation of the Animation 1 (defined in the screenplay shown in Fig. 2.24) starts with sentence "The dog is in the garden (CAMERASIDE)". Positional information gathered by the RE module makes the SE module aware of the fact that the dog is already in the garden location, and the side camera can be set. After the first sub-sentence, the screenplay specifies that the dog has to perform two actions, i.e., "wandering" and "wagging its tail" at the same time. The two actions can be activated with both gesture or voice commands. In particular, for the action "wander", it is possible to choose between performing the "walk gesture" (semantically linked to the "wander" verb), or pronouncing the voice input "wander" or "wander_around". The animation system will recognize an association between the verb "wander" defined in the screenplay and any of the issued commands (the walking gesture, or the "wander"/"wander_around" voice input). Regarding the second action, i.e., "wag", the performer can choose again between

[18] Animation 1: https://goo.gl/znkas2

the "wag" voice command, or "greet gesture" (a gesture in which the performer moves one hand to say hello). This is possible due to the affinity, from a semantic point of view, of the "wag" and "greet" verbs. In fact, a dog wagging its tail can be interpreted as if it is greeting someone or something. Similarly to the walking action, the system activates the "wag" action with a given parametric speed, if the action has been activated with a gesture, or with an arbitrary speed, if a voice command was used. As said, the screenplay defines that the two actions have to be executed simultaneously. Should the performer issue just one command, e.g., "wag", the system assumes that he or she wants to add improvisation features. Thus, only "wag" is activated and the system still waits for the performer to issue the two commands for walking and waging. It is necessary to note that when the "walk" action is active (alone or combined with other actions), the animator's orientation is used to control the character's rotations and a subset of his or her joints is retargeted to character's bones to make the dog walk in a more personalized way. Another aspect that can be pinpointed it that the "greet" command could potentially be used by the performer to activate different actions (e.g., to let the dog warmly welcome back its master). Moreover, both the commands were defined as parametric; hence, the performer could modify the speed of his or her gesture, making the character walk or wag its tail at a slower or faster speed. When the performer issues simultaneously the two required commands, the expected actions are triggered, as shown in Fig. 2.25a. The next sub-sentence that becomes active states "The dog exits from the garden (CAMERAFRONT)". A periodical check of the dog position in the scene allows the system to understand if an action has been correctly activated or not. As long as the dog remains in the garden, every action, e.g., making the dog jump, performed by the animator are interpreted as improvisations. When the system realizes that the dog has left the garden, the next sub-sentence becomes active and the camera view is automatically moved to the frontal one. From these examples, it emerges that improvisations can refer to both time and space information of expected and unexpected actions, e.g., letting the performer free to decide when the dog should leave the garden, the trajectory to follow, as well as other actions to perform in the garden. Once the dog has left the garden, it has to "walk" by "looking around". In order to activate the "look" action, the performer could either issue a "look"/"look_around" voice command or execute a "look" gesture. Similarly to what has been already seen, only when the system recognizes that the two actions are simultaneously performed it moves to the next sub-sentence, i.e., "The dog looks at the balls on the walkway". In this case, when the performer issues the "look" command (using either gestures or voice commands), the system has to execute additional tasks since, here, a direct object is specified for the action verb. The first task is finding a correspondence between the command issued and the "look" action. If a correspondence exists, the system verifies the presence in the environment of target objects (i.e., the balls),

and checks if at least one ball is on the walkway. If all the conditions are satisfied, the "look" action is triggered by choosing the closest ball on the walkway as a target (in the example shown in Fig. 2.25c the object named "Actor_Ball_2" is selected). If there are no balls on the walkway close to the character, a different action, i.e. "look_around", is triggered. This action is not valid for changing the current sub-sentence as it is considered as an improvisation not mentioned in the screenplay. By comparing the last two sub-sentences, it is observed that the same command ("look") could be used to trigger both the "look_around" and "look" (with target) action, thus confirming the benefits brought by the SE module that enhance the set of character's actions performed with a reduced set of commands. Similarly, the performer is allowed to add improvisation by deciding, for example, the number of balls the dog should look at, which ball is the target and when actions have to be performed. The next sub-sentence considered is "and smells the foods". Here, the system tries to find matches not only between the received commands and the current sub-sentence, but also with the previous one, since these actions can be triggered on multiple objects. In fact, the performer can decide to make the character smell food (by issuing, for example, the "sniff" or "smell" gesture/voice commands) as shown in Fig. 2.25b, or continue to look at other balls. In order to identify which are the objects in the scene that can be considered as food, the system leverages their semantic relations to understand that objects named "pizza" and "banana" are particular types of food. This way, it is possible to activate the action only when the character is close to them. When the dog "arrives close to the river", it has to "lie down" and a camera view change has to be done. In order to activate the "lie down" action, the performer can choose to pronounces the "lie_down" voice command or perform the "crouch gesture" which is semantically similar. Similarly, leveraging the semantic similarity, both the "stand_up" and "get_up" voice commands can be used to make the dog "get up". Afterwards, it is expected that the performer activates actions that make the dog "hops on a log that is floating on the river". To this aim, he or she can use both the "hop" or "jump" voice command. When the command is issued, the system is in charge of checking if there is a "log" object close to the dog on which jump to, in order to trigger an action with a specific target (Fig. 2.25d). Also in this case, the role played by the performer is fundamental, since he or she is free to decide the exact time at which the dog will jump, or if the character will execute other actions before performing the expected one. Similar considerations can be made when the system processes the final sub-sentence which asks the performer to make the dog "jump on the bridge".

A video showing the generation of Animation 2 is available for download[19].

[19] Animation 2: https://goo.gl/ZgxXdJ

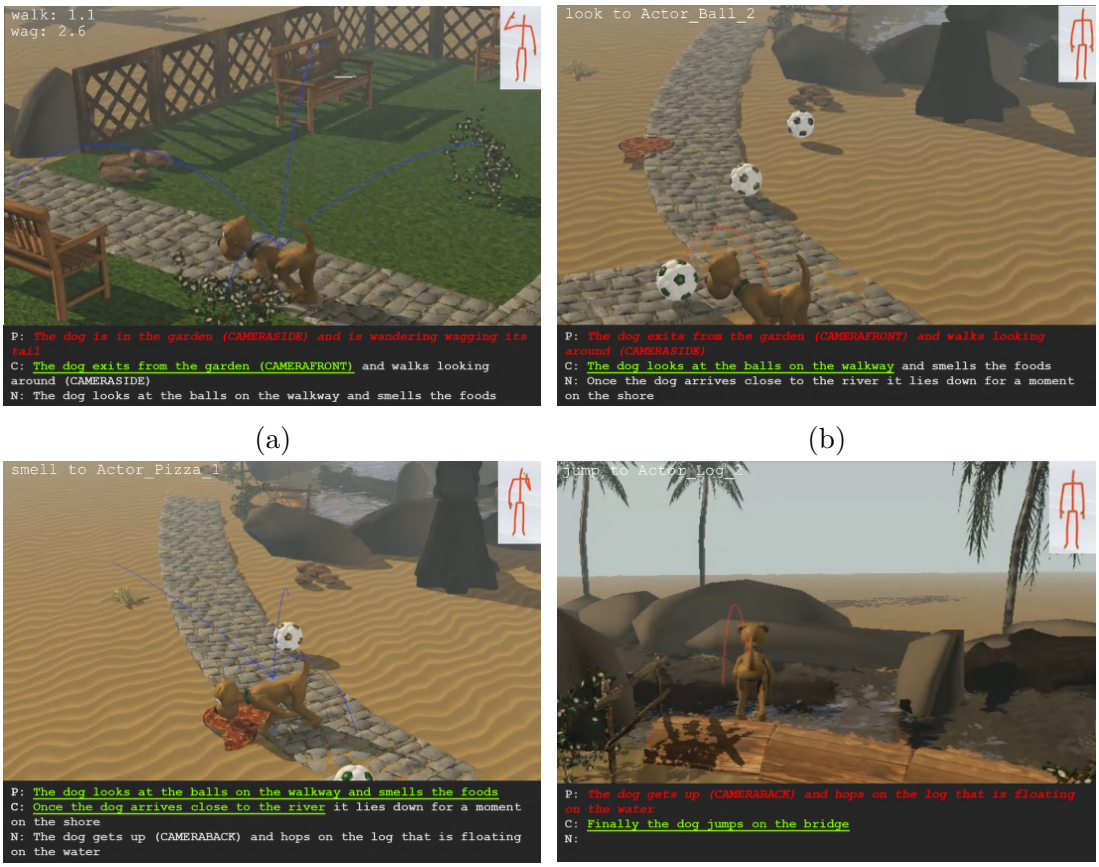


Figure 2.25: Screenshots taken from the system interface while a user involved in the study was creating Animation 1.

2.3.4 Results

As previously mentioned, a user study was carried out to evaluate the proposed system. The study involved both unskilled (UNUs) and skilled (SKUs) participants. The first category of participants was asked to create the two animations illustrated above, by using both the proposed system and Blender with a number of simplifications (discussed in the following). On the contrary, SKUs were requested to work only with the proposed system, and their results were compared with those obtained by UNUs.

The evaluation was conducted by taking into account both objective and subjective aspects. Objective evaluation was based on the time required to create the animations, subjective feedbacks were collected through a post-test questionnaire, organized in two parts. The first section aimed to study system usability according to the five attributes defined by Nielsen in [228]. The second part analyzed the

suitability of the system as a tool for one of the envisaged scenarios, namely, storytelling. In order to investigate this aspect, the "Dimension Star" [272], a model commonly adopted to observe possible strengths and weaknesses of a storytelling tool [253], was considered. Both the first and the second section of the questionnaire relied on questions to be evaluated on a Likert scale ranging from 1 (very low) to 5 (very high). Questionnaire is reported in the Appendix[20] of the original paper ([172]) which is available for download.

Moreover, from the collected data, it was possible to extract information on system's intuitiveness. Since a well-established methodology is currently missing to directly measure this factor [136], it was chosen to follow the definition of system's intuitiveness proposed in [233]. According to this work, which studied a methodology for measuring the intuitiveness of a simplified system for robot programming (quite comparable to the animation system proposed in [172]), this dimension can be considered as a combination of several contributions: efficiency (in terms of cognitive effort to accomplish the task), effectiveness (in terms of precision and completeness), and user's satisfaction.

Since one of the characteristics of the proposed animation system is the possibility to add improvisations, it was not possible to ask participants to recreate animations similar to a reference (as done, e.g., in [233]). For this reason, precision and completeness could not be measured in qualitative terms. Thus, it was decided to manage the effectiveness dimension by taking into account a specific statement in the questionnaire regarding efficiency/effectiveness. Finally, for what it concerns the efficiency and user's satisfaction dimensions, they were evaluated by considering the cognitive effort [272] and user's satisfaction [228] attributes.

Evaluation with unskilled users

Experiments were carried out with 25 participants (18 males and seven females), aged between 18 and 34. Participants reported no computer graphics skills. In order to compare the devised system with an already existing tool for animation targeted to storytelling, several alternative tools like Anitales, Animate it!, Iyan3d and Toontastic were considered. However, it was not possible to find a truly comparable system among these tools for the following reasons:

- character's actions had to be generated on-the-fly by moving the characters/object and/or their rigs, since they could not be picked up from an existing library;
- when a library with already defined actions was available, it provided an extremely limited set of both characters and predefined animations.

[20] Appendix of [172]: `https://goo.gl/2EWJNM`

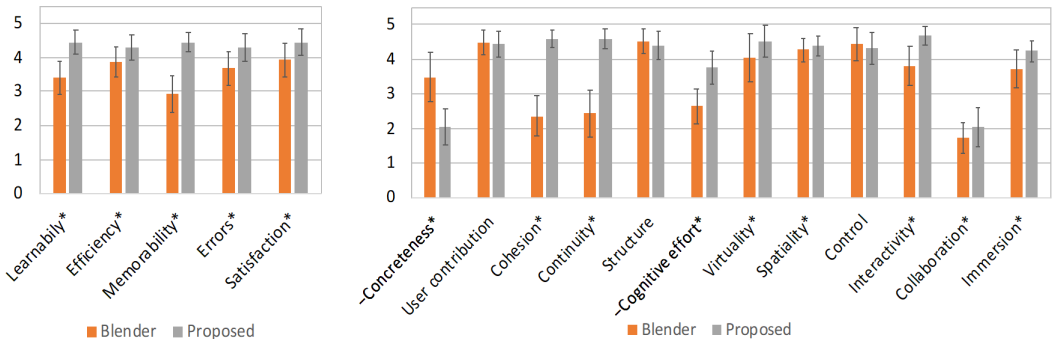
Hence, it was decided to use Blender with a number of ad-hoc limitations to make the experimental conditions comparable. Besides its adoption as RE module, Blender is also an extremely flexible open-source modeling and animation tool with a number of editors suited to different needs. Limitations intentionally introduced in Blender made this tool usable to users without prior experience in animation and allowed them to use the tool without the need to know the whole interface for completing the assigned task. Before starting the tasks, participants were shown the screenplays of the two animations to be produced and a rendering created for them. Moreover, they were given time to familiarize themselves with the interfaces of both the systems through a training phase. In particular, with respect to the Blender interface, they were informed about the use of the Blender's 3D view, the NonLinear Animation (NLA) editor and the Dope Sheet. These editors allow the animator to define the position and orient of the virtual character in the 3D space, activate/blend actions for it by selecting them from a scrollable list, move selected actions in the timeline, handle keyframes and visualize the generated animation. Regarding the proposed system, the main elements of the interface, as well as the list of valid commands, were explained. Participants could experience the command recognition and they were given the possibility to change, when needed, the basic configuration to match their preferences. On average, participants spent less time with the proposed system (M = 7 min 13 s, SD = 2 min 39 s) than with Blender (M = 12 min 30 s, SD = 2 min 01 s). When the training was completed, participants were requested to generate the two animations (i.e., Animation 1 and Animation 2) starting with either the proposed tool or Blender. The order of the animation to start with, as well as the system to use first, were randomly selected in order to limit possible biases due to learning effects. While executing the task, they were allowed to receive support and ask questions only for the first animation; while creating the second animation, they were requested to work autonomously. When operating with the proposed system, participants could make more than one trial, if they were not satisfied with the obtained result.

Statistical significance was analyzed using paired Student's t-tests. With respect to Animation 1, participants were significantly faster ($t(24) = 16.22$, $p < 0.01$, $d = 4.50$) with the proposed system (M = 2 min 13 s, SD = 57 s) than with Blender (M = 13 min 54 s, SD = 3 min 32 s). Similar results are also obtained for the Animation 2, where participants were more than six times faster ($t(24) = 15.07$, $p < 0.01$, $d = 4.20$) with the proposed interface (M = 2 min 27 s, SD = 51 s) than with Blender (M = 16 min 07 s, SD = 4 min 31 s). It is worth noting that in six cases (four while working on Animation 1, and two while working on Animation 2), participants stopped the creation of the animation, as the system recognized a wrong command. In these cases, they was allowed to start again the animation and completion time was calculated by summing all the attempts.

Subjective measures that were collected through the questionnaire are reported in Fig. 2.26. As with objective measures, statistical significance was analyzed using

paired Student's t-tests. The '*' symbol is used to indicate statistically significant results (i.e., $p < 0.05$). As reported in Fig. 2.26a, the proposed system was perceived as more usable than Blender (with the simplified interface) considering all the Nielsen's attributes. Best results are observed for the learnability (Blender: $M = 3.40$, $SD = 1.00$; Prop.: $M = 4.44$, $SD = 0.71$; $t(24) = -4.44$, $p < 0.01$, $d = -1.20$) and memorability (Blender: $M = 2.92$, $SD = 1.07$; Prop.: $M = 4.44$, $SD = 0.58$; $t(24) = -6.36$, $p < 0.01$, $d = -1.76$) attributes. A smaller effect is observed for the efficiency (Blender: $M = 3.88$, $SD = 0.88$; Prop.: $M = 4.28$, $SD = 0.74$; $t(24) = -1.85$, $p = 0.04$, $d = -0.49$), errors (Blender: $M = 3.68$, $SD = 0.99$; Prop.: $M = 4.28$, $SD = 0.79$; $t(24) = -2.27$, $p = 0.03$, $d = -0.67$), and satisfaction (Blender: $M = 3.92$, $SD = 1.00$; Prop.: $M = 4.44$, $SD = 0.77$; $t(24) = -2.32$, $p = 0.03$, $d = -0.58$) attributes.

Regarding system's suitability for storytelling and, in general, for animation production (Fig. 2.26b), statistically significant results can be observed for cohesion (Blender: $M = 2.36$, $SD = 1.15$; Prop.: $M = 4.60$, $SD = 0.50$; $t(24) = -9.33$, $p < 0.01$, $d = -2.53$) and continuity (Blender: $M = 2.44$, $SD = 1.36$; Prop.: $M = 4.60$, $SD = 0.58$; $t(24) = -7.11$, $p < 0.01$, $d = -2.07$), for which the proposed system was rated higher than Blender. This result is probably related to the capacity of the proposed system to intrinsically help the animator to keep a temporal coherency among the story events. As expected, participants perceived the proposed system as characterized by a lower cognitive effort (Blender: $M = 2.64$, $SD = 1.04$; Prop.: $M = 3.76$, $SD = 0.97$; $t(24) = -3.78$, $p < 0.01$, $d = -1.12$), probably guaranteed by the use of natural interactions. Moreover, the proposed system was preferred with respect to Blender for what it concerns spatiality capabilities (Blender: $M = 4.28$, $SD = 0.68$; Prop.: $M = 4.40$, $SD = 0.58$; $t(24) = -1.81$, $p = 0.04$, $d = -0.20$), i.e, the importance that system gives to objects in space, to space itself and to the degree of involvement of the virtual environment in the story development. Participants also expressed their preference for the degree of interactivity (Blender: $M = 3.80$, $SD = 1.12$; Prop.: $M = 4.68$, $SD = 0.56$; $t(24) = -3.56$, $p < 0.01$, $d = -1.00$) and immersion (Blender: $M = 3.72$, $SD = 1.10$; Prop.: $M = 4.24$, $SD = 0.60$; $t(24) = -2.40$, $p = 0.01$, $d = -0.59$) ensured by the proposed system, meaning that they felt more involved in the story. They judged the proposed system as more suitable for collaborative use (Blender: $M = 1.72$, $SD = 0.89$; Prop.: $M = 2.04$, $SD = 1.10$; $t(24) = -2.31$, $p = 0.01$, $d = -0.32$), though this feature was not explicitly available. The above benefits were paid with a higher concreteness (Blender: $M = 3.48$, $SD = 1.42$; Prop.: $M = 2.04$, $SD = 1.06$; $t(24) = 4.04$, $p < 0.01$, $d = 1.15$), meaning that contents of the story (objects and characters) provided by the proposed system were perceived as predefined, without the possibility to create them from scratch. Indeed, even though participants operated with a simplified version of Blender without using its full potentiality, it can be acknowledged that the proposed system is not intended to give users the same number of functionalities ensured by general-purpose



(a) (b)

Figure 2.26: Subjective feedback from UNUs regarding Blender and the proposed system based on (a) Nielsen's usability attributes and (b) "Dimension Star" model.

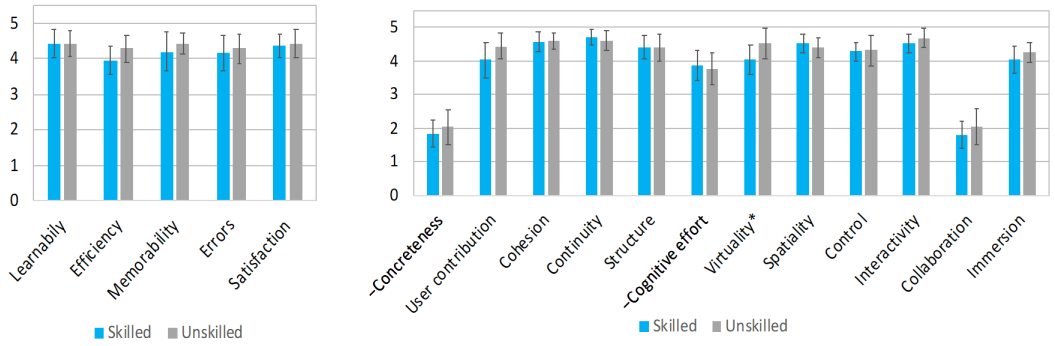
animation tools. Finally, with respect to intuitiveness, by computing the average for efficiency/effectiveness, cognitive effort, and user's satisfaction scores, it can be seen that Blender was perceived as being less intuitive than the proposed system. In fact, on a 1-to-5 scale, it reached a score of 3.48 ($SD = 0.76$) compared to 4.16 ($SD = 0.62$) obtained by the proposed system ($t(24) = -3.65$, $p < 0.01$, $d = -0.98$).

Evaluation with skilled users

Experiments were carried out with 25 participants (19 males and six females, aged between 22 and 33) selected among MS and PhD students of Politecnico di Torino. Participants reported moderate or good computer graphics skills, especially with Blender). Differently than with UNUs, participants performed the task ignoring Blender, hence they were asked to create the two animations solely with the proposed system. Another difference between UNUs tests was the lack of training (only a few instructions of the system were provided) as well as the fact that SKUs were not allowed to modify the basic configuration. These changes were considered in order to study the system's intuitiveness and to investigate to what extent it could be used without personalization. The overall procedure of the test was maintained as the previous one, as half of the participants were requested to start with Animation 1, whereas the other half with Animation 2. Only while creating their first animation users had the possibility to get support.

On average, the times for completing the tasks were 2 min 37 s ($SD = 58$ s) for Animation 1 and 2 min 27 s ($SD = 38$ s) for Animation 2. No statistically significant differences in terms of completion time are observed analyzing results achieved by UNUs ans SKUs using unpaired t-test Student's t-tests.

A further confirmation can be obtained from the subjective evaluation reported in Fig. 2.27. The figure compares the average scores assigned by UNUs from the



(a) (b)

Figure 2.27: Subjective feedback collected with the proposed system by SKUs and UNUs based on (a) Nielsen's usability factors and (b) "Dimension Star" model.

previous section with those of SKUs working with the proposed system. No statistical significance was found using unpaired Student's t-tests for all the dimensions except virtuality (SKUs: $M = 4.04$, $SD = 0.89$; UNUs: $M = 4.52$, $SD = 0.92$; $t(24) = -1.88$, $p = 0.03$, $d = -0.53$). This result could suggest that level of expertise in computer graphics do not affect the experience with the proposed tool, as scores are mostly comparable. Finally, computing the intuitiveness by averaging scores assigned by SKUs users to evaluate efficiency/effectiveness, cognitive effort, and user's satisfaction, a value equals to 4.07 ($SD = 0.61$) on a 1-to-5 scale was obtained.

2.3.5 Future developments

At present, the system leverages only skeleton data as source to transfer the animator's movements to the virtual character. However, future efforts could be devoted to provide animators with new personalization possibilities, e.g., by integrating more sophisticated motion retargeting algorithms like those based on performer's facial expressions [239]. Another future direction could pertain the possibility to consider screenplays containing a richer set of information; for example, sentiment analysis tools could be integrated with the aim to make the virtual character's actions reflect its mood as described in more detailed screenplays. Taking into account expectations reported by participants of the user study, a collaborative modality could be implemented to support multiple performers/characters. Finally, evaluations could be carried out by changing screenplays and characters, with the aim to investigate new features possibly required in more complex usage scenarios.

2.4 Concluding remarks

In this chapter, the focus was placed on the possibilities offered by alternative interfaces for the production of virtual character animation. In particular, the work in [173] was considered, which presented a system that allows the user to animate virtual characters through both the keyframing and performance-driven techniques, by combining in a unified pipeline the possibility to use reconfigurable TUIs and NUIs. Moreover, an automatic mapping algorithm was discussed, designed to aid novice animators to create the tangible props and define the retargeting rules between the input interfaces and the virtual character to be animated. Subjective results confirmed that the affordances offered by TUIs and NUIs make the animation significantly more intuitive than when using traditional interfaces based on M&K. Subjective results were confirmed by objective ones, which also revealed that the proposed system lets both skilled and unskilled user articulate and animate virtual characters faster than M&K. This finding could be also related to the possibility to control at the same time the character's pose and location in the 3D space. Results of experimental observations revealed that improvements in terms of efficiency were higher for novice users than for skilled users. Moreover, when unskilled users worked with the proposed system, they were able to complete the assigned task faster than skilled users working with M&K. However, the improved intuitiveness and the reduced completion time resulted in an increased physical effort and a less accurate posing.

The second system described in this chapter, originally presented in [172], aims to ease the production of virtual character animations by combining the performance- and NLP-based approaches. The system leverages a textual description of the scene (a screenplay) and a library of character actions. By combining data extracted from the screenplay, pre-defined actions in the library and information about the virtual context (e.g., the position of the character in the scene, nearby objects, etc.) with the performance of the animator (based on body gestures or voice commands), the system determines the action that should be played. Leveraging the NLP, the user can issue the same command to activate different actions, thus reducing the mental effort required by the performer. Moreover, the performer can deliver to the virtual character both temporal and spatial information for the execution of the actions, enabling the possibility to introduce personalizations. Experimental results obtained through a user study that involved both unskilled and skilled users demonstrated the usability, intuitiveness, and effectiveness of the proposed system, thus making the system particularly suitable for specific contexts like fast animation prototyping or interactive storytelling. However, limitations concerning the number of actions in the library could prevent its usage in high-quality animation productions, in which animators are more interested in a fine-grained control of the characters than in interface intuitiveness or simplicity.

Chapter 3

Is immersive virtual reality the ultimate interface for 3D animators?

Work described in this chapter was originally presented in [47, 171, 51] as well as in a conference paper which is currently under review.

3.1 Introduction

As stated in the previous chapter, computer-generated animations are used in a large number of application domains. However, their creation still represents a very time-consuming and skill-intensive task. Reasons for this are strictly related to the high level of expertise required by the involved tasks [312], as well as to the complexity of the adopted interfaces [59].

These limitations can be an issue not only for skilled users, who are probably interested in speeding up the production of high-quality animations, but also for novice users, who may search for a "quick-and-dirty" animation tool for sketching an idea and/or sharing it with other interested parties [79, 91, 330] As described in Chapter 2, one of the main issues to be addressed is the fact that although animators are requested to operate on 3D content, traditional animation software commonly offers tools and interfaces that are natively 2D [142, 236, 312]. To deal with this issue, the previous chapter focused on input interfaces, presenting new interaction paradigms able to ease the manipulation of (the interaction with) virtual characters. This chapter investigates a different aspect, which was partially disregarded in previous work, i.e., the system output. In fact, even the system output could be affected by the limited dimensionality of the visualization methods. It can be observed that traditional animation tools still rely, in most of the cases, on 2D displays; hence, to visualize the contents of interest from the required

perspectives, users have to continuously change the position of the virtual camera or simultaneously look at multiple views of the scene been animated [74]. Both solutions could make the existing animation tools rather complex to use, especially for novice users [312].

To cope with the above limitation, an increasing number of users with different backgrounds like, e.g., digital artists, filmmakers and storytellers, recently turned their attention to the possibility to adopt VR not only as means for visualizing virtual character animations, but also as a tool for creating them [103, 122, 245].

One of the main benefits brought by the use of VR for generating animations is the improved sense of presence, which allows users to feel as a part of the virtual environment being animated [312]. Moreover, the sense of presence has been proved to be capable of increasing animators' creativity and productivity [122]. Another reason to examine VR as a powerful tool for animation is the capabilities offered by this technology to make users interact with 3D objects by using input and output devices (e.g., controllers and head-mounted displays, or HMDs) that are natively 3D. The great interest shown by professional studios for the use of immersive environments, e.g., for articulating and animating virtual characters, is confirmed by prior work. An example is represented by PoseVR[1], a system recently developed by the Walt Disney Animation Studios. Various commercial products and research prototypes are already available on the main VR stores; however, most of them are generally targeted to non-professional users. Although these solutions are generally intuitive and easy to use, they do not provide the advanced functionalities offered by traditional animation tools like Autodesk Maya[2], Blender, etc. Tools developed for professional users often lack the integration with common animation suites [234, 243, 312]. In particular, in order to apply changes or reuse the animations generated within immersive environments, animators are generally requested to continuously perform import/export operations. These additional operations can slow down the entire animation process and can be perceived as highly distracting from the animators.

By moving from the above considerations, this chapter will describe studies that have been carried out with the aim to assess the capabilities of immersive environments for generating virtual character animations. The studies which will be presented in this chapter were designed by considering the lessons learned regarding the benefits provided by the use of 3D User Interfaces (3DUIs) as input interfaces. Studies discussed investigate the integration of VR into the character animation pipeline through consecutive steps. In particular, the first section of the chapter reports results achieved by introducing, at first, the VR technology in the execution of five representative animation tasks, and then studying its effects on

[1]PoseVR: https://www.technology.disneyanimation.com/projects/PoseVR

[2]Autodesk Maya: https://www.autodesk.com/products/maya/overview

the whole character animation pipeline. The second part of the chapter focuses on the character posing step, and will describe additional efforts placed on the design of new input interfaces based on tangible props and 3D sketches for articulating virtual characters' skeletons.

3.2 Immersive virtual reality-based interfaces for character animation

Although several VR-based tools for character animation are currently available on common VR stores or they have already been proposed by the research community, only a small number of works in the literature reported experiments aimed to validate the use of VR for creating animations [243, 245]. Hence, presenting results of user studies aimed to asses the effectiveness of VR to perform animation tasks over the entire pipeline are attracting a significant attention, as they could be helpful in the design of next-generation animation tools.

Starting from the above considerations, the goal of the two studies that will be discussed in the following (originally presented in [171] and [51]) was to investigate to what extent the use of VR could improve the various steps of a complete animation pipeline. To this purpose, a VR-based tool for creating animations in an immersive VR environment was developed. To design the tool, the common features offered by the latest VR-based animation tools were first analyzed, as reported in Section 3.2.1. Then, by considering the major components presented in the ACM's Computer Science Body of Knowledge for Computer Animation [199], a set of major functionalities to be included in the tool were defined, namely, keyframing and performance animation on diverse object properties, forward and inverse kinematics and spline manipulation. In contrast to most existing solutions, which were developed as standalone applications (thus preventing access to advanced functionalities usually offered by traditional animation software), the proposed tool, named VR Blender, was developed as an add-on for the well-known, open-source Blender suite. This choice allows animators to use the core animation functionalities of the underlying software in VR, by leveraging some of the effective interaction paradigms proposed in other VR-based solutions but also providing them with a larger set of features. The proposed tool is not intended as a replacement for traditional animation software but, rather, as an additional instrument that could be used to improve some (possibly, many) of the stages of the animation pipeline. Hence a possible workflow could be that animators start their project in Blender, continue it in VR and finalize it again in Blender (or other tools). Lastly, due to the fact that both the traditional and the VR-based interfaces are enabled at the same time, multiple users can view or work on the same 3D scene in a collaborative manner.

The study on the impact of VR on the animation creation process involved two stages. The aim of the first stage was to analyze, through both qualitative and

quantitative measurements, the impact of VR-based systems on animator performance, on interface usability and on the quality of produced contents. To this purpose, five representative animation tasks were considered. Although the results of the first stage were encouraging, the system used in the study was not designed to implement all of the steps of an animation pipeline, but just the most common ones (those experienced in the tests). Therefore, should the users be interested in developing a more general animation, at some point, they had to remove the headset and continue part of the work using traditional interfaces (i.e., M&K), since required functionalities were not directly accessible in VR.

Building upon the outcome of the first stage, in the second stage the attention was moved to providing users with all of the instruments needed for animating virtual characters and offering them a mechanism to define ways for accessing in VR all the remaining features. The character animation pipeline usually encompasses three steps, known as rigging, skinning and posing. In the first step, a so-called "rig", also known as "armature" or "skeleton", is assigned to the character's mesh, which is then used to manipulate the geometry of the character, giving it the intended pose for each frame. In the second step, the animator defines the influence of each skeleton's bone on the character's "skin", i.e., the vertices of the 3D mesh [18]. Finally, in the last step, the animator applies rotations and translations to bones in order to make the character assume the desired pose. These three steps usually involve tedious tasks which require that users possess significant skills and devote a lot of time in order to reach the desired quality [241]. For this reason, they can be considered good candidates for experimenting with VR. Furthermore, a system able to manage these three steps lets the animator produce a complete animation. Therefore, they represent good candidates also for studying the impact of said technology on a complete pipeline.

In order to support these steps, new functionalities were added to the tool. Effectiveness and usability of the new VR based functionalities for the specific pipeline considered in this stage were evaluated again using both subjective and objective observations, through a user study that involved both non-professional and professional participants (animators). Various dimensions were studied, including animation time, accuracy, precision, interaction naturalness, etc.

With respect to the issue related to the limited number of functionalities accessible in VR, while extending the VR-based interface it was noticed that the richness of available interaction patterns could result in an increase in the user's mental load and learning cost, possibly leading to a reduction in the effectiveness of immersive animation. For this reason, it was decided to not incorporate all the features available in the native interface to VR, but rather to provide the user with a methodology to extend the interface of the VR-based tool with custom functionalities. The first approach designed to deal with this issue involved the possibility to visualize the native windows of the Blender's interface in the VR environment.

This way, users are allowed to interact with the Blender interface native components like buttons, sliders, or combo boxes using ray casting in VR. However, this approach could reduce the sense of presence and present serious interaction issues related to the use of 2D interface elements in a 3D environment as well as to the limited resolution offered by VR displays [282]. Moreover, although this approach could make the operations for users with previous experience with Blender easier, duplicating Blender's interface components, which are notoriously hard to use for non-expert users, will automatically translate their complexity to VR. Taking into account the above limitations, the approach that was ultimately implemented involved the development of a script that allowed users to define native functionalities in VR, and customize how to access them by selecting various types of visual and controller-based interactions. The new features developed in the second stage were integrated in the original tool.

3.2.1 Related work

The possibility to express animators' creativity through tools based on VR and related technologies, including AR, 3DUIs, etc., is not so new. A first example can be dated back to more than 20 years ago, when a system was proposed ([74]) to extend the 2D sketch-draw animation paradigm to 3D by making use of VR. Simple animations could be generated by using a wand manipulator and visualized on a desktop display by wearing a pair of stereoscopic shutter glasses. The limitations of early work mainly regarded the technological characteristics of the hardware adopted, which constrained the complexity of 3D assets that could be generated and presented serious interaction concerns due, e.g., to tracking accuracy, latency, etc. More recently, the impact of the above issues was reduced by "simply" upgrading to more advanced (and expensive) VR setups, e.g., relying on multi-wall projection-based immersive environments with multi-camera tracking systems [141]. Nowadays, developments in consumer electronics has led to an incredible revolution in this market, making HMDs, gloves and similar accessories largely accessible (less expensive). These changes are opening ways for developing a whole new set of animation tools based on the above technologies.

By taking into account both commercial products as well as prototypes proposed in research papers, two major approaches can be identified. The first approach consists of extending existing animation suites with plugins/add-ons which basically allow the user to visualize the 3D viewport of the given animation tool and make it possible to interact with it in 3D. The second approach, which is by far more common, involves the development of ad-hoc animation tools which leverage the new interaction paradigms offered by VR.

Extending existing animation suites

An example of a tool extending an existing suite is represented by VR Viewport[3]. It is developed as a software add-on that can be installed in Blender to visualize its 3D Viewport, i.e., a key animation window, in VR using various types of HMDs. Interactions with the hand controller are not allowed. The visualization can be configured to work only with rotation data provided by the headset or using also the position. Another example is the plugin for Autodesk Maya named MARUI[4]. This tool allows users to create low poly models, sketches and animation in VR by using the HTC Vive's or Oculus Rift's controllers. The functionalities offered by the plugin are managed by means of 3D widgets. With respect to animation, the plugin includes functionalities for manipulating bones, inserting/deleting keyframes and navigating the timeline. All the remaining features of Maya can be accessed trough a set of controllers-activated shortcuts (which can be configured to activate frequently used functionalities) and menus representing the native 2D interface of the underlying software. Due to the complexity of the interface of Maya (whose functionalities are mostly maintained), the above solution is probably targeted more to professional than to novice users.

Ad-hoc VR- and AR-based animation tools

The approach that relies on ad-hoc animation tools makes it possible to leverage the features of today's immersive VR systems. However, this advantage is generally balanced by the difficulty to support all the functionalities natively available in existing animation suites, which need to be implemented from scratch. Therefore, tools following this approach often provide only a subset of common animation functionalities, generally addressing those needed for the specific animation tasks/techniques considered. In particular, most of the solutions are targeted to armature deformation-based animation (focusing in particular on characters), since the availability of 3D interfaces is considered as particularly helpful for this task.

For example, the system in [90] proposes a methodology for generating cyclic animations for 3D rigged characters using performance animation in VR. Users wearing a HMD and pinch gloves can choose the part of the model to be animated with one hand and record the animation by simply moving that part with the other hand. The work reported that novice users were able to create realistic layered animations for models with arbitrary topologies in a very short time. However, the system implements a very limited set of functionalities, basically including those needed for the interactive recording of parts' orientation over time using forward kinematics. Thus, according to the outcome of the work, it represents a hardly

[3]VR Viewport: https://github.com/dfelinto/virtual_reality_viewport

[4]MARUI: https://www.marui-plugin.com/)

valid solution for professional animators. In [22], the user can animate virtual characters through a multimodal interface enabling VR- and multitouch device-based collaboration. Through the proposed system, the movement performed by the user's body wearing a HMD is transferred to a virtual character, while another user can refine it using gestures on a touch screen. Besides motion capture, the system provides basic editing operations for refining the character's position, orientation and scaling.

Special-purpose methodologies focusing on specialized tasks are also proposed in the literature. For example, in [234], a VR application to define the control points of a path to animate an objects' movements through a 3D keyframe-based animations is presented. This approach supports a very specific animation task and combines the intuitiveness of manipulating objects using hands for the rough positioning in space with the interaction scheme based on virtual handles for fine adjustments. A similar approach was proposed in [28] to address the generation and editing of displacement animations.

Other work leverages AR for creating character animations. For example, the work in [81] investigates mobile AR-based animation. In this case, the keyframing technique is adopted. The user can choose the transformation to be applied to the virtual character by picking it from an on-screen interface, whereas a selected set of manipulations can be activated/controlled by framing his or her fingers with the device's camera.

Recently, several ad-hoc tools offering a richer set of functionalities have been presented. For example, AnimVR[5] offers functionalities to deal with hand-drawn animations in VR. It allows users to sketch scenes, objects and characters, and to intuitively animate them (or part of them) by applying deformations and transformations. Another example is Masterpiece Motion VR[6]. This standalone tool lets users create the character's skeleton and manage the skinning process in an immersive virtual environment. Unfortunately, the tool does not support any integration with common animation suites; hence, poses created by deforming the skeleton need to be exported to other tools if further animation steps are to be implemented.

Considerations

By analyzing the work reviewed above, it can be observed that, on the one hand, there are several VR animation tools integrated with software suites commonly used by expert animators and, in most of the cases, they only partially

[5]AnimVR: http://nvrmind.io/

[6]Masterpiece Motion VR: https://www.masterpiecevr.com/motion

leverage the characteristics of 3D input/output interfaces. On the other hand, ad-hoc tools generally provide an heterogeneous set of functionalities, with each tool supporting only a specific part of the overall animation pipeline. Moreover, even when a richer set of functionalities is offered by the given tool, animators are still asked to further import/export contents to/from other suites (e.g., for modeling, rendering, etc.); these continuous context switches could limit the added value of VR in the overall animation pipeline [26]. Furthermore, in both cases, the flexibility could represent a severe limitation for both novice users (since the tool could present high learning costs) and for professional users (who expect a tool able to provide advanced functionalities). Lastly, quantitative evaluations regarding factors that could impact on performance and user experience are reported only in a few cases (and not on complete pipelines), making it difficult to choose the tools better satisfying the requirements at hand.

3.2.2 VR Blender tool: First stage

This section will present the design of the VR Blender add-on and its components, focusing on the stage aimed to support common, representative animation functionalities [171].

Architecture

As said, VR Blender is a software package designed with the aim of making the most common tools of a well-known animation suite, i.e., Blender, accessible from within an immersive virtual environment.

Fig. 3.1 shows the overall architecture of the proposed system. The module named Virtual Reality Viewport relies on an existing library, i.e., the Virtual Reality Viewport library[7]. This library is responsible for showing, into an immersive environment, Blender's viewport through a HMD. The new module named *Virtual Reality Plugin* was developed to make available in VR the many common Blender functionalities, and to map on the underlying software the interaction performed with the controllers.

In [171], the HTC Vive VR kit[8] was used for development and testing, though in principle, the other VR systems supported by the Virtual Reality Viewport library can be used. The VR system is able to track both user's hands and the head/gaze movements in a room scale environment. This information is then provided to the Virtual Reality plugin to enable user interaction.

Interactions are based on the hand controllers of the HTC Vive. Names of buttons used in VR Blender for referring to available functionalities are conveniently

[7]Virtual Reality Viewport: https://github.com/dfelinto/virtual_reality_viewport

[8]HTC Vive: https://www.vive.com

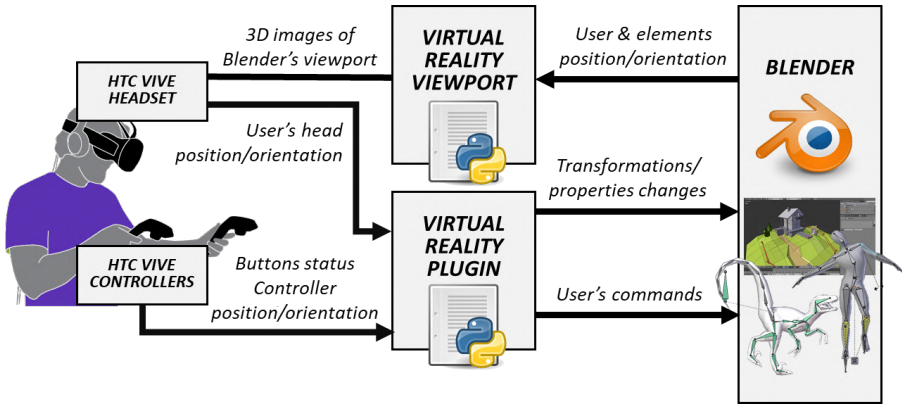


Figure 3.1: Architecture of the VR Blender system [171].

reported in Fig. 3.2. However, further details on the controllers and their functionalities can be found on the HTC Vive's website.

Leveraging the proposed add-on, every element (e.g., 3D objects, armatures, cameras, etc.) in Blender's 3D View can be animated. Once the user has created the 3D models and the scene has been set up in terms of lights, materials, textures, etc., the VR modality can be activated to start animating in VR. The VR mode can be used both to edit existing animations and to work on new ones. Similarly, animations created in VR could then be seamlessly edited using the native Blender's interface.

The activation of the VR mode can be accomplished directly from Blender's interface by pressing a button, grabbing the controllers and wearing the headset. Differently from other existing tools, no further operations, e.g., opening external software and tools, or importing/exporting models are required to work in VR

The user can interact with the system through a state machine that includes the following four states:

- Idle;
- Selection;
- Interaction;
- Navigation.

In the Idle state, the user is not performing any specific operation on the interface. Thus, the system is not receiving any specific input from the user. For example, this state is active when the animator is moving in the virtual environment, e.g., to change his or her point of view.

In the Selection state, the user can select/change the element to interact with at a given time (an armature's bone, a 3D object, a control point of a curve, a

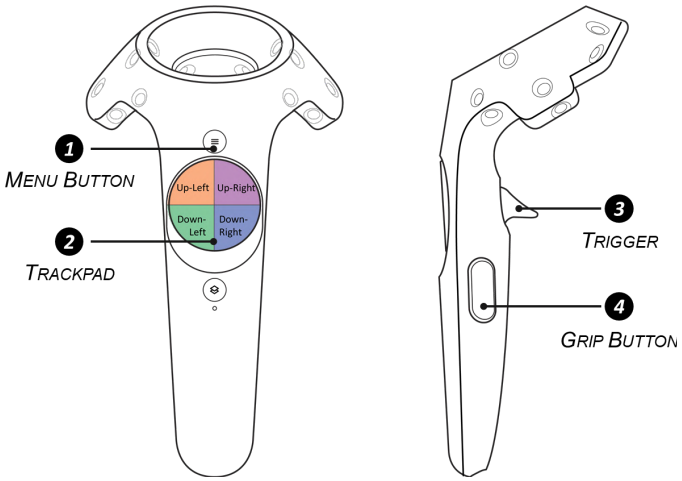


Figure 3.2: HTC Vive hand controllers.

camera, a material, etc.). The selection of a new element is performed by moving the controller on it and pressing the Grip button (see Fig. 3.2). Since the selection could represent an operation performed frequently, the user has the possibility to activate it with both controllers, hence selecting up to two elements (one per controller) at the same time.

The Interaction state allows the user to make changes/modifications to the scene by providing the corresponding inputs that, e.g., set the value of a specific property, modify the position/orientation/scale of an object, interact with controls/components of the graphical user interface available in the virtual environment or define a keyframe, etc. The user can activate this state by pressing the Trigger of either the right or left controller.

In the Navigation state, the user can apply transformations to the VR reference system in order to reach positions in the virtual scene that may not accessible due to the limited size of the HTC Vive's tracked space. By default, the origin of the VR reference system corresponds to the position of Blender's main camera. This origin is directly mapped to the center of the HTC Vive's room. The orientation of Blender's main camera provides the main axes for the reference system, whereas its scaling factor along the three axes identifies the ratio between virtual displacements in the VR environment (measured in Blender's units) and the corresponding real displacements in the tracked space (measured in meters). By default, a one-to-one mapping is set. In this way, one Blender unit corresponds to one meter in the real world. However, mapping can be configured by translating, rotating and scaling Blender's main camera, in order to satisfy requirements in terms of field of view imposed by the task the user intends to carry out. The Navigation state can be activated/deactivated by pressing the Menu button of the right controller.

A screenshot of the virtual environment when the system is in the Idle state

is reported in Fig. 3.3. In this figure, both the controls to manage the proposed VR system, as well as the elements that can be controlled are shown. In the next sections, more details will be provided.

Scene elements

The elements that can be manipulated and consequently animated through the proposed system include unarticulated 3D objects (e.g., the motorcycle in Fig. 3.3 and or articulated 3D objects (e.g., the rigged robot character), which can be controlled by manipulating individually each bone or end-effector in their armature. Moreover, custom cameras, in addition to the main camera, can also be manipulated as any other object.

Object's properties (e.g., transformations, material features, etc.) can be animated separately, by generating so called "actions", i.e., collection of frames in which the value for a given property or set of properties is defined.

Controllers

In order to access animation functionalities and interact with the elements of Blender's 3D View, the system provides users with a virtual representation of the HTC Vive controllers. The current system's state (i.e., Idle, Selection, Interaction, and Navigation) as well as the name of the selected element(s), are displayed in the text close to each controller. Textures displayed on the Trackpads are dynamically changed in order to reveal functionalities that can be activated at any given moment. A panel is added below the text indicating the object's name (as shown in Fig. 3.3 for the left controller), when the user selects a custom camera object. This panel allows the user to visualize the scene as seen by the selected camera with the aim of easing the creation of camera's animations. This approach is generally defined as a "camera-in-hand" metaphor.

Settings panel

The Settings panel (represented on left side of Fig. 3.3) becomes visible once the user presses the Menu button on the left controller. This panel allows the user to set up the animation parameters and select the actual animation mode and/or tool to work with.

Modes The system supports two modes in VR, named *Performance* and *Keyframing* mode. The first mode can be used to generate animations in real-time by transferring into the selected objects' or bones' properties the movements of the tracked controller(s). In this mode, keyframes are automatically registered while the user manipulates selected properties; hence, he or she can animate only two elements at

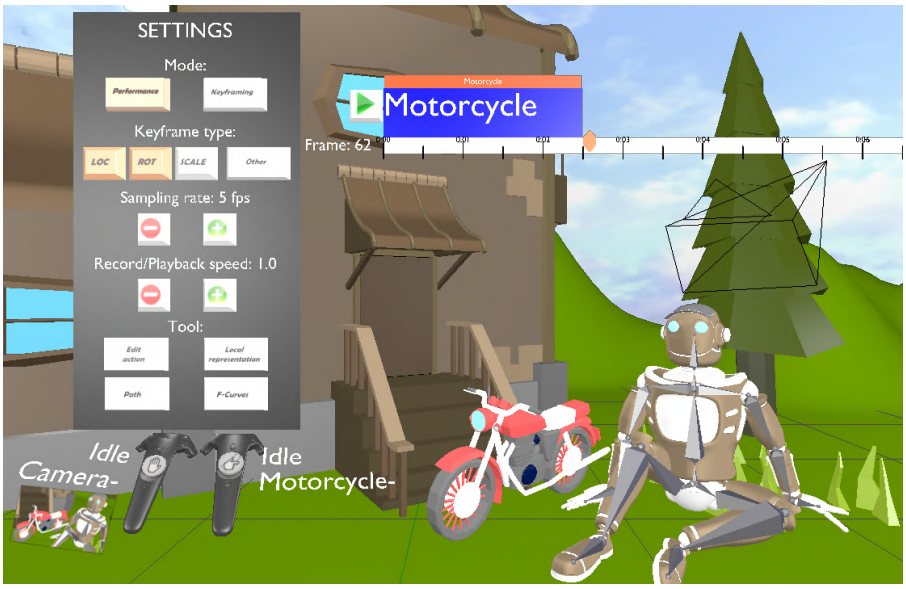


Figure 3.3: Overview of the VR animation environment.

a time. The user can select the properties to be animated by using the corresponding buttons under the *Keyframe type* label, i.e., *Loc*, *Rot*, *Scale*, and *Other*. These buttons allow the user to record keyframes for transformation properties that are frequently used, i.e., location, rotation, and scaling. Another button allows the animator to record keyframes for different elements' properties. The system supports the animation of the materials' diffuse color, transparency and shape keys' deformation intensity. However, in the future, other properties could be managed by adding new scripts able to define a mapping between Blender's scriptable properties and some of the interface's widget or controller's buttons. The *Sampling rate* control allows the user to set the speed at which frames are recorded. The proper number of frames (specified by the value of the *Sampling rate* control) recorded during the user's performance is automatically stored into a new action. The *Record/Playback speed* controls let the user define the speed at which frames are advanced during animation playback/recording. By choosing the Keyframing mode, the user can control the value of a given property at any frame. Interpolation between frames is automatically performed by Blender. Similarly to the Performance mode, in this mode the user is allowed to choose the properties to be saved in the keyframes, and he or she can animate two elements at a time.

Tools Both modes make available a set of tools that can be activated using corresponding buttons in the Settings panel. The set of tools supported includes the *Edit action*, *Local representation*, *Path* and *F-Curve* tools.

The Edit action tool provides the user with functionalities that allow the user to modify the timing of the animation, by delaying/anticipating actions in the

timeline, as well as extending/shortening their duration.

The Local representation tool implements the "world in miniature" metaphor, that can help animators, especially when they are working in Performance mode. When this tool is activated, a duplicate of the element selected with the right controller is added to the scene. This duplicate can be used as a proxy object to interactively transfer the transformations applied by the user to the original element, independent of the position and orientation assumed by the animator in the virtual environment.

The animation approach based on an element following a path can be implemented by using the Path tool. This tool allows the user to add and manipulate a path curve that an element in the scene has to follow. The animation is then generated by recording actions/inserting single keyframes on the evaluation time, i.e., the parameter that defines the position of the element along the given path.

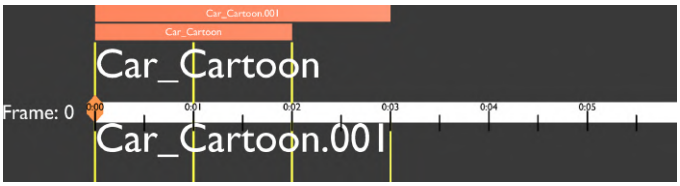
Finally, the F-Curves tool can be used to manually adjust the F-Curves, i.e., the interpolation curves generated by Blender when user records (with the Performance mode) or sets (with the Keyframing mode) new keyframes.

Timeline

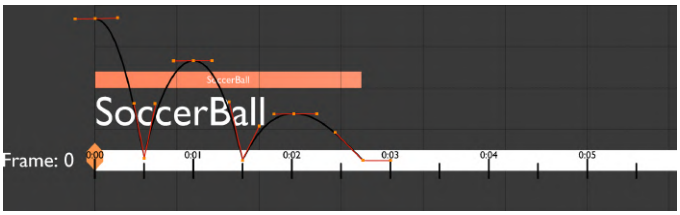
The timeline is in charge of controlling the timing of the animation. This component (depicted on the top-right of Fig. 3.3), shows the duration of the whole animation (white bar), and time information expressed in seconds. The current frame is denoted by an orange cursor on the white bar, as well as by a label on the left side. The names of the elements selected with the left and right controllers (if any) are indicated in white above and below the bar, respectively. Animations of the scene elements generated by the user are represented using colored bars. There is a single orange bar, on the top of the Timeline, for every element in the scene for which keyframes/actions are defined. This orange bar displays the name of the object and specifies the frames at which the animation starts/ends. For each orange bar, there is at least one blue bar that represents an action recorded for the selected elements. Thus, differently from the orange bars which are always visible, blue bars are displayed only when user selects the corresponding element. Blue bars are shown on the top or on the bottom of the timeline, depending on which controller has been used to select the element. Actions of the element selected with the right (left) are shown on the top (on the bottom). Each blue bar has a length that is proportional to the action's duration.

Similar to the controllers, the timeline's aspect is updated depending on the user's interactions. For example, in Fig. 3.4a, it is possible to observe the aspect of the timeline when the user is animating two bones belonging to the same armature in Performance mode. The same visualization is adopted when the user activates the Edit action tool. In this case, actions (blue bars) can be dragged and scaled.

(a) Performance Mode



(b) Keyframing mode



(c) F-Curve tool

Figure 3.4: Changes in the timeline's aspect depending on the current system's configuration.

When the Keyframing mode is selected, the timeline presents a more detailed visualization (shown in Fig. 3.4b) in which it is possible to distinguish keyframes constituting the various actions.

Finally, the visualization reported in Fig. 3.4c is used when the user enables the F-Curves tool. This configuration allows the user to manipulate/adjust the interpolation curves via interactive handles, by using an approach similar to the one used in the Blender's native interface.

Interaction design

As illustrated above, users can interact with the system's functionalities through the VR controllers and the graphics elements displayed in the 3D environment (like the Settings panel, the Timeline and the curves' handles). The controllers' configuration (i.e., the set of available functionalities, which are associated to buttons, textures and labels displayed in the VR environment onto the two controllers) changes according to the actions performed by the user.

Starting from the condition in which the Settings panel is visible, the virtual representation of the controllers is represented in Fig. 3.5a. The texture on the Trackpad button of the right controller indicates that Trigger button can be used

to interact with the Settings panel's widgets. There are no available functionalities mapped in the Trackpad and Trigger button of the left controller, as indicated by the texture. With the Menu button of the right controller, user can activate/deactivate the Navigation state, whereas the same button of the left controller displays/hides the panel (whose position is fixed in the VR environment).

When the Settings panel is hidden, the controllers' configuration changes according to the current state, and the mode or tool which has been activated. Once the Navigation state is activated, the configuration of the controllers presents the layout depicted in Fig. 3.5b. With respect to the right controller, the Grip button lets the user move the origin of the virtual environment (i.e., grabbing Blender's main camera) into a new position. The Trigger button can be used to change the orientation of Blender's main camera, applying a global rotation to the VR reference system. The Trackpad button allows the user to zoom-in/-out the entire scene. The left controller has no associated functionality, as represented by the texture of the Trackpad button. When the Navigation state is deactivated, the system automatically moves to the Idle state waiting for a new user's inputs. The configuration of the controllers when the system moves to the Selection or Interaction state depends on the specific mode or tool currently enabled.

Fig. 3.5c and Fig. 3.5d show the configurations for Performance and Keyframing modes when buttons for inserting keyframes on transformations are activated, respectively. A press on the Grip button of each of the controllers makes the system enter the Selection state, where the user can select the element to be animated (one per controller). During animation, the movement of the controllers is actually transferred to animated properties only when the Interaction state is activated by pressing the Triggers (separately for the elements selected with the left and right controller). The controllers' movement can be limited to specific axes. To this aim, a set of constraints have been defined, which can be chosen from a scrollable list (activated pressing Grip button while in the Interaction state). If the user selects the same element with both controllers and interacts with it by pressing both the Triggers of both the controllers at the same time (by basically executing an 3D pinch gesture), a scaling transformation is applied. In Performance mode, the Trackpad of the right controller is used to control the recording and playback of the animation. In particular, a press on the Trackpad's Up-Right button starts the recording of keyframes for the currently selected element(s), creating a new action. Another press of the button stops the recording and adds the action to the timeline. The Trackpad's Down button plays the recorded action, whereas the Up-Left button deletes it. For what it concerns the left controller, the Trackpad's Down-Right/Left buttons can be used to scrub the timeline by increasing/decreasing current frame by one. The Up-Left/Right buttons increase the scalar value controlling the transformation been animated. In Keyframing mode, the Trackpad of the right controller lets the user insert (with the Trackpad's Up-Right button) or

delete (with the Up-Left button) a key frame for the selected elements. The Trackpad's Down-Left/Right buttons allow the user to scrub the keyframes by setting the current frame to the next/previous key frame defined for the selected element(s). Functionalities of the left controller are the same of Performance animation.

Configurations for Performance and Keyframing modes when keyframes on properties other than transformations are selected are depicted in Fig. 3.5e and Fig. 3.5f. Only one element can be selected at a time using the right controller. The Trigger of this controller has the same functionality described for transformation keyframes (though scaling is not possible). The property actually modified for the selected element during interaction is chosen from a list which can be scrolled using the Grip button of the left controller. The Trigger of this controller can be used to interact with interface elements required to control non-scalar property values. For instance, to handle a material's diffuse color a color wheel and a color picker (moved with the left controller and activated with the Trigger) are used. Trackpads of both the controllers are configured like for transformation keyframes.

The controllers' configuration when the Path tool is activated is illustrated in Fig. 3.5g and Fig. 3.5h respectively for the Performance and Keyframing modes. With respect to the right controller, the Trackpad's down button lets the user create a new path curve for the selected object, if the path does not exist; otherwise, a new control point is added to the curve. The other buttons on the Trackpads of both the left and right controllers have the same meaning described for the previous configurations, but in this case the controlled property is the curve's evaluation time. The Grip button of the right controller is used to select a control point on the curve, whereas the Trigger lets the user move the control point in the 3D space. Concerning the left controller, the Trigger, can be used to translate or rotate the entire path in 3D.

Fig. 3.5i shows the configuration when the F-Curves tool is activated (independent of the mode). Curve's control points and handles can be selected using the Grip button on the right controller (entering the Selection state) and moving selected elements (one at a time) while keeping the Trigger pressed (Interaction state). The Grip button of the left controller is used to scroll the list of interpolation curves available for the selected element (for instance, if only location keyframes were defined, three curves would be available, one per axis). The position of the key frame on the timeline is changed by moving the control point horizontally; moving it vertically corresponds to increasing or decreasing the associated value (e.g., the location, rotation and/or scaling along a given axis, for a transformation keyframe).

Finally, as illustrated in Fig. 3.5j, when the Edit Action tool is activated, the user selects the action to be modified by bringing the controller close to the corresponding blue bar in the timeline and pressing the Grip button on either the left or right controller (two actions can be edited at the same time). With the Trackpad's Up-Left/Right buttons, actions can be delayed/anticipated, whereas with the Down-Left/Right buttons they can be speeded up/slowed down.

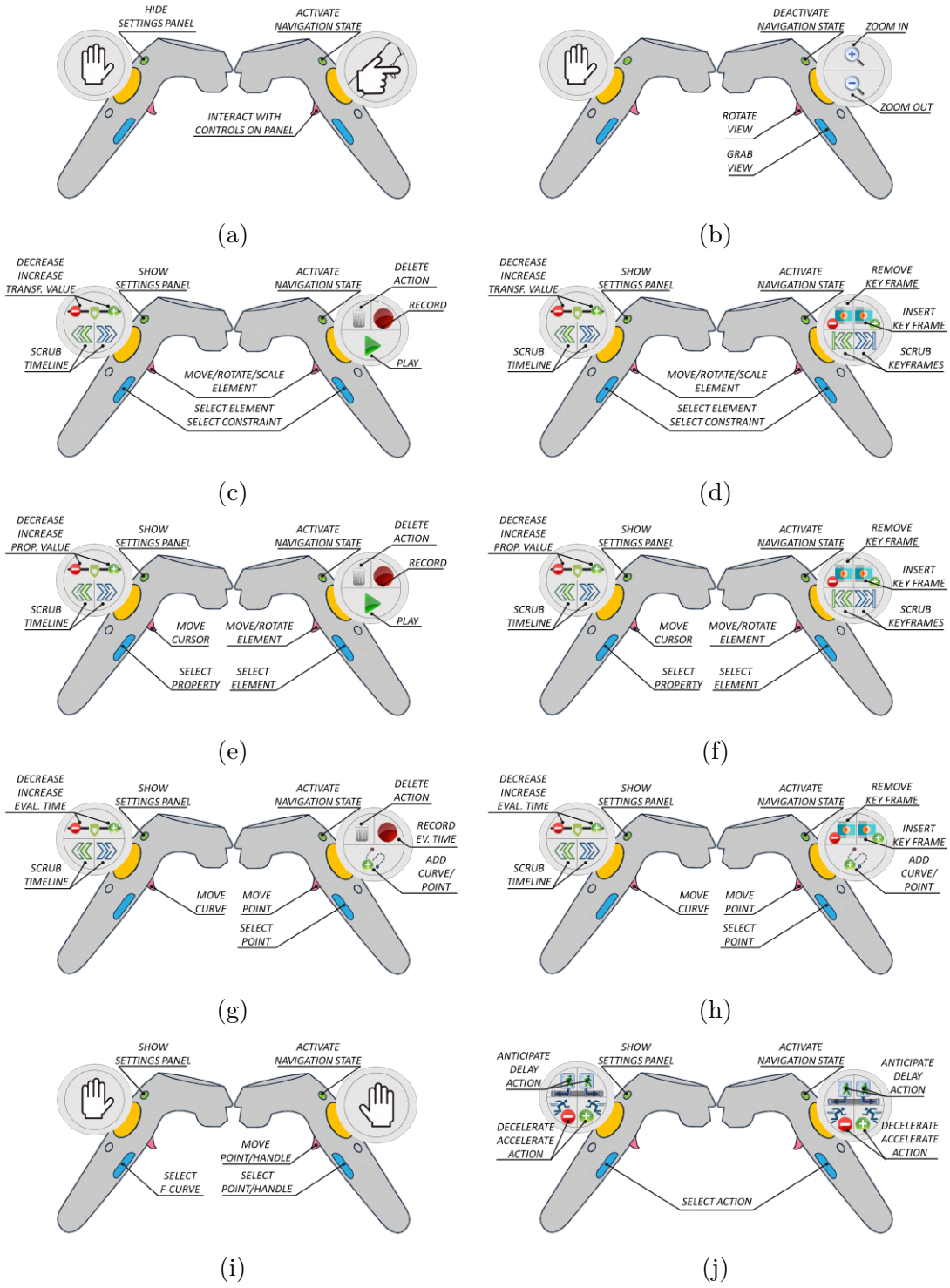


Figure 3.5: Controllers' configurations.

3.2.3 VR Blender tool: Second stage

This section will present the development of the VR Blender add-on focusing on the new modules, components and features introduced in the second stage [51].

Architecture

Fig. 3.6 illustrates the revised system's architecture of VR Blender. Changes made to already-existing modules and the new block introduced (i.e., the *Interface customizer*) with respect to the system presented in [171] are highlighted with light-green colored blocks.

The new functionalities enable the creation and the configuration of armature objects, that can be used to animate articulated virtual characters. Another feature is the possibility to customize the set of native Blender functionalities to make them available in VR. In particular, the first feature is supported through the introduction of three additional tools, named *Rigging*, *Skinning*, and *Constraints*.

The Rigging tool allows the user to configure an armature for a mesh in the scene. In particular, it is possible to define the armature's topology (position, orientation and length of the bones when the armature is in the rest pose), create new bone chains, subdivide bones to create new chains, and set parent relationships among the bones. The Skinning tool lets the user adjust the influence that bones have on the mesh's vertices by modifying vertex weights. Weights are shown by means of different colors, according to the standard visualization used in Blender's Weight painting mode. Here, the blue color indicates a weight equal to 0 (minimum weight, i.e., the bone does not influence the vertex), whereas vertices colored red are assigned a weight equal to 1 (maximum weight). Finally, the Constraints tool allows the user to set the parameters of an IK chain. For example, he or she can define a target bone, change the chain's length, etc.

These new tools can be enabled through the revised version of the Settings panel embedded in the VR interface. The new functionalities are grouped under the *Armatures* label, as shown on the left side of Fig. 3.7.

A new button, named *Bone selection*, has been introduced to allow the user to choose whether to limit the current operation performed (e.g., the insertion of a keyframe) to the whole armature or only to the selected bone. It is worth observing that new functionalities introduced in the revised version of VR Blender could be used also in the original tool; however, in that case, the user was requested to remove the headset (leaving the VR modality) and start operating with the traditional interfaces based on M&K. Thus, the user, forced to use the native Blender's interface, was loosing many of the advantages brought by the use of a 3DUI and by the immersive environment. The aim of the improvements made to VR Blender was to provide the user with tools to fully manage in VR a whole animation pipeline particularly suited to character animation.

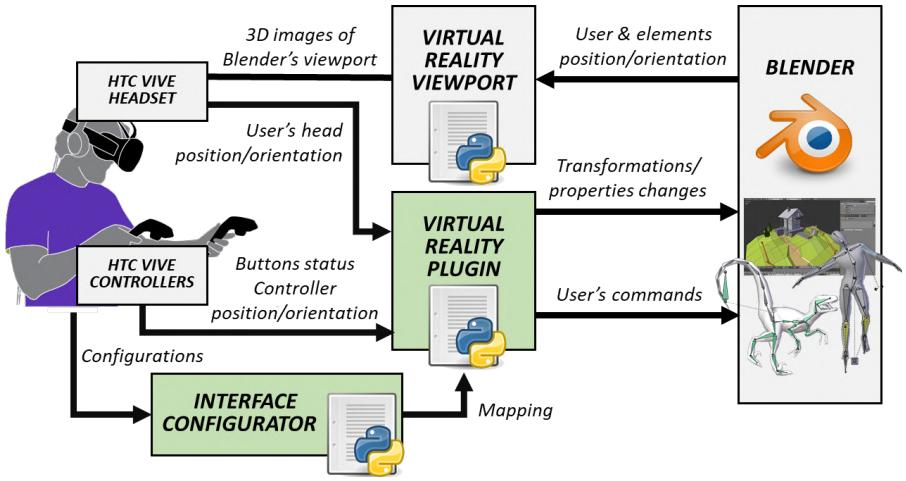


Figure 3.6: Updated architecture of the VR Blender system [51].

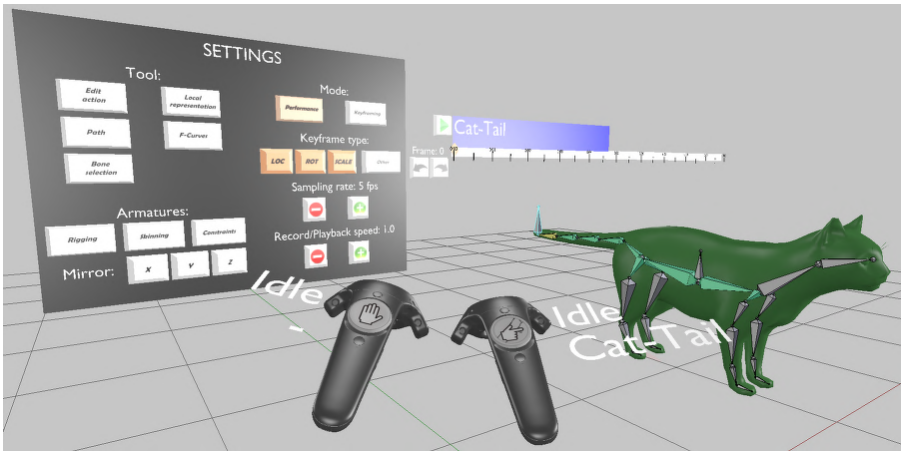


Figure 3.7: First-person view of the new VR Blender interface. The Settings panel includes the new widgets introduced to manage armature deformation-based animation tasks.

Regarding the second new feature introduced in [51] concerning interface customization, a new script (the *Interface customizer*) was developed. This new module allows the user to choose which are the native Blender's functionalities to be made available in VR, and to configure their mapping on the available VR-base interaction modalities. Further details will be provided below.

Interaction design

Controller configurations have been updated according to the changes introduced in VR Blender and the new tools made available. Similar to the original

version, the configurations change based on the active state, mode, and tool. Besides the new configurations defined to handle the new tools developed, differences affected also already-existing configurations, which are reported in Fig. 3.8.

Unlike the former version, the Trigger buttons of both controllers can be used to interact with the controls of the Settings panel, as shown by the texture of the two Trackpads in Fig. 3.8a. Moreover, here the Settings panel does not appear always in the same position in the virtual environment, but it is visualized in the position in which user presses the Menu button to make it visible. Another difference with respect to the controller configurations described earlier is the possibility to switch between a perspective and orthogonal view with the Trackpad buttons of the left controller when the Navigation state is active (Fig. 3.8b).

The configurations designed to handle the new functionalities integrated in the revised tool are reported in Fig. 3.8c, Fig. 3.8d and Fig. 3.8e. For example, Fig. 3.8c reports the configuration of the controllers when the user activates the Rigging tool. Pressing the Grip button of the right controller moves the system to the Selection state, where the animator can select the part of the bone, i.e., tail, head or entire bone, to be manipulated. Only when the user presses the Trigger button of the right controller, making the system enter in the Interaction state, movements of the right controller are transferred to the bone part selected. The Trackpad Up button of the right controller can be used to control the roll of the selected bone. In particular, with a press on the Trackpad Up-Left button the bone's roll is decreased, whereas the Up-Right button is used to increase it. The Down-Right/Left buttons are used to insert and remove bones, respectively. By default, the last bone inserted is automatically set as a child of the currently selected bone (if any); otherwise, it is set at level zero of the armature's hierarchy. Concerning the left controller, the Trackpad Up buttons allow the user to define the association between the mesh and the armature, by using two methodologies. The first (mapped on the Up-Right button) relies on Blender's automatic weight assignment, whereas with the second approach (Up-Left button) an empty group of vertices influenced by the bone is created for each bone. With the Down-Left button, the user specifies whether the selected bone is connected to its parent (in this case, only the orientation of the child bone can be controlled in the posing step) or not (in this case, also the location can be manipulated). In order to speed up the creation of long chains containing several bones with the same length, the user can press the Down-Right button that creates a sequence of two bones by subdividing the selected bone. Operations performed on the armature when the Rigging tool is active can be mirrored by selecting the widget on the Settings panel which corresponds to the symmetry axes desired. When at least one symmetry axis is selected, a mirrored copy of the selected bone is automatically added to the armature, if not existing. When the user moves one of the two bones (the original or the duplicate), the system automatically applies a mirrored transformation to the other bone. For example, if mirroring is enabled along the X-axis and the user applies a translation of 0.5 units along this axis to

the bone's head, the duplicated bone's head is translated by –0.5 units along the same axis.

Fig. 3.8d reports the controllers' configuration when the user enables the Skinning tool. Considering the right controller first, the Trigger button can be used to set the vertices' weight, making the system move to the Interaction state. In this state, all the vertices spanned by a spherical object attached to the right controller, later referred to as 3D brush, are inserted in the current vertex group. The use of a brush mimics the 2D circular selector used in the native Blender's interface. The weight for the newly inserted vertices in the group is specified by the user, as illustrated in Fig. 3.9. The current weight is reported on the text displayed on the right side of the controller. Moreover, when the system is in the Interaction state, all the mesh's vertices are visualized as black dots (as illustrated in the zoomed area in Fig. 3.9). In other states, vertices are hidden. The user can choose the vertex group to be modified by moving the right controller close to the bone which represents the new vertex group and pressing the Grip button. With the Trackpad Up-Left(Right) button the user can decrease(increase) the weight to be applied to the vertices spanned by the 3D brush during the interaction, whereas the Trackpad Down buttons allow the user to increase (with the Down-Right button) and decrease (with the Down-Left button) the size of the 3D brush. The Trigger button of the left controller can be used to translate/rotate the select bone in order to visualize in real-time the effect of the assigned weights. The Trackpad and Grip buttons have no associated functionalities.

Finally, Fig. 3.8e shows the controllers' configuration when the Constraints tool is enabled. Starting with the right controller, with the Grip button user can select a bone from the current active armature. The Trigger button can be used to apply a translation or a rotation to the selected bone. With the Trackpad Down buttons, the user can configure an IK chain by setting a target (i.e., the bone that controls the position of the end-effector) and pole (i.e., the component that controls the overall direction of the chain) for it. The first step for configuring the IK chain consists of selecting a bone that will be used as the end-effector. Then, the user gets close the right controller to another bone of the same armature and he or she can choose to press the Trackpad Down-Left button to create the IK chain by setting the last bone as pole or the Trackpad Down-Right button to set it as a target bone. With a press of the Trackpad Up-Left (Right) button, the user can increase (decrease) the length of the IK chain if the selected bone owns an IK constraint; otherwise, the value will be used to set the length of the next selected bone with that constraint. When the length is equal to 0, all the bones belonging to the same chain for which a parent relationship exists with the selected bone will be affected by the IK constraint; for values larger than 0, only a restricted number of bones equals to that value will be influenced. The Trackpad, Grip and Trigger buttons of the left controller have no associated functionalities.

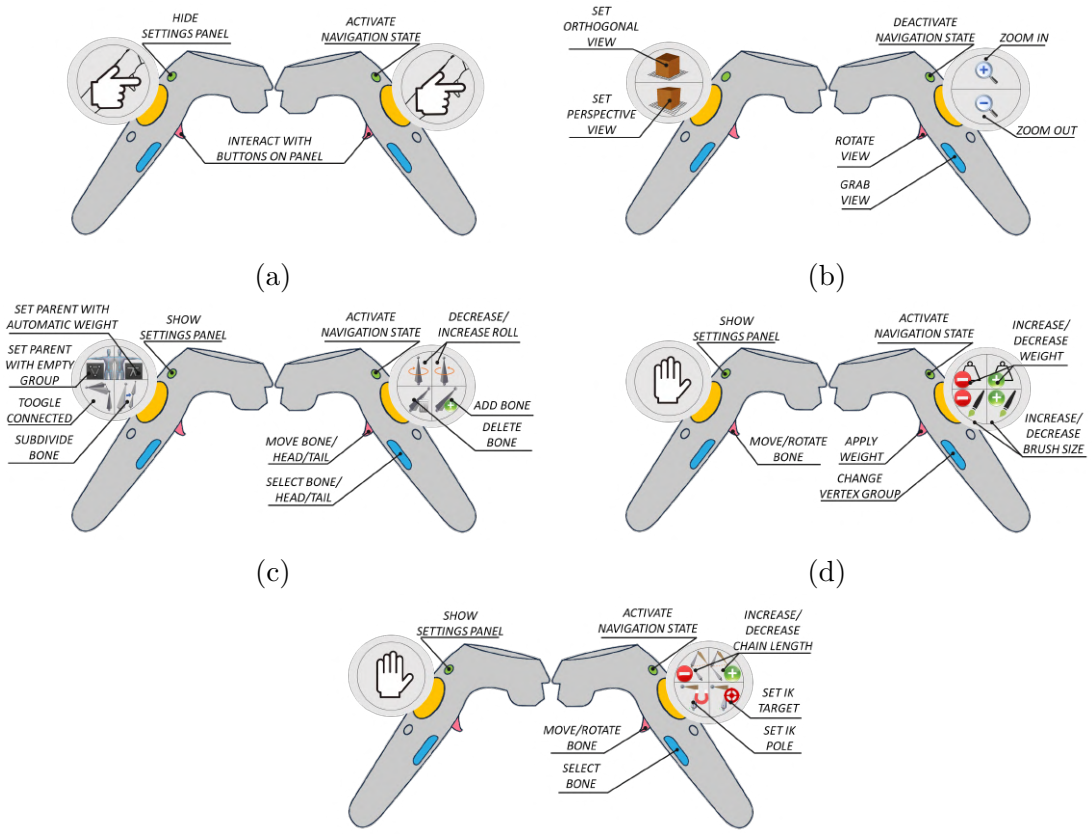


Figure 3.8: Controller's configurations enabling interaction with new tools.

Interface customization

In addition to the new tools, another important feature was introduced in [51] that provides the possibility to the user to customize the set of Blender's functionalities available in VR. Moreover, the user can choose the way to interact with the new functionalities, thus decreasing the cognitive load and learning cost related to the use of the VR modality. As said, this feature was implemented by the newly developed script named Interface customizer. Once installed, the script automatically adds a new module named *Custom VR Mapping* to Blender's Tool Shelf (i.e., the set of panels on the left of the 3D View in the native Blender's interface). By leveraging this panel, the user can create a mapping between a set of functionalities and the available VR interaction modalities that will be used to handle them in VR. After the setup, the interactions with new functionalities will be managed by the Virtual Reality Plugin, like all the states, modes, and tools available in VR Blender.

The script supports the configuration of two types of Blender's elements, namely

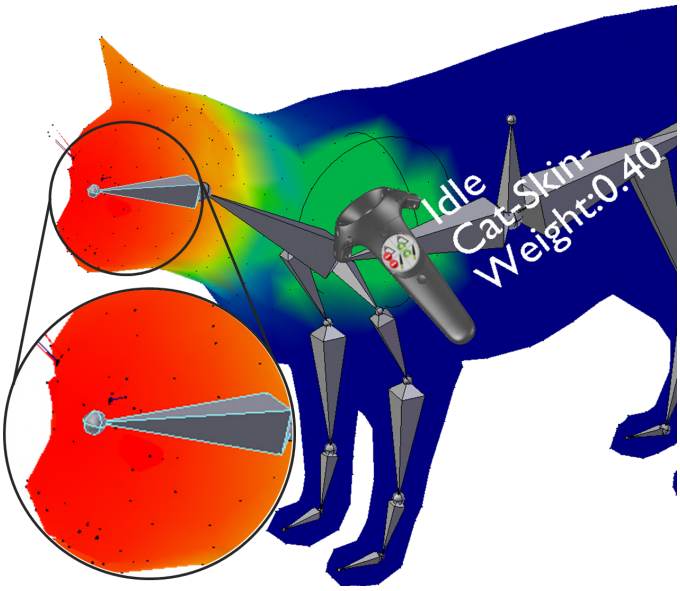


Figure 3.9: Interaction state when the Skinning tool is enabled. The bone currently manipulated and the weight set to new vertices are shows to the right of the virtual controller. All the mesh's vertices are displayed as black dots, as highlighted in the zoomed area.

functions and *controls*. Functions are those activities that imply the execution of a single operation (e.g., the addition of an object to the current scene, the playback of an animation, the definition of a modifier/constraint to the selected object/bone, etc.). In Blender's interface, functions are executed by pressing a button or using a keyboard shortcut. Controls are those variables, e.g., the location/orientation/scale of an object along a given axis, the parameter of a constraint, etc., that can be configured by leveraging the common components of the Blender's native interface, e.g. sliders, text boxes, check boxes, etc. For example, in Fig. 3.10a it is shown the panel that the user can leverage to configure a Limit Rotation bone constraint through the Blender's native interface. The definition of the bone constraint can be regarded as a function, whereas setting the parameters for configuring the constraint can be considered as controls. The setup of the Limit Rotation bone constraints requests to manipulate binary values and numerical or categorical inputs (those highlighted in red in Fig. 3.10a).

In order to configure the methodology to access parameters like those above in VR, the user can take advantage of the Custom VR Mapping panel (illustrated in Fig. 3.10b). As said, this panel is automatically included among the set of panels available on the Blender's Tool Shelf. At the beginning, only the *Add Function*, *Add Control*, *Clear* and the *Get in VR* buttons are displayed in the panel. The first two buttons can be used to configure a new mapping between a function or control and the way to handle it in VR. A click on the Clear button allows the user

to delete all the mappings defined. Mapping can be managed with a "*When-Do*" logic, implying that when a given condition occurs, the system does something. This logic can be implemented by configuring the When and Do boxes underneath.

The When combo boxes can be leveraged to configure the way the user can activate a function or modify a control in VR. In particular, when defining the logic for the functions, the user has two possible alternatives: the recognition of a specific event, i.e., *Key down*, *Key press* or *Key up*, triggered by the Grip button of one of the hand controllers, or a press on a *VR Button*, i.e., a 3D button component automatically added on the right side of the Settings panel in the VR environment (block highlighted in red with a label 1 assigned in Fig. 3.10c). Labels indicating the function enabled by pressing the button as well as the target are displayed, in order to help the user remember the configured mappings when operating in the immersive environment. For example, the labels "*Selected Object: Selected bone*" and "*constraints_add(type='LIMIT_ROTATION')* shown in Fig. 3.10c indicate that once user interacts with such a button a Limit Rotation bone constraint will be set for the currently selected bone of the active object (an armature).

The VR buttons are displayed only when the Settings panel is visible. The interaction with these buttons is performed through the Trigger buttons of the two controllers, in the same way the user interacts with other buttons of the Settings panel. Alternatively, the user can configure a When combo box for functions by leveraging the three events associated with the Grip button status of both the hand controllers. When the user presses the Grip button, a single Key down event is triggered, whereas the Key pressed event is generated continuously as the user maintains the button pressed. Finally, a single Key up event is fired when the button is released. When the user defines a mapping by making use of these three events, a new block is attached to the right side of the Settings panel (like block 2 in Fig. 3.10c). Differently than with VR button, in this case, the block displays only the label indicating the associated function and its target, since there are no interactable 3D components. As a matter of example, the mapping represented by the block 2 of Fig. 3.10c allows the user to call the function *constraints_clear()* that deletes all the constraints for the currently selected bone in the active armature. In this example, the given function is activated by pressing the Grip button of the left controller.

With respect to the controls, the When part of the mapping can be configured by choosing among two alternatives: the *VR Control*, i.e., a 3D interactable component or one of the events generated by interacting with Trackpads Up-Right/Left and Down-Right/Left buttons. In the case of VR Control, a new block is attached to the right part of the Settings panel (block 3 in Fig. 3.10c). The block displays some labels, indicating the name of the associated variable and its current value, as well as two buttons, that can be used to increase or decrease the current variable's value. When the mapping relies on the Trackpads' events, only labels are displayed (blocks 4–6 in Fig. 3.10c). The types of variables that can be managed are binary, integer,

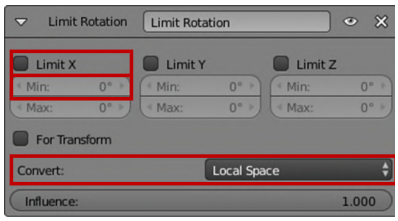
floating point, enumeration, and object. The type of the variable is automatically determined by the system when the user tries to modify it through the VR control or with the Trackpads. In this way, the system is able to find autonomously, a new valid value for the variable. For example, if the user tries to modify a binary variable, then the systems automatically inverts its value; if the variable is an enumeration, then the system sets the value to the next valid item in the list of categorical alternatives; if it is an integer, the value is increased or decreased by a fixed amount. Blocks 3–6 shown in Fig. 3.10c can be used to configure the parameters of a specific constraints for a bone named *Bone.001* in the *Armature* object, as indicated in the labels. In particular, block 3 allows the user to switch the value of the binary variable named *use_limit_x*, which for the Blender native interface corresponds to checking/unchecking the *Limit X* check box of the Limit Rotation constraint shown in Fig. 3.10a. The Trackpad's Up-Right/Left buttons of the right controller (blocks 4 and 5) can be used to define the value of the *min_x* variable which defines the minimum value along the above axis. Finally, with the Up-Right button of the left controller (block 6), the user can specify a different value for the field named *Convert'* (enumeration field highlighted in red in Fig. 3.10a), which defines the coordinate system to be used by the constraint.

With the Do text box, the user defines the reference to the function/control to be executed/set when the condition specified in the corresponding When block is satisfied. These references are expressed as Python commands, which Blender is able to interpret in order to implement the user-defined behavior. Moreover, the references inserted by the user are processed by the system to automatically determine the labels to show on the blocks attached to the Settings panel discussed above.

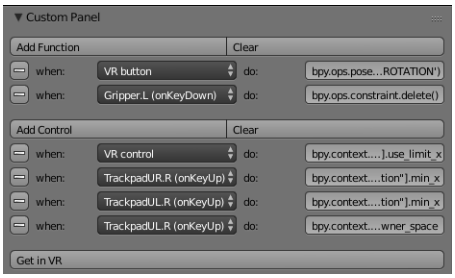
In order to make the system usable by different user categories (including those with limited or no expertise with Python), several alternatives are supported to retrieve the python code needed to specify commands:

- using the online Blender's API documentation[9];
- modifying the value to be controlled or executing the function by means of the native Blender's interface, then copying and pasting the last row displayed in the Blender's Info Editor;
- right clicking with the mouse on the corresponding graphics component of the native Blender's interface, then from the contextual menu that appears choosing the entry "Copy Python Command", or manually copying the string displayed when the mouse is hovering over that component.

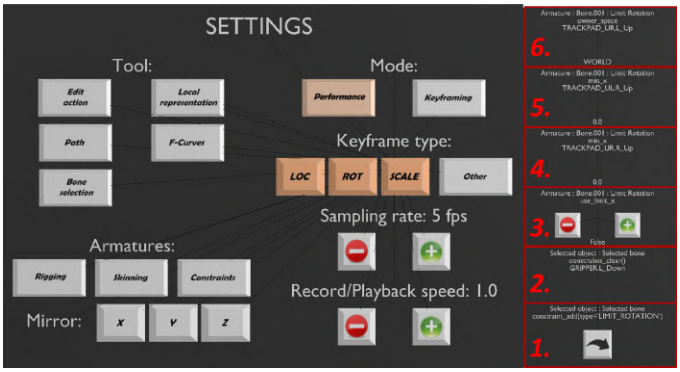
[9]Blender API documentation: https://docs.blender.org/api/2.79/



(a) Panel for configuring a Limit Rotation bone constraint.



(b) Custom VR Mapping panel.



(c) VR components attached to Settings panel.

Figure 3.10: Interface customization panels.

The system is also able to recognize keywords which allow the user to extend the effects that precise Python commands have on a specific target object also to different objects in the VR environment.

For example, the Python command that can be used to change the floating-point value of the *Min* text box for the Limit Rotation constraint shown in Fig. 3.10a (retrieved using one of the three methods mentioned above) is *bpy.context.object.pose.bones['Bone.001'].constraints['LimitRotation'].min_x*.

From the analysis of the above string, it is possible to notice that a specific bone (namely, "*Bone.001*") is used as the target for the operation. In order to extend the command to the last bone selected, the user should simply replace the bone name above with the keyword *SELECTED_BONE*. In addition to the previous keyword, the user can also use the keyword *SELECTED_OBJECT* to extend the specified command to the currently selected object. It is worth observing that Fig. 3.10c

presents both blocks for which a target bone/object is not directly specified but it is dynamically defined based on the bone or the object currently selected in VR, and the other blocks for which a precise target is given. The button on the left of each When-Do pair in the Custom VR Mapping panel (Fig. 3.10b) lets the user delete a given mapping.

When all the mappings have been defined and the user is ready to start animating in VR, he or she can press the *Get in VR* button on the bottom of the Custom VR Mapping panel (Fig. 3.10b) to attach the new blocks on the Settings panel or update previously created blocks. New mappings defined through the Custom VR Mapping panel on the Blender's native interface become automatically available also in VR. Their visibility depends on the status of the Settings panel. When the panel is hidden, the custom configurations are not visible and they have no effect, since the default mappings which depend on the currently active state, mode or tool, are restored. When the Settings panel is activated, custom mappings become visible and they can be used.

Several videos are available for download[10] to illustrate the flexibility of the devised interface customization mechanism. In particular, in the first video, the user creates a mapping to manage a Track to constraint, which makes the selected object follow another object. Both the objects, and the constraint, can be defined using custom When-Do blocks. The second video reports how to configure the parameters of a Limit Rotation bone constraint for bones belonging to a rigged character's armature. The third video shows a user managing very heterogeneous Blender's functionalities by setting custom functions and controls for, e.g., defining objects' shading, modifying the Timeline's start and end frames, and configuring a sub-surfacing modifier. It is worth observing that the devised approach can be easily used both with animation- and modeling- oriented functionalities.

3.2.4 Experimental evaluation: First stage

In order to investigate the possible advantages brought by the use of VR for executing common animation tasks, a user study was carried out involving students and academic staff of Politecnico di Torino.

Participants

In the user study 27 participants were involved (19 males and eight females), aged 21–43 ($M = 25.41$, $SD = 4.63$). Participants were grouped into two categories, depending on their previous experience in the computer animation field.

The first group included 15 participants that could be considered as non-professional

[10]Videos on interface customization: http://tiny.cc/ipvsaz

users (in the following referred to as NPUs), since they were students attending a course on computer animation with Blender.

The second group was composed by 12 participants that could be considered as professional users (in the following referred to as PRUs), because of their expertise acquired working in the field or teaching similar courses. On average, PRUs were also much familiar with VR technologies.

Tasks

Each participant carried out five representative animation tasks, working with both the proposed VR-based interface (VRI) and the native Blender's interface (BNI). Between each task, breaks were allowed. Tasks reported in Fig. 3.11 were designed with the aim of testing most of the functionalities implemented. Interfaces, as well as tasks, were tested in a random order, by trying to balance the coverage of all the configurations and limit biases possibly due to learning effects. All tasks requested participants to work on a given object (represented with a green color) in order to recreate the reference pose or the target animation (colored red). Some tasks presented a small spatial offset between the given and the target object, to make it clearer the visualization of the scene's contents and of tasks' objectives. The five tasks are described below.

- *Posing task* (Fig. 3.11a): users were asked to articulate the armature of a crocodile character to make it match the reference pose as closely as possible. The character is composed of 17 bones and 51 DOFs; both the forward and inverse kinematics methods had to be used to articulate it properly.

- *Keyframing task* (Fig. 3.11b): participants were requested to animate an unarticulated object, i.e., the car object, by defining three transformation keyframes for recording its location and orientation and three keyframes to specify the transition of its material's diffuse color between a dark and a light shade and vice versa.

- *Performance task* (Fig. 3.11c): the goal of this task was to recreate as faithfully as possible the animation of an eagle character. The character was controllable through three bones: two bones articulated the wings by leveraging the inverse kinematics, one bone controlled the location and orientation of the character.

- *Path task* (Fig. 3.11d): the goal of this task was to animate a planet's revolution by creating first an ellipsoidal path and then animating the evolution time through two keyframes set at the very beginning and end of the animation.

- *Interpolation curve editing task* (Fig. 3.11e): participants were requested to animate a bouncing ball by defining six location keyframes and then adjusting

the interpolation curves using the F-Curves tool to make the bounces on the floor sharper.

Videos showing the execution of the five animation tasks are available for download[11].

Before starting the experiments, participants were provided with instructions for operating with both the BNI and the VRI, by focusing on the main functionalities needed to complete the assigned tasks. Afterward, participants were left free to familiarize with the two interfaces by experimenting them for 5–10 minutes with other scene elements. During the execution of the tasks, participants were allowed to ask for support, since all them already knew the BNI but they were new to the VRI. No time limit was set.

Evaluation Criteria

The evaluation was carried out by considering both objective and subjective measurements. With respect to objective measurements, two metrics used in [47] were considered. The first metric, named completion time, measures the time needed to carry out the assigned task. The second metric, called animation accuracy, evaluates the differences between the user-generated animations and the reference in terms of Euclidean (for the position) and angular (for the orientation) distances averaged for each scene element and frame.

Subjective measurements were collected through an after-test questionnaire, which is available for download[12]. The questionnaire was organized into two sections. The first section assessed the overall usability of the two interfaces by measuring it through the System Usability Scale (SUS)[40]. The second section measured participant satisfaction in using the two interfaces. This section relied on questions presented in a previous work focused on 3D animation with non-traditional interfaces [243].

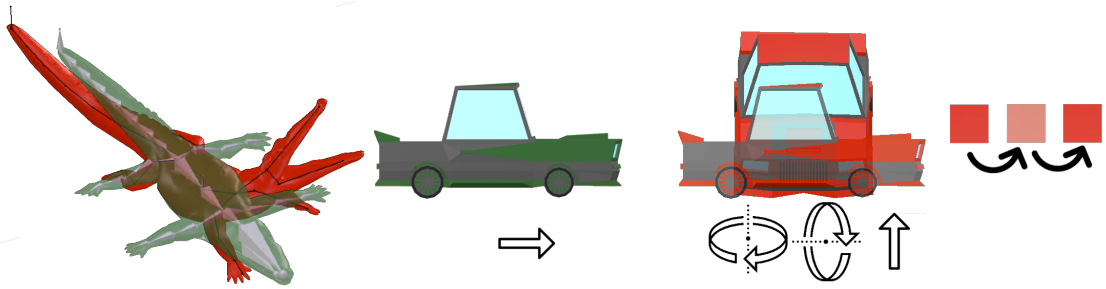
At the end of the experiment, participants were also requested to express their preferences for the two interfaces in the execution of each task.

3.2.5 Results: First stage

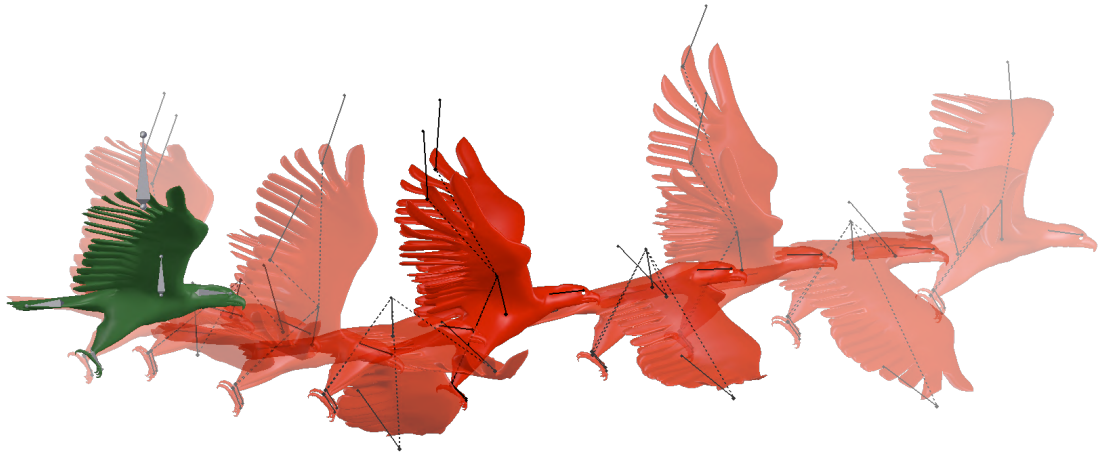
Results aimed to compare the performance of BNI and VRI will be presented by first describing objective results and then focusing on subjective results.

[11] Videos of the experiments: https://goo.gl/EBnP5E

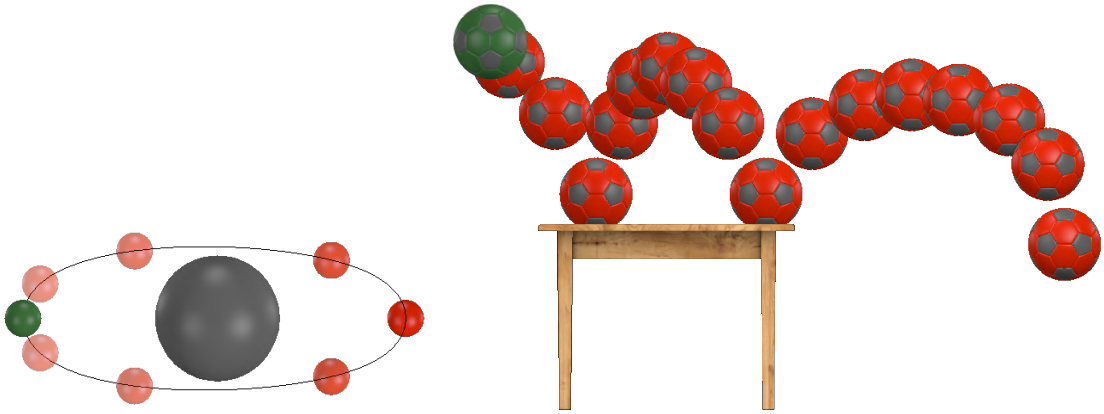
[12] Questionnaire: https://goo.gl/hhFCyi



(a) Posing task (b) Keyframing task



(c) Performance task



(d) Path task (e) Interpolation curve editing task

Figure 3.11: Tasks considered in the experiments.

Objective results

Fig. 3.12 reports measurements concerning completion time and animation accuracy for both the user categories. In order to analyze collected data, paired

sample Student's t-tests were used (all the assumptions for performing t-tests were satisfied). In the above figures, significant results ($p < 0.05$) are highlighted with the * symbol.

By focusing first on NPUs, it can be noticed that participants benefited in all the tasks from the use of the VRI in terms of completion time (Fig. 3.12a). With respect to the BNI, the VRI allowed users to be, on average, 35% faster in all the tasks. In fact, results were all statistically significant. In particular, analyzing the specific tasks, gain in terms of time was 24% in the Posing task (BNI: $M = 5$ min 56 s, $SD = 52$ s; VRI: $M = 4$ min 34 s, $SD = 29$ s; $t(14) = 4.10$, $p < 0.01$, $d = 1.93$), 20% in the Keyframing task (BNI: $M = 5$ min 35 s, $SD = 49$ s; VRI: $M = 4$ min 29 s, $SD = 33$ s; $t(14) = 2.63$, $p = 0.03$, $d = 1.55$), 51% in the Performance task (BNI: $M = 4$ min 14 s, $SD = 1$ min 58 s; VRI: $M = 2$ min 6 s, $SD = 45$ s; $t(14) = 2.81$, $p = 0.02$, $d = 1.42$), 48% in the Path task (BNI: $M = 4$ min 27 s, $SD = 1$ min 17 s; VRI: $M = 2$ min 21 s, $SD = 45$ s; $t(14) = 6.72$, $p < 0.01$, $d = 1.97$), and 52% in the Interpolation curve editing task (BNI: $M = 8$ min 0 s, $SD = 1$ min 25 s; VRI: $M = 3$ min 55 s, $SD = 46$ s; $t(14) = 10.43$, $p < 0.01$, $d = 3.57$). No statistically significant differences were found in terms of accuracy between the VRI and NBI (Fig. 3.12c). In general, data trended toward a higher accuracy with the VRI, suggesting that statistical significance might be obtained with a larger sample size.

Regarding PRUs, results confirmed the superior performance in terms of completion time of the VRI compared to the BNI, which was already observed for NPUs. On average, with the VRI users were 37% faster than with the BNI (Fig. 3.12b). In this case, the percentage of time saved was 37% in the Posing task (BNI: $M =$ 7 min 53 s, $SD = 1$ min 29 s; VRI: $M = 5$ min 1 s, $SD = 1$ min 22 s; $t(11) = 4.24$, $p < 0.01$, $d = 2.02$), 29% in the Keyframing task (BNI: $M = 5$ min 8 s, $SD = 1$ min 51 s; VRI: $M = 3$ min 39 s, $SD = 59$ s; $t(11) = 4.02$, $p < 0.01$, $d = 0.99$), 54% in the Performance task (BNI: $M = 3$ min 39 s, $SD = 1$ min 3 s; VRI: $M = 1$ min 42 s, $SD = 42$ s; $t(11) = 5.96$, $p < 0.01$, $d = 2.18$), 36% in the Path task (BNI: $M = 4$ min 37 s, $SD = 1$ min 3 s; VRI: $M = 2$ min 58 s, $SD = 36$ s; $t(11) = 4.95$, $p < 0.01$, $d = 1.92$), and 33% in the Interpolation curve editing task (BNI: $M =$ 5 min 28 s, $SD = 48$ s; VRI: $M = 3$ min 41 s, $SD = 1$ min 6 s; $t(11) = 4.70$, $p < 0.01$, $d = 1.86$). All the differences in terms of accuracy like for NPUs were not statistically significant (Fig. 3.12d).

Further insights can be obtained from the comparison of the results for the two user categories (using unpaired Student's t-tests). With respect to the completion time, it can be noticed that with the BNI, the NPUs ($M = 5$ min 56 s, $SD = 52$ s) were significantly faster than the PRUs ($M = 7$ min 53 s, $SD = 1$ min 29 s) in the Posing task ($t(17) = -3.30$, $p < 0.01$, $d = -1.53$); this means that they spent a lot of time in refining all the armature's DOFs. Conversely, in the Interpolation curve editing task, which is probably the task characterized by less intuitiveness, opposite results were achieved (NPUs: $M = 8$ min 0 s, $SD = 1$ min 25 s; PRUs: $M = 5$ min

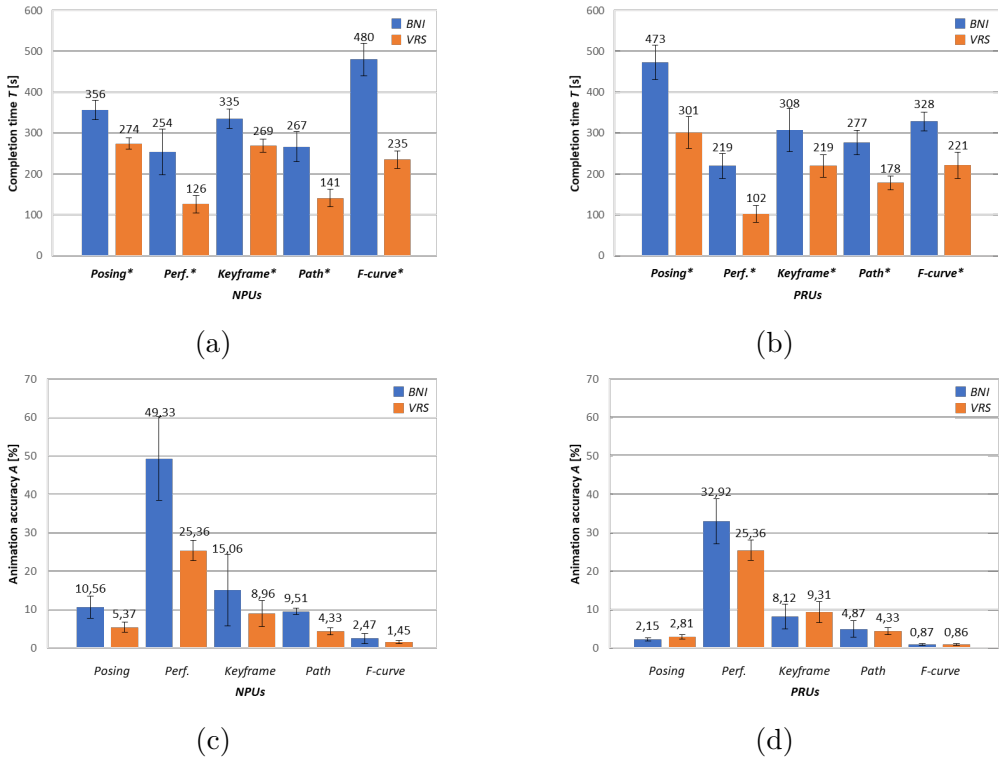


Figure 3.12: Objective results for non-professional and professional users. Average values (bars' height) and standard deviations (error bars) are shown.

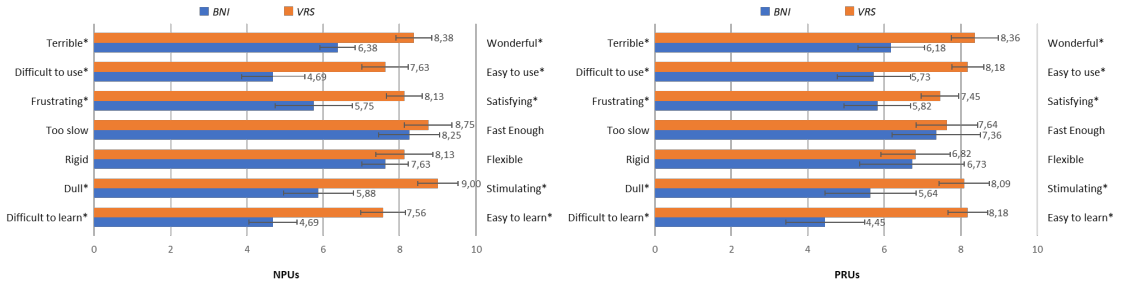
28 s, $SD = 48$ s; $t(17) = 4.94$, $p < 0.01$, $d = 2.30$). No other statistically significant differences were found for completion time. Concerning accuracy, by aggregating results for the two interfaces PRUs were in general more accurate than NPUs. In particular, considering individual tasks and analyzing the interfaces separately, it was found that the higher experience of PRUs with the BNI made these users be more accurate than NPUs in the Posing task (NPUs: $M = 10.56\%$, $SD = 6.29$; PRUs: $M = 2.15\%$, $SD = 0.91$ s; $t(23) = 3.75$, $p < 0.01$, $d = 2.30$)and in the Path task (NPUs: $M = 9.51\%$, $SD = 1.72$; PRUs: $M = 4.87$, $SD = 4.64$ s; $t(23) = 3.04$, $p < 0.01$, $d = 1.25$). No statistically significant differences were found for the VRI. It is worth observing that the comparable values obtained by the NPUs with the VRI with respect to PRUs with the BNI seem to suggest that VR-based systems could be useful in leveling animation skills among different user groups.

Subjective results

Concerning the first section of the questionnaire, which asked participants to rate the system's usability based on the SUS scale [40], it can be observed that VRI was considered as characterized by a usability higher than the BNI by both NPUs

(BNI: $M = 53.43$, $SD = 10.56$; VRI: $M = 75.00$, $SD = 9.40$ s; $t(14) = -5.97$, $p < 0.01$, $d = -2.16$) and PRUs (BNI: $M = 58.86$, $SD = 17.94$; VRI: $M = 75.91$, $SD = 10.97$ s; $t(14) = -4.86$, $p < 0.01$, $d = -1.15$). Analyzing scores assigned to individual SUS statements and focusing on statistically significant results, it can be noticed that the VRI was perceived as easier to learn than the BNI by both the PRUs (BNI: $M = 2.00$, $SD = 1.10$; VRI: $M = 4.00$, $SD = 0.77$; $t(11) = -8.56$, $p < 0.01$, $d = -2.11$) and NPUs (BNI: $M = 1.94$, $SD = 0.77$; VRI: $M = 4.06$, $SD = 0.68$; $t(14) = -10.54$, $p < 0.01$, $d = -2.92$). This is also confirmed by the fact that both the user categories stated they needed more information before starting to use the system with the BNI than the VRI (PRUs – BNI: $M = 4.27$, $SD = 1.27$; VRI: $M = 2.36$, $SD = 0.81$; $t(11) = 4.01$, $p < 0.01$, $d = 1.79$; NPUs – BNI: $M = 4.19$, $SD = 0.91$; VRI: $M = 2.31$, $SD = 1.08$; $t(14) = 7.32$, $p < 0.01$, $d = 1.88$). The two categories also reported that the VRI was found unnecessarily complex more than the BNI (PRUs – BNI: $M = 2.82$, $SD = 1.17$; VRI: $M = 2.00$, $SD = 0.89$; $t(11) = 2.32$, $p = 0.04$, $d = 0.79$; NPUs – BNI: $M = 2.75$, $SD = 1.06$; VRI: $M = 1.31$, $SD = 0.48$; $t(14) = 6.44$, $p < 0.01$, $d = 1.74$). Moreover, both the categories of participants judged the VRI easier to use (PRUs – BNI: $M = 3.00$, $SD = 0.77$; VRI: $M = 4.27$, $SD = 0.65$; $t(11) = -9.04$, $p < 0.01$, $d = -1.78$; NPUs – BNI: $M = 2.69$, $SD = 1.08$; VRI: $M = 4.06$, $SD = 1.00$; $t(14) = -5.06$, $p < 0.01$, $d = -1.32$). In general, the use of VR for animation was appreciated by both the users' categories, since they would like to use the VRI more frequently than the BNI (PRUs – BNI: $M = 3.18$, $SD = 1.17$; VRI: $M = 4.00$, $SD = 0.63$; $t(11) = -2.32$, $p = 0.04$, $d = -0.87$; NPUs – BNI: $M = 3.19$, $SD = 1.22$; VRI: $M = 4.31$, $SD = 0.70$; $t(14) = -4.70$, $p < 0.01$, $d = -1.13$). Finally, NPUs felt more confident in using the VRI than the BNI (BNI: $M = 3.19$, $SD = 0.91$; VRI: $M = 3.94$, $SD = 0.77$; $t(14) = -2.42$, $p = 0.03$, $d = -0.89$).

For what it concerns the satisfaction in carrying out the animation tasks evaluated in the second section of the questionnaire through the attributes presented in [243], results are reported in Fig. 3.13a and Fig. 3.13b. Focusing on statistically significant results (marked with *), the VRI was perceived as easier to learn (PRUs – BNI: $M = 4.45$, $SD = 2.16$; VRI: $M = 8.18$, $SD = 1.08$; $t(11) = -8.30$, $p < 0.01$, $d = -2.18$; NPUs – BNI: $M = 4.69$, $SD = 1.30$; VRI: $M = 7.56$, $SD = 1.21$; $t(14) = -7.90$, $p < 0.01$, $d = -2.29$) and to use (PRUs – BNI: $M = 5.73$, $SD = 2.00$; VRI: $M = 8.18$, $SD = 0.87$; $t(11) = -3.94$, $p < 0.01$, $d = -1.59$; NPUs – BNI: $M = 4.69$, $SD = 1.70$; VRI: $M = 7.63$, $SD = 1.26$; $t(14) = -7.48$, $p < 0.01$, $d = -1.96$). This finding seems to suggest that the proposed VR-based interface could be suitable for users with different levels of expertise in the field of computer animation. Both NPUs and PRUs expressed a higher appreciation for the VRI than for the BNI, which was perceived as more wonderful (PRUs – BNI: $M = 6.18$, $SD = 1.83$; VRI: $M = 8.36$, $SD = 1.29$; $t(11) = -4.07$, $p < 0.01$, $d = -1.38$; NPUs – BNI: $M = 6.38$, $SD = 0.96$; VRI: $M = 8.38$, $SD = 0.96$; $t(14) = -8.94$, $p < 0.01$, $d = -2.09$), satisfying (PRUs – BNI: $M = 5.82$, $SD = 1.83$; VRI: $M =$



(a) (b)

Figure 3.13: Subjective results concerning satisfaction based on criteria (items) in [243] for non-professional and professional users. Average values (bars' height) and standard deviations (error bars) are shown.

7.45, $SD = 1.04$; $t(11) = -4.50$, $p < 0.01$, $d = -1.10$; NPUs – BNI: $M = 5.75$, $SD = 2.08$; VRI: $M = 8.13$, $SD = 0.96$; $t(14) = -5.56$, $p < 0.01$, $d = -1.47$) and stimulating (PRUs – BNI: $M = 5.64$, $SD = 2.50$; VRI: $M = 8.09$, $SD = 1.38$; $t(11) = -4.37$, $p < 0.01$, $d = -1.22$; NPUs – BNI: $M = 5.88$, $SD = 1.89$; VRI: $M = 9.00$, $SD = 1.10$; $t(14) = -6.48$, $p < 0.01$, $d = -2.02$). No statistically significant differences were found between the two interfaces in terms of flexibility and perceived operation speed (although, VRI obtained the best results in terms of completion time according to the objective results presented above).

Finally, participants were requested to express their preferences for the two interfaces, which are tabulated in Table 3.1. On average, NPUs expressed a higher preference for the VRI than for the NBI in all the assigned tasks. Regarding PRUs, the VRI remained the interface that was most preferred, overall, though distances between the two interfaces were smaller. The tasks for which a high appreciation was registered were the Posing, Performance and Interpolation curve editing ones.

Comments collected during the experiments show that outcomes above are strongly related to design choices made in the interaction design, i.e., the way chosen to let the users access/control the various features. For example, participants would have preferred to experiment with the possibility to configure the mapping of the functionalities on the controllers (this need was then considered in [51]).

3.2.6 Experimental evaluation: Second stage

The experiments carried out in [51] were aimed at investigating two different aspects that can affect the performance of a user animating into an immersive environment: the precision of interaction and the user experience associated with the functionalities offered by the VR interface. It is worth observing that, in this case, the various steps of a (character) animation pipeline were considered. With the aim of investigating the above perspectives, two user studies were carried out

Table 3.1: Users' preferences for individual tasks and overall.

Tasks	**NPUs**		**PRUs**	
	BNI	*VRI*	*BNI*	*VRI*
Posing	0%	100%	19%	81%
Performance	25%	75%	10%	90%
Keyframing	32%	68%	45%	55%
Path	12%	88%	45%	55%
Interpolation curve editing	12%	88%	36%	64%
All	0%	100%	45%	55%

by involving students and academic staff at Politecnico di Torino. In the following, further details of the two user studies will be provided.

Interaction precision

There are two main factors that can negatively impact on the precision of interaction with VR elements, making the manipulation of 3D objects' parameters (like position, orientation, etc.) not accurate enough. The first factor concerns the precision of the mechanism adopted to track the hand controllers and the headset. The second aspect relates to the difficulties users are asked to deal with while operating in VR introduced by the characteristics of the visualization and interaction technologies used.

With respect to the first factor, assuming the adoption of the HTC Vive as the VR system, the literature already presents solutions (such as [35] and [226]) that experimentally evaluated the precision of the HTC Vive's tracking system. According to this previous work, different values of precision can be computed depending on the regions, i.e., in the center or on the boundaries of the tracked area, where the interactions take place. Moreover, precision can be influenced also by the size of the tracked area. The best conditions for the HTC Vive's tracking system reported a sub-millimetric precision [35]. Concerning the second factor, first of all, it is worth noticing that the Navigation state allows the user to dynamically modify the mapping between the real and the virtual reference systems, as described in the previous sections. For this reason, the evaluation needs to also consider the actual zoom level set to interact with the system, since a higher zoom could let the user, e.g., rotate/move more accurately an element, etc. Moreover, there is another aspect to consider, which is the fact that a higher zoom allows the users to visualize the VR environment in a clearer way, since the defocusing problem observed for the objects that are very close to the users' point of view becomes less relevant: hence, the user may perform the interaction in a better way too.

Tasks Based on the observations above, an experiment was designed to investigate the relation between the zoom level set in the VR environment and the corresponding system's precision. The design of the experiment considered the basic operations involved in the creation of an animation, i.e. positioning and rotating. Thus, two tasks were developed, referred to as *Positioning* and *Rotating* tasks. The goal of the first task was to move a given bone as close as possible to a reference bone, whereas the objective of the Rotating task was to apply a specific rotation to a given bone making it overlap as much as possible to a reference. With the aim of removing possible factors influencing the measurements, rotation (translation) of the controlled bone was disabled in the Posing (Rotating) task. To define the overall procedure, it was decided to adapt to the context of this experiment the approach already used in [35]. In particular, a sampling methodology was used asking a single user to perform each task with five different zoom levels. Therefore, the user had to repeat the positioning/rotating operation for each zoom level 500 times. Both these tasks were carried out requesting the user to manipulate bones in the center of the tracked space, where the precision of the tracking technology proved to be the highest possible for the considered VR system.

Evaluation criteria The evaluation criteria considered the Euclidean distance (for the Positioning task) and the angular distance (for the Rotating task) measured between the given bone and the reference. The estimation of system precision was then performed, like in [35], by calculating the mean and the standard deviation of the distances between the 500 samples collected for each zoom level and their centroid. Furthermore, by following the methodology in [226], the sample-to-sample jitter was analyzed by computing the root mean square (RMS) of the differences in the measured positions and orientations for each axis. This indicator was computed as

$$RMS_m = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}\Delta m_i^2} \tag{3.1}$$

where m is the considered measure (position or orientation, for a particular zoom level), n is the number of samples, and Δm_i is the difference between sample i and $i+1$.

User experience

In order to evaluate the user experience, a study was carried out with 23 participants (14 males and nine females), aged 21–34 ($M = 25.61$, $SD = 4.03$). Similar to the previous stage, participants were divided in two groups based on their previous experience with computer animation software, VR and related technologies. The first group was comprised of 13 participants who could be considered as NPUs,

since they were students attending a course on computer animation with Blender. The second group included the remaining 10 participants, who could be considered as PRUs; participants belonging to this group had expertise in teaching similar courses or because of working in the computer animation field. PRUs also reported that they were much familiar with the considered technologies, and most were accustomed to working in VR environments.

Tasks Each participant was requested to perform four different representative animation tasks using both the VRI and the BNI. Tasks were designed with the goal of testing the system with some of the key operations (from rigging to skinning, and posing) of a pipeline for armature deformation-based virtual character animation. In the first three tasks, the participants had to work with a cat character, which is freely downloadable[13]. The three tasks aimed to individually investigate the user experience for each of the three operations above, and they included 3D references which are generally not available to animators. The fourth task aimed to recreate more closely the real condition in which animators work while following a complete character animation pipeline. In this task, a human character[14] was used, and 3D references were replaced with 2D references that are commonly adopted by animators. The four tasks, illustrated in Fig. 3.14, were tested in a random order, balancing the number of users that were requested to start with the BNI and those with the VRI, in order to limit biases due to learning effects.The order of task execution was fixed, since it is the standard one adopted for the considered animation method. In the following, the four tasks are described in detail.

1. *Rigging task*: the objective of this task was to create an armature for articulating the cat character. By using the Rigging tool, participants were asked to create a new armature (displayed with bones having an octahedral shape) by following a close as possible the reference armature that is illustrated in Fig. 3.14a (with sticky bones).

2. *Skinning task*: this task asked participants to modify the weight of vertices assigned to a selected set of bones, in order to revise evident mistakes. Fig. 3.14b and Fig. 3.14c present an example of possible errors to be fixed. For instance in Fig. 3.14b the neck bone is mistakenly set to influence also vertices of the tail, whereas in Fig. 3.14c the left hind leg influences also the right hind leg. In order to complete this task, users had to use the functionalities provided by the Skinning tool. It is worth noticing that it was decided to provide participants with an initial set of colored weights to adjust, rather than starting to work from scratch, in order to make it possible the comparison of the

[13]Cat model: https://free3d.com/3d-model/low-poly-cat-46138.html

[14]Human model: https://free3d.com/3d-model/body-mesh-28679.html

users' performance during the skinning phase independently from the quality of the armatures created during the rigging step. Notwithstanding, more sophisticated skinning operations were investigated in the fourth task.

3. *Posing task*: in this task, participants were requested to articulate the armature of the rigged cat character. The armature is composed by 29 bones with 66 degrees of freedom, and it makes use of both forward and inverse kinematics to being controlled. In order to collect data that could be compared across users, users were provided with a 3D reference (in this case, a rigged model). The main goal of this task was to make the pose of the green cat in Fig. 3.14d overlap as much as possible with the pose assumed by the reference character (the red cat). Participants had to create two poses, respectively, at frame 0 and 20 (with the system configured in Keyframing mode). By interpolating the two poses an animation was actually generated, with intermediate frames automatically created by Blender.

4. *Complete pipeline task*: in this task, participants were asked to pass through the entire animation pipeline steps, by iterating the rigging, skinning and posing operations as needed. At first, users had to create the armature to be articulated in the remaining of the task by using the Rigging tool. Once the rigging was completed, participants had to use the Skinning tool to define the relations between the armature and the mesh. They were allowed to choose the preferred parenting method (with empty vertex groups or with the automatic assignment). Regardless of the method adopted, participants were forced to activate the Skinning tool in order to fix possible issues generated by the use of the automatic assignment, or to define the weights to be assigned to vertices in the case of empty vertex groups. Lastly, participants were requested to create an entire walking cycle animation by using the character they were working with till that moment. At any time, participants could return to a previous stage of the pipeline to correct possible errors (e.g., the lack of a bone in the armature or wrong weight assignments) in order to improve the quality of the generated animation. Differently than in the previous tasks, here the animation to be recreated was presented through a 2D video clip displayed in the virtual environment. This solution tried to mimic in VR the realistic usage conditions of the tool, since the use of a video as a reference is very common in the creation of animations with traditional software suites. It was chosen to not let the users free to create any animation, since in that case it would have not been possible to quantify performance in objective terms. In the video reference, there was no information either on the topology of the character's armature to be created or on the exact position and type of keyframes to be inserted. The above information should be inferred by changes in the video reference. Fig. 3.14e illustrates, on the left side, the human character to be animated and its condition at the beginning of

the task (i.e., it had a basic armature with three bones be extended/modified) and, on the right side, same of the frames captured from the video reference. The entire duration of the animation shown in the reference video was set to 60 frames.

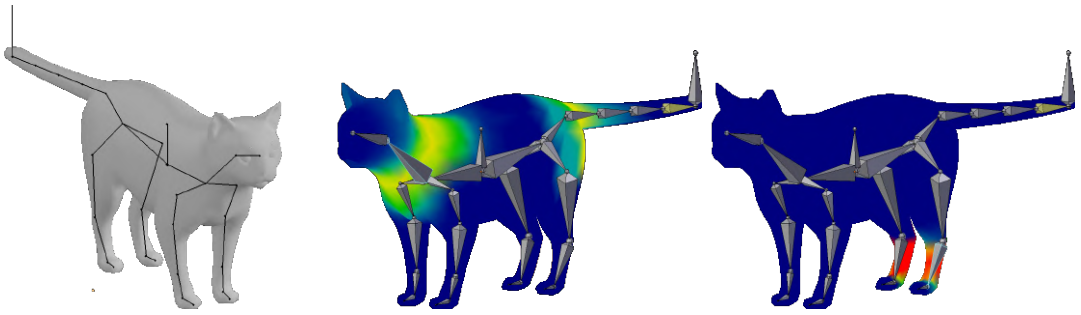
Before starting the experiments, participants were introduced to the functionalities of the BNI and the VRI that were needed to complete the assigned tasks. Participants were left free to familiarize with the two interfaces, by testing the same operations required in the task but on different scene elements. For participants without previous experience in VR, additional time was allocated in order to make them get accustomed with this technology. During the experiments, participants were allowed to ask for support, in order to possibly level the low experience in the use of the VRI with respect to the BNI. Similar to the first stage, participants were left free to decide when to consider the given task as completed, since it was chosen to set no time limit or quality threshold. Videos showing the execution of the tasks are available for download[15].

Evaluation criteria Users' performance was evaluated through both objective and subjective measurements, collected during and after the experiments, respectively. Two metrics, already used in previous work ([47, 171]) were considered for objective measurements, namely completion time (T) and animation accuracy (A). The first metric measures the time needed to complete the task and it could be computed for all four tasks. The animation accuracy estimates the difference between the reference and the result obtained by the users. In the fourth task, it was not possible to compare results across different participants since the number of bones in the created armature and the position of the keyframes inserted were arbitrary. Therefore, this metric was applied only to the first three tasks, where a "homogenous" reference was available. Due to the different goals of the tasks, it was also necessary to distinguish the definition of this metric for each task. In all these cases, the metric ranged from 0 to 100, where 100 represents a perfect result/match. For the Rigging task, the Euclidean distance between the bones of the reference and of the user-created armature (averaged on all the bones) was considered. For the Skinning task, this metric relied on weight distances, and was computed as

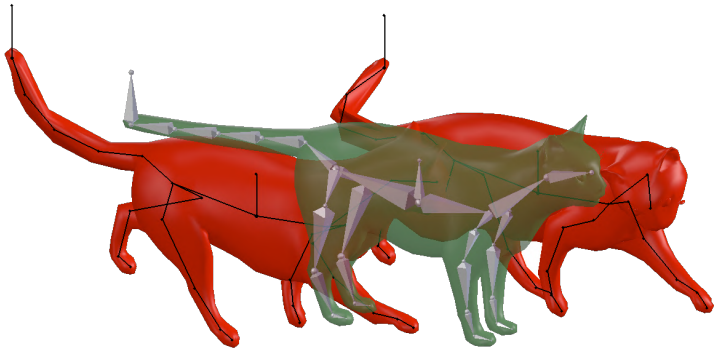
$$A = 1 - \frac{1}{\Omega} \cdot \sum_{i=0}^{N} \left(\sum_{j=0}^{M} weight_i(j) - \sum_{j=0}^{M} weight_i^*(j) \right) \tag{3.2}$$

where N and M indicate the number of vertex groups and vertices, respectively, whereas $weight_i(j)$ represents the weight assigned to the j-th vertex of the i-th

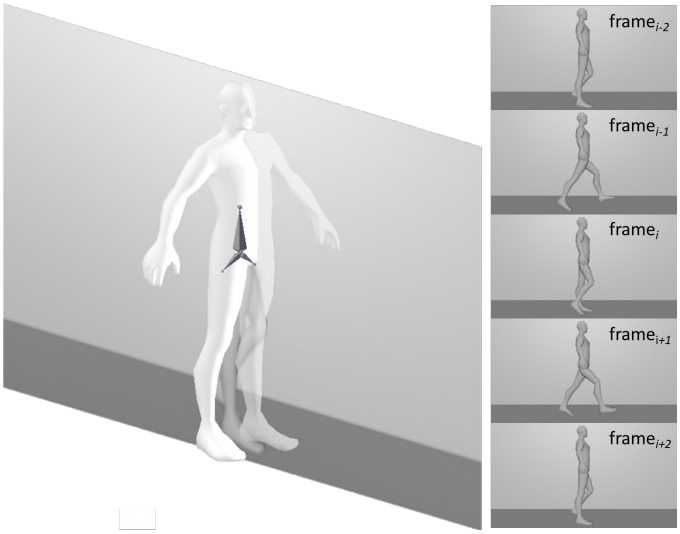
[15]Videos of the experiments: http://tiny.cc/1hxsaz



(a) Rigging task (b) Skinning task (c) Skinning task



(d) Posing task



(e) Complete pipeline task

Figure 3.14: Tasks considered in the experiments.

group. The * symbol is used to indicate reference weights (i.e., weights in the cat character with no mistake in the vertex groups), and Ω is a normalization factor computed as

$$\Omega = \sum_{i=0}^{N} \left(\sum_{j=0}^{M} weight_i^0(j) - \sum_{j=0}^{M} weight_i^*(j) \right) \tag{3.3}$$

where $weight_i^0(j)$ are the weights assigned to the j-th vertex of the i-th group at the beginning of the task.

For the Posing task, the animation accuracy was computed as an adaptation of the metric used for the Rigging task as:

$$A = 1 - \frac{1}{2} \sum_{i=0}^{N} \left(\frac{\delta_i}{\Delta} \right) \tag{3.4}$$

where N is the number of bones constituting the armature, δ_i is the Euclidean distance between the center of the bones in the reference and the user-controlled armatures, and Δ is a normalization factor calculated similarly to δ_i, but considering the controlled armature set in the rest pose (for frame 0) and in the pose at frame 0 (for frame 20).

In addition to the metrics described above a third metric, named *animation precision* (P), was considered in order to quantify the precision of the produced animation in numerical terms. Differently than in the case of interaction precision, here the metric also takes into account the possible impact on the user experience of the given animation functionalities. Similar to the second metric, it can be applied only when a "homogenous" reference is available, and different definitions are needed for each task. In particular, for the Rigging and Posing tasks, the metric was calculated by first determining a centroid for the coordinates of a specific bone as set by different users. Afterward, the distances between each bone and its centroid were estimated and averaged among the various users. For the sake of completeness, also standard deviation was considered. For the Skinning task, the computation included the following steps. First, the sum of the weights assigned to all the vertices in a given vertex group was computed for each group. Then, a centroid was computed by averaging the sums among the users. Finally, the mean, as well as the standard deviation of the distances between the sums and the centroid, were measured for each group.

Subjective measurements were collected through a post-test questionnaire completed in by the participants at the end of the experiment. The questionnaire (which is available for download[16]) included three sections. The first section was aimed to evaluate the overall usability of the two interfaces based on the SUS [40]. The second section included questions from a previous work [243], aimed to measure the users' satisfaction. Lastly, in the last section, participants were requested to express their preferences for the two interfaces considering each of the four tasks

[16]Questionnaire: http://tiny.cc/hf2saz

separately. Since the last task was composed by all the stages of an animation pipeline, preferences assigned to it could be regarded as an indication of overall users' preference.

3.2.7 Results: Second stage

Here, results aimed to compare the performance of BNI and VRI will be presented by first focusing on the precision of the VR-based interaction, then tackling user experience.

Interaction precision

Tables 3.2 and 3.3 report results concerning interaction precision. As expected, the more the zoom level is increased, the more distances of sampled values from the corresponding centroid decrease. The Pearson's correlation coefficient (ρ) was used to analyze the correlation between the zoom level and the values of the metrics. A visual analysis of the plotted data suggested to apply a transformation to the zoom level variable in order to obtain a liner scale. With this aim, it was chosen to apply a logarithmic function that remaps values (0.25, 0.5, 1, 2, 4) to (-0.6, -0.3, 0.0, 0.3, 0.6). Correlation coefficients obtained, confirmed a high inverse correlation between the zoom level and all the considered metrics.

Table 3.2: Interaction precision: mean values and standard deviations of the Euclidean and angular distances between the samples and the corresponding centroid for the considered measures.

Zoom level	**Location (*m*)**		**Rotation (*deg*)**	
	M	SD	M	SD
0.25×	0.0069	0.0038	1.0133	0.5730
0.50×	0.0029	0.0013	0.6739	0.3566
1.00×	0.0017	0.0016	0.5828	0.4008
2.00×	0.0010	0.0005	0.2653	0.1720
4.00×	0.0006	0.0003	0.2257	0.1622
Pearson's corr. coeff. ρ	-0.9034	-0.8855	-0.9711	-0.9269

User experience

Results concerning the user experience are presented by first focusing on objective results, then tackling subjective results.

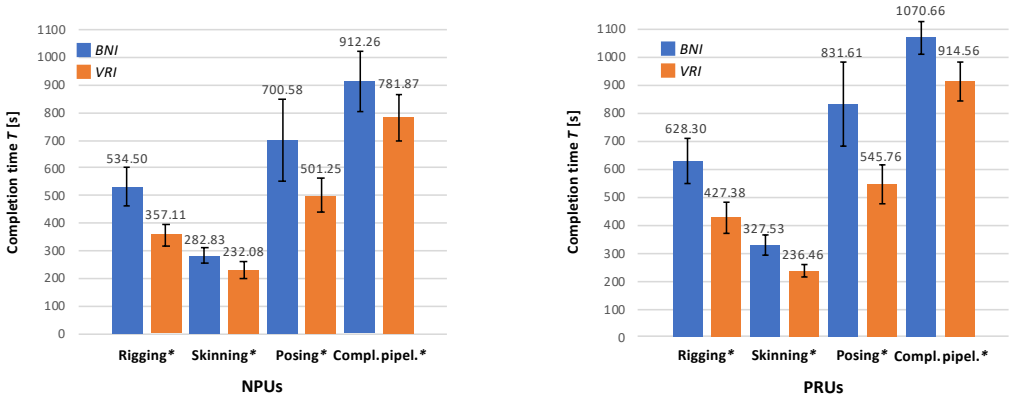
Table 3.3: Interaction precision: RMS values for the considered measures.

Zoom level	**Location (*m*)**		
	RMS(x)	*RMS(y)*	*RMS(z)*
0.25×	0.0006	0.0007	0.0007
0.50×	0.0002	0.0002	0.0003
1.00×	0.0002	0.0011	0.0002
2.00×	0.0001	0.0001	0.0001
4.00×	0.0001	0.0001	0.001
Pearson's corr. coeff. ρ	-0.8712	-0.8716	-0.8890
Zoom level	**Rotation (*deg*)**		
	RMS(x)	*RMS(y)*	*RMS(z)*
0.25×	0.0552	0.1444	0.0532
0.50×	0.0302	0.0894	0.0342
1.00×	0.0220	0.0746	0.0229
2.00×	0.0105	0.0392	0.0092
4.00×	0.0073	0.0350	0.0067
Pearson's corr. coeff. ρ	-0.9524	-0.9574	-0.9736

Objective results Results in terms of completion time, animation accuracy and animation precision for both NPUs and PRUs are shown in Fig. 3.15, Fig. 3.16, and Fig. 3.17, respectively. Statistically significant results (obtained by applying paired Student's t-tests, with $p < 0.05$) are highlighted with the * symbol.

Focusing first on results collected for NPUs, it can be noticed that the VRI outperformed the BNI in terms of completion time for all tasks. In particular, the gain in terms of time saved was 33% for the Rigging task ($p < 0.01$), 18% for the Skinning task ($p < 0.01$), 34% for the Posing task ($p < 0.01$), and 14% for the Complete pipeline task ($p < 0.01$). Concerning accuracy, results were statistically significant only for the Skinning task, in which users were more accurate (47%) with the VRI than the BNI ($p < 0.01$). As mentioned, data for the Complete pipeline task are not available, since a homogeneous reference was missing. Regarding animation precision, statistically significant differences were found only for the Posing task, in which the VRI allowed users to be more precise (94%) than the BNI ($p < 0.01$). The high variances measured for this task are strictly related to those already observed in Fig. 3.16a and they can be explained by the fact that, when working in VR, users found it easier to ensure that bones were properly aligned with the reference along all the axes.

With respect to the results obtained by the PRUs, it can be observed that trends for completion time and animation accuracy remained comparable to those



(a) (b)

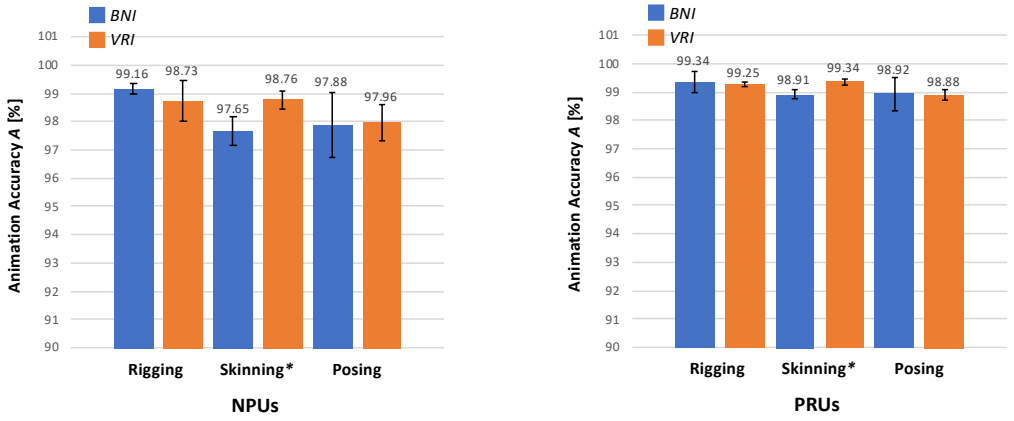
Figure 3.15: Objective results in terms of completion time; mean values are represented by bar heights, whereas standard deviations are shown using error bars.

observed for NPUs. With the VRI, PRUs were 32% faster in the Rigging task ($p < 0.01$), 28% in the Skinning task ($p < 0.01$), 34% in the Posing task ($p = 0.01$), and 15% in the Complete pipeline task ($p < 0.01$). Regarding animation accuracy, like for NPUs, statistically significant differences were found only for the Skinning task, for which the improvement with the VRI was equal to 39% with respect to the BNI ($p < 0.01$). It is worth observing that this result is particularly interesting, because it was achieved despite the fact that PRUs had previous experience with Blender and they never worked with the VRI. Reasons appeared to be related to the higher learnability and usability of the VRI, which were investigated more in detail through the subjective measurements. Differences concerning animation precision were not significant from a statistical point of view.

Comparing results achieved by the two user categories, conveniently reported in Table 3.4, differences were statistically significant (with unpaired Student's t-tests, $p < 0.05$) only for the animation accuracy in the Skinning and Posing task, though values seemed to suggest that PRUs were faster and more accurate than NPUs with both the interfaces. With respect to the animation precision, PRUs were found to be more precise than NPUs (the average distances from the centroids were lower) in the Rigging task using the VRI (0.0009 m, $p = 0.04$), and in the Posing task using both BNI (0.1533 m, $p < 0.01$) and VRI (0.0024 m, $p < 0.01$).

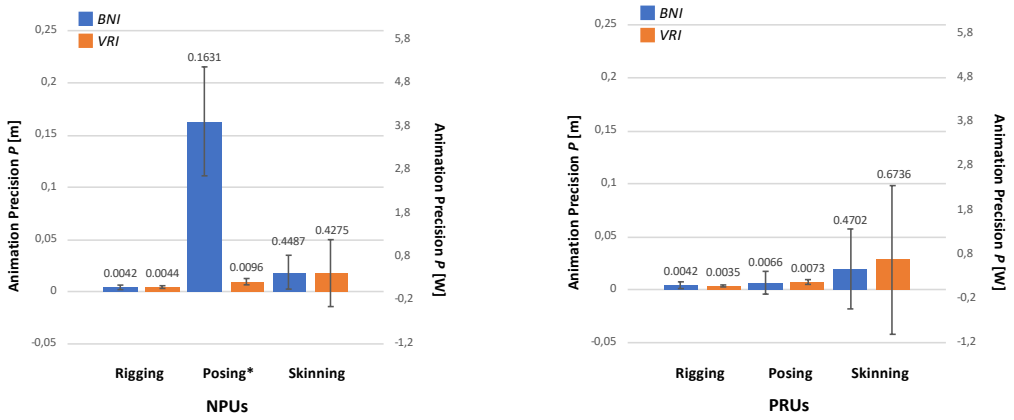
Finally, based on results reported in Table 3.5, it is observed that NPUs using the VRI were faster than PRUs using the BNI in both the Skinning ($p = 0.04$) and the Posing ($p = 0.04$) tasks. Concerning animation precision, statistically significant differences were found only for the Posing task, where PRUs were more precise than NPUs (0.0030 m, $p = 0.01$).

The above outcomes suggest that the VRI could be a powerful tool to help in leveling computer animation skills among users with different levels of expertise.



(a) (b)

Figure 3.16: Objective results in terms of animation accuracy; mean values are represented by bar heights, whereas standard deviations are shown using error bars.



(a) (b)

Figure 3.17: Objective results in terms of animation precision; mean values are represented by bar heights, whereas standard deviations are shown using error bars.

Subjective results As previously mentioned, the first section of the questionnaire aimed to evaluate the usability of the system based on the SUS scale [40]. Questions were expressed in the form of statements to be evaluated on a 1-to-5 scale (from strong disagreement to strong agreement). Evaluating and examining the SUS scores, it can be observed that both NPUs and PRUs perceived the VRI as characterized by a higher usability than the BNI (72.03 vs 45.57 for NPUs, 81.00 vs 67.20 for the PRUs). Both the user categories rated the VRI as "acceptable", whereas the BNI fall in the "not acceptable" range for NPUs and in the "marginally

Table 3.4: A comparison of objective results for the NPUs and the PRUs (when using the same interface). Times, percentages and distances are shown for the three metrics.

Task	$\mathbf{T_{NPUs} - T_{PRUs}}$		$\mathbf{A_{NPUs} - A_{PRUs}}$		$\mathbf{P_{NPUs} - P_{PRUs}}$	
	BNI	*VRI*	*BNI*	*VRI*	*BNI*	*VRI*
Rigging	93.81 s p=0.21	70.30 s p=0.10	–0.18% p=0.45	–0.51% p=0.23	–0.0001 m p=0.97	**0.0009 m** p=0.04*
Skinning	44.70 s p=0.19	4.37 s p=0.85	**–1.26%** p<0.01*	**–0.58%** p=0.01*	–0.0215 p=0.83	–0.2461 p=0.31
Posing	131.03 s p=0.27	44.52 s p=0.35	–1.04% p=0.13	**–0.91%** p<0.01	**0.1566 m** p<0.01*	**0.0024 m** p<0.01*
Compl. pipel.	158.39 s p=0.06	132.69 s p=0.08	Not available	Not available	Not available	Not available

Table 3.5: A comparison of objective results for the NPUs using the VRI and the PRUs using the BNI. Times, percentages and distances are shown for the three metrics.

Task	$\mathbf{T_{NPUs}(VRI)}$ – $\mathbf{T_{PRUs}(BNI)}$	$\mathbf{A_{NPUs}(VRI)}$ – $\mathbf{A_{PRUs}(BNI)}$	$\mathbf{P_{NPUs}(VRI)}$ – $\mathbf{P_{PRUs}(BNI)}$
Rigging	–107.12 s p=0.07	–0.61% p=0.22	0.0002 m p=0.83
Skinning	**–46.37 s** p=0.04*	–0.15% p=0.42	–0.0427 p=0.84
Posing	**–154.81 s** p=0.04*	–0.95% p=0.09	**0.0030 m** p=0.01*
Compl. pipel.	2.29 s p=0.10	Not available	Not available

acceptable" range for PRUs. The difference between the SUS scores assigned by the two user categories to the VRI was probably due to the greater confidence of PRUs with the VR technologies, whereas their greater experience with Blender made them perceive the BNI as more usable than the NPUs. Table 3.6 reports scores assigned to individual statements, and bold fonts indicates that an interface was evaluated more positively than the other. Focusing on the statistically significant results (marked with the * symbol), it can be noticed that both the user categories perceived the VRI as easier to learn than the BNI. Both PRUs and NPUs also found the VRI as characterized by a more appropriate level of complexity than the BNI, making the VRI an interface easier to use. Moreover, NPUs stated they would like to use the VRI more frequently than the BNI and they also perceived

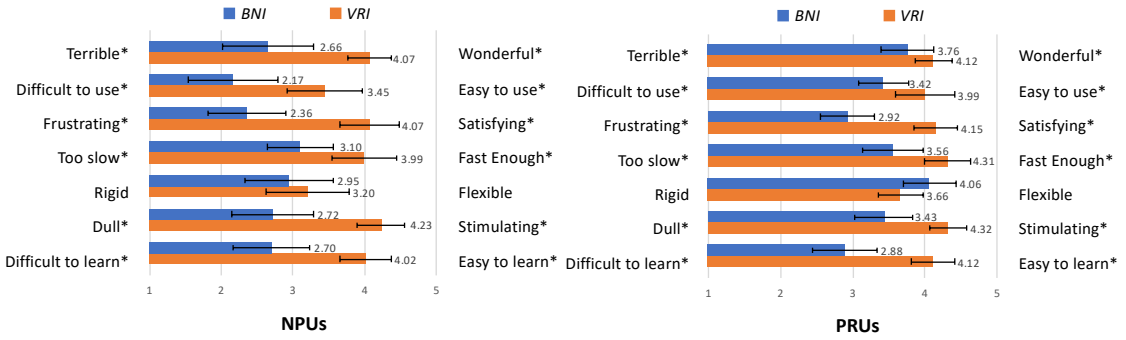
the animation system as less cumbersome when using the VRI than with the BNI.

Table 3.6: Subjective results concerning usability based on SUS statements [40].

Statement	**NPUs**			**PRUs**		
	BNI	*VRI*	*p*	*BNI*	*VRI*	*p*
1) I think that I would like to use this system frequently	3.22	**4.24**	<0.01*	4.13	4.44	0.67
2) I found the system unnecessarily complex	3.04	**2.16**	0.02*	2.43	**1.79**	0.04*
3) I thought the system was easy to use	3.01	**4.21**	<0.01*	3.15	**4.23**	0.01*
4) I think that I would need the support of a technical person to be able to use this system	2.36	2.69	0.41	2.03	1.75	0.09
5) I found the various functions in this system were well integrated	3.16	3.62	0.34	4.56	4.47	0.64
6) I thought there was too much inconsistency in this system	1.92	1.46	0.08	1.42	1.26	0.26
7) I would imagine that most people would learn to use this system very quickly	1.55	**4.23**	<0.01*	2.78	**4.12**	<0.01*
8) I found the system very cumbersome to use	3.26	**1.87**	<0.01*	2.20	2.01	0.2451
9) I felt very confident using the system	2.19	**3.45**	0.03*	3.99	4.02	0.90
10) I needed to learn a lot of things before I could get going with this system	4.32	**2.76**	<0.01*	3.65	**2.07**	<0.01*

The second section of the questionnaire asked participants to evaluate several aspects concerning their satisfaction, by rating a set of items proposed in a previous work ([243]). The original 0-to-10 scale was adapted to a 1-to-5 scale, to maintain coherency among the various sections of the questionnaire. Average values are reported in Fig. 3.18. Results confirmed findings from the first section of the questionnaire.

Focusing on statistically significant results (marked with the * symbol), the VRI was found as easier to use ($p = 0.02$ for NPUs and $p = 0.02$ for PRUs) and to learn ($p < 0.01$ for NPUs and $p < 0.01$ for PRUs). These results indicate that VRI



(a) (b)

Figure 3.18: Subjective results concerning satisfaction based on criteria (items) in [243].

could be used by users with different levels of expertise in the field of computer animation and VR. Regarding the remaining aspects considered in this section, both NPUs and PRUs expressed a higher appreciation for the VRI than for the BNI. In fact, the scores of VRI were closer to the "wonderful" ($p < 0.01$ for NPUs and $p < 0.01$ for PRUs), satisfying ($p < 0.01$ for NPUs and $p < 0.01$ for PRUs) and stimulating ($p < 0.01$ for NPUs and $p < 0.01$ for PRUs) items. Moreover, statistically significant differences were obtained in terms of perceived operation speed ($p < 0.01$ for NPUs and $p = 0.01$ for PRUs) with higher scores assigned to the VRI, as already observed for objective measurements. No significant differences were found with respect to flexibility.

Finally, the last section of the questionnaire asked users to express their preference between the two interfaces for the execution of each task. Percentage means are reported in Table 3.7. Both NPUs and PRUs preferred the VRI to the BNI for all the four tasks. Analyzing each task in detail, it can be observed that the Skinning task was the one which seemed to benefit more from the introduction of VR for both NPUs and PRUs (reasonably because of the possibility for the users to observe/manipulate in 3D the weights, without the need to continually change the point of view). For the remaining tasks, considering first NPUs, the percentage of participants who preferred the VRI with respect to the BNI was higher in the Rigging task than in the Posing and the Complete pipeline tasks. Regarding PRUs, this order was inverted. A possible reason for this change in the users' preferences was related to the fact that users that were not accustomed to VR perceived a higher physical effort in the tasks that required them to wear the VR headset for a long period (the Posing and Complete pipeline tasks are the longest ones). This aspect could have influenced the number of preferences for the VRI.

The percentage of participants who preferred the VRI in the Complete pipeline task was always the lowest one. In fact, based on comments collected at the end of the experiments for this task, it was observed that users (especially those with

limited experience with animation techniques) faced a number difficulties while working with a 2D video reference in a 3D environment.

Table 3.7: Preferences expressed by the users for the four tasks (percentages).

Task	**NPUs**		**PRUs**	
	BNI	*VRI*	*BNI*	*VRI*
Rigging	21.74%	**78.26%**	30.43%	**69.57%**
Skinning	4.35%	**95.65%**	17.39%	**82.61%**
Posing	26.09%	**73.91%**	21.74%	**78.26%**
Compl. pipel.	34.78%	**65.22%**	21.74%	**78.26%**

3.2.8 Future developments

From comments collected at the end of the experiments carried out in [171] and [51], it was possible to obtain cues for future developments.

Participants reported the lack of mechanisms in VR letting them quickly enable features that, in Blender, are generally activated via keyboard shortcuts (e.g., to quickly change among predefined views, to fine-tune or discretely adjust parameters' value, etc.). Several participants proposed to use ray casting for selecting objects and to replace panel-based approaches presently used in the tool with voice-/gesture-based commands. Participants also asked for more effective ways to provide clearer interaction feedback, e.g., on the system's conditions that allowed/not allowed to activate specific functions or when particular operations are needed (in [171] and [51], changes' in virtual controllers' textures are adopted). Other participants, especially NPUs, would prefer 3D icons instead of 2D textures on the Trackpad buttons, in order to immediately recognize the functionality currently activated and partially cope with the limited resolution of images showed by the HMD. Finally, focusing on outcomes of [51], participants proposed to manage also the modeling steps in VR, in order to take advantage of NUIs and immersion while working with Blender's tools for creating meshes that are natively 3D (e.g., the Blender's sculpting mode). This comment poses the attention on the importance to integrate VR approaches into a full animation pipeline, considering both modeling and animation. The possibility to configure new functionalities (which do not necessarily pertain animation) was partially taken into account in [51] through the introduction of the customization features. However, future efforts could be devoted to simplify the configuration process and improve custom VR-based representations (e.g., letting the user choose the textures to be shown on the controllers' buttons, group functions and controls into visual containers, etc.). In this respect, a dedicated user study could be designed to estimate the advantages possibly brought by customization in terms of mental load and learning cost.

3.3 A virtual character animation system based on a reconfigurable TUI and immersive VR

As previously described, the use of VR and related technologies offer animators opportunities and benefits with respect to improved visual and spatial awareness of the environment and available interaction methods [93]. However, with respect to the user input, VR-based animation systems risk falling short. Although several technologies able to track users' hands and body in VR are already available, the de facto standard for user interaction in many commercial VR systems is represented by the hand controllers [117]. It is worth observing that, if on the one side, a controller can offer significant advantages to the interaction being a 3D device, on the other side it still shares a number of characteristics with traditional input means. For instance, it is characterized by a high difference perceived between the shape and behavior of the input device and those of manipulated elements (the armature's bones). This aspect may lead animators to loose some of the benefits offered by VR due to a possibly decreased sense of immersion [117].

By considering the above observations, an animation system was presented in [47], combining the advantages granted by reconfigurable TUIs and by VR technology. In particular, the proposed system allows animators to manipulate a rigged virtual character by articulating an instrumented prop into an immersive virtual environment. The system is integrated within the well-know, open-source animation suite, namely Blender. The work builds upon the animation system presented in [173]. Such a system, already described in Chapter 2, lets the users manipulate a virtual character in real-time by using their body, voice, and a tangible interface. Due to the fact that the system supported only traditional output means, results of the produced animation could be observed only on a large screen or a projected display (hence, in the following it will be referred as the Projected System, or PS).

With respect to [173], the VR-based system illustrated in this section (later referred to as VRS) allows the users to generate animations while they are immersed in the virtual environment, thus letting them be more aware of the actual character's pose being manipulated. The activation of configurations to be animated, which in [173] was based on voice commands, is replaced by two different approaches based on gaze and proximity selection, which were expected to ensure higher usability. Finally, the system proposes new visual feedbacks designed to improve the users' understanding of the mapping between the interface elements and the controlled parts of the character. A user study was carried out in [47] to asses the effectiveness of the devised system, by asking participants to perform a character posing task and collecting both objective and subjective measurements.

3.3.1 Related work

The possibility to animate characters in VR is attracting greater attention every day. This is confirmed by the increasing number of commercial animation tools based on such technology that are already available at the consumer level developed for all the main VR platforms such as the HTC Vive and Oculus Rift. Focusing on tools targeted to character animation, a first example is offered by VIRTUAnimator[17], an application that lets users animate humanoid characters in a VR environment using hand controllers. Resulting animations need to be exported to other graphics tools in order to apply changes. Other examples are Tvori[18] and Mindshow[19], which offer the possibility to configure a virtual scene by filling it with simple shapes, characters, props and visual effects selected from a predefined library and to animate them. The Allright Rig[20] is a software library able to automatically reconstruct the rig of a humanoid character to be used in Unreal Engine. A plug-in allows the users to control the posing of the character in VR. Similarly, another tool supporting several VR systems (i.e., Valve Index, HTC Vive, Oculus Rift and Windows Mixed Reality) is called Merper VR[21]. It allows animators to articulate 3D models containing skeletal structure and skinning information into an immersive environment through the hand controllers. Animations can be generated by defining keyframes. Another plug-in developed for Unreal Engine, named Marionette VR[22], lets the users control a virtual marionette character in a VR environment. By using the Oculus Rift's controllers, animators can view their hands reproduced in a physics-enabled interactive space and interact with the character through marionette's ropes or applying forces to its geometry.

Although many of the tools above are characterized by several features that make them intuitive also for non-skilled users, they could present an important limitation regarding flexibility. In fact, the reduced set of characters that can be animated, as well as the number of operations allowed in these systems, could make them unsuitable for general-purpose animation scenarios. Furthermore, all these tools are available as either standalone applications or plug-ins to be installed in game engines like Unreal Engine or Unity. The lack of a direct integration with common animation suites like Blender, Autodesk Maya, etc. implies that users, who intended to reuse/manipulate their animations, need to perform additional operations taking take place in a separate step of the animation workflow. This

[17] VIRTUAnimator: http://store.steampowered.com/app/459870/VIRTUAnimator/

[18] Tvori: https://store.steampowered.com/app/517170/Tvori/

[19] Mindshow: https://store.steampowered.com/app/382000/Mindshow/

[20] Allright Rig: http://alexallright.com/allrightrig

[21] MeperVR: https://store.steampowered.com/app/725510/Merper_VR/

[22] Marionette VR: https://github.com/pushmatrix/marionettevr

fact could make the whole process more tricky and laborious.

As said in Section 3.2.1, an example of an integrated VR-based animation tool is MARUI. In this tool, the creation of animations represents just an example of the broad set of functionalities that can be managed in VR. However, interaction is still based on conventional 2D menus, windows and 3D widgets, manipulated through hand controllers.

VR- and AR-based systems for virtual character animation have been also studied by the research community. For example, a plug-in developed for Unity to create animations through the performance-driven approach in VR is presented in [312]. Similar to various commercial tools, in this system, users can record the movements of the hand controllers (tracked by the VR system) in order to move and animate characters and other objects into the immersive virtual environment by using inverse kinematics and performance animation. The functionalities of the system, such as selecting objects, or controlling the timeline, are enabled via controller-activated menus. Moreover, the system lets the animators create layered animations and use a virtual camera to generate effective shoots. No import/export functionalities are provided. A user study was carried out by involving only expert users. Obtained results confirm the possibility to reduce the time required for creating animations even though precision (in broad terms) resulted to be lower than with traditional animation suites.

It is worth observing that all the solutions summarized in this section leverages the hand controllers. Limited prior work considers/assesses the possible advantages (e.g., the improved connection between the real and the virtual worlds) brought by the use of non-traditional interfaces like, e.g., TUIs, within VR-/AR- environments. For example, in [93] and [243], the common VR hand controllers have been replaced by a cube-shaped TUI. AR markers attached to the faces of the cube handled by the user allow him or her to specify the animation to activate (from a pre-defined set) and configure its speed. The character's position can be managed by moving the cube in front of the camera which is framing the AR marker. The user can observe in real-time the produced animation in AR through a tablet device. One of the limitations of this approach is the reduced control possibilities offered to the user (limited by the adoption of the cube and its faces as input). Although the flexibility of the system could be improved by taking into account also gestures that rely on cube interactions (like cube shaking), the use of pre-defined animations could still represent a strict limitation. Moreover, as in most of the previous work, editing of the recorded animation is not allowed within the virtual environment. The authors of [117] proposed a custom-designed unit equipped with sensors, that can be assembled with low-cost off-the-shelf hardware. The unit can be mounted on common physical objects to track them in VR. Four prototypes (a simple cube, a stuffed animal, a treasure chest, and a wooden boat) were proposed and used in VR narrative animation scenarios to confirm the possibility of improving user experience by leveraging the active and passive feedback provided by the handheld

objects. Although positive results were obtained, this work totally disregarded character articulation, due to the type of sensors used in the TUI.

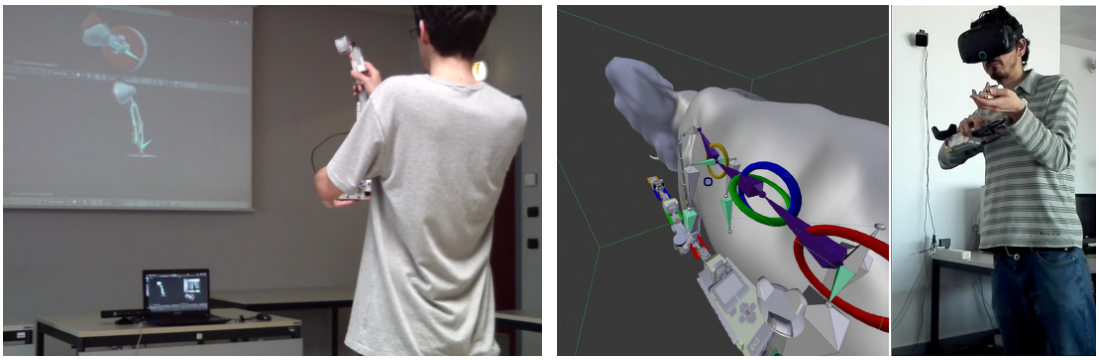
By moving from these considerations, an animation system was presented in [47] that combines both the advantages of TUI and VR technologies. As said, the proposed solution builds upon the PS developed in [173]. The novelty introduced in the former work involves the use of VR to allow users to be immersed in the animated environment, in order to improve the awareness of the mapping between physical and virtual elements and provide immediate feedback on the effects of 3D manipulations on the virtual character's pose. The different approach used by the two systems for scene visualization is illustrated in Fig. 3.19, which shows a user posing a virtual character. In the PS (Fig. 3.19a), user can observe the results on a multi-view wall projection, whereas in the VRS (Fig. 3.19b), the user wears a head-mounted display and operates in an immersive environment that includes the character to be animated and a virtual reconstruction of the TUI used to articulate it. Differently than in other TUI- and VR-based solutions mentioned in the above review, this system relies on a reconfigurable device, which proved to be particularly effective to decrease the time needed for character posing [173], Moreover, the TUI is expected to ensure a higher flexibility compared to custom props or common hand controllers.

3.3.2 Proposed system

As previously mentioned, the VR-based animation system that was originally presented in [47] builds upon a previous work reported in [173]. The high-level architecture of the system is represented in Fig. 3.20, where blocks and links modified/added to support the new features have been highlighted in green. In the following, more details on the individual blocks as well on introduced functionalities will be provided.

Input device

The block, named *Input Devices*, includes the user interfaces that can be used to gather information for controlling the position and orientation of bones belonging to the virtual character's armature and for managing the functionalities of the *Animation Software*. Similar to [173], several sensing technologies are supported, including both positional and rotational sensors, color and depth cameras, microphones, etc. In particular, the *Tangible Interface* is based on servo motors, sensors and bricks included in the Core and Expansion sets of the Lego Mindstorms EV3's Education Kit. As described in [173] the Lego Mindstorms' Intelligent Brick components is in charge of collecting data gathered by the servo motors and the sensors. Collected data are then sent to the *Interaction Agent* as JSON strings over a Wi-Fi/USB connection thanks to a third-party API. The *Body Tracking Interface* relies



(a) Projected System (b) VR-based system

Figure 3.19: TUI-based character animation.

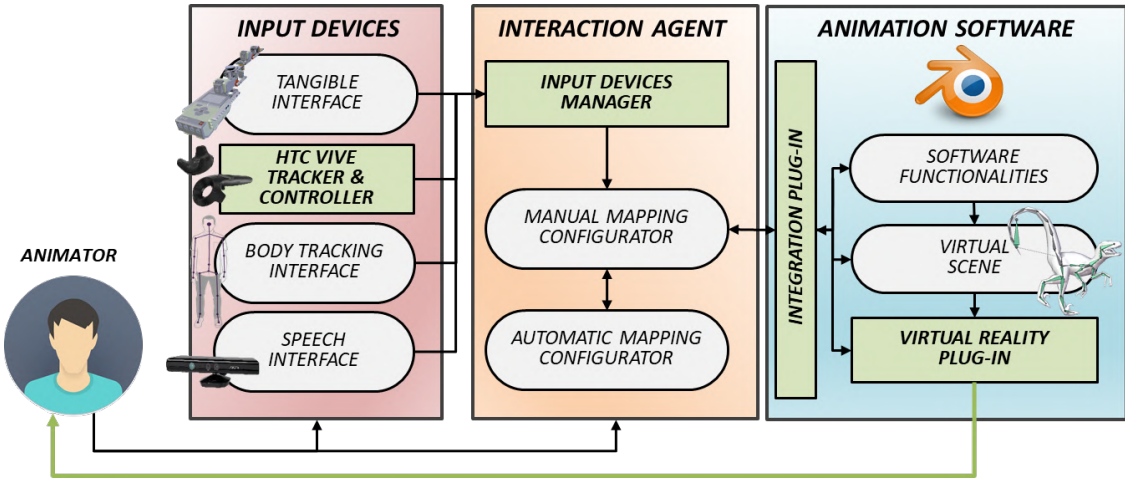


Figure 3.20: Architecture of the proposed animation system [47].

on the real-time skeleton data provided by the Microsoft Kinect device. The *Speech Interface* was implemented through the Microsoft Speech Platform library and it is responsible for recognizing voice commands used to enable animation functionalities.

With respect to the reference work, the proposed system includes a *VR Interface* based on the HTC Vive suite. This interface can be used to collect further position and orientation data derived from the elements tracked by the VR suite, i.e., the controllers and the trackers. In the work, a tracker was used, mounted on the tangible prop for tracking its position and orientation in the 3D environment. Moreover, the positional and orientation information related to the headset was leveraged to determine the user's point of view.

Interaction agent

Measurements provided by the Input Devices are managed by the Interaction Agent component through the *Input Devices Manager*. This component is in charge of converting received raw data into information that can be used for character posing. In [173] two approaches were proposed, which allow the user to map the available interface elements (i.e., servo motors, sensors, body joints, etc.) into the virtual character's DOFs. The first approach is based on a manual setup made by the user through a dedicated graphics tool named *Manual Mapping Configurator*. The second approach allows the user to take advantage of mapping rules automatically created by a so-called *Automatic Mapping Configurator*. In particular, an unsupervised algorithm optimizes the cost of assignment each individual interface element to a specific DOF of the character's armature, by taking into account several factors, like the similarity between the topology of the virtual character and the assembled interface, the correspondence between the DOFs (and related ranges) of the character's bones and interface elements, the presence of symmetries in the armature's topology, etc. The configuration automatically produced by the system may be refined by using the manual configurator in order to account for the requirements of different users. When the number of DOFs in the bones of the character's armature is higher than the interface elements available, the armature is subdivided into several partitions. This subdivision allows the animator to manage the subset of bones in a given partition with the same configuration used for the remaining of the armature without the need to reassemble it. To activate the control of a specific partition, the user can pronounce the corresponding voice command. Mapping rules generated both manually and automatically can be used to create and visualize the step-by-step instructions for assembling the Lego Mindstorms bricks that constitute the tangible prop. The Interaction Agent is in charge also of managing the vocabulary that is used to translate the recognized voice commands into *Software Functionalities*.

Animation software

The Animation Software handles the information regarding the scene where the animation take place (e.g., the position/orientation of 3D objects in the scene, armatures topologies, and their deformations, etc.). Moreover, it also provides a number of Software Functionalities, e.g., for selecting character's bones to be manipulated, inserting/ deleting/copying/pasting keyframes, navigating the timeline, visualizing the animation, enabling/disabling continuous keyframing (for performance-driven animation), selecting the view, etc. Like in [173], this component relies on the Blender open-source 3D modeling and animation suite. The Integration Plug-in, developed for a specific software, lets the Interaction Agent interact with selected Animation software (i.e., Blender).

In contrast to the previous work, this component includes a further block named

Virtual Reality Plug-in. This plug-in, developed as a Python script for Blender, allows the animators to visualize the Blender's 3D viewport through the HTC Vive headset, and offers them an actionable representation of the articulated tangible prop into the virtual environment. The use of VR enables the introduction of new features designed to ease the interactions with the system, which will be discussed in the following. The compatibility with the previous animation system, where the visualization modality is based on large-screen or wall projection, is guaranteed by changes in the developed Interaction Plug-in, which allow the users to easily switch between the VR-based interaction solution and the previous one.

3.3.3 VR-based features

Before describing the new features introduced in [47] in detail, the drawbacks of the previous work will be first summarized and then solutions developed to cope with them will be discussed.

Drawbacks of the reference system

The results reported in [173] demonstrated that, compared to traditional interfaces, the use of reconfigurable TUIs allows the users to animate virtual characters in a way that is perceived more intuitive and accessible to both subjects with and without skills in this field. However, although results demonstrated that completion time could be reduced by using this interface, it also showed that the accuracy of the final pose was lower compared to that achieved when working with M&K.

According to subjective measurements and empirical observations, these results could be related to multiple factors. First, animators were not allowed to precisely specify how to observe the virtual scene, since the system provided only a predefined set of views that could be either shown together in a split viewport or activated individually using voice commands. This aspect sets the attention on a relevant issue: although the reference system supported 3D input means, the visualization of the Blender's interface was still based on a 2D output (monitor or wall projection). This factor could influence the understanding of the actual virtual character's pose. Furthermore, the intuitiveness and effectiveness of the adopted solution could be reduced by the lack of a visual feedback representing the mapping between the interface elements being handled and the corresponding articulated DOFs. In [173], the only solution available to understand the effect that a manipulation of the interface element had on the character's armature was represented by a trial-and-error process, consisting in articulating the prop and looking at modifications in the armature. Another limitation was represented by the process needed for selecting the armature's partition to be controlled. Although in [173] a method was proposed based on voice commands, which let the users quickly change the current selection, it could be hard to use such a method when the number

of the partitions' names to remember is very high. Moreover, when a partition that had been already articulated in the intended pose is unintentionally activated, changes in the TUI would introduce unwanted modifications to the character's armature, making the system cumbersome to use.

The system proposed in [47] relied a shift to a VR-based interaction, since the use of VR was expected to provide the animator with an enhanced perception of the 3D space related to the new possibility of freely moving in the virtual environment in order to choose the best observation point to perform a given manipulation. Moreover, the use of a VR-based visualization made it possible to introduce other specific features to cope with the limitations of the system described in [173], which pertain the lack of manipulation feedback and intuitive methods for selecting armature's partitions. New features are discussed in the following.

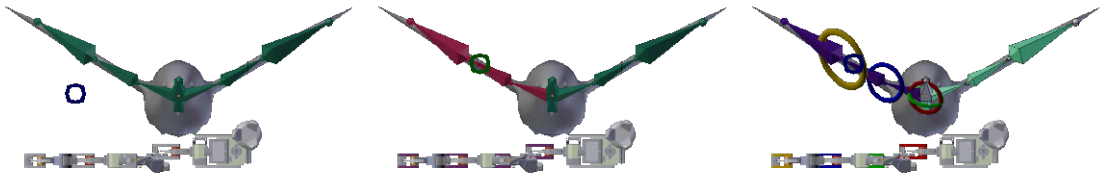
Visual cues

In the devised system, the user is allowed to articulate virtual characters while immersed in a VR environment. This characteristic is also leveraged to display the position and orientation of the tangible interface (tracked with the attached Vive tracker), and the configuration of its elements automatically reconstructed through the data continuously transmitted by the servo motors and sensors. The virtual reconstruction of the tangible prop is an essential feature to preserve the affordances offered by the use of a TUI in the VR environment.

The virtual representation of the TUI, shown in Fig. 3.21, is also leveraged to provide the user with additional visual feedback to cope with the limitations described above. The VR Plug-in is responsible for managing the visual cues to be shown. In particular, when the current selection does not contain any partition, bones that can be manipulated are represented using a green color (remaining bones are displayed using a gray color), whereas the TUI is drawn as it appears in the real world, as illustrated in Fig. 3.21a.

When the user starts the procedure for selecting a partition, all the bones belonging to it are visualized with a different color. The same color is used to represent the interface elements of the TUI that controls that partition (as depicted in Fig. 3.21b).

To make the rules defining the mapping between interface elements and bones' DOFs for a given partition visible, the user is requested to activate them with a specific command (more details will be provided in the following). As shown in Fig. 3.21c, once the user has activated the configuration to control a given partition, one or more circles are displayed for each bone to indicate the DOFs that can be controlled by the animator (one circle per DOF). Colors used to draw circles are the same used to represent the TUI's element which is mapped onto that DOF.



(a) No partition selected (b) Partition selected (c) Partition activated

Figure 3.21: Appearance of the character's bones and of the TUI during interaction.

Selection modalities

By leveraging VR, it was possible to introduce two alternative activation modalities, referred to as "gaze" and "proximity" selection. However, in the future new approaches, e.g., based on head-mounted depth cameras [214], could be considered.

The selection mechanism based on the gaze is commonly used in many VR-based applications. It leverages fixation time, i.e., the time that the user keeps looking at a fixed point. To this aim, a virtual "cursor" (shown in Fig. 3.21a) is attached to the user gaze, which remains always visible in the virtual environment. After looking at a bone for a certain amount of time, the partition containing that bone gets activated.

The approach based on proximity selection is similar. In this case, the virtual cursor is replaced by the position of the tangible prop. In particular, when the prop is kept close to a bone for a certain time (i.e., it is "snapped" to it), the partition including that bone is enabled. The position of the prop (i.e., the position of the HTC Vive tracker mounted on the prop itself) is represented in the virtual environment through a set of axes, as shown in Fig. 3.21a and Fig. 3.21b.

Fig. 3.21 shows the procedure based on gaze selection, as indicated by the presence of the cursor indicating the user's gaze, which is moved to select/control the right wing of the bird character.

Both visual cues and the selection methods are handled by the VR Plug-in, since this component is able to merge the bone data with information concerning the position and orientation of both the animator's head and the tangible prop. Preliminary studies on the two selection modalities suggest that selection based on gaze is perceived as more intuitive than proximity selection, probably because it is a common paradigm in VR applications. However, it is worth noticing that gaze selection could be difficult to use when bones are small or far from the user, since it requires animators to keep the gaze fixed on a small element for the fixation time set. On the other hand, proximity selection may work not well when bones are hard to reach in the virtual environment (too low/high with respect to the user, etc.). For these reasons, it was decided to support both the modalities. With the aim of avoiding the inadvertent activation of partitions while the user is manipulating another part of the character, selection needs to be activated through a voice

command.

3.3.4 Experimental evaluation

In order to asses the effectiveness of the proposed system, a user study that asked participants to perform a posing task using three modalities, i.e., by using the proposed VRS, the reference PS, and the Blender's mouse and keyboard native interface (later referred to as MK) was conducted.

Case study

The virtual character that participants were requested to interact with was the dyno character described in [173] for conducting the experiments. The same character was selected since it presents a high number of bones and DOFs to be controlled. The tangible interface assembled for this task is composed of four large servo motors. Its configuration is depicted in Fig. 3.22. The figure also reports for each bone the DOFs that should be controlled. Names on the left side of the figure correspond to the voice commands to be issued for activating one of the nine partitions when working with the PS. All the bones belonging to a partition are represented with the same color used for the names. Finally, arrows indicate a sample mapping between interface elements and manipulated DOFs of bones in a given partition (named "body", in the particular case). Positional data gathered by the HTC Vive tracker mounted on the Intelligent Brick were used to track the tangible interface in the 3D space and manage the position of the end effector controlling the character's tail (leveraging inverse kinematics). The reaming bones were controlled with forward kinematics.

In the experiments, the users had to make the initial pose of the character (shown in white/gray in Fig. 3.23) mimic as much as possible the target pose (in red).

Procedure

The user study was carried out with 20 volunteer participants (17 males and three females), aged between 22–34 years. Participants were selected among students and academic staff at the Politecnico di Torino, in Turin, Italy. Participants were divided into two groups depending on their skills in the field of computer animation. In particular, half of the participants were considered as skilled users, because of their expertise with computer animation suites achieved by attending and/or teaching 3D modeling and animation courses. Those in the second half were considered as unskilled users. All the participants had to complete the posing task with the three modalities, i.e., VRS, PS and MK. The Latin Order was used to select the modality a given user had to start and continue with, in order to reduce

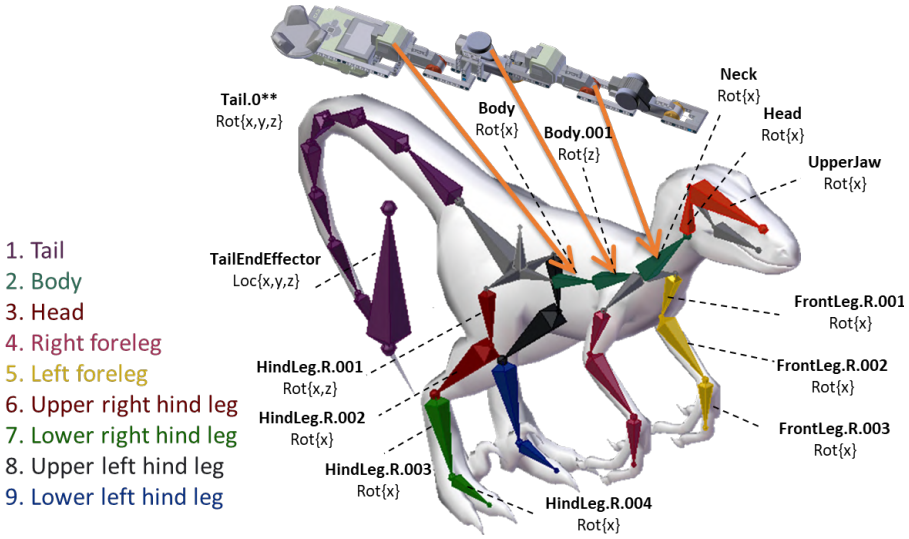


Figure 3.22: Dyno character used in the user study

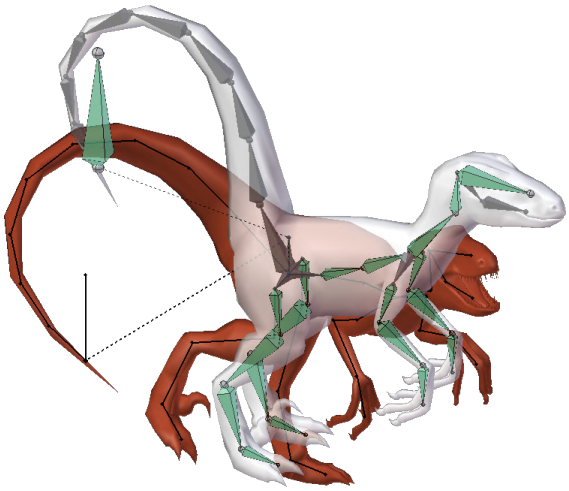


Figure 3.23: Dyno character's armature set in rest pose (white) and target pose (red)

possible learning effects. Before starting the task, the main functionalities of the animation system to be used for executing the task were introduced; afterwards, participants were given some time to get acquainted with the VR environment (specifically, they were allowed to familiarize with the interfaces used in the task by experimenting them with a different character to be articulated). For what it concerns the modalities in which participants had to use the PS and the VRS, the overall procedure included the following step:

1. pronounce a voice command that makes the task start (which also activates

the procedure for collecting objective measurements considered in the evaluation);

2. select an armature's partition to start with by issuing the corresponding voice command (when operating with the PS) or by using the gaze/proximity selection methods (when working with the VRS);

3. issue a voice command to reset the mapping between data gathered by the interface elements and the DOFs of the controlled bones;

4. correct the bones' position and orientation in the current partition by translating and articulating the tangible prop to match as much as possible the reference;

5. iterate from steps 2 to 4 until the target pose has been recreated and the two armatures overlap;

6. terminate the task by issuing a voice command.

Two videos showing a participant executing the task with the PS and the VRS are available for download[23].

Concerning the MK modality, participants were requested to operate with Blender's position and orientation handles in order to articulate each bone as needed.

Similar to [173], no time limit or accuracy threshold was set. In fact, participants were allowed to operate until they considered the task completed. However, an audio signal informed the users when they were approaching the target. The signal was activated when the pose reached a given threshold, which intentionally set to a high value (namely 20%) so that all the participants had to continue adjusting the pose despite the signal.

Performance metrics

To evaluate the effectiveness of the VRS and compare it with PS and MK, both objective and subjective measurements were considered. Objective measurements, computed for all the modalities, made use of the three indicators defined in [173]. The first indicator, the completion time, accounts for the time needed by a participant to execute the whole task., i.e., to reach the reference pose. The second indicator, the animation accuracy, evaluates the difference between the pose obtained by the user and the target pose in terms of Euclidean (for the position) and angular (for the orientation) distances. During the task, the value of this indicator decreases from 100% to an ideal 0% value (reached when the two armatures perfectly overlap). The third indicator, the amount of work, provides an estimation of

[23]Videos of the user study: https://goo.gl/6WcP4m

the work needed to articulate the character. This indicator is computed by measuring the area under the curve defined by the variation of the animation accuracy during the animation, from time $t = 0$ to $t = T$ (where T corresponds to the maximum time at which the minimum value of the animation accuracy was reached for one of the three systems). Then, the area is normalized by the animation time. In other words, the amount of work estimates how fast was the animator to converge towards the target pose: the lower the amount of work was, the faster the operation was.

Subjective measurements were collected by asking participants to fill in an after-test questionnaire, which is available for download[24]. Differently than in the objective evaluation which also considered the MK in order to gather a numerical ground-through under conditions which were slightly different from those in [173], the subjective evaluation was performed considering only the PS and VRS, since a higher appreciation of PS with respect to MK had been already demonstrated. The questionnaire was comprised of two sections. The first section investigated aspects pertaining to ergonomics associated with the interaction means based on the ISO 9241-400 standard. Users' preference for either the PS or the VRS was additionally collected. In the second section, specific usability factors related to interaction with virtual contents were studied based on the questions proposed in [153].

3.3.5 Results

Results that compare the performance of the VRS, PS and MK are presented by first considering objective results, then focusing on subjective results.

Objective observations

Average values in terms of completion time, animation accuracy and amount of work achieved by skilled (in the following SKUs) and unskilled (UNUs) users with the three systems are reported in Fig. 3.24. Statistically significance of the results was analyzed by means of the ANOVA and paired Student's t-tests ($p < 0.05$). For the ANOVA test, in addition to the significance values, the F statistic, and the degrees of freedom, it was chosen to express the effect size by means of the Partial Eta-Squared (η_p^2).

Starting from SKUs, on average users were significantly faster (Fig. 3.24a) with the VRS ($F_{(2,33)} = 4.03$, $\eta_p^2 = 0.24$, $p = 0.02$). Pairwise comparisons show that participants using the VRS ($M = 5$ min 4 s, $SD = 1$ min 14 s) were 18% and 22% faster than with both the PS ($M = 6$ min 12 s, $SD = 1$ min 7 s, $t(9) = -4.36$, $p < 0.01$, $d = -0.95$) and MK ($M = 6$ min 31 s, $SD = 1$ min 11 s, $t(9) = -2.53$, $p = 0.01$, $d = -1.20$). Moreover, participants were more accurate (Fig. 3.24b) with

[24]Questionnaire: https://goo.gl/A2fgLK

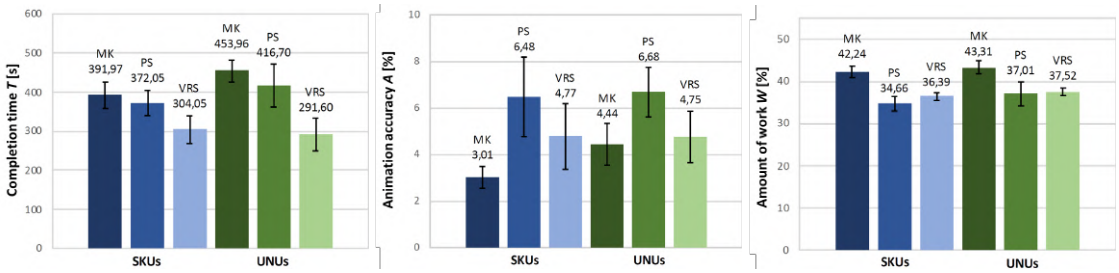
the VRS and MK ($F_{(2,33)} = 3.61$, $\eta_p^2 = 0.21$, $p = 0.04$). In particular, pairwise comparisons revealed significant differences between PS ($M = 6.48\%$, $SD = 3.60$) and both VRS ($M = 4.77\%$, $SD = 2.98$, $t(9) = 5.99$, $p < 0.01$, $d = 0.52$) and MK ($M = 3.01\%$, $SD = 0.99$, $t(9) = 2.77$, $p = 0.02$, $d = 1.32$). The VRS did not allow users to fill the gap with the MK; however, values obtained by the two systems were closer. With respect to the amount of work (shown in Fig. 3.24c), no significant differences were found between the VRS and the PS. This result was probably related to the fact that, although users were faster in setting a given DOF with the VRS since it was observable from a better point of view or can understand more easily how to control it thanks to visual cues, they also spent part of the time to reposition themselves in the virtual environment. In fact, repositioning is needed both to activate partitions by using either gaze or proximity selection (with their waiting times), as well as to understand the current mapping rules. These operations are not needed in the PS: e.g., partitions are selected by means of voice commands, and users did stand still in front of the projected wall. The time lost to perform the operations in the above makes the values of animation accuracy constant for a certain time interval. The presence of these time intervals results in an increased amount of work, since the area under the curve of the animation accuracy (which represents the computed value of the amount of work) does not have a continuous decreasing. Moreover, when computing the amount of work the contribution of time was removed through a normalization. For these reasons, similar values of the amount of work were obtained, notwithstanding the completion times were significantly different.

As expected, the amount of work with both the VRS and the PS was lower than with MK, confirming the potentialities of the TUI as a tool for sketching the character's pose.

Considering performance achieved by the UNUs (Fig. 3.24a), results confirmed the observations above. In particular, VRS allowed participants to complete the task faster than both PS and MK ($F_{(2,33)} = 9.66$, $\eta_p^2 = 0.40$, $p < 0.01$). In fact, pairwise comparisons show that participants were 30% and 36% faster with the VRS ($M = 4$ min 51 s, $SD = 1$ min 28 s) than with the PS ($M = 6$ min 12 s, $SD = 1$ min 7 s, $t(9) = -2.94$, $p = 0.01$, $d = -1.21$) and MK ($M = 7$ min 33 s, $SD = 58$ s, $t(9) = -5.53$, $p < 0.01$, $d = -2.31$).

With respect to animation accuracy (Fig. 3.24b) UNUs were more accurate when using VRS and MK ($F_{(2,33)} = 3.36$, $\eta_p^2 = 0.18$, $p = 0.04$). More specifically, pairwise comparisons show statistically significant differences between PS ($M = 6.68\%$, $SD = 2.25$) and both VRS ($M = 4.75\%$, $SD = 2.33$, $t(9) = 2.42$, $p = 0.03$, $d = 0.84$) and MK ($M = 4.44\%$, $SD = 1.88$, $t(9) = 2.77$, $p = 0.01$, $d = 1.15$). Considering the amount of work (Fig. 3.24c), the same considerations made for SKUs were still valid for UNUs.

Interesting outcomes can also be found by comparing the results of UNUs with those of SKUs. In particular, it can be noticed that SKUs were faster than UNUs



(a) (b) (c)

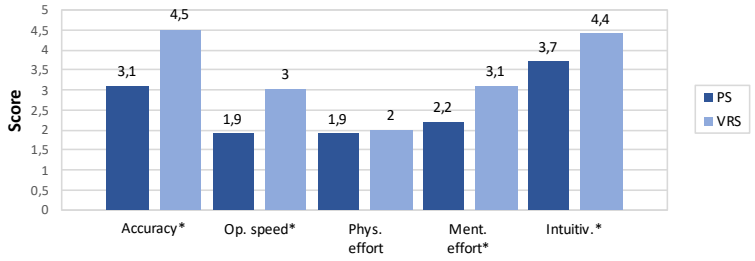
Figure 3.24: Objective results obtained by SKUs and UNUs.

when using both the PS and MK. However, when using the VRS, performance was almost the same, i.e., they were both faster and more accurate than with the PS. These results suggest that the VRS could be more effective than the PS in smoothing the differences in terms of computer animation skills needed. This could help confirm the helpfulness of the VRS for novice users. With respect to accuracy, SKUs performed better than UNUs only with MK, as expected. No differences were found in terms amount of work between the two user categories.

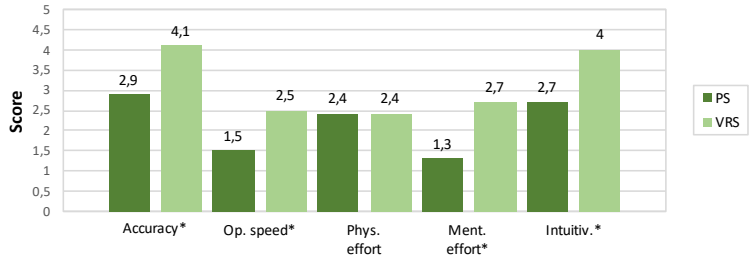
Subjective observations

The first section of the questionnaire requested that participants evaluate the perceived accuracy and operation speed, physical and mental effort as well as intuitiveness of VRS and PS, by rating them on a 1-to-5 scale. Moreover, participants had to express their preferences, by providing motivations for the choices they made. Fig. 3.25 report results for SKUs and UNUs, respectively. Values concerning physical and mental effort have been inverted on a better-to-worse, 5-to-1 scale (thus, a higher score can be interpreted as a lower perceived effort). Statistically significant results (based on paired Student's t-tests, $p < 0.05$) are marked with *.

Starting from preferences, overall, 19 out of the 20 participants expressed their preference for the VRS. According to the motivations provided, this preference was mostly due to two factors. The first one is the higher level of control over the animation system perceived when using the VRS. The second one pertains to the greater awareness of the virtual character when operating in an immersive environment. By focusing on questions that are statistically significant, it can be noticed that the VRS was perceived as more accurate (SKUs – PS: $M = 3.10$, $SD = 0.74$; VRS: $M = 4.50$, $SD = 0.71$; $t(9) = -4.12$, $p < 0.01$, $d = -1.94$; UNUs – PS: $M = 2.90$, $SD = 0.74$; VRS: $M = 4.10$, $SD = 0.57$; $t(9) = -4.13$, $p < 0.01$, $d = -1.82$) and faster (SKUs – PS: $M = 1.90$, $SD = 1.10$; VRS: $M = 3.00$, $SD = 0.94$; $t(9) = -2.28$, $p = 0.04$, $d = -1.07$; UNUs – PS: $M = 1.50$, $SD = 0.85$; VRS: $M = 2.50$, $SD = 1.08$; $t(9) = -2.53$, $p = 0.03$, $d = -1.03$) than the PS, confirming the objective measurements. Both the user categories also found



(a) SKUs



(b) UNUs

Figure 3.25: Subjective results concerning interaction means based on ISO 9241-400 factors.

the VRS as characterized by a higher intuitiveness (SKUs – PS: $M = 3.70$, $SD = 1.25$; VRS: $M = 4.40$, $SD = 0.84$; $t(9) = -2.68$, $p = 0.02$, $d = -0.66$; UNUs – PS: $M = 2.70$, $SD = 0.67$; VRS: $M = 4.00$, $SD = 0.67$; $t(9) = -3.88$, $p < 0.01$, $d = -1.94$) and, compared to the PS, it was found to require a lower mental effort (SKUs – PS: $M = 2.20$, $SD = 1.14$; VRS: $M = 3.10$, $SD = 0.57$; $t(9) = -3.25$, $p < 0.01$, $d = -1.00$; UNUs – PS: $M = 1.30$, $SD = 0.67$; VRS: $M = 2.70$, $SD = 0.82$; $t(9) = -3.28$, $p < 0.01$, $d = -1.86$). These results suggest that visual cues are perceived as useful, similar to the alternative methods introduced to select armature's partitions. Regarding the physical effort, no significant differences were found and scores were not particularly high. Notwithstanding, this result could be interesting since it apparently suggests that possible issues related to the use of immersive environment, e.g., concerning eye strain or motion sickness, did not influence the user experience. Comparing feedback provided on the same system by SKUs and UNUs, it can be noticed that advantages brought by the VRS were apparently more evident for UNUs than for SKUs (especially concerning mental effort and intuitiveness). This is probably due to the expertise of SKUs to work with multi-view visualization, in this case, used for the PS.

Questions in the second section asked participants to evaluate the two systems based on the nine usability categories defined in [153], namely, functionality, user input, system output, user guidance and help, consistency, flexibility, error correction/handling and robustness, sense of immersion/presence and overall system usability. Questions were expressed as statements to be evaluated on a 1 to-5 scale

(from strong disagreement to strong agreement). Participants were also requested to express an overall evaluation again in a 1-to 5 scale (from very unsatisfactory to very satisfactory) for each category. Table 3.8 and Table 3.9 report scores assigned to the overall evaluation questions by SKUs and UNUs, respectively. It can be observed that both the user categories preferred the VRS, since each category received, on average, a higher score (statistical significance based on paired Student's t-tests is reported).

Table 3.8: Aggregated subjective results concerning usability of the two systems for SKUs in terms of the usability factors defined in [153].

sFactor	**PS**		**VRS**				
	M	*SD*	*M*	*SD*	*t(9)*	*p-value*	*d*
Functionality	3.60	0.84	4.60	0.52	-3.35	<0.01	-1.43
User input	3.40	0.52	4.10	0.88	-2.69	0.02	-0.97
System output	3.20	1.03	4.30	0.95	-2.28	0.04	-1.11
User guidance and help	3.50	0.71	4.50	0.71	-4.74	<0.01	-1.41
Consistency	4.00	0.67	4.60	0.52	-2.71	0.02	-1.01
Flexibility	3.20	0.42	4.20	0.79	-4.74	<0.01	-1.58
Error correction	3.70	0.67	4.30	0.67	-2.71	0.02	-0.89
Sense of immersion	2.50	0.71	4.50	0.97	-4.04	<0.01	-2.35
Overall usability	3.00	0.82	4.30	0.95	-3.28	<0.01	-1.47

Table 3.9: Aggregated subjective results concerning usability of the two systems for UNUs in terms of the usability factors defined in [153].

sFactor	**PS**		**VRS**				
	M	*SD*	*M*	*SD*	*t(9)*	*p-value*	*d*
Functionality	2.90	0.99	4.30	0.95	-4.12	<0.01	-1.44
User input	2.50	0.71	3.90	0.57	-5.25	<0.01	-2.18
System output	2.70	0.82	4.20	0.92	-5.58	<0.01	-1.72
User guidance and help	2.90	0.99	4.10	0.57	-4.13	<0.01	-1.48
Consistency	3.10	0.74	4.20	0.92	-3.50	<0.01	-1.32
Flexibility	2.50	0.85	3.80	0.92	-3.88	<0.01	-1.47
Error correction	2.80	0.79	4.20	0.63	-3.77	<0.01	-1.96
Sense of immersion	2.40	0.84	4.90	0.32	-9.30	<0.01	-3.93
Overall usability	2.80	0.63	4.70	0.48	-10.58	<0.01	-3.38

Individual scores assigned to each question can be investigated by examining disaggregated data provided in the original paper.

Starting from the user input category, results indicate that the VRS was perceived as easier to use than the PS by both SKUs and UNUs. Moreover, participants

found it easier to move and reposition themselves in the virtual environment when using the VRS. UNUs additionally found that with the VRS they were able to limit the number of mistakes, they had a higher level of control over what they wanted to do and it was easier for them to select and move virtual objects during the experiment.

Regarding system output, the display and the field of view were considered as more appropriate with the VRS than with the PS by both user categories. Feedback shown was considered as more adequate with the VRS than the PS, as well as the information, which was found to be presented in a more meaningful way. Differently than with the PS, participants did not lack the sense of depth with the VRS. No statistical significances were found for the questions concerning nausea or eye fatigue, suggesting that participants felt comfortable with both the systems. Regarding scores assigned to statements about user guidance and help, no statistical significances were found for questions regarding difficulty to learn and need for further help. However, both SKUs and UNUs found the PS more difficult to use than the VRS.

With respect to consistency, both UNUs and SKUs stated that the sequence of inputs to execute a specific action matched better with their understanding of the task when using the VRS than the PS. This finding may be related to the different set of commands for selecting a partition, that in the VRS is achieved with gaze and proximity selection. UNUs also found that they were less confused and more confident that the system responded as they were expecting when using the VRS than the PS.

Regarding flexibility, VRS outperformed PS, since both categories of users found it easier to perform the task in the way (order) they chose, and they perceived as easier to tailor the system to their needs. Participants also found that with the VRS they were able to take shortcuts, probably because of the VRS support for different selection modalities and the possibility to move and observe the virtual character from the preferred perspective.

Regarding error correction/handling and robustness, both SKUs and UNUs stated that with the PS they were unaware of making mistakes. This result is probably due to the need, in the PS, to observe multiple views for determining whether a DOF was correctly set. In the VRS, this operation can be easily performed by moving a bit the head or body to change the point of view. Concerning the sense of immersion/presence, both user categories perceived the feeling of being part of the virtual environment when using the VRS, thus perceiving a better sense of scale than in the PS.

Finally, regarding overall usability, is can be observed that both user categories enjoyed more using the VRS than the PS, and found a real benefit in the use of VR as an interface between humans and machines. For both SKUs and UNUs, it was more difficult to manage three-dimensionality with the PS than with the VRS. Furthermore, UNUs found it easier to learn how to use the VRS than the PS,

they felt more in control, and they always had a clear idea about how to perform a particular operation. These latter findings confirm that benefits brought by the VRS could be even higher for UNUs than for SKUs.

3.3.6 Future developments

Future work could be aimed at improving pose accuracy by enhancing the performance of the technology considered to manage the articulation of the tangible prop, e.g., by considering other tracking methods. An alternative could be the introduction of methodologies to fine tune the mapping between TUI's and character's modifications during the interaction. Finally, experimental evaluations could be extended to consider diverse animation scenarios, e.g., facial animation, in order to characterize system performance under different conditions.

3.4 Posing character through 3D sketching

As revealed also by the earlier studies presented in this chapter, one of the most complex stages in the virtual character animation pipeline is represented by the posing step [170]. In this step, animators articulate the character's (skeleton) either in a direct or indirect manner by manipulating handles (or bones) which may be characterized by a high number of DOFs [115]. However, although animators are requested to operate on 3D elements, interfaces offered by common animation tools are natively 2D [312, 142, 61]. Focusing first on aspects regarding the user input, among the solutions already described above, it is possible to identify a new approach that leverages sketch-based interfaces. In fact, sketching is used in all of the phases of the creative process, from building up shapes and exploring motion with rough key poses to drawing storyboards [108]. Sketch-based interfaces have been explored by the research community for various computer graphics applications, since they allow for expressive, simple and intuitive, interaction in a way that is closer to cognitive processes [9].

Major developments in sketch-based research encompass modeling [184], mesh editing [300], rigging [36], posing [115], editing of articulated characters' motion [61], simulating crowds [206] and scene deformation [65]. However, considering that articulated figures have a relatively high number of DOFs to control, and taking into account the difficulty of editing 3D elements through 2D devices (like display, tablet, tabletop, etc.) or interpreting 2D line drawings in 3D, it is not surprising that fully sketch-based animation of articulated characters remains a challenging and open problem [61, 19]. Regarding system output, as illustrated in the above, the last decade was characterized by the progressive spread of VR that pushed the research community to study new ways to leverage this technology for addressing issues affecting computer animation [245].

By moving from the above considerations, in the following a system for character posing able to combine the benefits offered by sketch-based interfaces and VR technology is presented. With this system, animators can manipulate a rigged virtual character by sketching lines into an immersive virtual environment. This work was made in collaboration with researchers at the Hong Kong University. My contribution focused on the design and development of the VR system and the algorithm for character posing that will be detailed in the following.

So far, this work has been illustrated in a paper submitted to a conference which is currently under review.

3.4.1 Related work

As said, researchers have long recognized the great potential of sketch-based interfaces for a wide range of tasks in computer graphics. For example, the work in [113] proposes a mathematical definition of the *line of action* (LOA), i.e., a conceptual tool used by cartoonists and illustrators to help make their figures more consistent and more dramatic. The system allows animators to automatically align a 3D virtual character to a user-specified LOA by solving an optimization problem. By focusing on this simple abstraction, the animator can quickly adjust and refine the overall pose of his or her character from a given viewpoint. The work includes the description of an automatic way to determine the correspondence between the LOA and a subset of the character's bones. The well-known depth ambiguities problem of 2D sketches was addressed by constraining the transformations to the viewing plane.

In [115] a sketch-based posing system for rigged 3D characters was presented that allows artists to create custom *sketch abstractions*, i.e., a set of rigged curves that form an iconographic 2D representation of the character from a particular viewpoint, on top of a character's shape. When a new input sketch is provided, the system tries to minimize the nonlinear iterative closest point energy in order to find the rigging parameters that best align the character's sketch abstraction to the input sketch. The distinguishing characteristic of the method is that it does not prescribe the sketch representation a priori, but rather lets the animators encode the sketch representation that is most appropriate for the character to be deformed.

The authors of [19] present a sketch-based character posing system which is more flexible than the previous methodologies. The sketches provided as input for the character deformation could depict the skeleton of the character, its outline, or even a combination of the two. An optimization problem was formulated to match the subset of vertices from the character mesh with the points obtained by sampling the input sketch.

In [29], a method for 3D character posing able to reconstruct the pose of the character using *gesture drawings* and a rigged character model as input is proposed. The advantage of gesture drawings over other types of 2D inputs is the lack of

perceptual ambiguity. Unlike stick-figures, LOA, and outer silhouettes, gesture drawings allow artists to unambiguously convey poses to human observers. By identifying and leveraging the perceptual pose cues used by artists when creating these drawings, the system is able to automatically recover character poses that are consistent with artist's intent. The system handles complex poses with varying and significant part foreshortening, occlusions, and drawing inaccuracies.

The work in [108] refers to a methodology to infer a 3D pose from a monocular 2D sketch. The method does not make any external assumptions about the model, allowing it to be used on different types of characters. The 3D pose estimation is formulated as an optimization problem. In particular, a parallel variation of a Particle Swarm Optimization (PSO) algorithm is used to manipulate the pose of a preexisting 3D model until a suitable 3D pose is found. The pose is obtained by comparing the 3D rendering of the model and the input drawing. During the process, the user input is still required to pinpoint the joints on the drawing (operation which can be also performed by unskilled users).

3.4.2 Proposed system

From the analysis of the literature it can be observed that existing tools for posing 3D characters using sketches are still based on 2D input devices. As a result, the view in which the sketch is drawn represents a relevant factor in the creation of the pose, since it could affect the accuracy of the final 3D result. The basic idea of the proposed system is to convert the existing methodology from 2D to 3D in order to allow animators to draw sketches into an immersive virtual environment. Fig. 3.26 shows the basic usage of the devised system. Given a 3D rigged character and some sketches (drawn by the animator into the virtual environment), the system is capable to automatically align them by minimizing their distance. The system assumes that a skeleton is already defined for the character: hence rigging and skinning are not considered.

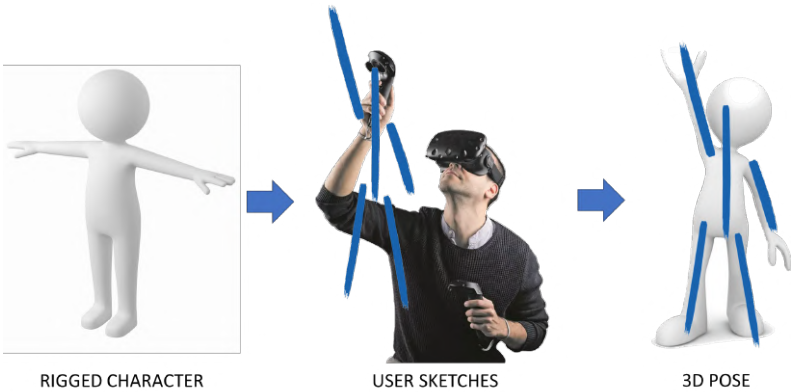


Figure 3.26: Basic usage of the proposed system.

Another drawback of the solutions in the literature is the fact that they were developed as standalone applications. Hence, integration with common animation suites such as Blender, Autodesk Maya, etc. can take place only in a separate step of the animation workflow, thus making the process more tricky. The proposed system considered this aspect by proposing an add-on for the Blender open-source graphics suite. The goal of the proposed system is not to replace Blender, but rather to offer an alternative tool that combines the affordances of posing characters with 3D sketching and the advanced functionalities targeted to character animation provided by traditional software, with the final aim to speed up the overall process.

The steps considered for the creation of the proposed system and the corresponding challenges to be solved can be summarized as follows:

- creating an immersive environment where the user can draw sketches;
- identifying and developing a methodology to find the best mapping between the sketches provided as input and the skeleton of the virtual character to be deformed;
- identifying and developing a methodology to find the transformations to be applied to the bones of the character's skeleton in order to align them with sketches.
- integrating the new methodology for character posing into a well-know animation suite.

The requirements reported above were considered for the design and development of the two main components constituting the system, i.e., a VR environment integrated in Blender for letting the users draw 3D sketches, and a matching algorithm in charge of articulating the character's skeleton in order to make it assume the pose represented by the provided sketches. In the following, more details about the VR environment for generating 3D sketches as well as the functioning of the matching algorithm will be given.

VR-based environment for 3D sketching

As previously mentioned, the graphics suite selected to host the posing tool based on 3D sketches is Blender. In particular, the devised tool was developed as a new add-on for Blender by leveraging two existing libraries: the Virtual Reality Viewport library and Pyopenvr SDK[25]. The Virtual Reality Viewport library allows users to visualize a 3D scene (containing the characters to be animated and the sketches) into an immersive environment through a HMD. Pyopenvr is a Python

[25]pyopenvr: https://github.com/cmbruns/pyopenvr

binding for the Valve's OpenVR virtual reality SDK to get the status of the HTC Vive's controllers, in order to implement specific behaviors when the user interacts with the controller's buttons.

The tool's functionalities can be activated into the immersive environment by pressing the controller's buttons, as shown in Fig. 3.27. Currently, functionalities are provided to:

- draw a stroke (right controller's Trigger);
- select the character to pose (right/left controller's Gripper);
- apply translation and rotation transformations to the selected character (left controller's Trigger)
- launch the matching algorithm (right controller's Trackpad Up);
- reset the transformations applied to the skeleton by setting the rest pose (right controller's Trackpad Right);
- delete all the strokes drawn by the user (right controller's Trackpad Down);
- delete the last stroke drawn by the user (right controller's Trackpad Left);
- activate the playback of the animation (left controller's Trackpad Up);
- navigate the timeline by increasing/decreasing the current frame (left controller's Trackpad Right/Left);
- insert a keyframe to record the orientation of all the bones in the character's skeleton for the current frame (left controller's Trackpad Down).

Visual feedback was introduced to ease the interaction with the system. In particular, on the right controller, a label shows the current operation performed, i.e., Idle (waiting for a new command), Selection (functionalities for changing the skeleton to be manipulated) and the current skeleton selected. On the left controller, a label indicates the current frame and the presence of a keyframe for the selected skeleton. If a keyframe has been set for the current frame, the text is colored yellow, otherwise, it remains grey (according to the convention used in Blender for representing keyframes).

At present, multiple strokes and multiple skeletons are supported. This implies that if the scene contains more than one character, all of them can be manipulated using this approach (one at a time) by drawing multiple strokes.

Fig. 3.28 shows several characters (whose geometry was kept intentionally simple) characterized by armatures with a different topology. Fig. 3.28a, Fig. 3.28b, and Fig. 3.28c, show the armatures in rest pose and the drawn sketches, whereas

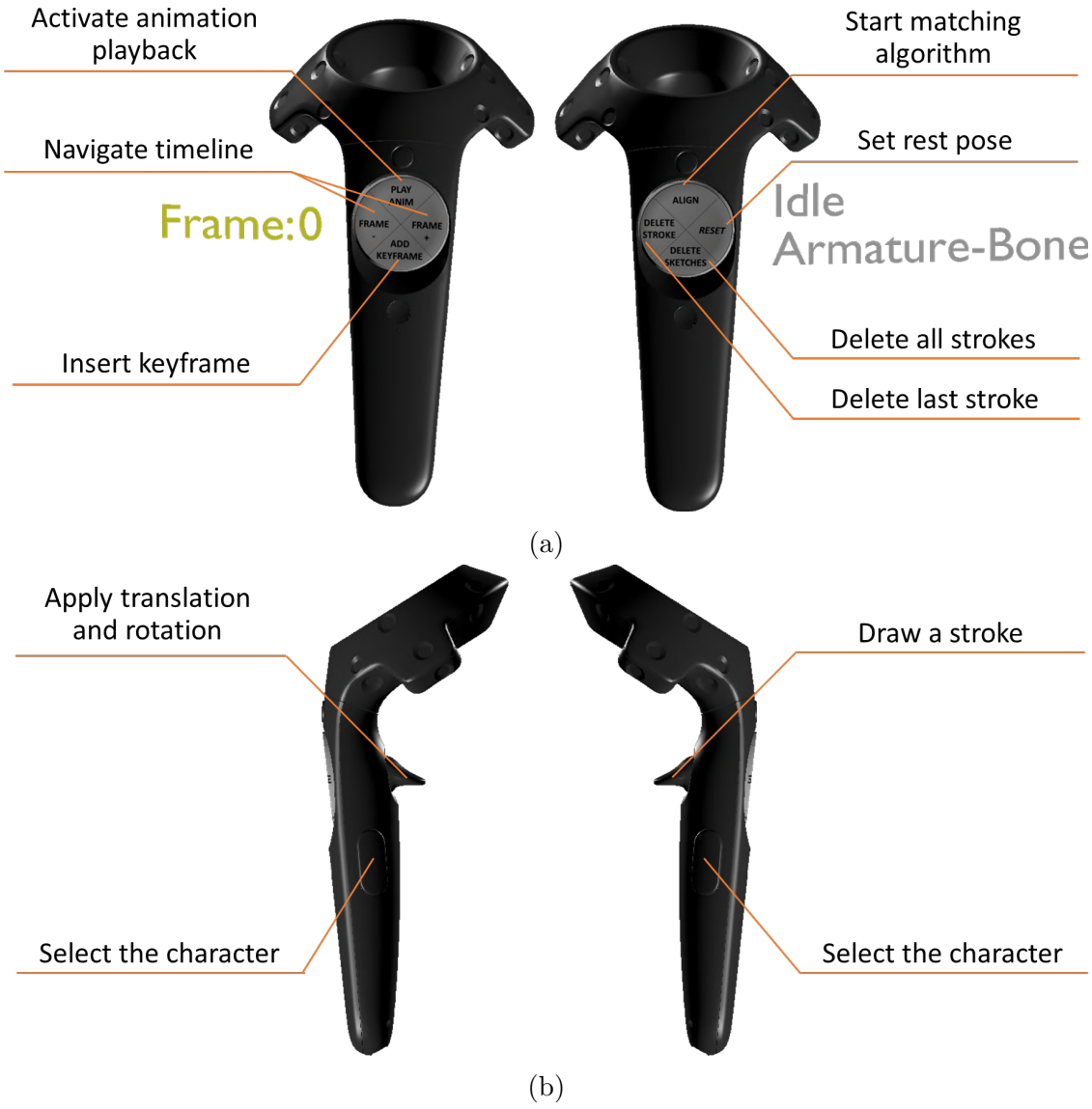


Figure 3.27: Functionalities available through the controllers.

Fig. 3.28d, Fig. 3.28e, and Fig. 3.28f, illustrate automatically computed poses. A video[26] is available for download, representing the current state of development.

[26]Video of current developments: https://bit.ly/35GdKqO

Matching algorithm

The algorithm used for the definition of the mapping was derived from the methodology presented in [19]. In that paper, the problem of identifying the mapping between the pose of a virtual character and the input sketch was formulated as an optimization problem:

Given two sets of points, the sampled input sketch $Y = (y_1, y_2, ..., y_M)$ and a subset of points belonging to the character model $V = (v_1, v_2, ..., v_K)$, find the correspondence, or match matrix, ω, and the amount of deformations in p which minimize the following expression

$$min(\omega, p) \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_{ki} \left\| y_i - v_k(p) \right\|_2^2 + \Phi(p) - \zeta \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_{ki} \tag{3.5}$$

subject to the following constraints

$$\sum_{i=1}^{M+1} \omega_{ki} = 1; \sum_{k=1}^{K+1} \omega_{ki} = 1; \omega_{ki} \in \{0,1\} \tag{3.6}$$

where:

- $\omega = \{\omega_{k,i}\}_{(K+1)\times(M+1)}$is the correspondence matrix consisting of two parts: the upper-left $K \times M$ part defines the correspondence, and the extra $K+1$-th row and $M+1$-th column are introduced to handle the outliers; the points in V and Y having no correspondences would be automatically determined as outliers;
- p is a vector conting the character posing parameters on the joints which deform points in V to Y in order to obtain a new pose $V(s)$ as closely as possible to Y;
- $\Phi(p)$ is a regularization term, used to add further constraints for searching candidate solutions in limited space;
- ζ is a scalar factor to weight the contribution of the third term of the equation, introduced to prevent treating too many points as outliers.

As reported in [19], solving Equation 3.5 directly is very difficult, since the objective function consists of a linear discrete assignment problem for correspondence and a least-squares continuous problem for deformation. The authors of [19] proposed to adopt an alternating strategy to find the correspondence parameter ω and the rig parameters p. Going back and forth between the correspondence and pose in an iterative manner can help to solve the problem, since the knowledge of one makes it easier the determination of the other.

By fixing p, it is possible to find the sub-optimal values for ω by leveraging two techniques: *Softassign* and *deterministic annealing*. Unfortunately, in [19] no

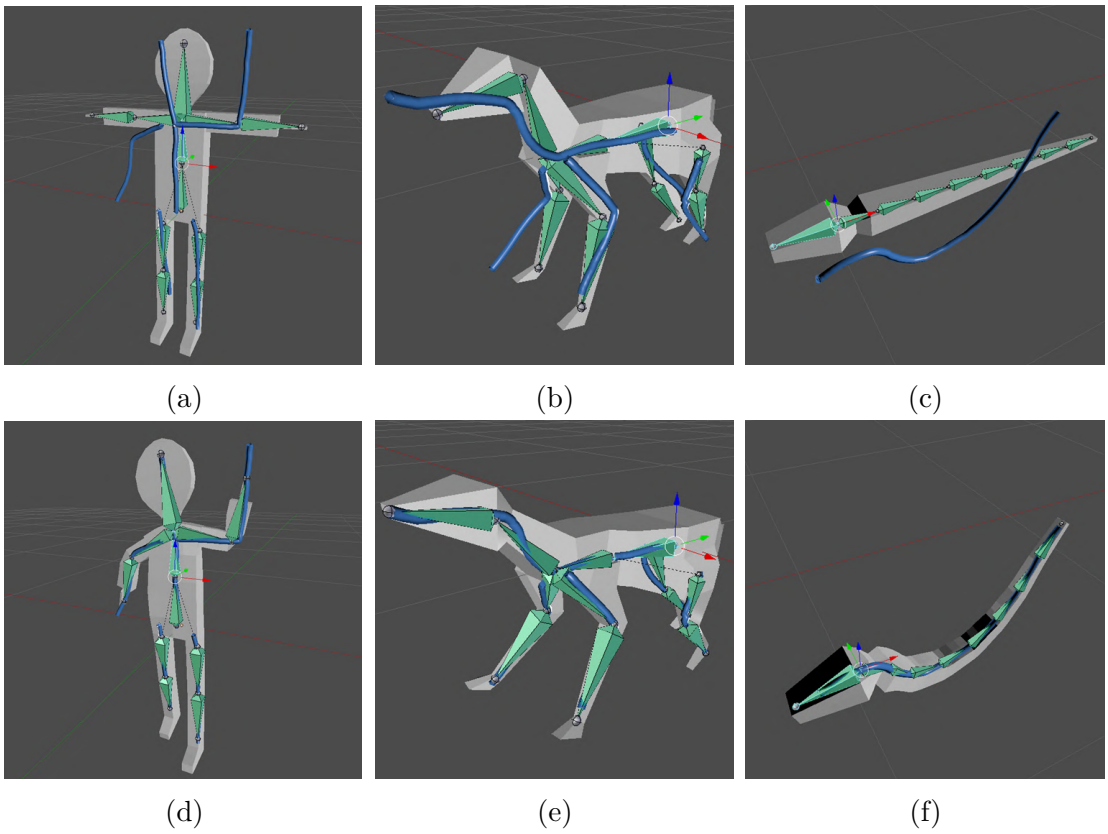


Figure 3.28: Examples of armatures articulated through 3D sketches.

technical details are provided on how to use these two methods for the particular problem. To implement them, the original paper presenting the use of these techniques for 2D and 3D point matching was considered [105].

The basic idea of the Softassign is to relax the binary correspondence variable ω to be a continuous-valued matrix in the interval [0; 1]. The continuous nature of the matrix basically allows fuzzy, partial matches between the two sets of points [105]. From an optimization point of view, this fuzziness makes the resulting energy function behave better [331] because the correspondences are able to improve gradually and continuously during the optimization, without jumping around in the space of binary permutation matrices (and outliers). The row and column constraints (Equation 3.6) can be enforced via iterative row and column normalization [283] of ω.

Deterministic annealing can be used to directly control the above fuzziness by adding an entropy term to the original assignment energy function (Equation 3.5) [105]. The newly introduced parameter β is called the temperature parameter. The name comes from the fact that as one gradually reduces β, the energy function is minimized by a process similar to physical annealing. At higher temperatures, the

entropy term forces the correspondence to be more fuzzy. The values obtained at each temperature are used as initial conditions for the next stage as the temperature is lowered. The process of deterministic annealing was used not only in [19] but also in a variety of optimization problems [64, 126].

Once the correspondence is found, it is possible to fix ω to obtain the parameters p which minimize the energy function (Equation 3.5). Differently than in [19] where the problem was solved using a Newton-Raphson scheme, in this case Blender's functionalities were used to get the transformation values that best aligns the character's skeleton to the corresponding points in the sketch.

3.4.3 Future developments

The proposal described above represents the baseline for a system that solves a computer graphics problem by leveraging new interfaces and VR technology. A possible extension could consist in the introduction of machine learning algorithms to make the system able to reconstruct the entire character pose or the overall animation by sketching only few lines of the pose.

3.5 Concluding remarks

The goal of the work described in this chapter was the evaluation, in both objective and subjective terms, of possible benefits and drawbacks brought by the use of VR in a computer animation pipeline, with the aim of determining whether this technology could be suitable to replace traditional interfaces. By developing the VR-based system in [171, 51], which was used in the experimental evaluations, it was possible to estimate the impact of a VR-based interface on representative animation tasks.

In [171], the results focused on five animations tasks. From the analysis presented therein, it was possible to conclude that, by operating with the hand controllers in an immersive virtual environment rather than with traditional M&K interface in front of a screen, it is possible to reduce the time required to complete the tasks for both experts and novice users. Users reported that they were more satisfied with the VR-based interface rather than the native one, and characterized the new interface with higher usability, since it was perceived as easier to learn and operate as well as more stimulating. Analyzing in more detail the results, it can be observed that the VR-based interface was largely preferred to M&K by novice users in all the tasks considered, whereas preferences expressed by experts were not as sharp, since in some tasks the interfaces were rated as almost comparable. Although this result can be already considered as a very relevant, there are still ways to improve the user experience for all the tasks, since the greatest advantage of the proposed solution is represented by the potential to let the users carry out

most of (if not all) the animation tasks by working within the same, integrated environment. The objective and subjective measurements collected in [51] confirmed the great befits brought by the VR-based interface also for what it concerns the more specific animation pipeline for virtual characters.

In [47], the focus was on the affordances offered by reconfigurable tangible devices manipulated into an immersive environment, and a comparison between a 2D representation of the virtual scene, possibly based on multiple views, and a 3D visualization leveraging immersive VR was presented. In this case, VR was used to enable new interaction paradigms for selecting the part to be animated and to provide users with additional visual feedback for better understanding how their actions on the tangible device translated into modifications of the character's shape. Results showed that, compared with a solution based on 2D visualization, the proposed VR-based system allowed both novice and expert users to reduce the time needed to complete the task, by also allowing them to obtain higher posing accuracy. A further finding suggests that the VR-based approach was able to reduce the impact of previous experience, allowing users with limited skills to perform similarly to skilled users. Finally, the VR-based approach was perceived as less mentally demanding and characterized by a higher usability from all the perspectives considered (except physical effort).

The possibility to articulate virtual characters through sketches-based interfaces into an immersive virtual environment promises to offer a new intuitive way for manipulating characters' skeletons by means of an alternative interaction paradigm that is already considered suitable for addressing a number of issues in the computer animation field. However, experimental evaluations should be carried out in order to effectively asses the benefit brought by the combination of the sketches-based interfaces and immersive VR.

Chapter 4

Interfaces and methods supporting the generation of graphics assets

Work described in this chapter was originally presented in [5, 46, 14, 50, 6].

4.1 Introduction

As with computer-generated animations, even the use of graphics assets, in general, is becoming an ordinary practice in many domains, ranging from interactive applications [273], to movies and video-games [88, 143], manufacturing [73, 251], data science [264], VR [338] and AR [13], among others.

A number of software suites, such as Blender[1], Autodesk Maya[2], and 3ds Max[3] represent the mainstream solutions chosen by professional users (e.g., modelers and animators) to design and develop virtual 3D scenes, since they provide a complete set of tools for manipulating 3D graphics [158, 232, 275].

However, as already described in the previous chapters, the high flexibility of these graphics suites and the huge number of functionalities are counterbalanced with a very steep learning curve [197]. This fact could represent a constraint for their effective use, especially for novice users [51]. Even for professional users, the manual generation of 3D graphics content requires a lot of effort and represents an extremely time-consuming task [266].

In general, operations needed for generating 3D graphics assets require to manipulate several DOFs (3D position and orientation). However, traditional input

[1]Blender: https://www.blender.org/

[2]Autodesk Maya: https://www.autodesk.com/products/maya/overview

[3]3ds Max: https://www.autodesk.com/products/3ds-max/overview

devices, such as M&K, can manage only two degrees of freedom at a time [47], which translates into an increase in the users' mental effort, ultimately affecting system output. In fact, in order to have a clearer idea of what the virtual scene contains, the users must simultaneously look at multiple views of the same scene or use shortcuts to quickly modify the current point of view. As a result, graphics assets produced by users with limited skills are often unrealistically simple [323], and/or require a significant amount of time to be created. Therefore, the high number of skills needed, and the concerns regarding the use of traditional software suites could prevent users with limited skills from quickly produce 3D assets, although this could represent a fundamental operation also for them [56] (e.g., to present new a concept to more expert users, or to create a draft project to be refined by using more accurate methods).

The objective of the work presented in this chapter is the development of new solutions allowing users with limited computer graphics skills to produce computer-generated graphics assets. In particular, a system supporting the automatic creation of a 3D virtual scene from a 2D image, originally introduced in [5], will be presented first. Afterwards, the application of two tools based on AR and VR to specific use cases, namely constructive art [46, 14] and sport training [50, 6], will be discussed, as they present opportunities to show the effectiveness of alternative approaches to support the generation of 3D content related to the considered domains. In fact, these use cases may request users who, in principle, are not familiar with computer graphics tools to manipulate computer-generated graphics assets.

4.2 Automatic generation of affective 3D virtual environments from 2D images

Among the numerous possibilities offered by traditional computer graphics suites, one of the most common operations performed is the definition of the objects' layout in the scene. The limitations regarding the limited dimensionality of the user input and system output have a huge impact on the user's experience/performance particularly for this task.

With respect to the user input, the research community is devoting greater attention to new methodologies capable of generating content automatically by processing, e.g., text [54], images [313] and audio clips [289]. In general, the process of generating 3D scenes through automatic tools involves two main steps: definition of the content, and scene synthesis [183]. In the first step, a computer-based system identifies the objects to be inserted in the scene and the spatial information needed to correctly place them according to specific rules (such as physical and/or relational constraints). In the second step, a scene is generated and rendered by a graphic engine by taking into account the information retrieved in the first step. Some work in the literature strictly focused on the first step, disregarding the step concerning

scene synthesis. When implemented, the second step generally produces scenes with poor visual quality (e.g., choosing for the 3D mesh of the recognized objects primitive shapes (like cubes or spheres) with the aim to show only the occupied volumes and spatial relations identified in the first step) [183].

Systems able to define not only the layout, but also a convincing visual representation of the 3D scene are attracting greater attention because of their applicability in real scenarios. The most common methodologies proposed in the literature involves the creation of 3D scenes by providing a text description as input. Although this approach has been used previously [54, 56, 68, 277], it may not represent the best choice for fast prototyping. For example, writing a description of the scene to be recreated could be slower than using a picture that represents it. For this reason, in [5], it was decided to investigate the use of a 2D image as input for the system, making it automatically infer content to be inserted in the scene from it.

Considering system output, as discussed in Chapter 3, the attention of the research community is devoted to new possibilities offered by VR technology and the benefits it could bring to the computer graphics domain. In particular, in the specific context considered, the possibility to explore the automatically generated 3D scene in an immersive virtual environment could help users to better understand the actual objects' displacement in the scene. This feature could be leveraged by non-professional users to, e.g., provide expert developers with more accurate suggestions on how the scene should be refined. Furthermore, the use of VR makes it possible to introduce more intuitive techniques for 3D interactions [51] that, in this case, could be leveraged by non-professional users to modify the scene's layout.

Another aspect that is often disregarded is the emotional relevance of the virtual environment, that becomes particularly relevant when the generation of 3D immersive experiences is specifically tackled. In fact, it was proven that emotional relevance could significantly impact on the users' sense of presence [127]. Different shapes, lights, materials, and textures are able to convey measurable effects on the humans' mind [69]. The increasing interest in creating affective virtual environments is confirmed by the growing number of work in the literature [17, 167, 223, 240, 289].

Considering these aspects, in the following a system for the automatic generation of 3D scenes from a single 2D image targeted to non-professional users will be presented. The system was not meant to replace traditional graphics suites, but rather to augment by offering a new tool that allows unskilled users to quickly generate a draft of a 3D scene. Non-professional users can benefit from this system to share with expert developers a possible setup of the environment that they can work with to apply refinements. The generated scene can also be explored within an immersive environment, with the aim to provide users with more insights about objects' layout that could be communicated to expert users for improving the quality of the final outcome. Thus, the system was implemented as an add-on for Blender. This integration allows professional users to directly manipulate the

scenes automatically generated by the system without the need for import/export operations. Using intuitive 3D interaction techniques, the users can also apply changes to the objects' layout within the immersive environment. The VR system adopted in this work is the HTC Vive. Lastly, the system tries to take some steps towards the integration of aspects characterizing affective VR environments into the scene generation process. The work was developed in collaboration with researchers at the University of Hong Kong. My contribution focused on the development of the system and in the execution of a user study that will be detailed in the following.

4.2.1 Related work

The possibility to automatically generate 3D scenes is not new. For example, the WordsEye system presented in [68] allows users to easily create a 3D scene from a text description by leveraging a database containing 3D models and poses. To this aim, the system first parses the input text, then analyzes the input from the semantic point of view, and finally identifies low-level descriptors (e.g., 3D objects, poses, spatial relations, color attributes, etc.) to be added/used to/in the scene. The final result is a static scene, in which key issues of semantics and graphical representation have been considered letting the users disregard additional concerns related to the generation of convincing animations. The framework in [277] was developed to support the generation of a 3D scene given a text description or an audio clip, by arranging 3D objects retrieved from a database. To define objects' placement, the proposed algorithm assumes that spatial relations (in, on, under, above, in front of, etc), together with a number of possible modifiers (to the left/left of, towards), have been included in the input text/audio. More recently, in [54], another system able to generate a 3D scene from a text description has been proposed. First, the system infers the objects to be placed in the 3D scene from the text. Then, in contrast to previous work, the most likely objects' layout is determined based on spatial arrangements previously used/proposed by the same system. Refinements applied by the user to adjust the obtained layout are considered by the system to improve next estimations. Another example is the ScenSeer system proposed in [56]. As with the work described above, the system infers the objects to be included in the scene by parsing a text description. However, in this case, the system leverages a spatial knowledge base (obtained by combining an existing database of 3D models and 3D scenes) to determine the objects' layout and identify further objects to be inserted even though they are not explicitly mentioned in the text. The user is allowed to add, remove, manipulate and replace objects in the scene by issuing other text commands.

All the methodologies presented so far leverage textual descriptions as input. However, for fast prototyping purposes, this approach does not represent the optimal solution, since writing the description of a scene could be tedious and time-consuming. Considering this aspect, the system described in the following will focus

on the use of images as input to recreate a 3D environment. The literature presents a number of approaches that tried to faithfully generate a 3D scene depicted in an image. For instance, the solution proposed in [313] is able to recognize perspective cues (such as perspective lines or distorted planes) in a single 2D image to create a 3D reconstruction (composed by flat textured planes representing walls, floor, and ceiling) of both indoor and outdoor environments. The work in [85] describes tools implemented for Matlab to build a 3D reconstruction of an environment from a set of calibrated images. Compared to [313], the resulting scene appears to be more realistic, since possible objects found in the environment are considered. However, the 3D scene generated is represented through a single mesh with a huge number of vertices. This aspect makes this approach (commonly referred to as photogrammetry) not suitable for fast prototyping. The possibility to recognize objects in the scene was also addressed in [246], where a neural network was used to identify and reconstruct two classes of objects: boxes and spheres. The network receives the 2D image as input, and the resulting output is a 3D textured VRML, X3D or WebGL file representing the recognized object. However, the limited set of recognized objects represents a limitation for general-purpose applications.

It is worth noting that all the work reviewed above proposed standalone tools, making the integration in the existing 3D graphics suites difficult to achieve. The literature provides a few examples of systems able to combine both the advantages of automatic 3D scene generation and functionalities offered by standard graphics suites. For instance, the framework presented in [197] converts a text description into a 3D scene that can be visualized in Autodesk Maya. In order to represent the common sense knowledge, i.e, common properties of an object, such as its name, the typical position and orientation in the scene, a knowledge base is used there. The result obtained by combing data in the knowledge base and information extracted by input text, is represented as XML file describing the 3D scene. The file can later be loaded in the considered graphics suite to visualize the scene.

Examples of work dealing with the automatic generation of immersive environments are presented in [290] and [289]. The system proposed in [290] leverages the real world as a template for modeling the virtual environment. To this purpose, the system first combines depth and color images representing the surrounding environment in order to generate a 3D map. Then, it identifies walkable areas and obstacles that are replaced by a corresponding virtual counterpart in VR. The virtual counterpart does not represent the exact object recognized in the real environment, but rather it is a different object which occupies the same bounding volume. To determine which objects have to be included in the scene, users are requested to specify a context like, e.g., scene settled on an island, in a volcano, in the space, etc. Afterwards, the system retrieves required objects from a set containing items related to the given context. The work in [289] describes a system called Auris, which automatically creates a VR environment from audio clips and lyrics provided as input. In particular, nouns are extracted from the lyrics through the Stanford

part-of-speech tagger [204]. Nouns are then used to influence the design of the global scene, the objects to be included, and their materials. The output generated by the system is a psychedelic and surreal environment that can be explored by the users within an immersive environment. A new aspect considered in this work is the possibility to make use of emotions to dynamically affect the lighting of the scene and the textures to be applied to objects. The general mood to be conveyed by the scene is determined by a neural network trained to distinguish between two classes (happy and sad moods) by using the Mel-frequency Cepstral Coefficients (MFCC) features extracted from the audio clips.

Considering the above review, a system was designed able to combine four main features:

- the automatic generation of a 3D scene from its 2D, image-based representation;
- the integration with a well-known graphics suite for editing purposes;
- the possibility to explore and manipulate the created scene in a virtual environment;
- the introduction of emotional aspects in the automatic scene generation process.

4.2.2 Proposed system

The overall architecture of the system introduced in [5] is represented in Fig. 4.1. The standard workflow begins with the definition of the input, i.e., the source image and the mood that the scene has to communicate, provided through the graphical user interface provided by the *Scene Creator add-on*. Then, the add-on is in charge of combining the input data and the results of Google Cloud Vision APIs, which are used to extract the context and objects in the image. Models to be included in the scene are gathered from a *Models database*. When the 3D scene is composed, it can be further manipulated in the well-known 3D computer graphics software Blender. Lastly, the user is allowed to explore the generated 3D scene into an immersive virtual environment and apply changes to the position, orientation and scale of objects through the *Virtual Reality add-on*.

Input

As mentioned, in the first step of the generation of the 3D scene, the user has to define the input for the system. The user must specify two main inputs: the parameters which set the emotion to conveyed by the scene and the source image to be used for inferring its contents. The mood to be conveyed can be specified through

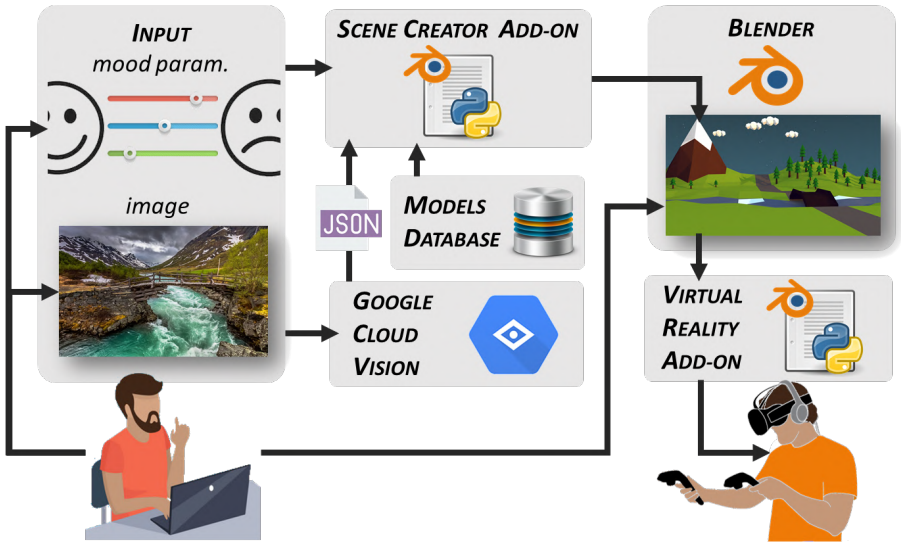


Figure 4.1: System architecture [5].

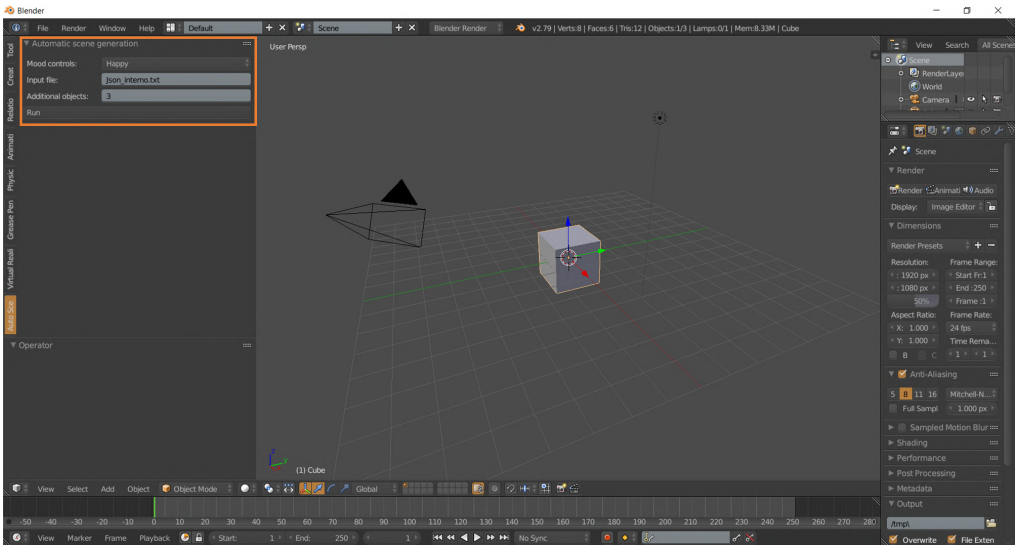


Figure 4.2: Automatic scene generation panel which is the interface of the Scene Creator add-on.

the Scene Creator add-on. This add-on is made up of two different parts: the back-end and the front-end. The back-end handles the logic behind the generation of the scene, whereas the front-end can be used to select the mood that the scene has to suggest by means of a panel named *Automatic scene generation* (shown in Fig. 4.2). After the installation of the Scene Creator add-on, the Automatic scene generation is automatically included in the Blender's Tool shelf, i.e, the set of panels, belonging to the native Blender's interface, placed on the left of the Blender's 3D View editor.

The Automatic scene generation panel shown in Fig. 4.2 is composed by the following controls.

- *Mood controls*: a combo-box that lets the user choose the emotion to be suggested. At present, the system supports two emotions: happiness, and sadness.
- *Input file*: a text-box that can be used to indicate which is the file to be considered as source image.
- *Additional objects*: a text-box for defining how many additional objects have to be inserted in the 3D scene in addition to those already recognized by the Google Cloud Vision APIs into the source image; more details about the type of objects additionally inserted into the scene will be provided in the following.
- *Run button*: the button that activates the process for the automatic generation of the 3D scene by running the functionalities of the back-end.

In order to extract the context from the source image and retrieve the objects to be inserted in the scene, the user provides as input an image to the Google Cloud Vision APIs.

Google Cloud Vision

The Google Cloud Vision APIs is a cloud-based library developed by Google to let developers take advantage of pre-trained machine learning models for image labeling and classification. With these APIs, it is possible to identify several elements in the source image, like objects, faces and texts. Furthermore, the APIs are able to provide developers with helpful metadata, built upon the elements recognized in the source image. The APIs are available from the Google Cloud Platform web page, section AI & Machine Learning Products[4]. A free trial of the APIs is offered on the web page, to test the basic functionalities through a dedicated web-based interface, without the need to install and activate any software. As reported in the online API documentation[5], the output of the Google Cloud Vision processing is a JSON file containing different fields. Data are structured as follows:

- *cropHintsAnnotation*: data containing the 2D coordinates of the corners for possible crops, i.e., regions of the image that can be removed since they could represent unwanted objects;

[4]Google Cloud Platform: https://cloud.google.com/vision/

[5]Google Vision API documentation: https://cloud.google.com/vision/docs/

- *imagePropertiesAnnotation*: attributes that specify the dominant colors found in the source image;
- *labelAnnotations*: list of labels that represent the broad categories (objects, locations, activities, products, etc.) to which objects identified in the source image belong to;
- *localizedObjectAnnotations*: list containing data representing general information for each object, e.g., the name, its position, and its bounding box, i.e., the 2D coordinates of the rectangular region that contains the object;
- *safeSearchAnnotation*: data that can be used to identify in the source image the presence of possible explicit contents, e.g. adult or violent contents;
- *webDetection*: web references, i.e., links, to web resources, which match with the contents of the source image.

The JSON file generated by the Google Cloud Vision APIs is used in the following steps as a descriptor for the source image. In particular, the two lists *labelAnnotations* and *localizedObjectAnnotations* are taken into account for generating the 3D scene, since they contain all the data needed for selecting the objects to be synthesized. In the future, other information contained in the JSON file could be considered to improve the results of the generation process. For instance, the dominant colors could be used to influence the materials assigned at the object in the scene, whereas web references could be used to get more insights about the source image.

Models database

In order to synthesize the scene, objects are retrieved from an existing database of 3D models. For each object, the database contains the 3D mesh (saved as Filmbox .FBX files), its materials, textures, and metadata. Metadata include the following information (as an example, possible values are reported for the object named “waste bin”):

- alternative names for the object (dustbin, garbage pail, trash bin, wastebasket);
- categories the object belongs to (park, garden, backyard);
- information to establish relations with other objects in the scene.

For the use cases that will be described more in detail in the following, a database containing 28 objects belonging to two specific environments (an indoor and an outdoor setting) was considered.

Scene Creator add-on

The back-end of the Scene Creator add-on was developed as a Python script for Blender. This script receives as input: the JSON file generated by the Google Cloud Vision APIs describing the source image and the parameters configured by the user to represent the mood. The content of the JSON file, as well as the parameters for the mood definition, are combined to generate the 3D scene. To this aim, an algorithm was implemented which determines the objects to be included and their position in the scene. The main steps of that algorithm can be summarized as follows.

1. The initial condition of the scene is restored by removing all the objects, materials, textures, and lights which may have been generated by a previous run of the algorithm. Moreover, all the parameters and variables used by the algorithm are set to their default values.

2. The JSON file is generated by submitting the source image to the Google Cloud Vision APIs for processing.

3. The JSON file generated in the previous step is parsed to extract the information contained in the *labelAnnotations* and *localizedObjectAnnotations* lists.

4. From data extracted by the *labelAnnotations* list, the system determines whether the image represents one of the two settings supported by the system, i.e., indoor or outdoor scene. In particular, if the list contains labels referring to the "home" context, like room, bedroom, living room, an indoor setting is assigned. For indoor settings, a predefined setup consisting of four walls, ceiling, and floor is automatically generated in the 3D scene. For outdoor settings (associated with labels like grass, park, garden, etc.), only a mesh representing the ground is added.

5. The *labelAnnotations* and *localizedObjectAnnotations* lists are parsed to identify a match between objects in the database and objects recognized in the source image. If the user increases the number of objects to be added in the scene by setting a value greater than zero in the Additional objects text-box, the algorithm tries to find additional matches between the object category (specified in the object metadata) and the labels in the *labelAnnotations* list.

6. All the objects to be added in the scene are organized in a graph-based structure. Each node corresponds to an object, whereas edges represent the spatial relations among objects. Relations are retrieved from the metadata contained in the Models database.

7. The graph is explored, and for each node (object), the algorithm defines its position in the 3D scene based to a set of rules; these rules avoid objects overlapping, and ensure that spatial relations and physical constraints are satisfied.

8. The current node is marked as explored, and the corresponding object is inserted in the scene.

9. Steps 7 and 8 are repeated until all the nodes are processed.

10. Lights are inserted and their parameters, as well as the materials and textures of all the objects belonging to the scene are configured according to the mood selected by the user (the influence of the mood will be described later).

11. The main camera for rendering the scene is added in a specific location for all the scenes generated by the algorithm, whereas its orientation is automatically adjusted in order to have at least one object in the field of view.

Blender

The output generated by the algorithm is a Blender scene that includes textured 3D objects, lights, and a camera. The scene is ready to be used and it does not require additional operations to be rendered. However, the user is allowed to further adjust the objects' layout and the textures applied to them by making use of Blender's native interface. For example, the user can manipulate existing objects, add new ones, modify their materials, assign new textures, configure a different camera, etc.

Virtual Reality add-on

Once the scene has been created, the user can explore it as an immersive environment by activating the VR mode. This modality is supported by the Virtual Reality add-on, a Python script developed for Blender. This add-on relies on the Virtual Reality Viewport library and the Python bindings for Valve's OpenVR SDK, named Pyopenvr. The former library was leveraged to visualize Blender's viewport in VR through a HMD. The functionalities offered by Pyopenvr were leveraged to gather the position/orientation of the Vive's controllers and the buttons' status. The above data are then used by the developed add-on to make the user interact with the virtual contents through the Vive's controllers. With the aim of making the coordinate system of Blender's 3D view aligned with the virtual environment, the center of the generated scene is considered as the origin of the virtual environment. By handling the tracked data gathered by the VR system, the user's movements in the physical, real space are transferred to the avatar's position in the virtual environment letting the user explore the 3D scene. Currently, the

system supports a one-to-one mapping, in which one Blender unit corresponds to one meter in the real world. The interactions with 3D objects passes through two stages, which are described below.

- Object selection: this step is performed by moving the virtual representation of the Vive's controllers (reconstructed in the VR environment by leveraging tracking data) close to the object to be selected, and pressing the Grip buttons of one the controllers.
- Object manipulation: this step allows the user to manipulate the position, orientation, and scale of the selected object by pressing the Trigger button and operating with the controllers. Once the Trigger button is released, the object preserves the last transformation applied with the controllers. In particular, position and orientation can be controlled by moving/rotating the controllers, whereas scaling is obtained by performing a 3D pinch gesture with both the controllers.

4.2.3 Spatial relations

One of the challenges to cope with was the lack of information that specifies the position of the objects in the scene. The use of an image as input for the generation process introduces the issue of inferring the spatial relations among objects placed in the scene. In contrast to previous work, in which the objects' relations were assumed to be already described in the text provided as input, the Google Cloud Vision APIs are not able to provide this type of information.

To infer spatial relations, it was decided to select 100 images representing indoor and outdoor settings by searching them on the web using specific keywords. Then, each image was submitted to the Google Cloud Vision APIs processing to retrieve the objects. Lastly, for each object extracted from the image, at least one spatial relation was manually associated, by selecting it from a predefined set. This set was composed by specific relations defined in the text-based scene generation system proposed in [54]. The set includes the following relations: left, right, above, below, front, back, on top of, next to, near, inside, and outside. The possibility to generate realistic 3D scenes by using this high-level descriptive information was already demonstrated in [183, 277]. The outcome of the process described above is the list of objects recognized in these environments and their possible relations. These data were considered during the design of the Models database. In particular, all the recognized objects have been included in the database and, for each object, a new entry was added to the metadata to indicate the probability of finding a specific spatial relation with another object in the database. When the two objects are recognized in the input image, the most recurrent relation is assigned. For example, by analyzing the 100 images related to the indoor setting, it was found that for the "lamp" and "nightstand" objects there was a high probability to observe

the relation "the lamp is on top of the nightstand" (and viceversa). Therefore, if a lamp and a nightstand are recognized in the source image, the above relations is set for the two objects.

The metadata also contain information regarding the bounding boxes that represent, for each possible relation observed, the 3D regions where related objects have to be located. For example, in Fig. 4.3 the spatial regions defined for the nightstand object are shows. In order to define the position of the lamp (that has to be) on top of the nightstand (step 7 described in Section 4.2.2), the algorithm first gets the bounding box of the nightstand representing the "on top of" relation. Then, the system generates a random 3D coordinate into the given region and assigns that coordinate to the lamp object. Finally, the z coordinate which represents the vertical position of the lamp is modified using the algorithm proposed in [323] to respect physical constraints and prevent the presence of a floating lamp over the nightstand. If no relations are defined, the object is randomly positioned in the scene by respecting only the physical constraints.

As with in previous work ([323]), the proposed system supports hierarchical relations among the objects. For example, for the relation "the lamp is on top of the nightstand", the nightstand could represent the parent node whereas the lamp can be regarded as a child node. The system preserves this hierarchy also in Blender's data structure, in order to transfer further transformations possibly applied by the user from the parent object to its children, as common in many modeling tools.

Alternatives of the scene are generated even if the user runs the algorithm by providing every time as input the same source image, number of additional objects and mood. This aspect is mainly related to the randomness in the placement of objects in the assigned regions. Differences in the generated scenes can be introduced also by the fact that setting a number of additional objects greater than zero makes the algorithm randomly choose the objects to be inserted by selecting them from a set of possible alternatives that match the scene context. For example, the three scenes shown in Fig. 4.4 were created with the same source image and by setting the value of additional objects equal to three. The Google Cloud Vision APIs recognized into the given source image the bed, the mirror, the cabinet, the cabinetry and the nightstand. These objects were included in all the scenes generated by the system. Other objects, such as the lamp, the carpet, the library, etc, were automatically included in the scenes as additional objects.

4.2.4 Mood influence

As previously mentioned, the system in [5] supports two opposite emotions, namely happiness and sadness. The definition of the affective aspects that can be used to convey emotions leverages the main outcomes of prior work, namely [289] and [166]. According to this work, a scene conveys a happy mood when lights

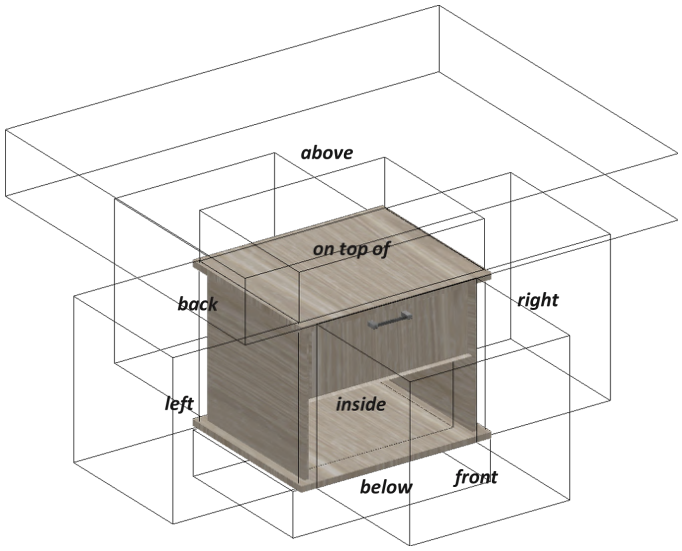


Figure 4.3: Spatial regions defined for the nightstand object.

intensity is very high, objects' materials have very light, pure and saturated colors, and textures present curved shapes or patterns with a high number of small rounded elements (e.g. particles). Conversely, a sad scene presents low intensity lights, contains few objects, and features materials with dark tones and low saturation as well as textures with dot patterns. Considering the above observations, a mapping can be created between emotions and the various parameters that control the scene. The algorithm presented in Section 4.2.2 is able to handle lights, together with objects' materials and textures. In particular, when a mood is set, specific materials and textures containing the requested features are automatically assigned to each object by selecting them from a predefined set of available options. Furthermore, the number, and the intensity of lights in the scene, are configured to match the mood selected. As said, Fig. 4.4 presents three different scenes generated from the same source image: it is possible to observe differences in terms of materials, image textures and lights introduced by the selection of a happy (Fig. 4.4b, Fig. 4.4c) and sad (Fig. 4.4d) mood, respectively.

4.2.5 Use cases

In order to present results that can be achieved by using the algorithm proposed in Section 4.2.2, two use cases were designed. In the first use case, an indoor environment was considered. As illustrated in Fig. 4.5a, the source image represents a bedroom with a number of furniture elements. By parsing the *localizedObjectAnnotations* list, generated by the Google Cloud Vision APIs, five objects (highlighted with a blue bounding box in Fig. 4.5a) were recognized: mirror, bed, cabinet, cabinetry and nightstand. These objects were automatically included in the 3D scene as

(a) (b)

(c) (d)

Figure 4.4: Scenes generated from the same (a) source image, including a bed, a mirror, a cabinet, a cabinetry and a nightstand, and by setting the number of additional objects to three. Results obtained by setting (b, c) happy mood and (d) sad mood for the same source image.

highlighted in Fig. 4.5b. Since the value of Additional objects field was greater than zero (it was set to two), the *labelAnnotations* list was also considered to increase the number of objects in the 3D scene. Categories identified in this list refer to the following label: bedroom, furniture, room, bed frame, bed sheet, interior design, wood, hardwood, floor. Considering these categories, two objects (the small table and the carpet highlighted in Fig. 4.5b with a red bounding box) were added to the 3D scene. Spatial relations were retrieved from the objects' metadata. For example, it is possible to observe the following relations: the mirror on top of the cabinet, the nightstand to the right of the bed, the carpet near the bed, etc. As said, the mood to be conveyed influences the objects' materials and textures, as

well as the intensity of lights. For example, in Fig. 4.5b it is possible to notice the very bright hue assigned to the walls' color or the texture of bedding presenting circular shapes.

In the second use case, the source image represents an outdoor environment with a picnic table on a garden (Fig. 4.5c). From this source image, the Google Cloud Vision API extracted only the bench object highlighted with a blue bounding box. However, the *labelAnnotations* list contained ten categories referred to the following labels: picnic table, outdoor furniture, outdoor bench, tree, outdoor table, leisure, picnic, recreation, park, sunlounger. From these categories, it was possible to identify and add to the scene (for a number of additional objects set to four) the objects that are shown in Fig. 4.5d with red bounding boxes, i.e., tree, picnic table, carpet, and food. The scene was created with a sad mood, as suggested by the low-intensity lighting and the textures with dark tones.

It is worth observing that from the analysis of many images representing indoor and outdoor settings, it appears that the number of objects recognized by the Google Cloud Vision API is generally higher when the source image represents an indoor than an outdoor setting. Moving from this consideration, the use of the *labelAnnotation* list becomes necessary in order to obtain a rich scene for the outdoor settings. Thus, in order to include more objects from the *labelAnnotation* list, the number of additional objects should be higher for outdoor settings than for indoor settings. However, the resulting 3D scene may differ more from the source image since a higher number of additional objects could introduce elements that might not be contained at all in the source image.

A video showing the use of the system is available for download[6].

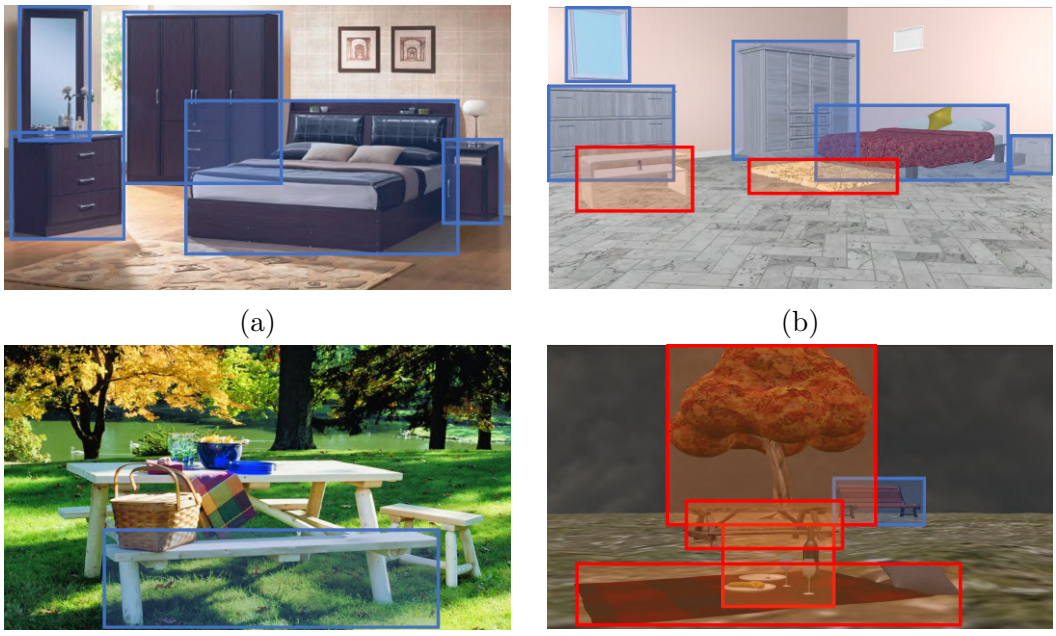
4.2.6 Experimental evaluation

To evaluate the usability of the automatic scene generation system, a user study was carried out by involving 12 participants (six males and six females), aged between 21 and 32 ($M = 24.81$ and $SD = 3.32$) who were selected among students and academic staff at the University of Hong Kong. All the participants were considered as non-expert users, since they had low expertise in the use of computer graphics suites and VR systems.

Procedure

At the beginning of the study, participants were introduced to the overall procedure to be carried out during the experiment. Then, they were requested to

[6]Video of the system: https://bit.ly/2moaIVT



(a) (b)

(c) (d)

Figure 4.5: Considered use cases; (a) indoor source image (source: https://bit.ly/2kW2HHh) containing five objects recognized by the Google Cloud Vision APIs (blue bounding boxes), (b) result obtained with the happy mood and Additional objects (red bounding boxes) set to two, (c) outdoor source image (source: https://bit.ly/330wm6O) containing one object (blue bounding box), and (d) result obtained with the sad mood and Additional objects (red bounding boxes) set to four.

complete a consent form and a demographic questionnaire designed for evaluating their previous experience. The experiment consisted of two tasks designed to evaluate different aspects of the proposed system.

The first task (later referred to as T_1) was designed to evaluate the system's usability. Considering this aim, each participant was first provided with basic instructions that allowed him or her to use the system for generating a 3D scene. Then, the participant was asked to generate a scene from scratch by choosing a source image (selected from a predefined set of available images), setting the number of additional objects and the mood to be suggested by the scene. No restrictions were imposed on the number of trials as well as input parameters to be set. The participant was left free to execute the task at his or her own pace, as no time limit was defined.

The goal of the second task (T_2) was to widen the analysis of the system by taking into account three different aspects:

- the similarity between the source image and the generated 3D scene;

- the match between the mood set and the conveyed one;
- the spatial awareness.

In the second task, participants were asked to explore four scenes within an immersive environment generated by the proposed system. The considered scenes were generated by using the same number of additional objects and by defining different source images and moods as input. The first two scenes (later referred to as *Indoor setting #1* and *Indoor setting #2*) are shown in Fig. 4.6a and Fig. 4.6b. They represent two indoor settings created by setting the happy and sad mood, respectively. The other two scenes, named *Outdoor setting#1* and *Outdoor setting #2* are shown in Fig. 4.6c and Fig. 4.6d. They represent outdoor settings generated to convey a sad and happy mood, respectively. As suggested in [289], the exploration was considered fulfilled when the participant spent at least 30 seconds for each scenario in the immersive environment (average time among all the participants was 57 seconds). Latin Order was used to choose the scenario to start and continue with, in order to reduce possible biases related to learning effects. At the end of each task, a post-test questionnaire was delivered to participants in order to evaluate the specific aspects mentioned above.

Metrics

The evaluation of the proposed system involved both objective and subjective measurements. The objective measurements leveraged two indicators: completion time, and object placement. The first indicator, collected in T_1, considers the time spent by a participant to create a scene, and it was used to estimate the effectiveness of the devised system. The object placement indicator was collected only in T_2. As in [295], this indicator measures the capacity of the participants to memorize where objects were located within the scene, thus providing information about the users' spatial awareness provided by the VR environment. In order to measure this indicator, participants were requested to specify on a map representing the VR environment just experienced which objects they observed in five specific positions. Fig. 4.7 shows the five positions considered for all the scenarios. The indicator was then calculated as the percentage of objects correctly recognized in their correct position.

Subjective observations were collected by asking users to complete a questionnaire at the end of each task. The questionnaire (available for download[7]) included two sections. The first section (administrated after T_1) measures the usability of the system according to the SUS scale [40]. Questions were expressed in the form of statements to be evaluated on a 1-to-5 scale (from strongly disagree to strongly

[7]Questionnaire: https://bit.ly/2lYzFHk

(a) Indoor setting #1

(b) Indoor setting #2

(c) Outdoor setting #1

(d) Outdoor setting #2

Figure 4.6: Scenarios considered in task T_2.

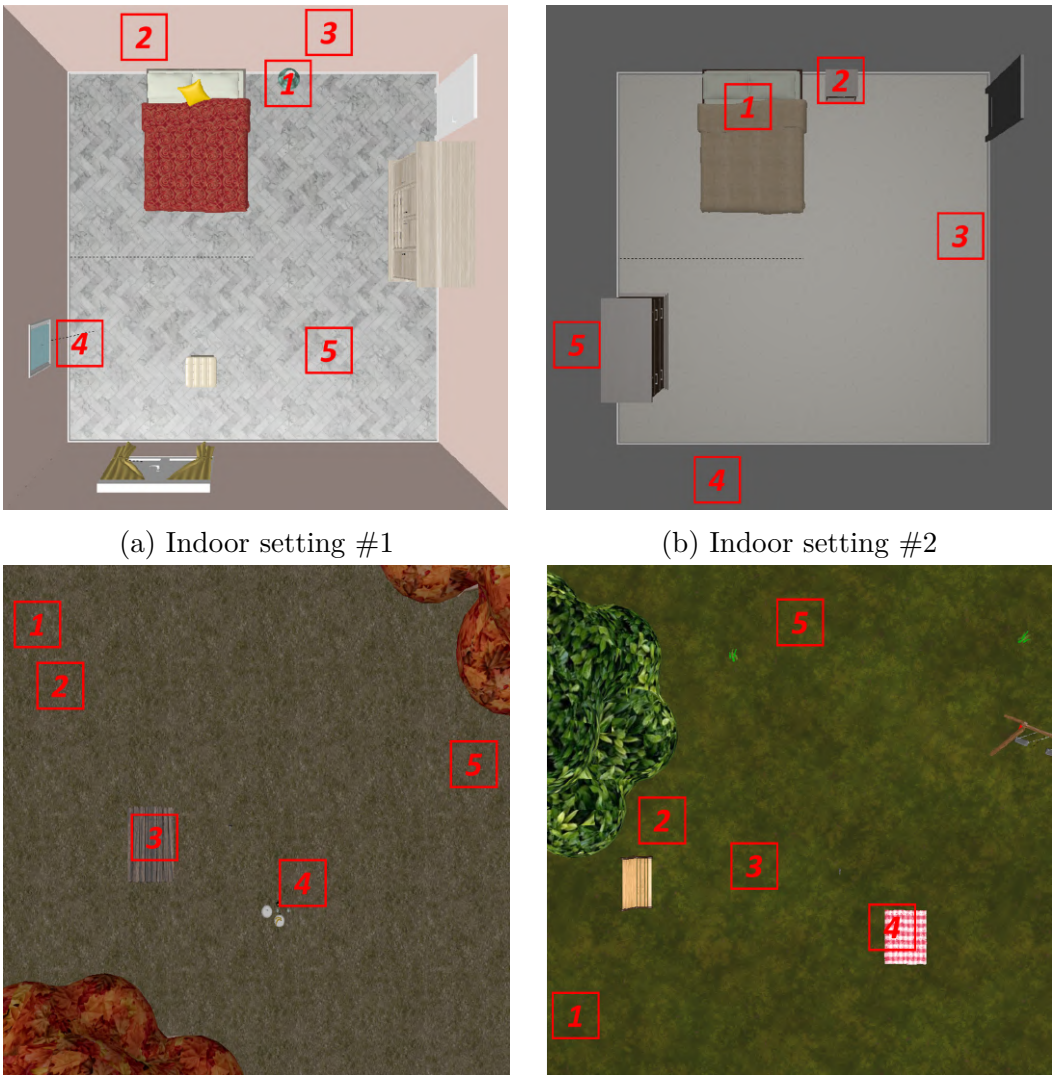
agree). The second section (filled in after the execution of T_2) invited users to rate the similarity between the source images and the generated 3D scenes by specifying a score in a 1-to-5 scale like in [55]. Finally, as proposed in [289], participants were requested to evaluate the emotion conveyed by each scenario by providing a score in a range from 1 (very sad) to 5 (very happy).

4.2.7 Results

Data extracted from the demographic questionnaire completed at the beginning of the study showed a low expertise of participants in using graphics suites and VR technology. In particular, considering the use of graphics suites, 36.4% of the participants stated that they never used these suites, 45.5% used them sometimes, 18.2% once a month or once a week. With respect to VR, 45.5% of the participants never used VR systems, 54.5% use them sometimes or once a month. In the following, the results of objective and subjective measurements collected at end of each task will be presented.

Objective measurements

Starting from T_1, on average participants spent slightly more than 70 seconds each to generate a 3D scene from scratch ($M = 71.08$ s and $SD = 35.10$). Regarding



(a) Indoor setting #1

(b) Indoor setting #2

(c) Outdoor setting #1

(d) Outdoor setting #2

Figure 4.7: Maps showing the five positions considered for evaluating users' spatial awareness in the four scenes.

T_2, after having explored the 3D scene in the immersive environment participants were able to correctly remember the objects' layout, since, on average, the percentage of objects correctly recognized was quite high ($M = 72.5\%$, $SD = 0.15$). This result could be related to an improved spatial awareness guaranteed by VR technology.

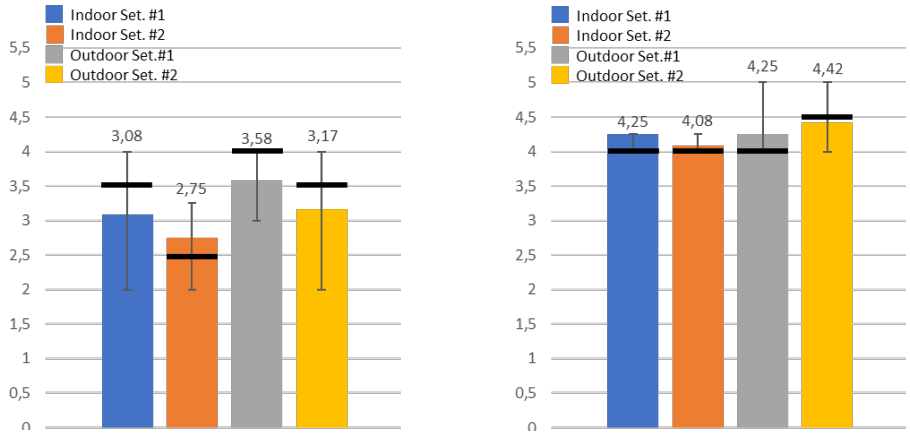
Subjective measurements

After T_1, the usability of the proposed system was measured by leveraging the SUS scale. Overall, participants found the system as characterized by a high usability, confirmed by the SUS score which was equal to 77.92. According to [40], such a score corresponds to grade B in the SUS scale (adjective rating equal to "*Good*"). The average scores assigned to each statement are reported in Table 4.1.

Table 4.1: Subjective results concerning usability based on SUS [40].

Statement	**Score**
I think that I would like to use this system frequently.	3.42
I found the system unnecessarily complex.	2.00
I thought the system was easy to use.	3.92
I think that I would need the support of a technical person to be able to use this system.	1.42
I found the various functions in this system were well integrated.	3.75
I thought there was too much inconsistency in this system.	1.58
I would imagine that most people would learn to use this system very quickly.	4.33
I found the system very cumbersome to use.	2.08
I felt very confident using the system.	4.08
I needed to learn a lot of things before I could get going with this system.	1.25

Fig. 4.8 shows the average scores (bars heights), medians (black lines) and quartiles (errors bars) obtained by analyzing ratings provided by the users for what it concerns scene similarity and mood perception in the four settings. Scores of mood perception reported for the Indoor setting #2 and the Outdoor setting #1 have been inverted, in order to remap values on a scale in which higher values indicate a higher similarity with the mood conveyed by the generated scene. Participants, on average, perceived the generated scenes as a good representation of the source images, as confirmed by values which are close to 3.14 (representing neutral similarity) With respect to mood perception, the scores (which are in general higher than 4.0) confirmed the capability of the system to convey different emotions in the four considered settings.



(a) Scene similarity (b) Mood perception

Figure 4.8: Subjective results.

4.2.8 Future developments

At present, the system is characterized by several limitations concerning flexibility, i.e., possible scenes that can be created. This aspect is mainly related to the limited number of objects and related annotations (for defining the spatial relations) available in the Model database. Future work could consider the possibility of using alternative techniques based, e.g., on machine learning, to identify the spatial relations among objects directly in the source images. Possible evolutions could be focused on increasing the number of 3D objects in the database, as well as on considering different settings. Moreover, future efforts could be devoted to compare the proposed system with related work, as well as with traditional software, by considering additional/alternative objective and subjective measurements in the evaluation. Finally, other data extracted from the image by the Google Cloud Vision APIs could be considered, e.g., the dominant colors of the image or the web references, with the aim to improve the similarity between the source images and the reconstructed 3D scenes.

4.3 Augmented reality for constructive art

The previous discussion devoted to the construction of the 3D scene presented possible concerns related to the manipulation of 3D objects through traditional 2D interfaces. Even the domain referred to as constructive art presents similar issues. In fact, in this type of application, the users are requested to manipulate/interact with 3D components in order to build artworks. New solutions based on different technologies as well as alternative interaction paradigms have been proposed to address the above limitation.

Among the most promising approaches are those leveraging AR technology. In fact, today AR is becoming extremely common for many applications, ranging from magazines' augmentation (e.g., SPAM Magazine[8]), to advertising (e.g., greenpub[9]), online catalogues (e.g., IKEA[10]) and art exhibitions (e.g., Aidan Gallery[11]), to name a few.

For many years, the use of AR has been limited to a few domains, since the low performance of the primitive handled devices named Personal Digital Assistants (PDAs) that were used to run AR applications had a huge impact on user experience, and their usage required the availability of a server to perform computationally intensive operations. [256]. Today, due to the great technical advancements of electronic devices, it is possible to run standard AR applications on common (and less expensive) mobile devices that ensure high performance and provide heterogeneous data gathered by the numerous sensors, like RGB cameras, gyroscopes, accelerometers, and touch screen.

With respect to the domain of constructive art, usually, the interaction scheme used for AR applications includes the following steps:

- a tag/marker associated with a given virtual scene/object is placed in the real world;
- the system superimposes virtual contents over the tag/marker framed by the camera;
- the artist manipulates the 3D objects in order to assemble the artwork.

It can be noticed that an intuitive interface is needed to make the artist focus mainly on the creative process rather than addressing issues related to the use of the interface. The main concern regarding the manipulation of 3D contents through AR techniques usually refers to the approach adopted for making the user control all the DOFs needed for translating/rotating/scaling 3D objects in the scene as well as configuring specific parameters. This task could become even more complex when the user performs the interactions on the same device used to display the scene.

Several interfaces have been proposed for 3D object manipulation in virtual scenes [250]. However, such work has mostly focused only on the objects positioning, disregarding methods that can be used to configure specific parameters of the object itself.

[8]SPAM Magazine: http://spam-magazine.com/whatis.htm

[9]greenpub: https://www.augment.com/portfolio-items/greenpub/

[10]IKEA Place: https://bit.ly/2UAeoBx

[11]Aidan Gallery: http://www.aidangallery.com/

Moving from these considerations, two different interfaces have been proposed to allow artists to perform two interaction tasks commonly executed in constructive art applications. In particular, the first task was aimed to create a 3D sculpture by manipulating/interacting with a number of virtual assets. The second tasks regarded a scenario in which a set of particle systems needs to be configured in order to produce a given visual effect. The two interfaces were developed as AR applications for mobile devices and explored two alternative approaches for performing the above tasks.

4.3.1 T4T

The idea, originally presented in [46] and referred to with the acronym T4T (Tangible interface 4 Tuning 3D object manipulation tools), was to make use of a customizable 3DUI allowing the users to choose the way to manipulate object parameters in the applications of interest. In particular, the users can take advantage of a TUI (built by using common objects) that can be used as a control knob for finely tuning the above parameters. If the parameter to be controlled includes object positioning, the knob manipulation enables the users to achieve more precision than by using only the mobile device; otherwise, if the parameter concerns the particle systems manipulation, the knob provides the users with a more physical way for operating.

Related work

This section focuses on prior work that enable fine control of the objects' parameters, by removing the need to directly manipulate the mobile device used to view the virtual scene. Using handheld devices as user input for manipulating virtual objects has been largely studied in the past, and several methodologies have been proposed already. For instance, the work in [207] presented a technique based on a viewpoint cursor to convert 2D input into 3D interactions to be used within an immersive spatial AR environment. The idea behind this technique is similar to the approach used, e.g., to highlight projected contents in a conference through a laser pointer. The difference with respect to this approach is that, in this case, the viewpoint cursor is strictly bound to the user's field of view. This solution, referred to as cursor-based manipulation, has been longly considered a commonly accepted mechanism for manipulating objects in virtual scenes. However, it could present limitations in terms of accuracy, since an accurate object manipulation requires fine control of the handheld device, that may be difficult to achieve, e.g., because of hand(s) vibrations. Moreover, this technique could hardly let the users control other object's parameters or particle systems.

A possible way to cope with these requirements could be represented by approaches able to combine the cursor-based manipulation with other interaction

techniques. For example, the solution in [33] proposed to map the object parameters to different hand gestures. Although this solution let the users achieve more accurate result, it presents severe limitations in term of scalability, since a huge number of gestures to remember could translate into a high mental workload for the users and/or poor intuitiveness. A possibility to limit this issue could be the use of TUIs [154]. A physical prop may offer the same level of flexibility reached by the previous solution. In fact, the mapping between tangible manipulation and object parameters could be configured by the user. Furthermore, the affordances offered by the use of physical interaction means could be leveraged to obtain higher accuracy.

Considering the advantages and disadvantages of the solutions described above, the proposed interface integrated the well-known cursor-based interaction mode with a tuning mode implemented by using a TUI. The tangible prop can be configured to control both objects' manipulation and desired parameters. By leveraging the proposed solution that combines the use of an external interaction mechanism with the cursor-based interaction mode, the user is allowed to operate on virtual objects without the need to manipulate the device that is being used to display the virtual scene.

Architecture

The overall architecture of the proposed interface is represented in Fig. 4.9. The target devices for the proposed interface are handheld devices equipped with standard components (i.e., touch screen, accelerometers and gyroscopes). In particular, the interface has been developed as an Android application for tablet devices by leveraging the well-know Unity game engine[12]. The tablet display can be used to visualize the image stream captured in real-time by the camera augmented with virtual assets. A marker placed in the real environment is use to determine the origin of the virtual scene in which virtual assets will be added. The Vuforia extension for Unity is used to maintain stable the tracking of the marker. The tangible prop is created by placing additional markers on top of any common object, e.g., a paper glass.

Interaction approach

The interaction passes through three phases, referred to as *marker tracking*, *cursor mode*, and *tuning mode*. In the first phase, the user has to frame the marker with the handheld device's camera. When the virtual scene is generated and assets are added to the current frame, the system automatically moves to the cursor mode.

[12]Unity: https://unity.com/

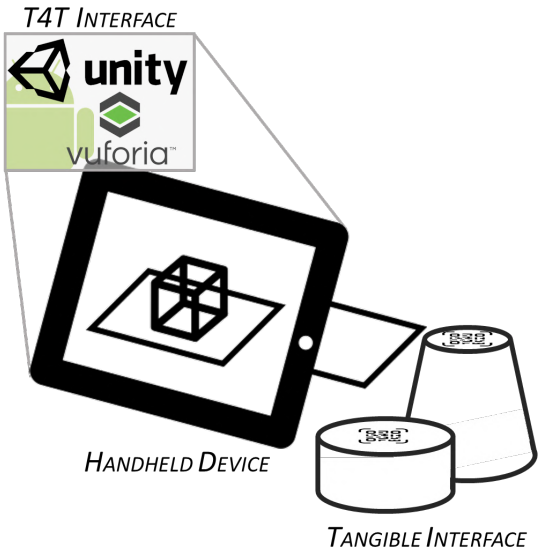


Figure 4.9: Architecture of the T4T interface [46].

The cursor is represented as a circle shown on the screen that can be used to interact with the objects in the virtual environment. To move the cursor, the user is requested to grab the tablet in a new position or rotate it. A raytracing technique lets the user select a particular object in the scene. The position of the cursor is determined by the intersection between a ray emitted from the center of the camera view which is perpendicular to the camera plane and the first object hit by the ray. A floor plane has been included in the scene in order to visualize the cursor when intersections with objects in the scene are not found. In order to select an object, the user has to keep the cursor on the object's surface for a certain amount of time. Conversely, shaking the device deselects all the objects. Once an object is selected, a pop-up menu appears showing available functionalities (Fig. 4.10). The interface supports six functionalities: two targeted to the manipulation of translations, three for controlling the rotations, and the last one for handling uniform scaling. With respect to the translation, the two functionalities allow the user to move the selected object on the horizontal plane ("Move on plane") or the vertical axis ("Move on Z") just by moving the device. Rotation is accomplished by performing the common two-finger rotation gesture on the tablet display. Similarly, scaling is performed with a pinch gesture.

If an object is selected, the user can activate the tuning mode for enabling a fine-tune manipulation of the parameters. The available functionalities offered by this modality are shown in the pop-up menu displayed in Fig. 4.11). Manipulation starts when a tangible prop is introduced by the user in the field of view of the device's camera. Parameters can be interactively modified by mapping the tracked rotation of the prop to the selected function. The current value of the manipulated

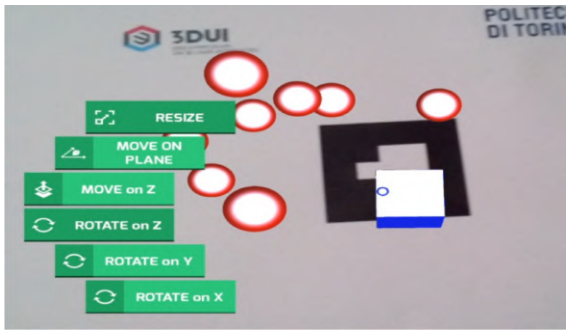


Figure 4.10: The T4T interface in cursor mode. The object detected by the cursor with raytracing is selected, and can be manipulated by moving the device as well as using finger gestures.

Figure 4.11: The T4T interface in tuning mode. Parameters of selected object can be tuned by manipulating the tangible prop acting as a control knob.

parameter is represented within a circle, on top of the prop, thus giving continuous feedback to the user. A custom shaped progress bar in the form of a rim surrounding the circle is used as a graphical indicator of the same quantity.

The sensitivity of the tangible manipulation for each parameter can be adjusted by performing a long-tap on the corresponding menu item and configuring the desired value with a slider. For example, uniform scaling operation on a specific object can be performed through the following step:

- selecting the object of interest with the cursor;
- tapping on the scaling function in the menu;
- rotating the tangible prop until the scale factor of the object reached the desired value.

Some examples of available functionalities can be obtained by considering the selected study cases (creating a 3D sculpture and manipulating a particle system). In both the scenarios, the tangible interface can be used for translation and rotation

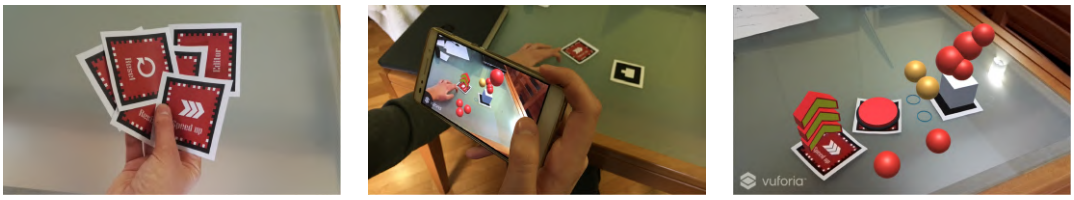
around each axis and to perform uniform scaling. In the first scenario, shown in Fig. 4.10, the capabilities of the interface are leveraged to scale objects on each separate axis. The use of the proposed interface applied to the second use case is illustrated in Fig. 4.11. It can be specifically observed the possibility provided by the interface to tune particle system's parameters, such as particle size, emission rate, speed, etc.

4.3.2 HOT

The goal of the HOT solution (acronym of Hold your Own Tools for AR-based constructive art) proposed in [14] was to make the user perceive the coexistence between real and virtual worlds. To this purpose, the interface provides a minimal workspace removing all the always-visible (overlapped) menus, which could "digitize" too much the view. Functionalities of the interface are moved into a collection of tool tags (Fig. 4.12a), that, once framed by the camera of the mobile device (Fig. 4.12b), allow the user to activate a given function. Each 2D tool tag has a 3D virtual counterpart (Fig. 4.12c) designed to suggest the provided function. The handheld device allows the user to interact with the 3D tool added to the scene in order to activate and use the associated functionality. Tags are collected in a portable deck, that can be expanded to support new tools/functionalities. The manipulation of virtual objects in six DOFs relies on the device's multi-touch capabilities.

Related work

In this section, attention is devoted to methodologies based on TUIs and multi-touch devices for 3D object manipulation, which promise to reduce the distance between the real and virtual world. The solution proposed in [32] leverages a tangible AR interface for interacting with virtual objects through a set of tags with embedded functionalities. Each functionality is activated by physically placing the tags in the real world. The authors of [189] defined a set of interaction "modes" that allow the user to control the six DOFs by leveraging the technique of "directness-independence". This technique consists in using the user's fingers that are not necessary on the object being manipulated. The proposed solution combines the two approaches reviewed so far, but also introduces new aspects to make the interaction easier. In particular, as in [32], a set of tool tags are used, but the need to physically manipulate them is removed, thus preserving the essential role played by the device's screen. The concept of directness-independence is leveraged to let the user manipulate objects that could be inconvenient to interact with since they are, e.g., too small with respect to the device's display. The concept of interaction modes presented in [189] was also preserved, even though the way to activate modes was changed. In particular, in [189] modes were selected using finger gestures; in



(a) (b) (c)

Figure 4.12: (a) A deck of tool tags; each tag can be used to access one or more control functionalities. (b) A tag is inserted in the (real and virtual) workspace. (c) Tags are accompanied by actionable virtual counterparts in the augmented environment, which evoke associated functionalities.

the latter work, modes are activated based on the number of fingers touching the screen and their state (stationary or moving).

Tag toolkit

The overall architecture of the proposed interface is illustrated in Fig. 4.13. As with the interface proposed in [46], the interface in [14] was developed as an Android application for tablet devices. The software was implemented by leveraging Unity and the Vuforia library. A marker placed in the real environment is used as a reference system to build the artworks. In addition to this tag, used as "pedestal" for the artwork, the devised solution leverages a set of supplementary tool tags, which can be used to introduce in the virtual scene ad-hoc tools for creative operations.

In order to demonstrate the effectiveness of this approach, a set containing four different tags has been considered. Tool tags can be used for the following task:

- modify the manipulation speed of selected objects;
- apply 3D scale transformations;
- control objects' parameters;
- reset the virtual scene.

Once a tool tag enters in the camera's field of view, the associated functionality is enabled and its virtual representation rendered: as in [32], textual cues on the printed tag and on the 3D geometry help the user to recall the corresponding functionality.

Once the virtual tool is displayed, the user can start to operate with the device to actually "realize"/"leverage" the functionality associated to that particular tool tag. When the user interacts with the tool, its virtual representation changes state, to provide immediate feedback. When the tag exits the camera's field of view, its virtual representation disappears. For example, if the user needs to accurately

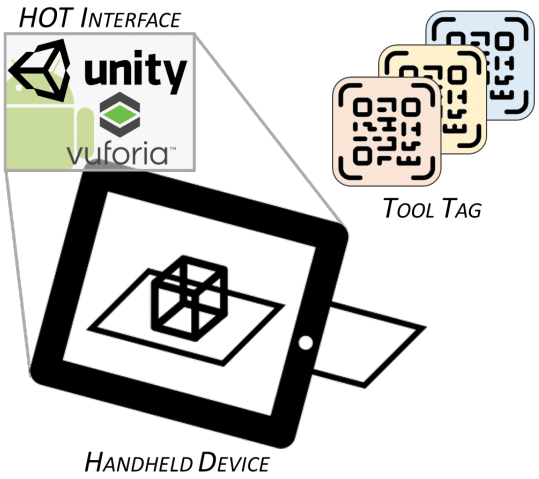


Figure 4.13: Architecture of the HOT interface [14].

manipulate the position and orientation of an object, he or she can use the speed tag (Fig. 4.12b). The tool is displayed on the screen as a graduated arrow. By tapping on it, objects' manipulation modality changes to discrete steps. Similarly, the tag tool for scaling is composed by three arrows: a slide gesture on a given arrow applies a scale transformation to the object along the direction pointed by that arrow. The tool tag devised to control objects' parameters is represented as a 3D menu in the augmented scene. This way, it is possible to avoid a menu with a long list of items, since new items corresponding to new sub-menu elements can be developed and added to the set of available tags. Lastly, the fourth tool tag can be used to refresh/restart the scene. The assigned functionality is activated by pressing the button associated with this tag. As said, in the future the number of tool tags recognized by the system could be increased. For instance, a new tag could be added to implement a sort of database, to store groups of objects. Another tag could be introduced to make it possible to manipulate the objects' materials and textures.

Object manipulation

The design of the proposed interface enabling the manipulation of 3D objects considered the following features.

1. Selection mode: an object can present two states, i.e., selected or not selected. Interactions performed by the user influence all the selected objects simultaneously. Multiple objects manipulations are allowed. The approach based on raycasting [220] is adopted for selecting objects. When the user touches any

other point of the screen, all the objects are deselected.

2. Feedback elements: visual cues were introduced in the virtual scene in order to make the interaction with the system for building artworks more effective. For example, when an object is added to the current selection, its colors slightly change and a new shape is rendered on the working surface to simulate a shadow. This shadow can be intended as a visual cue to inform the user about the objects' height above the plane and their depth with respect to the camera's view. In order to match the tactile feedback provided by the real world, virtual objects cannot pass through the working surface (i.e., the tag used as pedestal for the artwork). Audio feedback is also associated to the selection process. Lastly, collisions among virtual objects are indicated through short vibrations of the device, aimed to simulate real impacts.

3. Directness-independence: according to [189], the techniques adopted for interacting with the system can be classified as indirect, since it is possible to manipulate objects without touching them. Differently than with direct techniques, the choice made allows the interface to properly support different screen sizes and ways to hold the device.

4. Interaction modes: four different interaction modalities have been developed, which are activated depending on the number of fingers touching the screen and their current state. Table 4.2 shows the devised modalities, characterized by different numbers of fixed (f) and moving (m) fingers and gestures. These modalities ensure a full control of translations and rotations on six DOFs. One additional mode is considered in order to perform uniform scaling (complementing the scale tag).

Axes used as a reference system for the translation can be interpreted as follows: the y axis is orthogonal to the working plane, whereas the x and z axes are parallel to it.

In order to adapt the translation over the xz-plane to the user's point of view, axes are dynamically evaluated at each frame. In particular, the x and z axes are rotated with respect to the original pose, following the camera rotation, around the y axis. Rotation axes adapt to a non-static viewpoint as well. They are updated at each frame to follow camera's local axis.

4.4 Virtual reality for sport training

The last decade has been characterized by the progressive introduction of new technologies to support different activities related to the sport practice. Examples can be found in various sports, from basketball, to soccer, football, etc. More and

Table 4.2: Interaction modes.

Mode	Interaction	Effect
1m	Pan	Translation (x,z)
2m	Pinch	Translation (y)
1f + 1 m	Stick + Pan	Rotation (x,y)
1f	Horizontal Tilt	Rotation (z)
1f + 2m	Stick + Pinch	Scale

more researchers are devoting attention to methodologies to improve the effectiveness and efficiency also of more traditional (less technological) domains [53, 268, 309].

One of the most promising technology explored so far is VR, which seems to be capable of bringing significant changes to the way a number of sport-related tasks are carried out [192]. Many solutions have been already proposed to take advantage of VR in sports, e.g., to improve the players' performance [53], to enhance their action-reaction skills [94], and to support psychological training used to mentally prepare them for an important event [286], but also to let common users experience extreme sports [89].

One of the most promising fields in which VR is becoming a core technology is the analysis and planning of game tactics [106]. In the training sessions focused on tactic analysis, players are requested to memorize a lot of information, regarding, for instance, the positions to keep in the court, the movements to perform when a particular situation occurs, etc. For this reason, methodologies able to make these training sessions more effective are becoming of paramount importance.

The system that was originally presented in [50] and will be discussed in the following addresses precisely this aspect, by proposing a VR solution targeted to basketball which allows coaches to create/define all the 3D graphics assets (3D models, animations, etc) needed for representing tactics and to visualize them into an immersive environment. This work was made in collaboration with the technical manager of the Auxilium CUS basketball Torino's youth sector, who provided important advice on sport-related aspects.

4.4.1 A participative system for tactics analysis in sport training based on immersive virtual reality

The possible benefits brought by the use of VR for sport training are confirmed by the huge number of prior work in the literature targeted to various sports and different training phases [109]. Regarding training sessions for tactics analysis, they ordinarily involve the use of physical books containing illustrations of the players' movements as well as of paper or tactics boards where coaches draw actions [309].

However, these representations are generally too abstract. Players, especially the youngest ones, may find it difficult to mentally recreate the exact situation in which a given tactic should be applied [309]. For this reason, these training sessions are generally supplemented with many hours of on-field training. VR is a powerful tool to address these issues, since it gives the possibility to the players to immerse themselves in a virtual experience mimicking the real one [192].

Moving from the above consideration, a new system named "VR Playbook" was presented in [50]. The VR Playbook system allows coaches and players to create, visualize and study tactics without the need to use 2D instruments, such as tactics boards or diagrams on paper. This choice was made, under the hypothesis that, by bringing the players into an immersive environment, they can learn movements and actions in a more effective and engaging way.

The VR Playbook system combines features available in common commercial software for the creation and editing of sport tactics with 3D visualizations based on immersive VR. Furthermore, spatio-temporal data (3D spatial coordinates in time) of the players and of sport equipment (e.g., the ball) collected during previous matches (through tracking methods) can be loaded in the system in order to visualize/analyze players' movements from different viewpoints. Recreated actions can also be modified and players' movements revised, e.g., to correct mistakes made during the game.

Considering the fact that, in general, training sessions devoted to tactics analysis involve multiple users [106], a networked architecture was designed, which lets multiple users (coaches and players) simultaneously join the same session. It was chosen to use simple VR solutions based on Google Cardboard rather than more complex (more expensive) systems like the Oculus Rift, the HTC Vive or the Windows Mixed Reality-based headsets, in order to easily increase the number of users participating to the training session. Although the devised system could be applied to empower training sessions of a number of sports, so far, a prototype implementation has been created for basketball.

Related work

The possibility to analyze previous matches and visualize new tactics in order to defend and attack as best as possible against the opponents' strategy in the next game is a common feature in many of the tools and applications already available on the market. Examples of such tools targeted to a competitive sport like basketball are, e.g., Dartfish[13] and LongoMatch[14]. These applications allow the coach to edit video and add labels to game footage available in a shared database or captured

[13]Dartfish: http://www.dartfish.com/

[14]LongoMatch: https://longomatch.com/en/

in real-time. The results of the video analysis/editing can then be discussed with the entire team. For this reason, this category of applications is mainly used to generate match reports supporting the study of the tactics adopted by the opposing team or the behavior of single players in the previous games.

Another category of applications includes tools like FastDraw[15] and Basketball Playbook Home[16]. These tools allow coaches to easily draw and manage basketball plays and drills. Although the actions of previous games can be created by using these tools, the visualization is still based on two dimensions. Moreover, only Basketball Playbook Home is able to automatically create animations of the players' movements between frames. In fact, the outcome leveraged by coaches is a collection of static diagrams.

Products developed by companies such as STRIVR[17] and Beyond Sports[18] offer the possibility to use training tools based on immersive 360° videos [320] and VR, respectively, for different sports like football, soccer and basketball.

The effectiveness of VR for training has been demonstrated in several work developed by the research community [131, 315]. For example, in [130], a training software based on immersive VR (named SIDEKIQ) was proposed to enhance the performance of football players. A minimalistic interface lets coaches generate football game plays that can be visualized either on a desktop screen or using an immersive display (headset or CAVE). A user study was conducted by involving 17 football players in a three-day evaluation. The goal of the evaluation was to monitor the scores assigned to each player after executing a number of trials designed to assess his or her decision making skills. Results showed the effectiveness of VR for training, since all the participants were capable to improve their scores obtaining, on average, a 30% overall improvement. However, a possible limitation is represented by the impossibility to make multiple players collaborate during the training session, since the tool was developed for one-to-one training sessions. Differently than in [130], in [106] a strategy analysis tool for soccer that allows multiple users to jointly analyze a game from different viewpoints into a virtual space representing the court is presented. Users can manipulate virtual objects through a tabletop interface. The visualization of a simple representation of the game can be observed through a wall-mounted display. Movements of the virtual objects follow the timeline information defined by the user or loaded from a file. Although a simple representation of the game is used whereby realistic animations of players' movements were missing, from the results of the user study reported by the authors it appears that the system could ease the recognition of tactical errors, since users are more aware of their

[15]FastDraw: https://goo.gl/WsUSyC

[16]Basketball Playbook Home: https://www.jes-soft.com/playbook/

[17]STRIVR: https://www.strivr.com/use-cases/sports/

[18]Beyond Sports: https://www.beyondsports.nl/

spatial position in the court.

Notwithstanding, the introduction of realistic animations and the use of high-quality graphics (in the previous work, players' body was represented by a conic shape, players' head by a sphere, etc.), could make these training systems more engaging and effective for learning purposes. For instance, the tactics simulation software for basketball presented in [335] leverages accurate 3D models of the venue and of the players. Moreover, professional animation tools were used to generate high-quality animations representing players' movements, such as pass, dribbling, shooting, breakthrough, defense, etc. The drawback of this work is the lack of a mechanism allowing the user to create tactics and to modify existing games, since actions to be played can only be chosen from a predefined list.

Taking into account the above considerations, a participative system based on wearable VR was developed, which provides the user a tool for the creation and editing of tactics and a methodology for visualizing games based on immersive VR. The system presents realistic 3D models and animations, which were created using professional tools (Autodesk Maya and the Perception Neuron Axis motion capture suit, respectively).

Proposed system

The core components of the VR Playbook system are illustrated in Fig. 4.14. As shown in this figure, the overall architecture is comprised of two main components: the Coach and the Player application. Both the applications have been developed by leveraging the Unity game engine, since it is able to support different platforms.

Coach application This component allows the coach to generate and edit tactics to be submitted to the players, through an intuitive interface targeted to tablet devices (Fig. 4.15). Interaction relies on touch gestures, which let the coach easily activate the various functionalities of the application. Functionalities are handled by the modules analyzed below.

- *Touch Input Listener*: This module detects touches performed by the user on the tablet screen and recognizes common gestures executed with fingers on the display. In particular, when a press or release gesture is identified, that gesture is used to define the behavior that a virtual player has to follow. Players are represented with a blue/red numbered icon depending on the team they belong to (as shown in Fig. 4.15). The Touch-Script library[19] was used to deal with touch input. The library supports multi-touch gestures and provides mechanisms for collecting information about gestures' execution,

[19]Touch-Script library: https://assetstore.unity.com/packages/tools/input-management/touchscript-7394

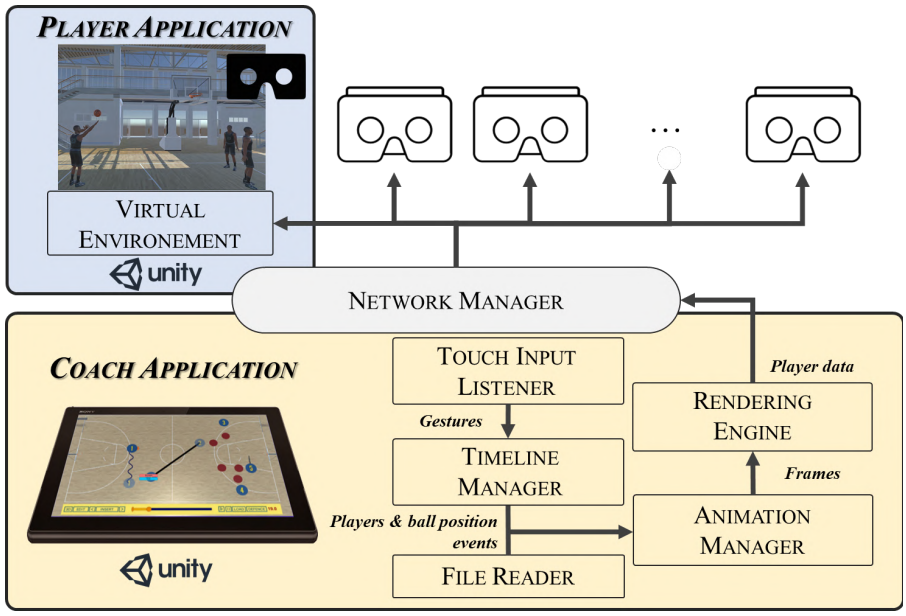


Figure 4.14: Architecture of the proposed system [50].

like, e.g., the beginning or end of a gesture. This feature is leveraged in the proposed system, since, for instance, when the user starts to press a player's icon, a method activates the drawing of the lines describing player's behavior according to the official International Basketball Federation (FIBA) symbols. Different behaviors can be specified, by choosing them from a contextual menu at the end of the interaction.

- *Timeline Manager*: This module is responsible for converting the user's gestures performed on the tablet screen into basketball events (e.g., players' movements, a ball passing, a defense position, etc.). Events are collected and stored in a data structure based on frames, which can be interpreted as the timeline of either the whole game or of a single action. Each frame records information regarding the 2D position and orientation of the players, the 3D position of the ball, and the time at which given basketball events occur. As previously mentioned, the user can define the movement of a specific player by first performing a press gesture on the numbered icon representing it, then dragging the icon to the desired position and finally releasing it. At the end of the release gesture, the coach can decide the type of the movement to be executed by choosing, at present, between "cut" (movement performed with a shot in order to beat the defender and receive the ball) and "screen" (attack movement without the ball, performed with the purpose of creating an advantage for a teammate) if the player is not the owner of the ball. If the selected player is the ball owner, a movement with the ball is automatically added to

the timeline (as for player 1 in Fig. 4.15). The owner of the ball can be specified by dragging and releasing the icon of the ball (the small orange dot) close to a player. If the owner of the ball changes between two different frames, a ball passing event is detected. The system represents this event through a dotted line that links the two players involved in the gameplay. For what it concerns the players' movements, the start position is represented with a transparent icon (as shown for player 1 and player 2 in Fig. 4.15). The multitouch feature allows the coach to control more than one player at the same time. The "*Insert*" button in the bottom part of the interface lets the coach insert a keyframe in the timeline. The system automatically interpolates values recorded in two consecutive keyframes in order to compute in-between frames . Navigating the timeline, i.e., moving backward and forward in the current frame, is accomplished through the "<" and ">" buttons or the slider in the bottom part of the interface. The presence of orange ellipse(s) over the slider indicates the presence of a keyframe registered by the coach for that frame. The current frame is shown in the bottom right corner of the interface. The "*Defence*" button allows the coach to enable/disable the visualization of the opposing team (the red players).

- *File Reader*: With the proposed system, the coach can manipulate tactics generated in previous training sessions, and based on data collected by tracking the ball's and players' positions during previous, real matches. The former functionality allows the coach to revise a single tactic; the latter can be used to analyze through computer-generated game animations previous matches played by the team in order to highlight correct movements, but also to possibly correct wrong ones (through direct editing). A methodology developed for the automatic generation of animations from tracked position of the ball and the players will be described in detail in the following. In both cases, it is assumed that data are recorded in a file containing spatio-temporal information and events. Therefore, a mechanism is needed to parse it and generate a timeline structure comparable to that which would be normally generated by manual editing operations. To read the file, the coach can press the "*Load*" button. The tactic is automatically saved, for further modification, when the coach activates the animation playback with the "*Play*" button.

- *Animation Manager*: This module is responsible of handling the 2D and 3D visualization of the gameplay. A press on the "*Play*" button creates the path that the players and the ball have to follow to interpolate the position data specified in the keyframes. The playback of the created animation can be visualized not only on the tablet device managed by the coach, but also on players' devices connected to the live session. The "*Pause*" button stops the visualization on all the devices. Depending on the events recorded in the timeline for each frame, a different animation is played during the playback

to visualize for instance, a throw, a dribbling, or a defense play. This module, implemented by making use of the Unity Animator component, is able to blend different animations (e.g., the walk and run animations are mixed depending on the velocity of the player) and to control the transition between different animations when specific events have to be reproduced by the player during the game (e.g., a transition from dribbling to ball passing). The "*3D*" button switches from the 2D to the 3D visualization. In the latter modality, the 2D icons representing players are replaced by 3D human models. When the coach presses the "*Edit*" button, the edit mode is enabled. This mode allows the coach to apply modifications to the current tactic. When the coach modifies the tactic, all the animations displayed on the players' devices connected to the live session are automatically paused.

- *Network Manager*: In order to implement the network communications, it was decided to use the capabilities offered by the Photon Unity Networking (PUN) Framework SDK[20]. PUN can manage multiple client devices connected through a Wi-Fi connection. Each client receives data representing the tactics and visualizes them in a VR environment through a wearable headset (worn by each player). The server side of this network architecture is represented by the Coach application. Theoretically, up to 100 clients can connect at the same time to the Coach application. Moreover, this module, which is included both in the Coach and in the Player application, is also responsible for network configuration and for the synchronization of the virtual objects, so that an animation played on the server is visualized at the same time, with the same pace on all the connected devices.

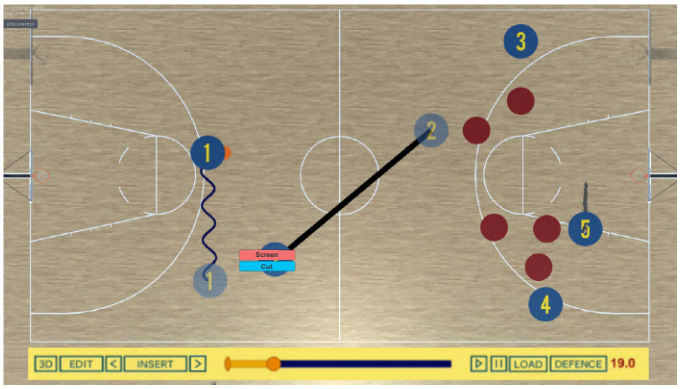


Figure 4.15: Graphics interface of the Coach application.

[20]Photon Unity Networking: https://www.photonengine.com/en/PUN

Player application The target devices for the Player application are those supporting Google Cardboard. The main aim of this application is to provide players with an intuitive interface to visualize whole games or gameplays into an immersive VR environment. Once the connection is established between a Player and the Coach application, the player can activate the live session through the interface shown in Fig. 4.16a. By using the gaze, each player can choose the number of the virtual player he or she wants to impersonate in the immersive environment. This way, the player can observe the action from a first-person perspective, as shown in Fig. 4.16b. When the coach activates the edit mode, the top-view visualization illustrated in Fig. 4.16c is enabled. This visualization is expected to limit motion sickness which could be experienced when the coach drags the player's representation during the editing. Moreover, top-view visualization allows players to observe the tactics being created and, at the same time, provides them with a broad view of the court where the position of teammates and opponents can be better identified.

Usage scenario

In the following, a typical usage workflow for the proposed system is presented by reporting its main steps.

1. The coach prepares a training session through his or her device by either specifying a number of keyframes that control the behavior of a player during one or more gameplays, or by loading a file with previously recorded data.

2. The coach enables the edit mode to modify players' behavior at some instants of time.

3. Multiple players connect to the live session through their VR headsets in order to visualize the tactics being created.

4. The coach terminates the tactics manipulation by disabling the edit mode and starts the playback of the animation.

5. Players and coach visualize together the gameplays and the movements to be analyzed or learned.

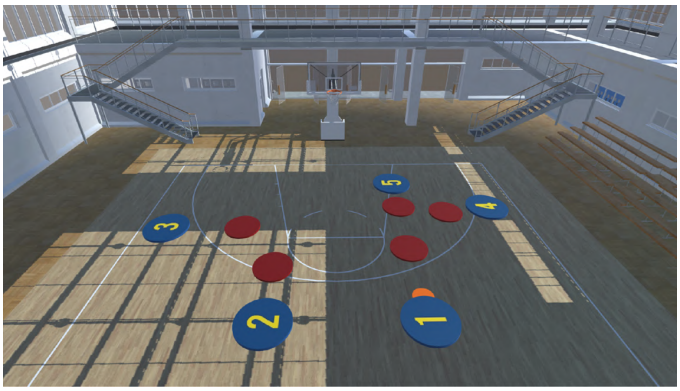
6. Steps from 2 to 5 can be repeated to adjust the tactics based on coach-players interaction.

Future developments

Future work could be devoted to asses the effectiveness of the devised system through a user study with coaches and players. In particular, a possibility could be to progressively introduce the VR-based system during the training sessions, with

(a) Main menu

(b) First-person visualization



(c) Visualization from the top-view

Figure 4.16: Player application.

the aim to collect preliminary feedback on users' acceptance and usability of the system. Aspects to be considered in the evaluation could consider, for instance, the users' feeling after a prolonged use (e.g., in terms of motion sickness and physical effort), the impact of the selected visualization methods on spatial awareness, the

importance of graphics contents and animations quality, etc. Afterward, experiments aimed to evaluate the effectiveness of the proposed approach with respect to the training efficacy could be considered. From the point of view of the tool itself, new animations could be generated to further improve the realism of the simulation and completeness of the tool. Moreover, the possibility to introduce dedicated interfaces enabling coach-player communications directly within the VR-based system could be verified. Lastly, the applicability of the proposed VR-based training methodology to other sports could be possibly investigated.

4.4.2 Automatic recognition and reproduction of sport events from spatio-temporal data

The recent advancements in tracking technology made the spatio-temporal data collected during matches or training sessions of various sports available for further processing [258]. Several methodologies based on different sensing techniques have been proposed in the literature to record the movements of players and other equipment (e.g., a tennis ball, a baseball bat, etc.) with high sampling rates [70, 112, 147]. As said, nowadays the analysis of tracking data is becoming a fundamental aspect in many competitive sports, since it can provide coaches with helpful insights about the previous game, which can be used for the automatic recognition of the opposing team's strategy [309], the generation of commentaries for matches [336], etc.

Moving from these considerations, the work which was originally reported in [6] investigated the use of machine learning for the automatic recognition of players' actions from spatio-temporal data; recognized actions could then be used, e.g., in a VR-based application for activating the corresponding animations (although information inferred from data could be used in other contexts for different purpose).

The proposed solution builds on previous work targeted to soccer [258]. With respect to the reference work, new features are extracted to:

- consider aspects disregarded in the reference work;
- integrate data not present in the reference dataset (like, for instance, the vertical position of the ball);
- take into account the different characteristics of basketball with respect to soccer, with the aim to improve recognition accuracy.

The result of the elaboration has been integrated in an immersive VR tool to make the user visualize animated reconstructions of previous basketball matches for tactic analysis and training. In particular, events recognized through the machine learning algorithm are provided in input to the VR system discussed above, which uses them to activate corresponding player's animations.

Related work

Methodologies designed for the automatic recognition of sport events have been already proposed in the literature. For example, the inductive learning techniques described in [336] can be used to automatically generate commentaries for football matches within a management simulation game named Championship Manager. To this aim, three classification techniques (i.e., Decision Tree, K-Nearest Neighbors (KNN), and Naive Bayes) are considered in order to find the mapping between game states and commentaries. The algorithm proposed in [301], is able to create automatic footage annotations by tracking the players and the ball in a tennis match. In particular, by analyzing the movements of the players and the ball, the algorithm is capable of classifying a player's action distinguishing backhand and forehand strokes with high precision and recall rates. In [213], players' tracking data are leveraged to identify offensive strategies in basketball through a linear Support-Vector Machines (SVM) classifier and a rule-based algorithm.

The solution proposed in [258], which is the reference work for the methodology discussed in the following, suggested the use of three machine learning algorithms, i.e., SVM, KNN, and Random Forests (RF), to classify events in a soccer match, like passes or receptions. The dataset used for the evaluation refers to matches of the German Bundesliga, and includes the following data:

- the timestamp;
- the two-dimensional coordinates of the ball;
- a list of game events (e.g., fouls, substitutions, offsides, etc.), and the identifiers of the players involved.

The classification was performed by considering a number of features computed with the raw position data of the ball. To train the classifiers, the dataset was manually annotated by identifying considered events in the footage of three matches.

By building upon the promising results reported in [258], an improved technique for the automatic classification of sport events from spatio-temporal data was developed; the devised technique extends the capabilities of the previous approach to a different sport, i.e., basketball. In particular, after having experimented with the same algorithms and features proposed in the reference work on a dataset containing position data from National Basketball Association (NBA) matches, additional features were introduced, which proved to significantly enhance the classification of basketball events.

Dataset

The dataset contains the spatio-temporal data collected at 20 Hz during matches belonging to the 2015–16 season of the NBA[21]. Data are subdivided into matches and actions (referring to a given match). For each action of each match, the position of the ball, and the position of each player, are recorded. Files in the dataset, stored as a *.csv* file, provided the following data:

- $team_{id}$: identifier of team to which player belongs to (this value is equal to -1 if the tracked object is the ball);
- $player_{id}$: identifier of the tracked object (-1 if the tracked object is the ball);
- x_{loc}, y_{loc}, z_{loc}: 3D spatial position of the tracked object (the z coordinate is provided only if the tracked object is the ball);
- $game_{clock}$: remaining time of the match;
- $shot_{clock}$: remaining time of the 24 seconds provided to a team for finalizing an offensive action;
- $quarter$: quarter of the game;
- $game_{id}$: identifier of the match;
- $event_{id}$: identifier of the action in the game.

The coordinate system adopted to represent positions in the dataset is shown in Fig. 4.17. The axis referring to the x_{loc} and y_{loc} fields are normalized in the $0-100$ and $0-50$ range, respectively for the x and y axis. The origin of the reference system, i.e., the point with (0,0) coordinates, is associated to the bottom-left corner of the court. Annotations were added to the dataset in order to associate sport events to the corresponding spatio-temporal data, by manually identifying them in the footage of the match between the San Antonio Spurs vs. the Minnesota Timberwolves, played on December 23rd, 2015. As in [258], passes and receptions were considered. Other events identified in the footage, including shots, dribbles, etc. were marked with the label "other". Some events belonging to the category "other" were randomly deleted, in order to balance the number of occurrences of the three events. At the end of the process, the annotated dataset contained 180 entries per event category.

[21]Dataset: https://github.com/sealneaward/nba-movement-data/tree/master/data

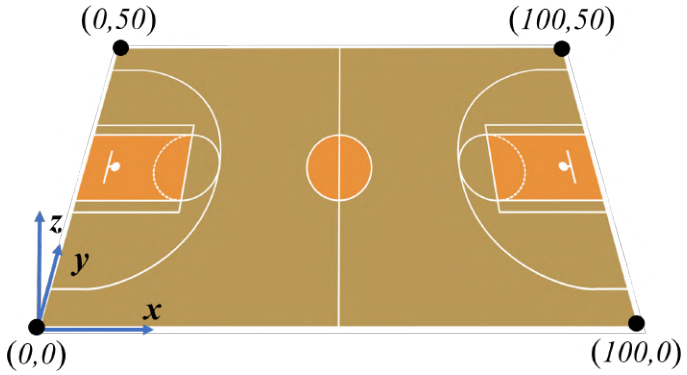


Figure 4.17: Basketball court and coordinate system used.

Features

According to [258], a sport event can be identified in a dataset containing spatio-temporal data by analyzing the values of several features that characterize it. Features were extracted and/or computed by running a script developed in C# on the above data. The script, for each time t in the dataset, generates a vector containing the computed value for each feature. Features used in [6] can be grouped in five categories. The first category contains the so-called "two-dimensional" features. These features were determined from the reference work. The remaining categories contain newly developed features generated from scratch or as extensions of the features in the first category. In particular, the features in the second category were determined by adapting previous features in order to consider only the movement of the ball along the z axis; hence, they will be referred to as "vertical". Features in the third category represent the extension made to deal with "three-dimensional" spaces. Features in the fourth category take into account the position of the players; thus, they will be referred to as "players"' features. Lastly, features in the fifth group were estimated by aggregating data (i.e., computing the means and variances for previous features values within a given time window). Features in this category will be referred to as "aggregated" features.

Before analyzing in detail the above features, it is worth providing some basic definitions. The position of a tracked object o at time t is represented as $p(o,t)$. Similarly, $p_x(o,t)$ and $p_y(o,t)$ correspond to the position along the x and y axes of the tracked object. The distance between two consecutive positions can be defined as:

$$d(o,t_1) = p(o,t_2) - p(o,t_1) \tag{4.1}$$

where t_1 and t_2 are different time samples and $t_1 < t_2$.

Two-dimensional features

The features in this category are computed by considering only the position of the ball in two dimensions. In the following, the subscript 2D will be used to refer to features of this category. In the reference work, the z dimension was not used since it was not available in the dataset.

- *Velocity*: The two-dimension velocity, was introduced as an indicator of the ball's momentum. It can be computed by dividing the length of the direction vector $d(o, t_1)$ by the time interval between two adjacent samples as in the following equation:

$$Vel_{2D}(o, t_1) = \frac{|d(o, t_1)|}{t_2 - t_1} \tag{4.2}$$

- *Acceleration*: Like for the velocity, the acceleration was included as an indicator of the ball's momentum. It can be measured as:

$$Acc_{2D}(o, t_1) = \frac{Vel_{2D}(o, t_2) - Vel_{2D}(o, t_1)}{t_2 - t_1} \tag{4.3}$$

- *Acceleration Peaks*: Given the high sampling rate of the data, the same value for the acceleration could be measured in several consecutive time samples. Therefore, the authors of the [258] proposed two features referred to as Acceleration Peaks. These features combine consecutive acceleration values by selecting the highest and the lowest ones among adjacent values, respectively. The procedure for computing the actual maximum and minimum peaks is composed of two stages. In the first stage, the sum of two consecutive accelerations is calculated, ignoring negative and positive values, i.e., by setting them to 0, for the computation of the maximum and the minimum peak, respectively:

$$AP_{2D\_max}(o, t_2) = \sum_{x \in t_1, t_2} \max(0, Acc_{2D}(o, x)) \tag{4.4}$$

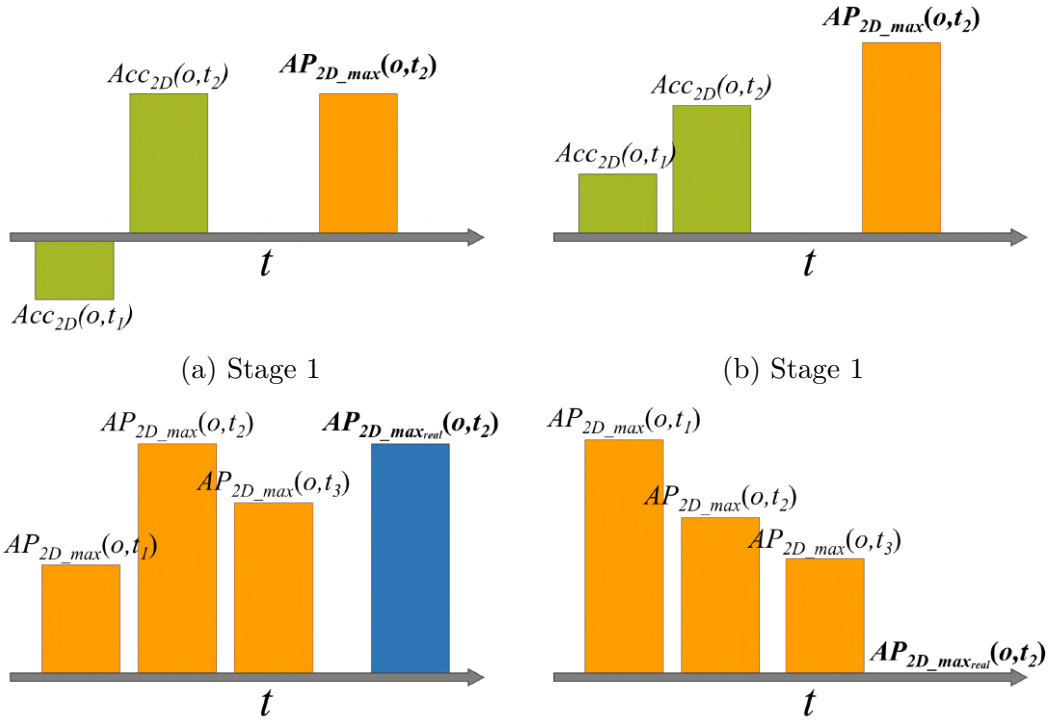
$$AP_{2D\_min}(o, t_2) = \sum_{x \in t_1, t_2} \min(0, Acc_{2D}(o, x)) \tag{4.5}$$

  Fig. 4.18a and Fig. 4.18b show two examples of $AP_{2D\_max}(o, t_2)$ computation. In the first case (Fig. 4.18a), $Acc_{2D}(o, t_1)$ is ignored and set to 0 since it is negative; in Fig. 4.18b, both the acceleration values are considered when computing the sum.

  In the second stage, in order to prevent the detection of a peak in two consecutive samples, real acceleration peaks, $AP_{2D\_max_{real}}(o, t_2)$ and

$AP_{2D\_min_{real}}(o, t_2)$, are computed setting them to $AP_{2D\_max}(o, t_2)$ and $AP_{2D\_min}(o, t_2)$ only if the value of the feature at time t_2 is higher than values at t_1 and t_3, otherwise they are set to 0.

Fig. 4.18c and Fig. 4.18d show the resulting $AP_{2D\_max_{real}}(o, t_2)$ for two different sequences of $AP_{2D\_max}(o, t_2)$ values.



(a) Stage 1 (b) Stage 1

(c) Stage 2 (d) Stage 2

Figure 4.18: Examples of Acceleration Peaks calculation.

- *Direction Change*: This feature estimates the variations in the trajectory of the ball during the game, comparing the angle between two consecutive direction vectors (as shown in Fig. 4.19a). This feature was considered to improve the recognition of events like passes or shoots, assuming that the trajectory of the ball is strongly influenced when these types of events occur.

  The direction change $DC_{2D}(o, t_2)$ of object o at time t_2 is measured by applying the arccos() function as follows:

$$DC_{2D}(o, t_2) = \arccos\left(\frac{d(o, t_1) * d(o, t_2)}{|d(o, t_1)| * |d(o, t_2)|}\right) \tag{4.6}$$

- *Distance to Target*: During a match, the goal of the player is to throw the ball into one of the baskets (nets, in the reference work) in order to earn points. Therefore, it can be reasonable to assume that the ball moves towards one of these targets. Moving from this observation, the contribution of this feature can be leveraged to distinguish between passes and shots. The distance of object o at time t from the target is calculated as:

$$DT_{2D}(o,t) = |p(o,t) - b(o,t)| \tag{4.7}$$

where *b(o,t)* represents the target position determined considering the direction of the ball with respect to the x axis. As shown in Fig. 4.19b, this position could be either the point T_1 with coordinates (0, 25) if the ball moves towards the left side of the court, or T_2 with coordinates (100, 25) if the ball moves towards the right side. The figure shows different distances to target (represented by solid lines colored in blue) computed considering the direction of the ball (represented by an arrow at each data point). For object in P_1, characterized by a horizontal velocity equal to 0, target could be determined; hence, the feature value is set to ∞.

- *Cross on Target Line*: This feature is computed by considering the distance between the target and the position in which the ball would cross the end line if the ball maintains the current trajectory up to the line. Fig. 4.19c shows a data point P_1 and its direction vector d_1. If the ball continues to follow the trajectory represented with a dashed line, it would reach the end line in C_1. The distance between C_1 and the target position T_1 is the actual value of this feature (highlighted in blue). Position of C_1 can be calculated as:

$$\begin{pmatrix} b_x(o,t) \\ ctl \end{pmatrix} = p(o,t) + s * d(o,t) \tag{4.8}$$

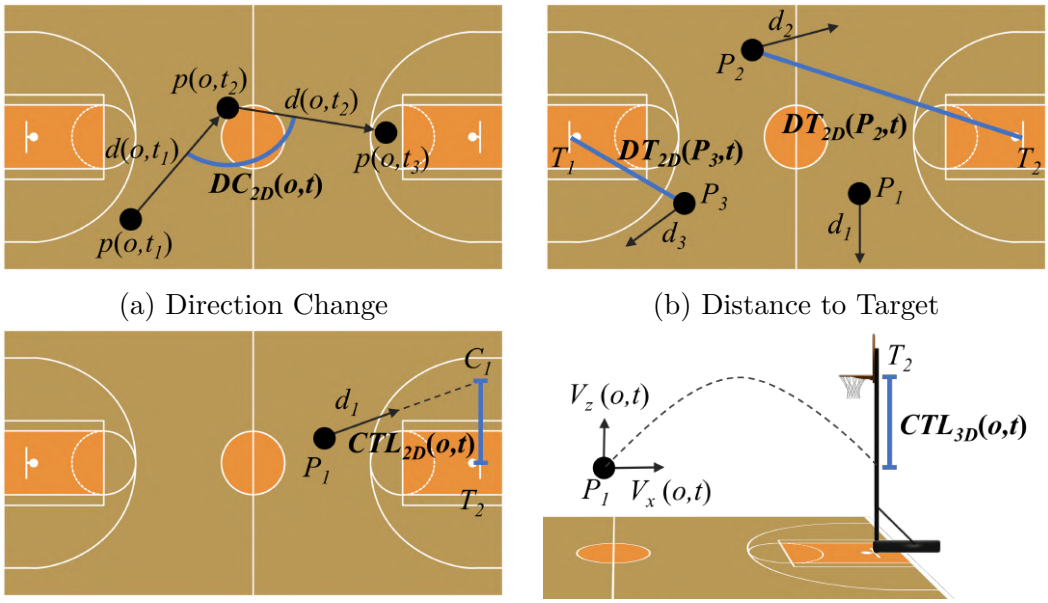
where s is a factor that, if it is multiplied by the direction vector of the object o at time t and added to the position of the object o at time t, allows to reach the end line. From all of the above, it is possible to compute $CTL_{2D}(o,t)$ as:

$$CTL_{2D}(o,t) = p_y(o,t) + d_y(o,t)\frac{b_x(o,t) - p_x(o,t)}{d_x(o,t)} \tag{4.9}$$

where the subscript identifies the axis considered.

Vertical features

This category contains a subset of the features described above, but recomputed considering only the z coordinate:



(a) Direction Change

(b) Distance to Target

(c) Cross on Target Line (2D)

(d) Cross on Target Line (3D)

Figure 4.19: Examples of two-/three dimensional features calculations.

- Velocity ($Vel_V(o,t)$);
- Acceleration ($Acc_V(o,t)$);
- Acceleration Peaks ($AP_{V\_max_{real}}(o,t)$ and ($AP_{V\_min_{real}}(o,t)$).

The remaining features belonging to the first category cannot be recalculated, as the single dimension considered for computing these features does not allow to identify the direction of the ball.

Three-dimensional features

This category includes features that are computed by considering the three coordinates. The considered features, indicated with the subscript 3D, are reported in the following.

- Velocity ($Vel_{3D}(o,t)$);
- Acceleration ($Acc_{3D}(o,t)$);
- Acceleration Peaks ($AP_{3D\_max_{real}}(o,t)$ and $AP_{3D\_min_{real}}(o,t)$);
- Cross on Target Line ($CTV_{3D}(o,t)$).

For what it concerns $CTL_{3D}(o,t)$, a parabolic trajectory is assumed (as shown in Fig. 4.19d), and the value of this feature (represented with a blue line) is calculated as:

$$CTL_{3D}(o,t) = -\frac{1}{2}gt_{line}^2 + Vel_z(o,t)t_{line} + p_z(o,t) \tag{4.10}$$

where g is the gravity acceleration, $Vel_z(o,t)$ is the component along the z axis of $Vel_{3D}(o,t)$, and t_{line} is the time that is required for the ball to reach the end line; t_{line} is defined as:

$$t_{line} = \frac{b_x(o,t) - p_x(o,t)}{Vel_x(o,t)} \tag{4.11}$$

where $Vel_x(o,t)$ is the component of $Vel_{3D}(o,t)$ along the x axis.

Players' features

The two features belonging to this category take into account the relation between the position of the ball and the players. Therefore, the contribution of the players' features is strictly influenced by the way the ball's position changes in close proximity to a player. This behavior could be a valid descriptor for some basketball events.

- *Ball-Player Distance*: This feature measures the distance between the ball and the closest player at a given time t. It is computed as:

$$BPD(o,t) = |p(o,t) - p_{player}(o,t)| \tag{4.12}$$

  where $p(o,t)$ and $p_{player}(o,t)$ is the 2D position of the ball and of the closest player at time t, respectively.

- *Team of Closer Player*: This feature indicates for each time t which is the name of the team to which of the player closest to the ball belongs to.

Aggregated features

This category contains a set feature computed by aggregating consecutive samples measured for the above features. The aggregation is performed by computing the average and the variance of samples within two time windows, named *before-window* and *after-window*. The aggregation can be used to monitor the features' dynamics. The size of the two windows has been experimentally defined and includes 20 samples (i.e., one second) before and after the current time. The features considered for the aggregation are: $p_z(o,t)$, $Vel_V(o,t)$, $Acc_V(o,t)$, $DC_{2D}(o,t)$, $BPD(o,t)$. Considering them and based on the values in the two windows, the features reported

in the following are computed. The AVG or VAR subscripts represent the operator applied (i.e., average and variance, respectively), whereas "b" and "a" subscripts refer to the before and after window:

- $AVG_b p_z(o,t)$;
- $AVG_a p_z(o,t)$;
- $VAR_b p_z(o,t)$;
- $VAR_a p_z(o,t)$;
- $AVG_b Vel_V(o,t)$;
- $AVG_a Vel_V(o,t)$;
- $VAR_b Vel_V(o,t)$;
- $VAR_a Vel_V(o,t)$;
- $AVG_b Acc_V(o,t)$;
- $AVG_a Acc_V(o,t)$;
- $VAR_b Acc_V(o,t)$;
- $VAR_a Acc_V(o,t)$;
- $AVG_b DC2_D(o,t)$;
- $AVG_a DC_{2D}(o,t)$;
- $VAR_b DC2_D(o,t)$;
- $VAR_a DC_{2D}(o,t)$;
- $AVG_b BPD(o,t)$;
- $AVG_a BPD(o,t)$;
- $VAR_b BPD(o,t)$;
- $VAR_a BPD(o,t)$.

Performance evaluation

In order to investigate the effectiveness of the devised features for sport events recognition from spatio-temporal data, they have been used in combination with the three machine learning algorithms considered in [258], i.e, the SVM, KNN and RF classifiers.

A vector containing the values of all corresponding features was created for every time t. Each vector indicates the occurrence of a sport event during the game. Each event is characterized by particular values computed for each of the defined features. For instance, passes are probably characterized by a significant acceleration peak and high values for the feature describing direction changes, whereas in the case of receptions, the ball should present a strong negative acceleration with the distance between the ball and the closest player (probably the ball's owner) that remains almost the same.

The data science software platform named Rapidminer[22] was used to execute the three algorithms. The focus was put on the recognition of three events: pass, reception and other ball events, though in basketball rather than in soccer. The quality of results achieved was assessed through the computation of the accuracy, precision, recall, and F-measure. Cross validation with 20 partitions and linear sampling were used to cope with the reduced size of the dataset.

Algorithms were executed by considering different combinations of the features of the five categories. At the beginning, only the first category was considered, in order to qualitatively compare results obtained on the new dataset with those in [258]. Afterward, the vertical and the players' features were considered. Then the two-dimensional features were replaced with the three-dimensional ones. Lastly, the aggregated features were included. By changing the set of features considered, the overall accuracy increased from the initial value of 33.68% obtained when using only the first category, to the 76.67% when including the last set of features. Table 4.3 reports recognition results for each event at each stage, obtained with the machine learning algorithm which achieved the best performance (i.e., KNN).

Application scenario

The proposed method was used to extend the capabilities of the VR Playbook tool described in Section 4.4.1. As previously mentioned, the tool was designed to allow coaches and players to create tactics and visualize previous basketball games in an immersive environment.

The original implementation of the tool allows coaches to manually specify the players' events in the timeline, e.g., based on available game footage or by resorting to their memory, in order to visualize the actions of a previous match. Players'

[22]RapidMiner: https://rapidminer.com/

Table 4.3: Results of KNN for the recognition of basketball events.

Features	**Event**	**Precision**	**Recall**	**F-measure**	**Accuracy**
Two-dim.	Generic	0.38	0.40	0.39	33.68%
	Pass	0.32	0.30	0.31	
	Reception	0.68	0.67	0.67	
Vertical & Players	Generic	0.93	0.98	0.95	74.31%
	Pass	0.66	0.61	0.63	
	Reception	0.64	0.65	0.64	
Three-dim.	Generic	0.93	0.98	0.95	74.93%
	Pass	0.66	0.61	0.63	
	Reception	0.64	0.65	0.64	
Aggregated	Generic	0.93	1.00	0.96	76.67%
	Pass	0.69	0.65	0.67	
	Reception	0.68	0.67	0.67	

trajectories could be defined by drawing arrows on the touchscreen, linking the starting and ending points of a given action.

Alternatively, the system provides the users with the possibility to load a dataset to recreate real movements. However, the lack of annotations regarding the events' timing and type, make the animations created poorly realistic, since positional data could only be used to reconstruct a run cycle animation for players.

The methodology described in this section could be used to automatically extract players' events from a dataset containing only spatio-temporal data. Extracted events can be exported in a format ready to be parsed and imported by the VR Playbook tool. In this way, the quality (in terms of animation realism) of the reconstructed match can be improved, since the exact time a given animation shall begin/end is automatically determined, and a more accurate relation between the players' hands and the ball can be defined and used, for instance, to blend the run and pass animations.

Fig. 4.20 shows how the devised methodology (represented by the module named Event Recognizer) was integrated in the architecture of the VR Playbook. It can be easily noticed that the integration is transparent to the users, since automatically extracted events are treated as manually defined ones, and coaches can further edit them using the same tablet-based interface.

An example comparing the quality of animations that could be created using only dataset's raw data and the devised methodology is shown in Fig. 4.21a and Fig. 4.21b.

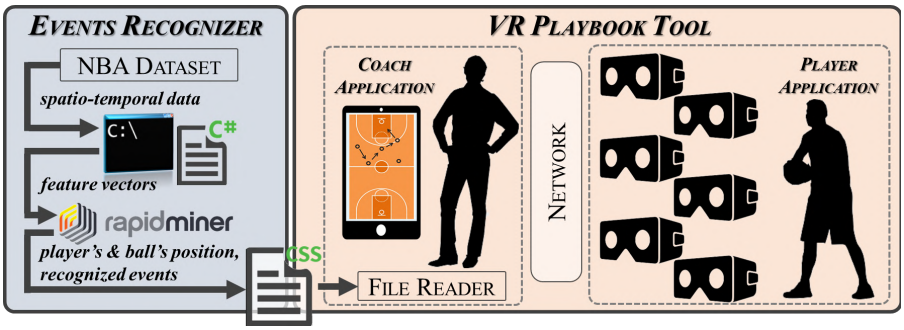


Figure 4.20: Integration of the event recognition methodology developed in [6] into the VR Playbook tool.

(a) Only raw position data

(b) Automatically recognized events

Figure 4.21: Frames of a 3D animation created using different conditions.

Future developments

Future work could be devoted to analyze the impact of introducing new features as well as new classification methods (e.g., based on deep learning) on the achievable accuracy. Moreover different small-scale basketball events (like throws, screens, cuts, etc.) and large-scale phenomena occurring during the game could be considered (e.g., to predict dangerous actions, to identify tactics, to spot mistakes made by a player in executing a tactic, etc.). Effort could be possibly devoted also to manage the introduction of improved techniques for animation blending to further enhance the quality of the generated animations, making the VR-based visualization systems suitable also for sport applications different than training. Lastly, a user study could be planned to validate the effectiveness of the proposed VR training system with coaches and players of a basketball team.

4.5 Concluding remarks

In this chapter, it was shown how new technologies, VR and AR in particular, and automatic procedures leveraging machine learning can be helpful tools to support the creation of computer graphics assets by users with limited skills in the given domain.

In particular, the system presented in the first part of the chapter allows non-professional users, e.g., storytellers, screenwriters, etc., to easily assemble a full 3D scene for fast prototyping or for sharing ideas [5]. In the second part of the chapter, innovative approaches have been applied to specific use cases concerning constructive art and sport training, which present similar requirements. In fact, target users with limited computer graphics skills, i.e., artists in [46, 14], coaches and players in [50, 6] are still considered, and specific graphics assets have to be managed (artworks and basketball tactics, respectively). Results reported above showed promising effects/advantages brought by the use of these approaches in the different domains considered, even though more objective evaluations should be carried out in order to fully asses their effectiveness.

Chapter 5

Leveraging graphics assets: Interactive applications development

Work described in this chapter was originally presented in [4].

5.1 Introduction

In the previous chapter, it was shown how new technologies and automatic procedures could be leveraged to make the generation of computer-generated graphics assets easier. However, to obtain the entire (or at least a more complete) picture, it is necessary to consider also aspects that concern the interaction with these assets and how to make them accessible to end-users. This was the goal of the work reported in the following.

An interesting domain where it is of paramount importance the way to make users able to access 3D graphics assets is represented by interactive applications development. These applications, for instance, be intended to enhance the users' experience in public exhibitions [266, 321]. For example, interactive installations have been already made available to the visitors of several museums, like the Virtual Archaeological Museum (MAV) in Italy[1], the National Museum of Zurich in Switzerland[2], the National Museum of Singapore[3], and the Cleveland Museum of

[1]MAV: https://www.museomav.it/museum/?lang=en

[2]Ideas of Switzerland, National Museum of Zurich: https://www.landesmuseum.ch/ideas-of-switzerland

[3]Story of the forest, National Museum of Singapore: https://www.nationalmuseum.sg/our-exhibitions/exhibition-list/story-of-the-forest

Art[4] in the USA.

Another usage example is represented by the systems being adopted, e.g., for marketing purposes, in trade fairs [329] and shopping areas [148, 187]. Several companies have created ad-hoc solutions to advertise or sell their products; for instance, one could mention the VR experience developed by Mastercard and Swarovski for the Atelier Swarovski in the USA[5], the AR app developed by IKEA[6], the virtual tour designed to visualize the new cabins of All Nippon Airways planes[7], the immersive test drive simulation developed for the Volvo XC90[8], etc.

The idea, shared among these different application domains, is to replace static contents with virtual assets that visitors/customers can interact with through NUIs in order to make their experience at the museum/store more engaging and enjoyable. Historical landmarks, ancient or modern artworks, and commercial products, and any other objects can be displayed and manipulated through projected holograms/walls, VR/AR systems, tangible/gestural interfaces, etc. The possibility to manipulate these technologies has been proved to be capable to enhance users' understanding, making knowledge more accessible to people [23]. Furthermore, the improved engagement brought by these interactive exhibitions could increase the interest of visitors, customers, etc., bringing them back several times to the museums or the stores [7]. Finally, there are studies such as the work in [144], which showed a connection between interactive experiences and a number of aspects related to the field of HCI, like creativity, embodiment, affect, and presence. This strict connection may translate improvements made to one of these fields into another.

From the above observations, it appears that the design and development of these applications are becoming of fundamental importance. However, such processes are still very time-consuming and skill-intensive, and require significant computer skills [266, 321]. For this reason, the research community is devoting more and more attention to the creation of tools able to make the development steps easier, independently of the particular application domain as well as of the device chosen for visualization and interaction.

Research activities considering this topic generally fall under the broad umbrella of end-user development (EUD). The goal of the EUD is to make end-users (for example, in this case, the curator of the museum exhibition or the sales manager) able to develop and manage complex systems, interfaces and applications without

[4]ARTLENS, Cleveland Museum of Art: http://www.clevelandart.org/artlens-gallery

[5]Atelier Swarovski: https://mstr.cd/2wZcAce

[6]IKEA's Place app: https://bit.ly/32cbghp

[7]All Nippon Airways's Aeronautics VR: https://mbryonic.com/portfolio/ana/

[8]Volvo's XC90 Test drive VR: http://framestorevr.com/volvo2

possessing professional software development skills [21]. Nowadays, there are various methods, techniques and tools that are oriented to EUD, as reported in the survey reported in [201].

Among EUD solutions, a significant attention is posed on end-user programming (EUP) techniques, which are aimed to let the users create their programs by leveraing programming paradigms suited to their skills [41]. EUP techniques include different approaches ranging from programming by example [31], to natural language programming [334], visual programming [21], etc. The approach which is attracting greater attention from both researchers and developers is the one based on visual programming languages (VPLs) [244]. These languages have already been adopted for educational purposes, e.g., [39] and [310]. The main advantage is the possibility to replace the blocks of code written following a given syntax with visual elements that represent the behavior of the program. The common concepts of traditional programming languages (like variables, functions, etc.) are replaced, e.g., by colored blocks that can be plugged each other or connected through arrows/links to define the intended application logic by using an intuitive interface. Benefits brought by the use of VPLs have been confirmed already in terms of users' engagement, satisfaction, motivation and performance [39, 249].

As it will be shown in the following, based on the above considerations a system was developed, named Visual Scene Editor (VSE), which allows users with limited programming skills (later referred to also as unskilled users), to develop interactive 3D applications by defining involved graphics assets and their behaviors/interaction capabilities. The applications generated through VSE can be displayed on any traditional computer or large display, as well as in immersive VR environments and on AR headsets. Target users (e.g., visitors of an exhibition or customers in stores) may interact with the generated contents by using traditional interfaces (like, M&K) or NUIs, e.g., based hand/body gestures, voice, gaze, etc. At present, the current implementation leverages computer and holographic displays, together with hand gesture-based interactions.

The design and development of the VSE rely on a previous experience made with a VPL-based tool named Leap Embedder (LE) [266], which was developed tackling the same need. In fact, the main aim of the LE was to ease the usage of the BGE[9], a real-time game engine already adopted to develop interactive 3D applications in various domains, ranging from cultural heritage [42, 123], to production control [186], molecular modeling [314], etc. The BGE natively provides users with a visual interface, that can be used to set connections between events recognized by the system (e.g., interaction on a virtual asset) and actions to perform within the graphics environment (e.g., in response to that interaction). Although the BGE interface is very flexible in terms of applications that can be developed, it

[9]BGE: https://docs.blender.org/manual/en/2.79/game_engine/

is also difficult to use. For this reason, the LE was aimed to ease the original graphics notation of the BGE and, at the same time, extend its capabilities by integrating hand gesture-based interactions (gathered through the Leap Motion sensor[10], hence the name of the tool). Notwithstanding, innovations introduced in the LE maintained the original *object-centric* paradigm of BGE (and other tools) used for the definition of the application logic. This approach relies on so-called "logic bricks", which are defined for individual interactive objects. Each object could communicate with others only through message passing. This choice had a huge impact in terms of usability, limiting the advantages guaranteed by the use of the simplified notation.

The design of the VSE tool integrated the outcome of the LE's experience with helpful observations proposed in the literature concerning VPLs in various application contexts, in order to propose an alternative to LE. In particular, the VSE introduces a different interaction paradigm, in the following referred to as *scene-centric* (hence, the acronym of the tool), capable to support the development of 3D interactive applications by achieving better performance with respect to both the LE and the BGE. It is worth observing that, although the VSE was focused on public exhibitions, interactive applications that can be created with this tool (like those developed with the BGE and the LE) could be used, in principle, in other domains in which users with limited application development skills are involved.

5.2 Related work

This section first discusses the use of interactive applications in public exhibitions. Then, it describes the application domains whereby EUD approaches are used, with a focus on systems based on VPLs. Lastly, more details regarding the LE and the BGE are provided, in order to better introduce the design of the VSE and its contribution.

5.2.1 Applications for public exhibitions

The possibilities offered by the use of interactive applications to improve the user's experience in public exhibitions are confirmed by several solutions proposed in the literature. Starting with solutions focused on museum visits, a first example is provided in [285], where a virtual environment allows users to virtually explore a reconstructed heritage scenario. The system provides users with a virtual tour guide capable to propose diverse information depending on the actual visitor's interests. In [262], an AR location-based mobile game was proposed. The game, which was designed to support the visit of a real museum, requested the users to collect

[10]Leap Motion sensor: https://www.leapmotion.com/

as many points as possible by accomplishing a set of objectives (e.g., physically reaching some points of interest or solving quizzes). Although adults appeared less attracted by this game, probably due to its cartoon-style interface, results confirmed the possibility to provide a compelling learning experience through the proposed system, which was more effective than a traditional portable audiovisual guide. Still considering solutions targeted to cultural heritage, other work addressed outdoor settings, including the two solutions presented in [62] and [52], which are aimed to improve the fruition and enjoyment of archaeological sites. In particular, the system in [62] proposed a CAVE-like environment that allows visitors to virtually visit an ancient Greek temple by making use of 3D visualization, immersive sounds, and haptic interactions to improve the user experience. The immersive VR system in [52], in turn, supports interaction with archaeological remains.

Work reviewed so far confirm the benefits of using these new technologies to support learning and to improve the user experience offered by public exhibitions focused, e.g., on cultural heritage. However, the growing diffusion of these alternative means for communicating information is posing new challenges to the research community regarding the way to develop these interactive applications. In fact, a number of solutions in the literature proposed approaches aimed to support developers in the creation, manipulation, and validation of interactive exhibitions.

For example, the authors of [321] proposed a tool targeted to museum staff or to the curators of an exhibition aimed to simplify the development of web-based or of VR- and AR-based cultural heritage applications based on predefined templates. The types of templates supported by the system included a tree-based visualization with metadata and a virtual gallery walkthrough with on-request display of artwork details. The authoring interface only allows the user to specify which elements have to be included in the exhibition and where to place them (in the template visualization). Different interactions that may be needed in more complex applications developed for similar purposes were not integrated. In [133], a set of guidelines to be followed during the design of virtual environments targeted to cultural learning scenarios was proposed. Similarly, the work in [8] presented the design steps to be considered for the whole development cycle of a serious game supporting cultural heritage scenarios. The literature contains other work focusing on the attributes that can influence the user experience, e.g., in an interactive VR-based showroom [329], on determining the human's senses to be stimulated in a multi-sensory art exhibition [311], on how to limit new phenomena like cybersickness in VR applications recreating 3D stores [187], on how visitors move [195] or behave [100, 222] in virtual environments, etc.

5.2.2 End-User Development

Solutions that can be referred to as EUD techniques have been effectively applied in a number of domains. Various applications have been developed, for example, in

the field of smart environments and Internet of Things (IoT) [75, 102], that require systems able to manage a huge number of devices that, once connected each other, generate a significant amount of data that need to be handled somehow [302]. EUD techniques have also been applied to the cultural heritage context to allows domain experts to be directly involved in the development of interactive exhibitions [10, 101]. In fact, their contribution is proven to be capable of enhancing the quality of the generated output [212]. Techniques using EUD have been also leveraged for the creation of general-purpose video-games: examples are, e.g., the tool designed in [137] to let children develop interactive applications using an agent-based framework, the framework proposed in [215] that allows end-users to define the behavior of Non-Player Characters, and the solution described in [252] to support the retargeting of existing serious games to other domains. Another interesting application domain is represented by the possibility to program autonomous systems: in this context, the environment presented in [181] supports the control a humanoid robot through a simplified rule-based notation that makes accessible also sensors and actuators without the need to know the underlying technical details. Even in the field of mobile applications development there are examples of EUD techniques, including the environment designed to create mobile applications on handheld devices proposed in [71], or the tool that allows users to create complex pervasive applications by leveraging simple drag-and-drop interactions [96]. Finally, EUD techniques also focused on the creation of Web sites and Web applications; examples are, e.g., the tools described in [169, 159], which ease the generation of Web pages by adopting a template-based approach, and the ontology-based strategy designed in [307], which is able to generate functional prototypes from formal specifications.

5.2.3 Visual Programming Languages

As previously mentioned, one of the approaches of EUD currently attracting greater attention is represented by VPLs. A VPL is a high-level language that lets users create software and other computer-generated products by making use of visual graphic elements, removing the need to follow the canonical text-based syntax of traditional programming languages [151].

One of the first examples of VPLs dates back to the first half of the sixties, when the system called Sketchpad was developed at MIT by I. Sutherland [296]. The system allowed users to generate 2D contents by moving an optically-tracked prop on a computer display. Five years later, the authors of [82] proposed an ancestor of the so-called flowchart-based VPLs. As in [296], the prototype allowed users to create flowcharts by using a tracked probe and visualize the generated diagrams on a computer screen.

The technological progress rapidly improved the efficiency and effectiveness of the examples above, making them pass from handling a few graphics primitives (like lines, circles, etc.) to several complex shapes, thus letting users create richer

representations and handle larger problems. Technical developments also provided the possibility to extend the range of possible domains where VPLs can be applied. In fact, these languages started to be adopted at schools for educational purposes, e.g., to improve students' problem-solving skills and make programming more accessible. For example, in [155], mechanics and robotics concepts were taught through the Lego Mindstorms' VPL-based environment named NXT Software[11]. In [249], a different educational software developed by Lego called WeDo[12] was employed to make younger students practice the basics of computer programming. Experimental results demonstrated that students learned these concepts more effectively when they used WeDo than with traditional lessons. In [310], a gaming environment called CodeCraft supports students in learning the fundamentals of coding. The game environment leverages a problem-based learning approach, whereby users are requested to solve a series of puzzle games that involve 3D virtual components by using a VPL. Another VPL-based framework was proposed in [39] to make students study distributed programming concepts. The framework allows the users to execute code remotely on different machines, by sending messages with a structured data payload. Experiments carried out with participants with no programming skills showed the effectiveness of the devised framework, since the majority of the participants were able to develop simple but functioning applications.

With respect to the application of VPLs to EUP, in [317], a VPL-based paradigm for programming the behavior of a robotic arm is proposed. The interface offers the possibility to define the actions to be executed by the robot by dragging-and-dropping predefined routines and combining them in order to create a single sequence of instructions. User studies were conducted with the aim to compare the VPL-approach with two commercial programming environments (by ABB Automation Company and Universal Robots). Results showed that the proposed VPL lets the users program the robot faster than with the other environments, preserving the same level of accuracy.

Regarding public exhibitions, in [293], a VPL-based tool is presented that allows end-users to develop engaging exhibits by using visual blocks. The tool leverages an event-based approach that integrates pluggable and colored blocks that represent callbacks. The above tool was tested in a cultural heritage scenario, that required expert users to develop a Web-based application that could be used by the visitors to interact with 2D contents through hand gestures.

[11]NXT: https://bit.ly/3cHR5gc

[12]WeDo: https://bit.ly/3081ZeZ

5.2.4 Leap Embedder

The work considered in the above review shows that EUD-based approaches have been experimented in a number of heterogeneous domains. However, it appears that tools targeted to the creation of 3D interactive applications for public exhibitions are still receiving poor attention in this perspective.

Even when this specific domain is considered by supporting the management of 3D elements, tools available are usually able to manage only a few predefined application configurations. Although, in principle, general-purpose tools designed, e.g., for 3D video-game development, could be leveraged to this purpose, they usually require users to directly manipulate 3D geometries, manage lighting and cameras, etc. This aspect could prevent unskilled users to make use of them, since they are considered too complex. On the other hand, when special-purpose tools targeting other domains are considered, they probably do not provide enough contents and interactions required to create the applications of interest.

The tool LE proposed in [266] appears to be one of the few tools that were designed to fill this gap. The development of a 3D application with LE relies on 3D assets previously generated by leveraging a professional modeling and animation suite. To this purpose, the graphics suite named Blender[13] was chosen, since it also includes an integrated real-time game engine (BGE).

The BGE can be used to develop 3D interactive applications by working with contents created with Blender and/or imported from different graphics suites. The application logic in BGE can be defined by using an event-driven VPL-based syntax that relies on three different kinds of logic bricks: "sensors", "controllers" and "actuators". These components can be wired in order to define the logic associated to each game object. Objects can share part of the logic by communicating with each other through the exchange of messages. Python code can be attached to the components in order to define possible custom scripted controllers.

The basic idea of the LE was to remove the need to use the above notation by replacing it with another visual syntax encompassing a reduced set of logic bricks referred to as "blocks". Due to the fact that the target application domain for the produced contents was represented by public exhibitions, in designing the LE it was considered the need for the users to easily integrate NUI-based interaction modalities (in particular, hand gestures). The logic of the interactive applications developed with the LE can then be imported in the BGE for execution.

The LE's GUI is depicted in Fig. 5.1. The general settings are shown on the left and top sides (1). The center area (2) contains several tabbed panels: each panel represents an object and it can be used to define the object's behavior by means of the said blocks. Lastly, a list representing the available 3D objects is shown on the right side (3). It can be observed that a per-object visualization of the application

[13]Blender: https://www.blender.org/

logic is adopted, which is similar to the one used in BGE.

The recognition of events is performed through the so called "Gesture" and "Wait for" blocks: the former block type is meant to support the definition of user interactions replacing the BGE's sensors, whereas the latter block type is used to support the message exchange making the object waits until a new message is received from another object. The BGE's controllers are removed and the support for scripted logic is not provided, since the given target users of the tool would reasonably lack programming skills. Actuators are the same provided by the BGE. However, a new design of the "Message" actuator was introduced to support multiple destinations (still requesting user configuration), since the communication with different objects is very common in interactive applications.

A user study aimed at comparing the LE and the BGE was conducted in [266] by involving three user categories, i.e., beginner, intermediate and skilled (based on their previous knowledge of Blender). Results obtained in the creation of interactive applications with the two tools revealed that beginner and intermediate users were not capable to complete the assigned task with the BGE, which was judged as difficult to use since it leveraged a very sophisticated notation and interaction paradigm for defining the application logic. Skilled users completed the task with both the tools: however, they were faster with the LE than with the BGE, even though they already knew the latter tool.

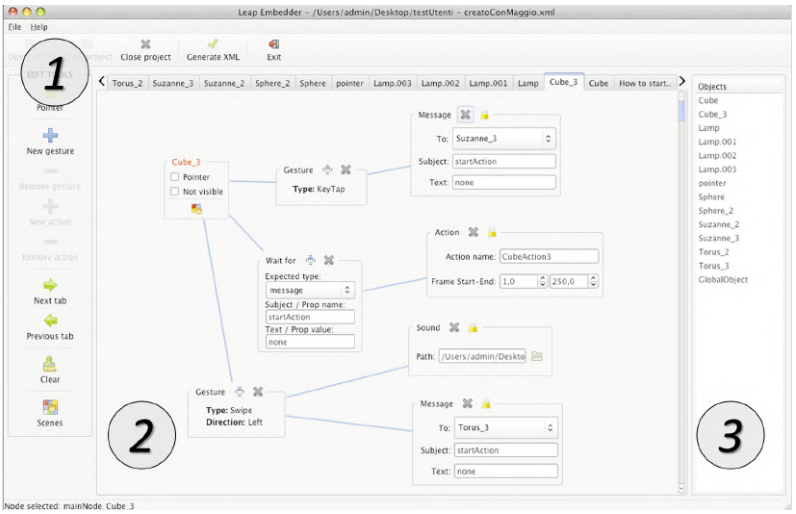


Figure 5.1: Interface of the LE.

5.3 Visual Scene Editor design

This section discusses the main limitations of the LE and introduces the solutions that have been devised to cope with them during the design of the VSE.

5.3.1 LE limitations

In order to identify possible weaknesses of the LE, two experts in the computer interaction field with years of experience in using 3D computer graphics software (e.g., Blender, Autodesk's Maya, etc.) and video-game/interactive application development environments (e.g., Unity, Unreal Engine, etc.) were requested to carry out the same tasks described in the user study of [266]. At the end of the experience, they were asked to provide comments aimed to determine possible aspects of the LE that could be considered to enhance the performance of unskilled users as well as the usability of the tool.

A first concern regarded difficulties in distinguishing components that should be used to define the application logic. It was also lamented the huge amount of parameters to be configured in each block (basically moving the original complexity of the BGE there). In fact, although the LE design makes it easier for inter-object (especially one-to-many) communications, the used approach is still based on messages that have to be configured explicitly by the users. Finally, the object-centric approach adopted in LE was criticized; the effects of adopting this approach do not impact only on the application logic, which is distributed into single objects, but it also affects the visualization which is "centered" (like in the BGE) on a the selected object. Although the possibility to distribute the application logic could be regarded as a plus from a programmer's perspective, its effect on the visualization is that it does not allow users to observe the behavior of multiple objects at the same time, making it difficult to define and control relations among them.

Based on the above comments, three possible areas were identified which requested intervention:

- *visual elements identification*;
- *inter-object communication*;
- *relations visualization*.

Areas, and the solutions proposed to cope with the limitations described, are detailed in the following sections.

Visual elements identification

With the aim to improve the recognition of the visual elements, it is possible to draw inspiration from VPL-based tools that leverage visual blocks. Such tools take advantage of shapes and colors to make the users capable to clearly distinguish the user interface's elements and logic's components. Text labels usually appear on the shapes to describe their function, whereas colors are used to group the shapes that share a similar purpose. Furthermore, such tools offer the possibility to organize the layout of the interface elements by using panels and windows.

Several tools characterized by this feature already exist. Examples of tools based on shape and/or color are miniBloq[14], ToonTalk [152], Snap![15], etc. Among them, it is possible to identify the tool that is indeed attracting most of the attention, i.e., Scratch [203].

Scratch was created to support primary school students to acquire computer programming skills by creating 2D graphics programs. Scratch's notation, which has been already used for developing interactive exhibits [293], is based on different colored blocks, which can be combined to form a structured "script" implementing a functional program. The composed script can be visualized in the right panel (1) shown in Fig. 5.2. Each block denotes a specific operation and its shape indicates which are the other blocks that can be connected to it. Blocks are grouped into several "categories", each represented with a specific color that allows users to clearly distinguish them. Categories, and individual blocks available for a specific category, are shown in the center panel (2) in Fig. 5.2. The left side of Scratch's interface contains the "stage" area, which allows users to visualize a preview of the final application (3) and the "sprites" (4), i.e., the 2D graphics elements that can be used/controlled in the application.

The effectiveness of Scratch's notation and the panel-based organization of its interface have been already investigated in the literature. For example, the work in [238] presents a comparison between Scratch and a traditional programming environment based on text language. Experimental results showed that unskilled users preferred to generate applications with Scratch, expressing interest in continuing to develop with such a tool. Similar outcomes were achieved in [263], where benefits of using Scratch were confirmed for both learning coding concepts and developing computational thinking abilities.

Inter-object communication

As mentioned, the message exchange approach in the LE still required users to manually configure parameters of the messages (e.g, their payload, sender, receiver(s), etc), making the communication between objects very cumbersome. This limitation can be overcome by adopting the *event-condition-action* approach proposed in [75].

Examples of tools making use of this approach are, e.g., AgentCubes [137], GameSalad[16], Click Team[17], etc.

[14] miniBloq: http://blog.minibloq.org/

[15] Snap!: https://snap.berkeley.edu/

[16] GameSalad: https://gamesalad.com/

[17] Click Team: www.clickteam.com

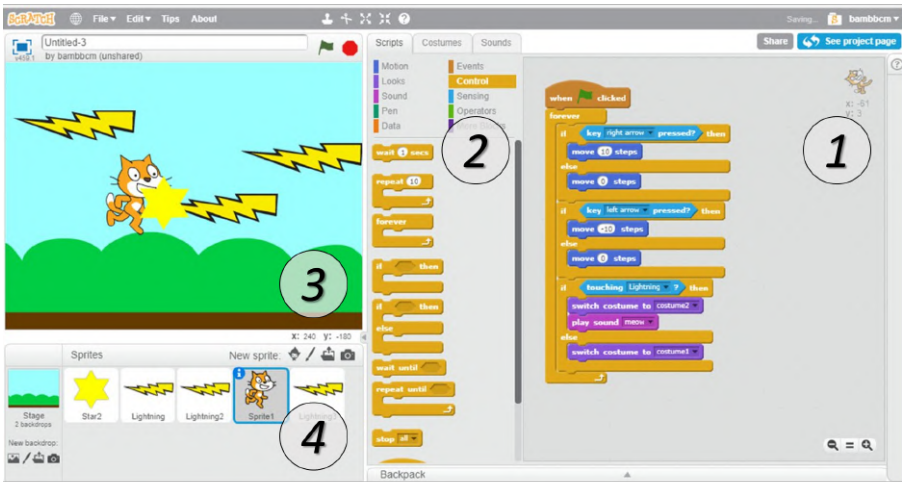


Figure 5.2: Interface of Scratch.

Among the various tools available, the solution adopted in Kodu[18] was considered. This tool is a VPL-based environment that has been developed by Microsoft Research with the aim of making video-game programming accessible to children. Kodu lets users create interactive 3D scenarios by also leveraging sophisticated features typical of professional game engines, like, e.g., camera control, collision detection, etc. [202]. The possibility to use Kodu not only as a tool for programming video-games but also to learn more general computer science concepts has been studied and its effectiveness evaluated in prior work [292]. One of the main aspects of Kodu is that it is completely event-driven [95]. Users are allowed to define so-called "rules" (1), which are evaluated according to the "if this happens, do that" paradigm. As shown in Fig. 5.3, rules are specified using graphics "tiles" (2), which represent the building blocks of the language. Tiles can be combined by assembling them into a "When-Do" strips. As with the Scratch's scripts, the Kodu's rules can be assigned to active objects. The "+" operator (3) is used to connect the "When" part of a rule to its "Do" part: when the events specified by the tiles on the left side of the "+" operator occur, the actions on the right side are executed.

Relations visualization

As mentioned, both the LE and the BGE leverage an object-centric interaction paradigm, which makes it difficult for users to visualize/understand the relationships among objects. In order to overcome this limitation, a different paradigm was

[18]Kodu: https://www.kodugamelab.com/

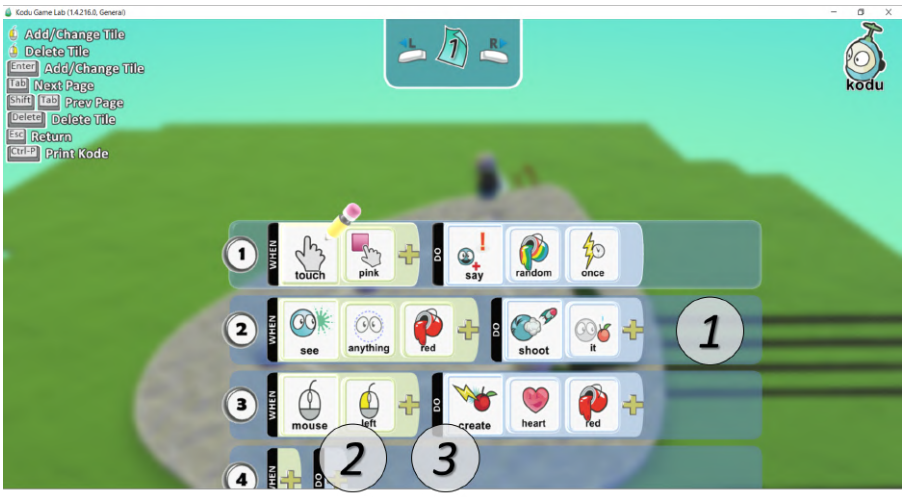


Figure 5.3: Interface of Kodu.

considered in the design of the VSE named scene-centric, which was expected to ease the modality in which users can take a look at the communications occurring among objects.

This approach provides the user with a high-level representation of the application being developed, referred to as "scene". A scene can be considered as a container of 3D elements that the users can possibly interact with at a certain time when the application is executed. Besides serving as a visual container for the virtual objects, a scene is also expected to be an alternative way to organize the user's work. Each project can be made up of multiple scenes, each containing one or more interactive/not-interactive assets. It is worth observing that the concept of a scene as a 3D objects' container is used also in the LE and the BGE. However, these latter tools do not offer the possibility to understand if the behavior being developed by the user refers to an object contained, e.g., in scene 1 or scene 2. In fact, this information has to be remembered by the users while developing; the alternative for them is to continuously switch among the windows describing the scenes and the windows used for specifying the objects' behaviors.

5.3.2 Interface design steps

Considering the limitations of the LE described in Sec. 5.3.1, a new tool was designed to let unskilled users develop 3D interactive applications in an effective, efficient and satisfying way. A new GUI, and a new graphics notation, were designed and developed by considering the feedback provided by the experts after their experience with the LE, the outcomes of the literature review focusing on specific tools and the established usability principles proposed in [227].

The GUI of the VSE and its visual notation can be considered as the final

result of an interactive process that encompassed continuous refinements and improvements. The first mockup, which is depicted in Fig. 5.4, was based on Scratch's panel-based layout. Moreover, since its first mockup, the GUI was designed considering the Kodu's "When-Do" paradigm, represented in Fig. 5.4b.

Different content organizations (e.g., the position of contents in panels, the links between them, etc.) and interaction methodologies (e.g., drag-and-drop, selection from a list, etc.) were examined and modified when considered as not appropriate, reaching the final version of the main windows illustrated in Fig. 5.6 (more details on the final GUI will be provided in the following).

For instance, one of the concerns affecting the first mockup was related to the difficulties associated with configuring the parameters required for the definition of the "Interface Logic", i.e., the set of relationships between events recognized and actions to be activated. This problem was associated to the limited space allocated in the GUI to perform this operation (bottom-right panel in Fig. 5.4b). In the final implementation, an adjustable panel was devoted to this purpose. Another issue was associated to the difficulty of spotting the interactions among different objects (defined by the wires which connect the "When" block of an object to the "Do" block of another one). The impact of this limitation, already present in the strips used by Kodu for defining the rules, was increased by the message-exchange approach adopted in the BGE and the LE. In fact, when dealing with messages, users are required to specify (and remember) the content of the messages to send to target objects, without the possibility to visualize them at the time of defining the messages themselves. In order to cope with this issue, it was decided to leverage the link-and-wire paradigm that lets users specify relations (later referred to as "Links") among "When-Do"-based blocks defined for the different objects. Because of this decision, a different content organization was adopted which allows users to visualize more than one object at the same time in the said scene-centric paradigm.

Another aspect disregarded in the first mockup was the possibility to make use of colors as suggested by the Scratch's interface. Their usage in the VSE's GUI was introduced to make it easier for the users to recognize a specific scene (and consequently the inter-objects relations) and to visually distinguish "When" buttons from "Do" ones.

5.4 System overview

This section illustrates the overall architecture of the VSE tool, by describing in detail a typical usage workflow as well as the GUI's components.

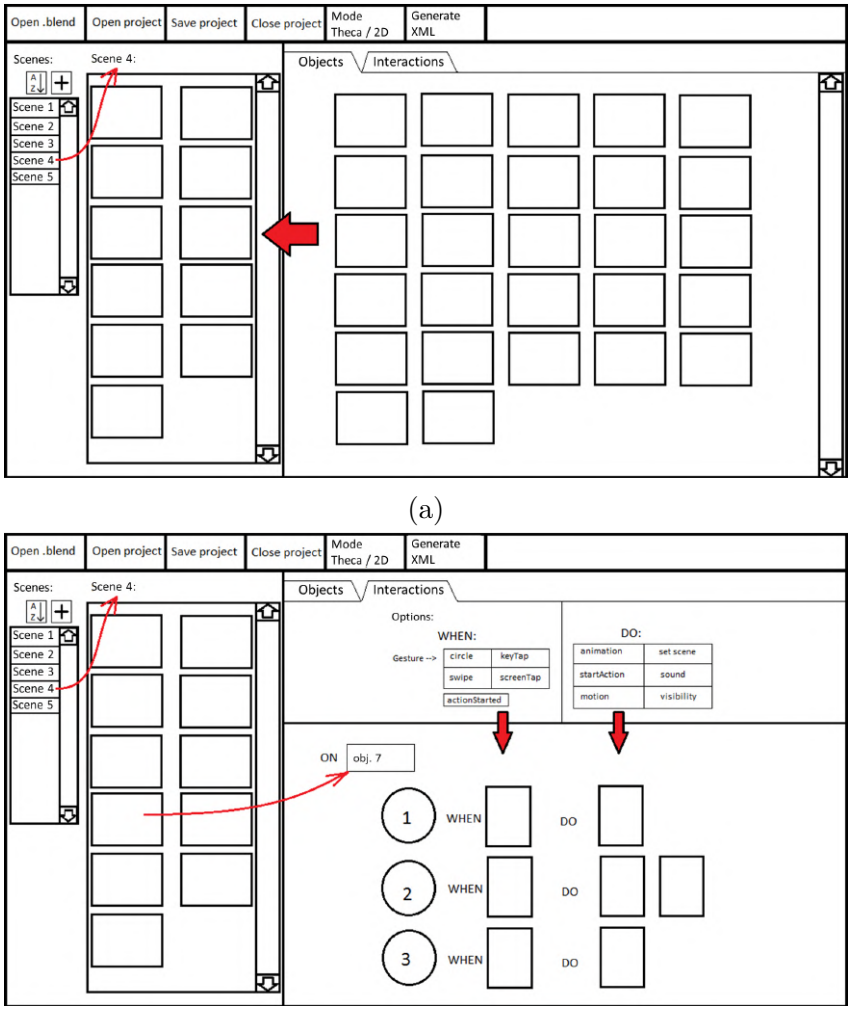


Figure 5.4: Initial mockup of the VSE's GUI.

5.4.1 Architecture and usage workflow

The high-level architecture of the tool and the expected usage workflow are shown in Fig. 5.5. The development of the interactive applications takes place through the steps reported in the following.

First, a file "*.blend*" including assets (like video and audio clips, text descriptions, static or animated 3D models, etc.) is loaded. No programming skills are required for the generation of the assets, since this step involves the use of modeling and animation suites like, e.g., Blender or Maya. In principle, any graphics tool can be used to generate the assets, since the only requirement is that they can be exported to a Blender- (VSE-) compatible *.blend* file or, alternatively, that they can be saved using a filetype supported by Blender. Assets can be also obtained

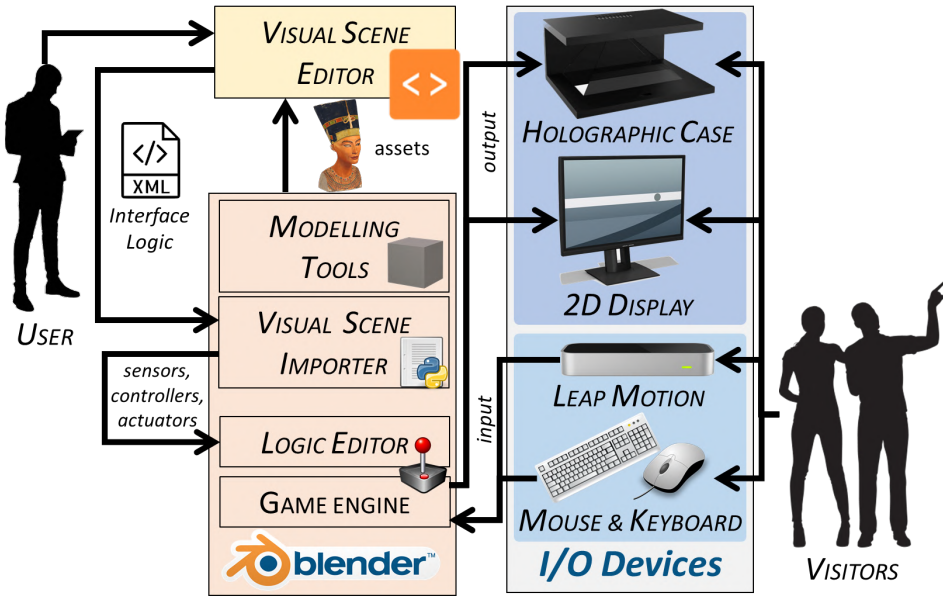


Figure 5.5: High-level architecture and usage workflow of the VSE.

from a number of online catalogues, e.g., BlendSwap[19], Free3D[20], and Blender for Architecture[21] (for 3D models), or Bendsound[22] and Free Music Archive[23] (for audio files). Imported assets will form the library of resources that can be possibly included in the 3D interactive application being created.

When the library is loaded, the user can start to arrange the available objects into the scenes, by selecting objects to be available for interaction at a given time. Once the scenes are defined, the user can specify the mechanisms to make the application move from one scene to another, and the behaviors (like the playback of animations or the appearance of a text description) to be activated when specific events (like an input provided by the user, a collision between two objects, etc.) are recognized.

Once the user completes the definition of the Interface Logic (or part of it), he or she can export the project to a XML file. This file can be leveraged later both to introduce changes in the application logic using the VSE, or it can be imported in Blender to visualize the interactive application through the BGE (for previewing, e.g., during development, testing or deployment). An add-on, named Visual Scene

[19]BlendSwap: https://www.blendswap.com/

[20]Free3D: https://free3d.com/it/3d-models/blender

[21]Blender for Architecture: http://blender-archi.tuxfamily.org

[22]Bendsound: https://www.bensound.com/

[23]Free Music Archive: https://freemusicarchive.org/

Importer, which is developed in Python, is in charge of the import operations. This add-on automatically converts the Interface Logic defined with the VSE into the visual description based on sensors, controllers and actuators which is used by the Blender's Logic Editor.

Once the import is terminated and the corresponding sensors, controllers, actuators and needed connections have been created, the user can further apply changes to the scenes' layout or to the Interactioon Logic through Blender's native windows (e.g., the 3D View and the Logic Editor). It is worth noting that, although the current implementation of the framework considers only the BGE to run the interactive application, other frameworks/game engines, including Unity and Unreal Engine, can be supported in the future, without the need to introduce any change in the VSE. In fact, a new framework can be integrated by re-implementing only the script responsible for translating the XML file generated by VSE into the target application logic (i.e., the Visual Scene Importer). A different add-on with respect to that used by the LE was developed since VSE adopts a more sophisticated export format.

A real-time rendering of the interactive application generated automatically by the BGE can be displayed through the selected output devices. At present, a 2D display and a holographic case (more details are provided in Section 5.5) have been considered, although other output devices based, e.g., on stereoscopic displays, VR-/AR- systems, etc., can be easily supported.

The current implementation lets the users interact with 3D assets by leveraging hand gestures or traditional interfaces (like M&K). However, supporting alternative interaction methods that can be managed by the host computer (like body gestures, voice commands, etc.) is possible as well, since the BGE can be fully scripted.

From Fig. 5.5, which shows the high-level architecture (and expected usage workflow) of the devised framework, it can be observed that the VSE can fully replace the LE tool proposed in [266].

5.4.2 GUI's functionalities and visual notation

As said, the development of the VSE's GUI passed through an iterative process that lead to the final version shown in Fig. 5.6. The core components of the GUI are described in the following.

Library

When the system has loaded the *.blend* file or the XML file representing an existing project, the GUI displays, on the right side, a list containing all the objects which can be inserted into the scenes (1). Each object is represented trough a small icon (showing a preview of the object's appearance when rendered in the 3D program) and a text label indicating its name. The list includes not only 3D

meshes (like Cube, Sphere, Suzanne, and Torus, in Fig. 5.6) but also, e.g., lights (like, Lamp, Lamp.001, etc.) which could be added in the scenes to define a given lighting. Like in traditional graphics suites, if a given object presents a hierarchical relationship with other objects (i.e., it has one or more child objects, which inherit transformations applied to their parents), then a tree-based visualization is used to represent it within the library. When the user inserts the parent object to the scene, all its children are automatically added as well. In order to filter objects based on their name, the user can take advantage of a search bar placed on the top part of the library. When an object is selected (like the Torus in the figure), a larger version of the preview can be visualized on the bottom-right side of the GUI.

Scenes

As said in Sec. 5.3.1, the use of the object-centric approach introduces a number of limitations. For this reason, a scene-centric visualization is leveraged which provides the users with the possibility to visualize, at development time, the high-level view of an entire "scene". A different color is associated to each scene in order to help the user to visually assign objects to scenes and easily recognize scene-to-object relations.

Like the first mockup (inspired by Scratch), also the final version of the VSE's GUI is split into several panels, each providing different functionalities. This way, users have the possibility to visualize all the panels at once, without having to switch among different windows or tabs (as in the LE or other tools, like, e.g., that in [102]).

The scenes, as well as the Interface Logic defined for them, are represented in the panel labeled as (2) in Fig. 5.6. Depending on the complexity of the project, the number of scenes and assets could make it difficult to simultaneously visualize all the scenes with their assets and relations. For this reason, the VSE supports two scene visualization types: collapsed or expanded. However, only a single scene at a time can be expanded letting the user configure it. The collapsed visualization, applied for example to the scenes named Schema, Start and Scene_2 in Fig. 5.6, shows only the name of the scene, the color assigned to it, and two buttons that allow the user to delete and edit it. Once the scene is visualized as expanded (like scene Scene_3), all the remaining scenes are automatically rearranged in the panel in order not to be hidden by the current scene being manipulated.

On the top side of an expanded scene (3), a toolbar is shown in order to provide the user with several configuration functionalities. For example, he or she can specify the color to be assigned to the scene (which will be also used for representing Links) and set its name (by default, colors and names are automatically defined by the system). A checkbox lets the user set the current scene as the first to be visualized when the BGE starts the execution of the interactive application. The boundaries of the scene set as start scene are highlighted with the same color

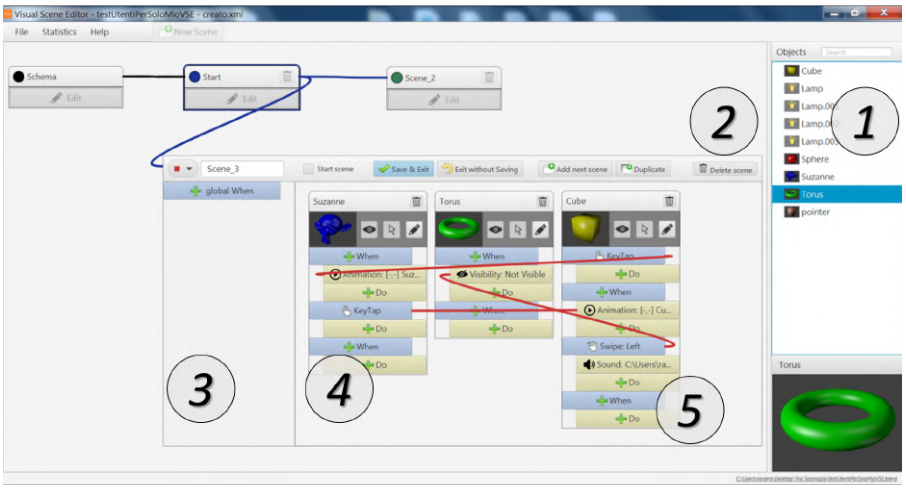


Figure 5.6: Final implementation of the VSE's GUI: 1) object library, 2) scenes, 3) selected scene, 4) scene's objects, 5) object's "When-Do" blocks, 6) menu bar, and 7) status bar.

assigned to it (as it happens for the scene named Start in Fig. 5.6). A button on the right side of the toolbar can be used to duplicate the current scene, creating a copy containing the same objects and Interface Logic.

In the region labeled as (4), the user can control how the objects appear in the given scene and their behavior can be defined by using the "When-Do" blocks (5). A smaller panel (on the left side of this region) lets the user define global behaviors enabled for the whole scene. Functionalities that can be accessed through a dedicated button will be discussed later.

Objects

To insert a new object in the scene, the user is requested to drag-and-drop in the scene the desired object by selecting it from the library. As for the scenes, objects can be represented either with a collapsed or expanded visualization. However, in this case, more than one object can be visualized as expanded at a given time. This way, the user can define (and see at the same time) relations among objects.

The collapsed visualization shows only the name of the object and the two buttons that can be used to delete it and to activate the expanded visualization.

In the expanded visualization, a title bar is shown on each object presenting its name and the delete button. Under the toolbar, a preview of the object is shown, together with three buttons. The first button is meant to modify the object's visibility in the scene. The second button makes the object a *pointer*, i.e., the visual representation of the target of user's interaction, for the current scene. For example, in the current implementation, if the user selects the mouse and the 2D

display as input and output devices, respectively, the movements of the mouse are automatically transferred to the pointer object and the traditional arrow cursor is replaced with the pointer object itself. When the holographic case and the hand tracking input are selected, the position of the cursor is managed by the 3D position of the tracked user's hand. The last button enables the collapsed visualization for the object.

The bottom part of each object can be used to set the Interface Logic through the "When-Do" blocks. By default, each object presents an empty "When-Do" block pair. When the user specifies a condition by clicking the corresponding "When" button, a new block pair is automatically created; moreover, if an action is defined for a "Do" block, a new "Do" block is automatically added under the same "When" block. Drawing inspiration from the different shapes used by Scratch to denote specific operations, it was decided to use diverse types of indentations for the "When-Do" in order to provide the user with a clear distinction between conditions and actions.

When the user has set a "When-Do" block, an icon and a short description of the selected action/condition appear on the button (for example, the KeyTap event for the "When" block set for the Cube object, or the Animation playback of the "When" block set for the Suzanne object, in the Fig. 5.6).

The types of events recognized as well as the actions integrated in the VSE will be presented in Section 5.4.2.

Links

The "When-Do" approach detailed above and adopted also in Kodu can be used only when the conditions to be recognized concern the same object on which a given action has to be activated. For this reason, it was decided to adapt the Kodu's event-driven approach by proposing the use of the *linking-and-wiring* paradigm to replace the Kodu's rule strips. This paradigm was leveraged in many environments, like AudioMulch[24], Audulus[25], Nuke[26], Node-RED[27], and SpaceBrew[28], among others, and its effectiveness has been already investigated in various domains [193, 287]. The approach relies on the concept of Link, which, as said, has been introduced in order to overcome the inter-object communication method based on the message exchange approach adopted in the BGE and the LE.

Links are lines that enable inter-object communications, which can be used to

[24] AudioMulch: `http://www.audiomulch.com/`

[25] Audulus: `audulus.com/`

[26] Nuke: `https://www.foundry.com/products/nuke`

[27] Node-RED: `https://nodered.org/`

[28] SpaceBrew: `https://docs.spacebrew.cc/`

activate an action associated to an object that is different from the one which detects a given condition, thus simplifying the creation of the logic. A new Link can be defined by drawing a line (with a drag-and-drop operation) that connects the "When" block of an object to a "Do" block of another object. This mechanism simplifies also the communication of events that involves multiple objects, since it is possible to define Links that connect the same "When" block of an object to several "Do" blocks of different objects.

Using Links, the user can also trigger the change of a scene, or reload it; in this case, the link has to be defined between the "When" block of an object and the corresponding scene's toolbar.

The color used to represent a Link is the same as the color used to draw the scene that contains the object that the connection originates from. For instance, in Fig. 5.6, all the Links between objects of Scene_3 are red since they start from an object belonging to that scene (whose color is red), whereas the Link that has been set to move from the Start scene to Scene_3 is blue, since it starts from an object (not visible in the figure, due to the collapsed visualization) included in the former scene (which is assigned the blue color).

A right mouse click on the "When-Do" block displays a menu that lets the user delete the block and/or remove the outgoing Links.

Templates and scene behaviors

In the left-top corner of the scene panel (2) in Fig. 5.6 it is possible to observe the presence of a scene named Schema. This scene can be considered as a sort of template which can be leveraged by the user to specify objects/behaviors that are present/valid in/for all the scenes without the need to repeat their definition in each scene. It is automatically generated when the user creates a new project, and it can be visualized/managed as a normal scene. The difference with respect to the other scenes is that the Schema scene cannot be renamed, deleted or moved. Furthermore, the scene cannot be duplicated or set as the initial scene. A practical use of the Schema scene could be that of defining the same lighting conditions for all the scenes or implementing the logic for controlling the cursor. If two different behaviors have been defined for the same object in the Schema scene and in a normal scene, the definition in the Schema scene is disregarded (overridden).

As anticipated, it is also possible to configure a so-called "GlobalWhen" blocks for each scene. These blocks are used to manage events that involve the whole scene and not just a particular object that belongs to it. As matter of example, the action of modifying the currently displayed scene when a specific gesture is performed on a given object is a typical operation that involves the use of a canonical "When-Do" block. However, if the user is interested in changing the current scene independent of where the gesture is actually performed, he or she can delegate the definition of this condition to the "GlobalWhen" block.

When-Do blocks

Dedicated windows have been developed in order to let the user configure each "When-Do" block. For what it concerns the "When" condition, the interface presents three tabbed panels (Fig. 5.7), grouping semantically related conditions to be recognized. The user can specify the hand gesture to be recognized in order to trigger an event through the Gesture panel (Fig. 5.7a). At present, the recognition of hand gestures is based on the Leap Motion sensor. Hence, the panel presents only the four gestures supported by this device, i.e., circle, swipe, key tap and screen tap. The documentation provides more details about the gestures supported by the Leap Motion SDK[29]. Mouse events are mapped onto recognized gestures.

The Timing panel (Fig. 5.7b) can be used to consider conditions influenced by time. In particular, the OnLoad option is used to trigger an event when the scene is loaded (if specified in a GlobalWhen block) or when the object is inserted into the scene (in the case the condition is specified in the "When" block of an object). Continuous events are triggered when the user specifies the Always trigger, whereas Delay triggers the event after the specified time interval.

The Proximity panel (Fig. 5.7c) is intended to control events that consider spatial conditions (two objects that collided or are close to each other at runtime).

The actions to be executed upon recognition of an event can be configured through the "Do" block by means of a window that includes two tabbed panels (Fig. 5.8).

The Action panel (Fig. 5.8a) is meant to manage three types of actions to be executed on the object (the playback of an animation, the reproduction of a sound and the change of the object's visibility) and the corresponding parameters (the name of the animation or the file of the sound to be played, the starting and end frame of the animation, and whether the object is visible or hidden).

The Objects panel (Fig. 5.8b), can be used to define actions that involve the introduction/removal of objects to/from the scene. Intuitively, the Delete option cancels the object, Add inserts a new object in the same location of the current object, whereas Replace removes the current object and adds a new one (specified by the user) to the scene.

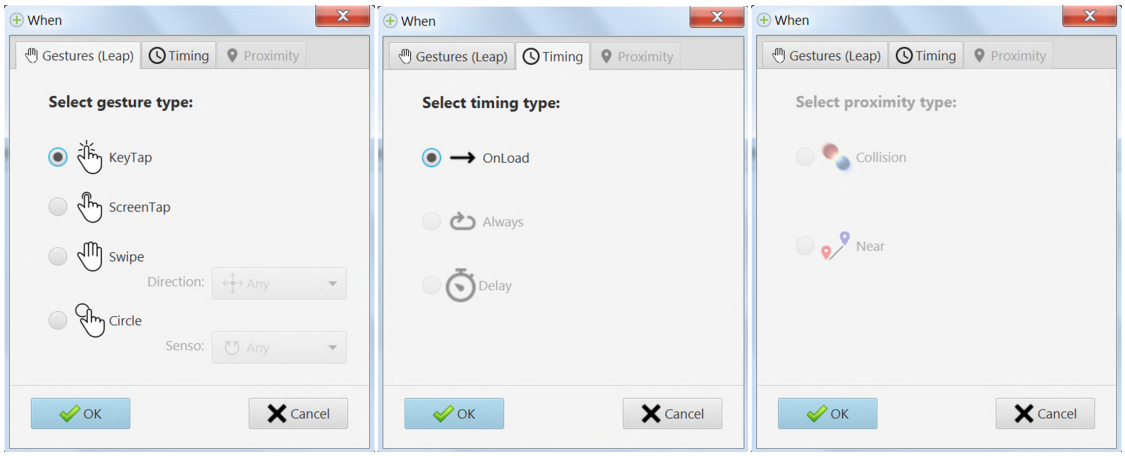
5.4.3 Software modules

The VSE tool has been developed using the JavaFX graphic library[30]. The Model View Controller (MVC) design pattern was used to structure the software modules that are shown in Fig. 5.9.

The View contains the following modules:

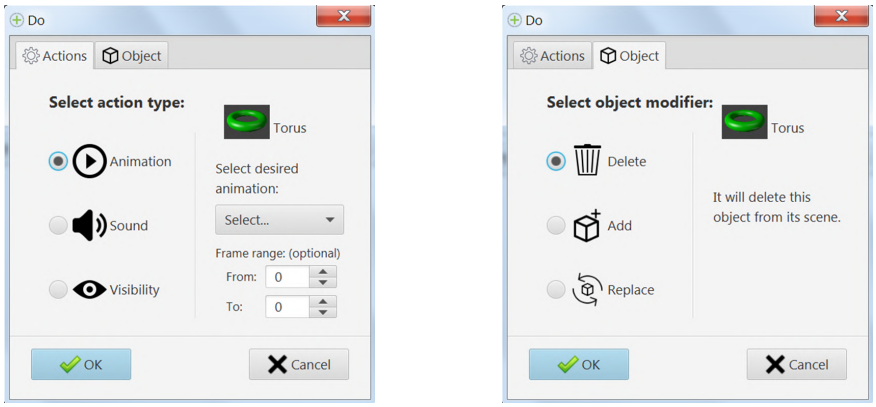
[29]Leap Motion SDK documentation: https://bit.ly/2XOMXHP

[30]JavaFX: https://www.oracle.com/technetwork/java/javafx/overview/



(a) Gesture panel (b) Timing panel (c) Proximity panel

Figure 5.7: Window for specifying the parameters of a "When" block.



(a) Action panel (b) Object panel

Figure 5.8: Window for specifying the parameters of a "Do" block.

- Project Window: it handles the graphics components regarding the main window;
- Scene Container: it is the visual container that the user can leverage to assemble the current scene;
- Node: it represents one of the assets added to the current scene and configured by the user;
- When/Do Block: it lets the user configure and inspect each of the "When" conditions and "Do" actions for a given asset.

The Model includes the modules listed below:

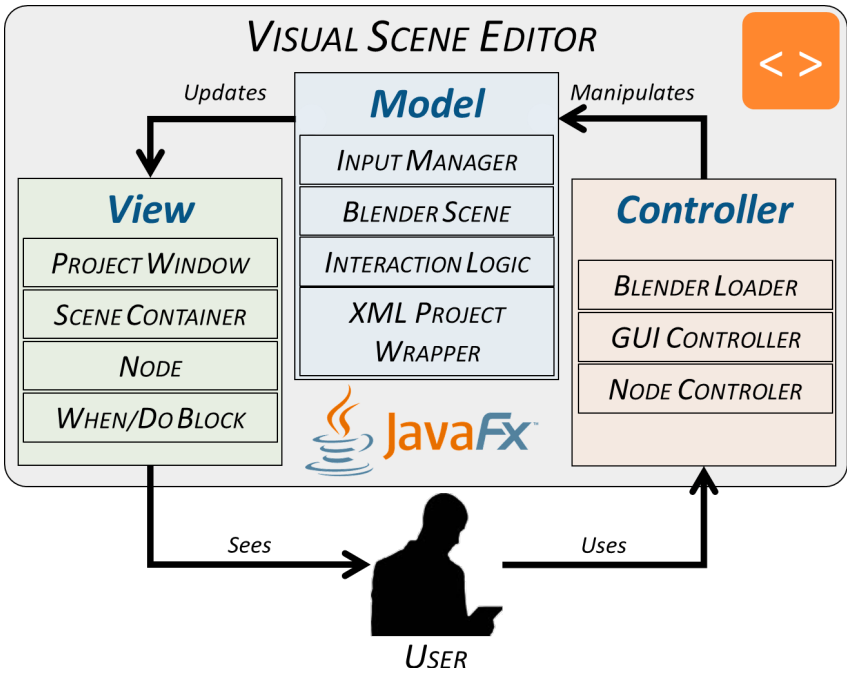


Figure 5.9: Architecture of the proposed VSE tool: software modules.

- Input Manager: it is the data structure used to describe the possible input device(s) to be handled by the interactive application;
- Blend Scene: it includes information regarding the library of objects that can be considered in the development of the interactive application;
- Interface Logic: it is the data structure containing the logic of the application being developed;
- XML Project Wrapper: it holds the information needed to generate the XML file exported from a VSE project, whose data are represented in a format that can be loaded in the BGE.

Finally, the Controller relies on the modules below:

- Blender Loader: it includes functionalities required to import into the VSE the assets created with Blender (or exported in a compatible format);
- GUI Controller: it manages the behavior of the components in the main window;
- Node Controller: it is responsible for controlling the assets configuration.

Figure 5.10: Overall setup of the system used for the virtual exhibition of Queen Nefertiti's bust.

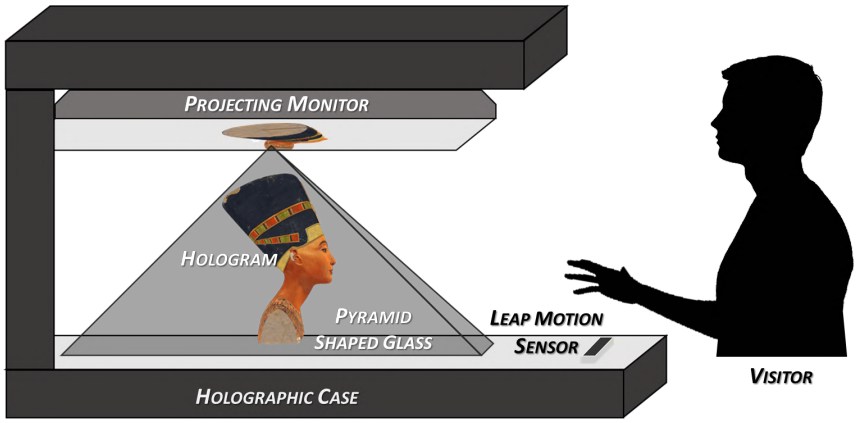


Figure 5.11: Schema of the Pepper's ghost effect used for the holographic case.

5.5 Use case

An interactive application targeted to a possible virtual exhibition of a known artwork was developed in order to present the capabilities of the VSE in a real-world scenario. To this purpose, an ancient Egyptian artifact, namely the bust of Queen Nefertiti exhibited at the Berlin's Neues Museum was considered. The assets include the animated 3D meshes of the artifact and of part of its (interesting) details, text descriptions of the historical background and videos.

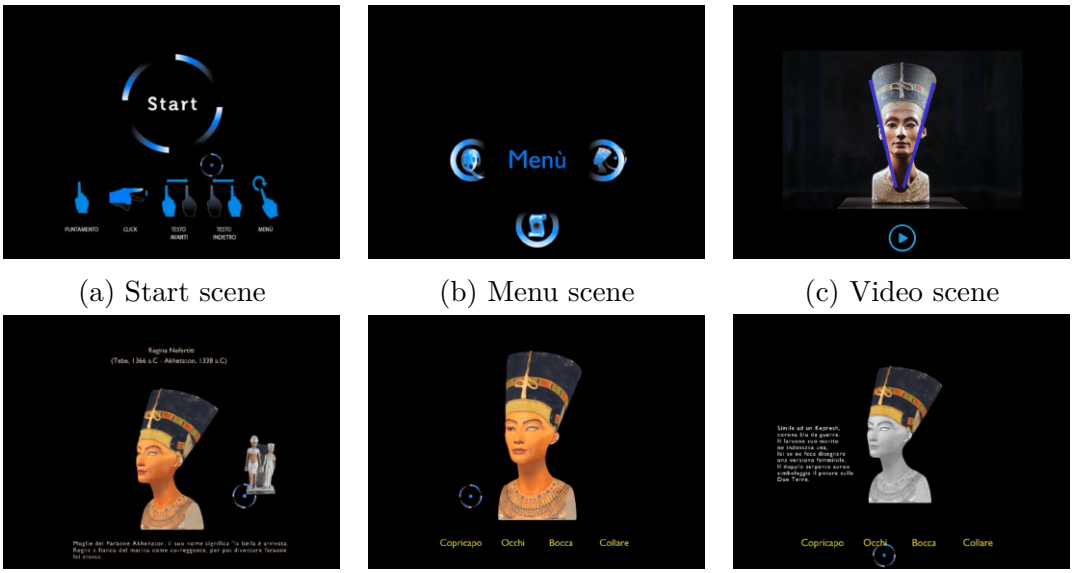
The setup considered for the use case is shown in Fig. 5.10. The output device selected is a custom-made holographic case. This device leverages the well-known Pepper's ghost effect (Fig. 5.11), which is able to recreate 3D objects by projecting digital images on a pyramid-shaped glass using a display (the projecting monitor) which is hidden to the visitors in the top of the case.

User interaction relies on hand gestures tracking. A Leap Motion sensor is mounted in the case and is used to collect tracking data representing the user's hand movements in the 3D space in front of it.

The interactive application developed with the VSE consists of nine scenes, which are described in the following.

- Start (Fig. 5.12a): it shows a brief description of the gestures recognized by the system that let the visitor interact with the 3D contents; once the visitor executed a key tap gesture on the Start label represented in the center of the holographic space, the application is started and the next scene (Menu) is shown.
- Menu (Fig. 5.12b): it can be used by the visitor to select the next scene to be visualized by choosing it among three alternatives: Video, History and 3D Model; a key tap performed on one of the corresponding icons loads the new scene, whereas a circle gesture lets the visitor return to the Menu scene.
- Video (Fig. 5.12c): it displays a video describing the artwork.
- History (Fig. 5.12d): it holds a sequence of textual descriptions, which introduce the history of Queen Nefertiti; texts can be scrolled using a swipe gesture.
- 3D Model (Fig. 5.12e): it presents an animated 3D model of the Queen Nefertiti's bust; the animation, which is automatically played when the scene is loaded, makes the bust rotate clockwise; a tap gesture performed by the visitor on the labels indicating the collar, eyes, mouth and head displayed on the bottom of the holographic projection loads a new scene where the selected part is highlighted on a grayed model using colors (see below).
- Collar, Eyes, Mouth and Head (Fig. 5.12f): the part of the artwork that is selected by the visitor in the previous scene is highlighted, and a text description is shown presenting further information about the details of the bust: a circle gesture allows the visitor to close the current scene and return to the 3D Model scene.

The behaviors defined in the interactive application and the presentation of this specific artifact have been chosen since they represent an interesting example of how scenes can be organized to showcase different types of assets, i.e., graphics widgets (like buttons, panels, labels, etc.), 3D models (the model of the Queen Nefertiti's bust and its details), animations (e.g., to rotate the bust in the 3D Model scene), and videos (as in the Video scene).



(a) Start scene (b) Menu scene (c) Video scene

(d) History scene (e) 3D Model scene (f) Head scene

Figure 5.12: Representative scenes created for the Queen Nefertiti's use case (the Collar, Eyes and Mouth scenes are similar to the Head scene).

A video is available for download[31] showing the generated interactive application. Three videos presenting the development of the application by means of the BGE, the LE, and the VSE tools are also available for download[32], in order to qualitatively compare the different levels of complexity of both the development process and the resulting logic.

5.6 Experimental evaluation

In order to asses the performance of the VSE with respect to other tools, two experiments were conducted by involving participants characterized by different levels of expertise in (interactive) application development and design of public exhibitions. In the following, the composition of the user groups for the two experiments is first described. Then, the tasks to be performed and the experimental procedure pursued are presented. Lastly, the criteria considered in the evaluation are detailed.

[31] Video of the resulting Queen Nefertiti application: http://tiny.cc/bydtbz

[32] Queen Nefertiti application, video of the creation process: http://tiny.cc/axdtbz

5.6.1 Experiments and participants

The first experiment is a preliminary study (later referred to as S1) intended to set a sort of baseline for the evaluation of the tools' performance. It involved three expert users in computer programming, precisely, 3D game/application developers with years of experience with several languages and graphics suites, including Blender and the BGE.

The second experiment was intended as a user study (later referred to as S2) to investigate tool effectiveness and usability with possible end-users. In this case, the user group consisted of 14 volunteer participants aged between 22 and 29 years ($M = 25.93$, $SD = 1.94$), selected among students enrolled in the B.Sc. degree on Design and Visual Communication at Politecnico di Torino[33] in Turin, Italy. They were expected to be skillful in the fields of visual communication and product design, with basic programming expertise and no experience with the considered tools. Based on available statistics, during the degree program (e.g., in internships, as well as in course and thesis projects) and after graduation, it is expected that many of them will be involved in working activities in the field of the considered scenario.

5.6.2 Tasks

The interactive application to be created in both S1 and S2 was a simplified version of the planner tool available on the IKEA website[34]. The original tool lets IKEA's customers design/customize a sofa by assembling a number of components, e.g., loungers, poufs, etc. The user can decide the size and location for each component and, once completed the assembly, can observe an animation for the created product.

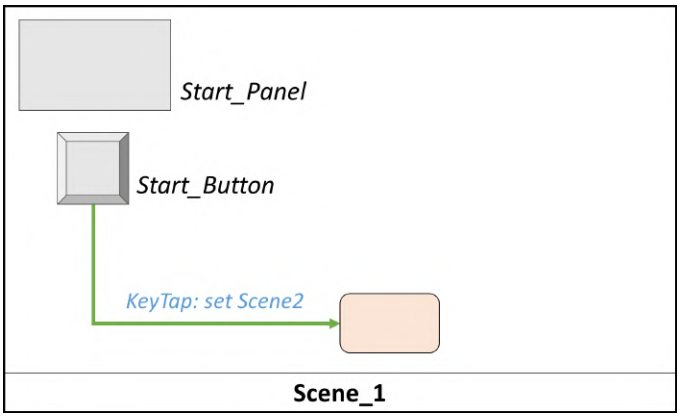
Participants were tasked with developing three scenes (labeled from 1 to 3), each including various interactive 3D objects. In each scene, they were asked to define various conditions (corresponding to different user interactions) and enable corresponding actions (i.e., changing the visibility of an asset, playing an animation, etc.).

Fig. 5.13 illustrates a high-level representation of the application's workflow, whereas Fig. 5.14 shows a set of screenshots representing the resulting application when imported and visualized in the BGE.

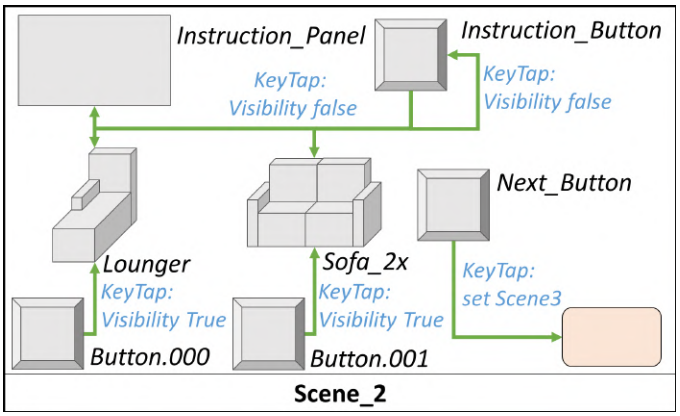
Scene1 (Fig. 5.14a) represents the start scene, and can be considered as an introduction to the actual application. As shown in Fig. 5.13a, the scene includes only two objects: Start_Panel and Start_Button. The interaction to be defined in

[33]Design and Visual Communication B.Sc.: https://didattica.polito.it/laurea/design/en

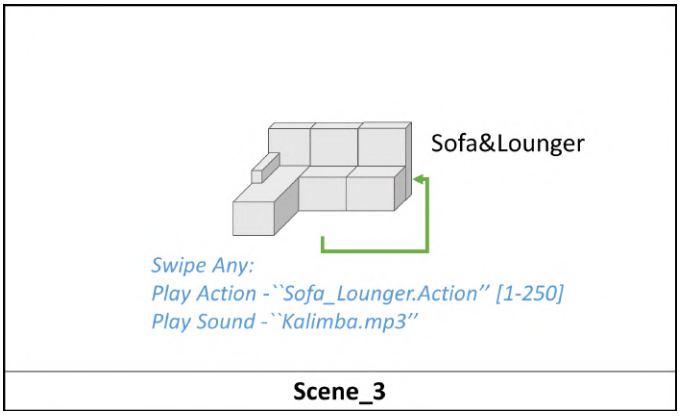
[34]IKEA's planner: https://bit.ly/2OsIG6P



(a) Scene1



(b) Scene2



(c) Scene3

Figure 5.13: Overview of the scenes to be created.

this scene is a key tap gesture on the Start_Button, which triggers a transition to Scene2.

Scene2 is the scene that allows the customer to visualize the instructions for

using the application (Fig. 5.14b) and to assembly the sofa by selecting components from a list of available elements (Fig. 5.14c). It was chosen to partially define the logic of this scene, limiting the number of components to be handled and the interactions to be implemented by the participant, in order to reduce the completion time of the overall experiment and make the task not repetitive. Thus, as shown in Fig. 5.13b, participants were asked to add to the scene and configure only the following assets: Instruction_Panel, Instruction_Button, Button.000, Button.001, Sofa_2x, Lounger, Next_Button. The interactions to be managed in Scene2 are:

1. a key tap gesture on Instruction_Button sets the visibility to false for the Instruction_Button itself, as well as for the Instruction_Panel, the Lounger, and the Sofa;

2. a key tap gesture on Button.000 and Button.001 changes the visibility to true for the Lounger and the Sofa_2x, respectively;

3. a key tap gesture on Next_Button triggers a transition to Scene3.

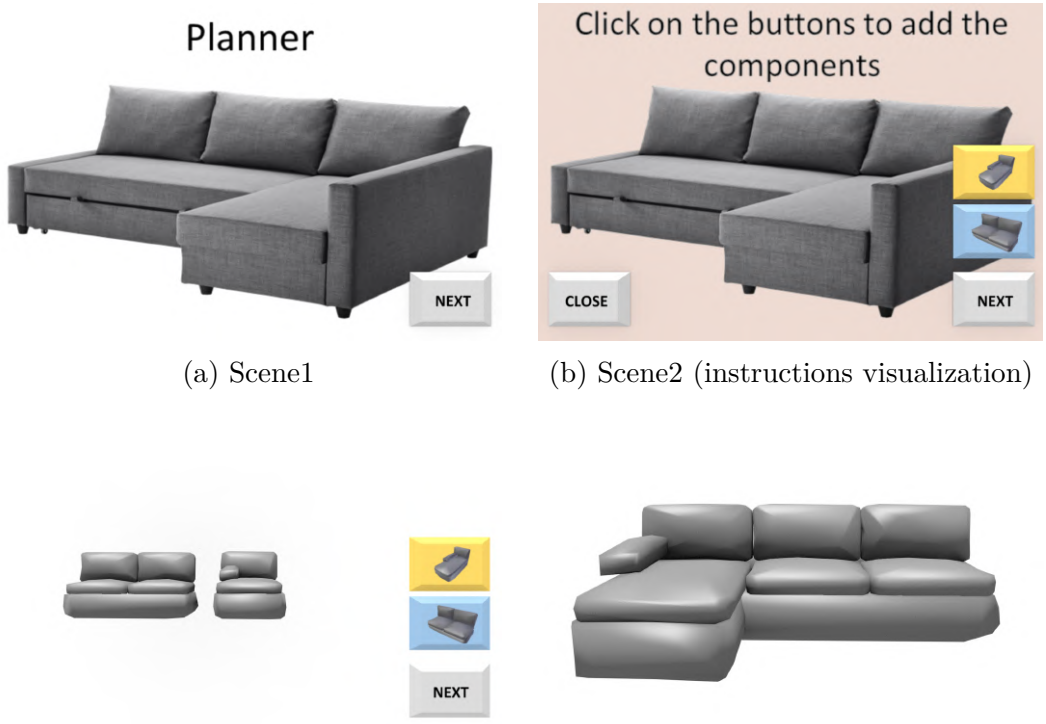
Scene3 (Fig. 5.14d) is the scene in which the customer can observe the assembled sofa. It contains only the Sofa&Lounger object, as depicted in Fig. 5.13c. The interaction to be implemented enables the reproduction of a sound and the playback of an animation when a swipe gesture is recognized over the Sofa&Lounger object.

5.6.3 Methodology

The capabilities of participants involved in study S1 made it possible to evaluate the creation of the above scenes with the VSE and with two other tools, i.e., the LE and the BGE. Considering the results obtained in this study (detailed in Sec. 5.7.1), in S2 it was decided to require participants to operate only with the VSE and the LE.

The experimental procedure consisted of several steps. At first, experiments were presented to all the participants. Then, instructions on how to work with the tools were provided (when needed), by letting participants familiarize with the considered interfaces. When the participants felt they ready for starting the experiment, they were requested to first create the three scenes by using one of the tools. Afterwards, they were asked to repeat the operations with the other tool(s). The order defining the interface to start and continue with was continuously changed, in order to reduce the impact of learning effects in the evaluation.

The creation of the application included two steps. In the first step, instructions detailing scenes composition and interactions to be defined were described to the participants by showing them the workflow in Fig. 5.13. Then, only for study S2, participants were invited to take some notes about the application to be developed, by filling in a table-based template with their annotations (samples templates are



(a) Scene1

(b) Scene2 (instructions visualization)

(c) Scene2 (sofa assembling)

(d) Scene3

Figure 5.14: Scenes of the application created by participants of the user study as rendered by the BGE.

reported in Tables 5.1 and 5.2 for the VSE and the LE, respectively). This step was meant to simulate the design process that is leveraged by the user to mentally organize contents following the working schema adopted by the specific tool to be used for implementing the interactive application. Before taking notes, users were left free to change the structure of the template (e.g., adding/removing columns, etc.), in order to provide them with the possibility to better align notations with their mental structures representing the working schema to be adopted. In the second step, participants were requested to develop the three scenes. Progress was monitored by a supervisor. Users were not allowed to receive any help or suggestions from the supervisor. One they completed the assigned task, participants were informed about the possible presence of errors and they had to fix them in order to develop a fully working set of scenes.

Participants could visualize a preview of the application being developed by creating the XML file (with the export procedure of the VSE and the LE) and importing it in the BGE. However, the presence of critical errors like, e.g., forgetting to set a cursor or specifying the start scene, could lead to failures during the export of the XML file from VSE and LE, preventing the visualization of the preview.

When	**On**	**Do**	**On**
KeyTap	*Object*1	*Set Visibility False*	*Object*2

Table 5.1: Template to use with the VSE. Interaction "on key tap gesture on Object1, change the visibility of Object2 to false" is represented.

Sender	**Msg Subject**	**Msg Text**	**Receiver**	**Action**
*Object*1	*OnKeyTap*	*Set Visibility False*	*Object*2	*Change visibility*

Table 5.2: Template to use with the LE. Interaction "on key tap gesture on Object1, send a message to Object2 for setting its visibility to false" is represented.

Three videos showing a user performing the above tasks with the BGE, the LE, and the VSE are available for download[35].

5.6.4 Metrics

The differences between studies S1 and S2 led to the definition of two sets of metrics for the evaluation of the performance.

In particular, for study S1, objective measurements included two metrics, namely, the amount of time and the number of visual components (bricks/blocks and connections) used by the skilled users to develop the interactive application when operating with the three tools. With respect to the amount of time, two different intervals were measured during the experiment: the first one is the time that the user spent to obtain his or her best results, whereas the second one considered the time needed to fix possible errors recognized by the supervisor. These two time intervals were then summed up to obtain the overall completion time for developing the application. Moreover, at the end of the experiment, skilled users were asked to express feedback /comments on their experience through an interview.

For study S2, objective metrics included completion time and number of errors made. The two definitions of completion time presented above were complemented with a third definition aimed to take into account the time needed by participants to complete the table template used to describe the application to be implemented. Subjective aspects were evaluated trough a post-test questionnaire structured in the following four sections[36]:

- Q1: demographics users' information, experience with 3D and programming languages/tools (visual or not);

[35]Videos of the experiments with the three tools: http://tiny.cc/8biqkz

[36]Questionnaire: http://tiny.cc/w9nqkz

- Q2: evaluation of system usability based on the SUS questionnaire [40];
- Q3: evaluation of the task load based on NASA Task Load Index, or NASA-TLX [118];
- Q4: users' preferences.

Regarding Q4, participants were asked to express their preference for the two tools on a 1–5 scale (with 1 representing high appreciation for the VSE, 3 neutral, 5 high appreciation for the LE) by accounting four different aspects, i.e., scene, object and interaction/relation management, and overall preference. Comments or reasons provided by the user to motivate their choice were also collected. Users were requested to fill in the questionnaire after having performed the tasks with both the tools.

5.7 Results

In this section, results obtained in the two studies are presented and analyzed.

5.7.1 Study S1

Objective results obtained in the preliminary study are reported in Fig. 5.15. In particular, in Fig. 5.15a, the overall completion time, including both the time to complete the tasks (to the best of participants' perception) and the time required to fix errors (highlighted by the supervisor) are shown. Fig. 5.15b and Fig. 5.15c present the number of visual elements (blocks and connections, or Links, respectively) used to assemble the three scenes.

It was decided to analyze the results in an explorative way, by taking into account means and variances, since the limited number of collected data prevented the use of more appropriate statistical tools.

With respect to the completion time, the large differences observed among the average values and the small variances obtained (VSE: $M = 2$ min 34 s, $SD = 25$ s; LE: $M = 9$ min 39 s, $SD = 29$ s; BGE: $M = 20$ min 27 s, $SD = 55$ s), suggest that VSE allowed participants to complete the task faster than with the other tools. Moreover, based on the number of elements (blocks and connections) used it can be inferred that, in absolute terms, the VSE was the tool that required participants to assemble/join a lower number of components. Connections were not considered for the LE, since they are generated automatically by the tool when the user configures the related blocks. Given the relatively high number of blocks (twice those of the VSE), the selection is expected to require a considerable amount of time, slowing down the process for defining the logic. With the VSE, users were roughly seven time faster than with the BGE, and defined a considerably lower number of both

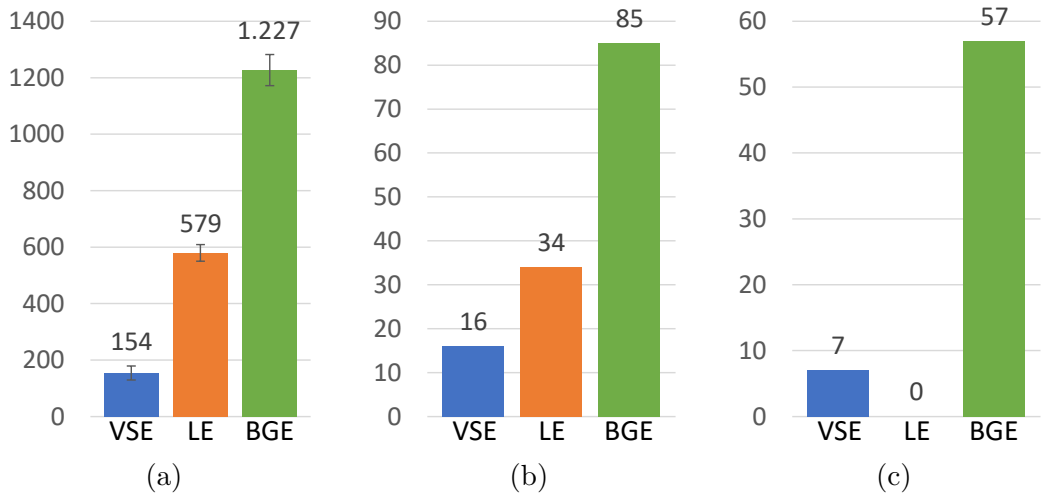


Figure 5.15: Objective results for study S1 (skilled users); (a) average completion time for developing the application (in seconds), (b) number of blocks, and (c) number of connections or Links used.

blocks and connections. This result is probably due to the need of defining Message sensor-actuator pairs (and related connections) in place of a single Link connecting the "When" and "Do" blocks when two different objects had to communicate.

Comments collected at the end of the experiments can be summarized as follow:

- the VSE provided the possibility to simplify repetitive operations that characterize both the BGE and the LE, e.g., the execution of an action on an asset when the recognition of the condition is performed by another asset; this advantage is related to the fact that the VSE is able to automate tasks that users are requested to execute manually with the latter tools (e.g., setting up the message subject and content for both the sender and the receiver);

- although the LE contained similarities with the BGE (a software that participants already knew), the VSE was found to be easier to learn and use by users who were not familiar/confident with message-based communications and/or with programming paradigms;

- differently than both the BGE and the LE, the VSE offered the possibility to obtain a high-level view of the overall application being created, and the interactions defined for the assets; this aspect of the VSE could be leveraged to let the user understand and manage the design of the application's workflow better than with the other two tools.

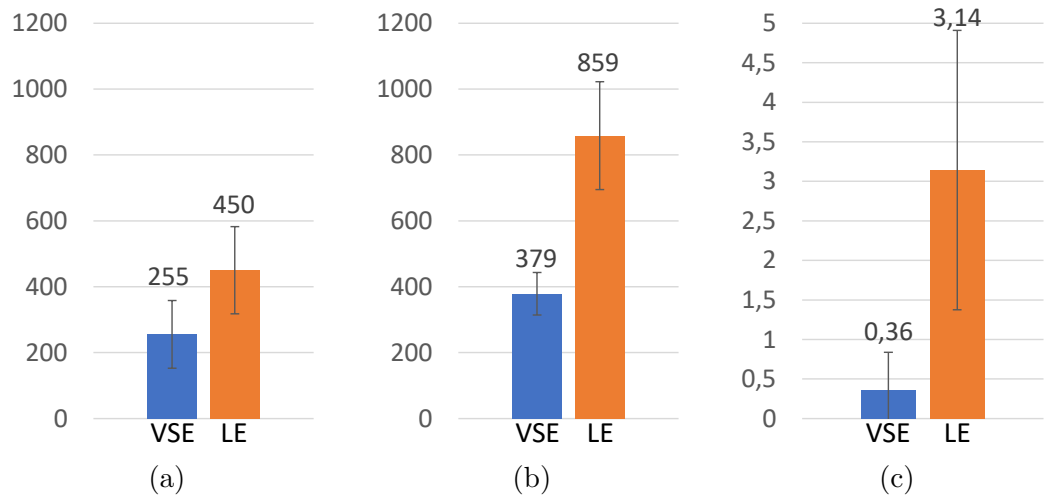


Figure 5.16: Objective results for study S2 (unskilled users); (a) average completion time for representing the application's workflow (in seconds), (b) average completion time for developing the application (in seconds), and (c) number of errors.

5.7.2 Study S2

The objective results achieved in the second study in terms of completion time and number of errors are illustrated in Fig. 5.16. In this case, statistical significance was analyzed using paired Student's t-tests ($p < 0.05$).

It can be easily observed that participants were able to complete the assigned task in almost half of the time when operating with the VSE compared to the LE. In fact, participants were, on average, 43.29% faster with the VSE (M = 4 min 15 s, SD = 1 min 46s) than with the LE (M = 7 min 30 s, SD = 2 min 17s) to complete the table template ($t(13) = -3.56$, $p = 0.03$, $d = -1.58$), and 51.58% faster with the VSE (M = 6 min 19 s, SD = 1 min 7s) than with the LE (M = 14 min 19 s, SD = 2 min 50s) to develop the actual application ($t(13) = -10.54$, $p < 0.01$, $d = -3.72$). Moreover, with the VSE they were able to reduce the number of errors, which were, on average, approximately 11% lower than those made with the LE (VSE: $M = 0.36$, $SD = 0.5$; LE: $M = 3.14$, $SD = 1.83$; $t(13) = -5.64$, $p < 0.01$, $d = -2.07$). These results can be explained by the fact that users spent less time with the VSE than with the LE to adjust the visualization, since they could easily see at the same time all the connections among objects. Furthermore, since the LE is based on the concept of message inherited by the underlying BGE, participants were still forced to manually set all the message parameters. This operation required a considerable amount of time (compared to the mechanism used in the VSE, which is based on drawing a Link between the "When" and "Do" blocks), and also led to the introduction in the development process of a possible source of errors.

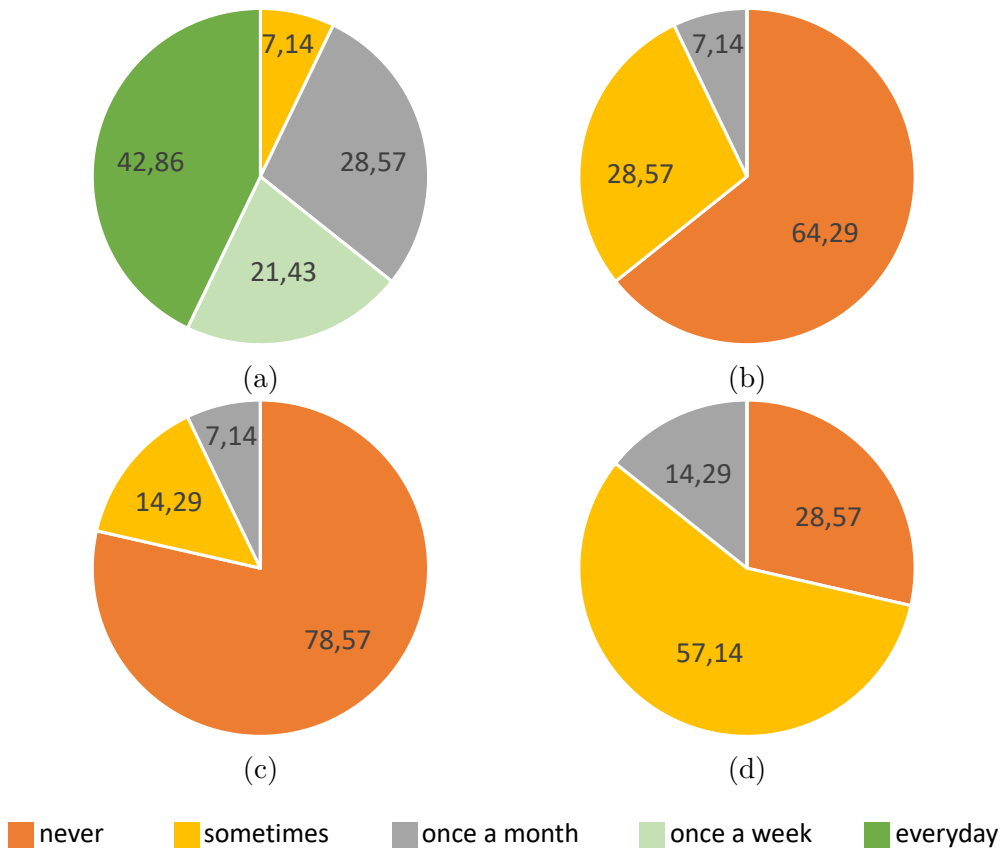


Figure 5.17: Results of section Q1 of the questionnaire concerning users' experience; experience with (a) graphics suites, (b) tools for node-based programming, (c) VPLs, and (d) tools for video-game or interactive application development.

Regarding subjective results, section Q1 of the questionnaire confirmed that the majority of the participants had familiarity with graphics suites such as Blender and Maya as well as with video and photo editing tools like Adobe Photoshop[37] and Adobe Premiere[38] (Fig. 5.17a). In fact, these types of software are regularly used in the course of the considered B.Sc. program, where students learn how to manage (visual) product design and communication. A small percentage of participants had little or no previous experience with node-based programming tools and VPLs, as reported in Fig. 5.17b and Fig. 5.17c. Lastly, roughly 70% of participants said to have little experience with tools and languages for developing video-games and/or interactive applications, as shown in Fig. 5.17d.

[37] Adobe Photoshop: https://www.adobe.com/products/photoshop.html

[38] Adobe Premiere: https://www.adobe.com/products/premiere.html

Concerning the results of section Q2, participants found the VSE as characterized by a higher usability with respect to the LE. In particular, the VSE achieved a SUS score higher than LE (VSE: $M = 79.11$, $SD = 12.23$; LE: $M = 53.57$, $SD = 18.31$; $t(13) = 7.38$, $p < 0.01$, $d = 1.64$). According to [40], the score obtained by the VSE (i.e., 79.11) corresponds to the C grade of the SUS scale, which classifies the usability of the tool as "Acceptable", whereas the score of the LE (i.e., 53.57) is equivalent to the F grade, corresponding to a "Marginally acceptable" usability. Based on the provided feedback, the possibility provided by the VSE to automatically handle some key operations (e.g., creating inter-object communications) reduced the number of operations to perform. This advantage made the participants perceive the VSE as easier to use and more satisfying than the LE. Moreover, the advantage of getting an overall view all the interactions at the same time allowed participants to learn the VSE faster than the LE.

These outcomes were also confirmed by the results of section Q3, which investigated the task load by means of the NASA-TLX tool. In particular, participants stated that the VSE was characterized by a lower workload than the LE (VSE: $M = 33.48$, $SD = 9.86$; LE: $M = 59.79$, $SD = 10.20$; $t(13) = -10.14$, $p < 0.01$, $d = -2.62$), allowing them to achieve better results (performance) with less effort (mental demand) as well as with fewer errors to be dealt with at the end of the experience (frustration).

Finally, preferences reported at the end of the task in section Q4 appear to confirm the above findings (Fig. 5.18). In particular, approximately 85.5% of the participants, on average, expressed their preference for the VSE rather than for the LE for all the aspects investigated. Only slightly less than 10% of the participants preferred the LE, since they perceived the GUI of this tool clear and easy to understand even though the message mechanism was consider as difficult to manage.

5.8 Future developments

Future work can be focused on extending the number of possible interactions supported by the tool, regarding, for instance, the integration of new input devices (e.g., smartphones or smartwatches, with their embedded gyroscopes, accelerometers, microphones, etc.), other sensors (to manage, e.g., body gestures and/or voice commands, etc.), or an improved set of hand gestures (e.g., gestures performed with both the hands). The possibility to control a more comprehensive set of conditions and actions (e.g., the change of the object's size and shape, etc.) could be also taken into account. According to feedback received, the system can be provided with the capability to natively support the visualization of the preview of the interactive application being created, without the need to export the logic being created to the graphics engine for visualizing it. Finally, efforts could be devoted to developing custom import scripts capable to translate the logic contained in the

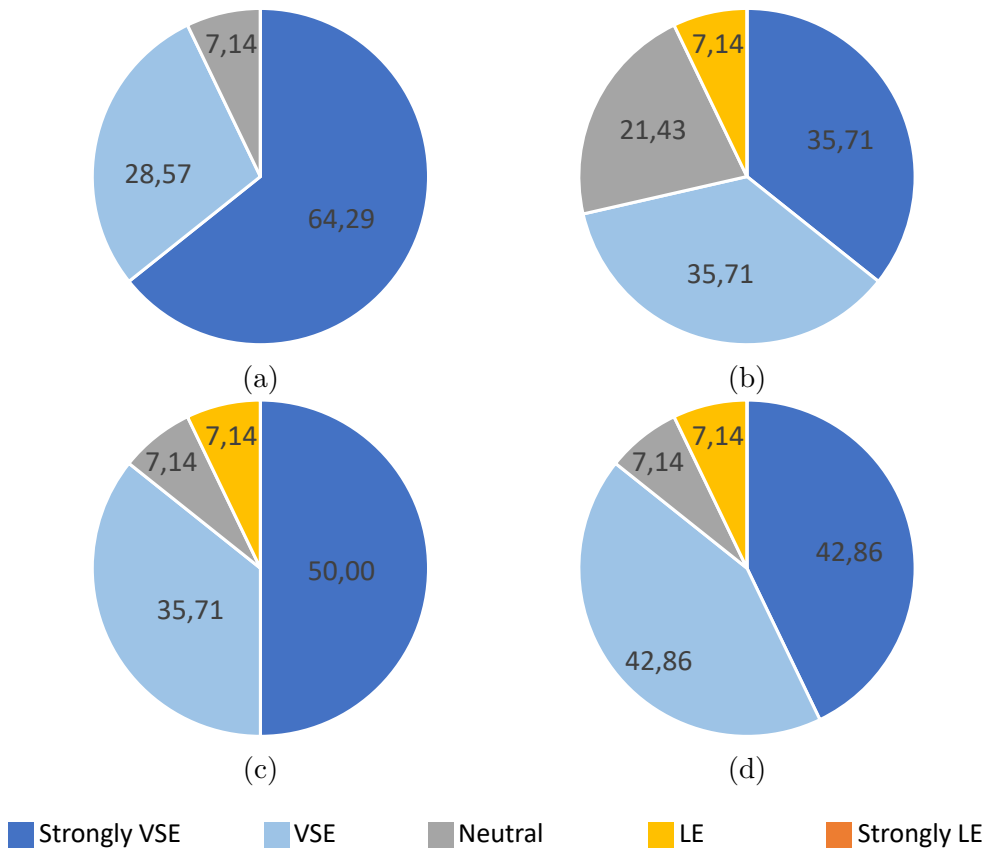


Figure 5.18: Results of section Q4 of the questionnaire about users' preferences; (a) scene, (b) object, (c) interaction/relation management, and (d) overall preference.

XML file generated by the VSE into data required for reconstructing scenes into other real-time 3D graphics engines, like Unity or Unreal Engine.

5.9 Concluding remarks

This chapter presented the VSE tool that could be leveraged by users with limited to no programming skills to develop 3D interactive applications for public exhibitions. The design of the VSE considered the principles of EUD and discussed the pros and cons of related VPL-based environments like Scratch, Kodu, BGE and LE. Differently than previous tools, the proposed one adopts a scene-centric approach for managing asset connections, which lets the users better visualize the elements belonging to the various scenes and their interactions.

Experimental results obtained by involving participants with both 3D graphics and programming skills revealed that the intended interactive applications could be created faster and leveraging a lower number of blocks when operating with

the VSE than with LE and BGE. Regarding the target end-users (i.e., users with little to no programming experience but with a background in design and visual communication), the VSE allowed them to carry out the assigned tasks in a shorter time, as well as making a lower number of errors than the LE. The advantages of the VSE were also confirmed by subjective results, which indicated a greater usability for all the dimensions considered.

As reported above, the proposed tool was used for the creation of two different applications: the first one (the use case) was targeted to visitors of a possible cultural heritage interactive exhibit, whereas the second one (used in the evaluation) focused on requirements that could be set by either physical or online stores. Despite such focuses, the VSE could be possibly used to develop 3D interactive applications targeted to other contexts.

Chapter 6

Augmented, mixed and virtual reality: A new perspective for human-machine interaction

Work described in this chapter was originally presented in [43, 49, 48, 34, 45].

6.1 Introduction

In the previous chapters, the early stages of the graphics assets production pipeline was addressed by presenting interfaces and tools designed to ease the generation of (the access to) graphics contents. However, as previously described, the developments in CG are posing various challenges related to its use in heterogeneous application domains [25, 218, 255, 261, 319]. In this thesis, the broad domain of CG usage has been investigated by focusing on aspects regarding the HMI. In particular, it has been studied how graphics assets, as well as innovative interfaces, can be leveraged to support users' interactions with machines by providing them with helpful feedback on the operations being executed.

The paramount importance of providing effective feedback is confirmed in various domains. For example, in the context of HRI, the lack of an intuitive feedback can make an interface difficult to use, leading to an unnatural and confusing user experience with high mental workload even when operators are skilled users [114]. Another example is given by applications targeted to training purposes, where the possibility to teach/explain concepts in a more meaningful way has been proved to be effective in stimulating learners, enhancing their motivation [86]. Feedback provided by the system can be visual, tactile, auditory and, in general, may involve all the senses together.

Moving from these considerations, this chapter describes interfaces based on AR and VR technology which have been designed to provide the users with feedback

that can be leveraged to support their activities in different domains. In particular, the analysis will first consider a work originally reported in [43]; this work focuses on haptic feedback in reconfigurable TUIs that can be used to interact in a more natural and realistic way with graphic assets within an immersive virtual environment. Then, other work originally reported in [49] and [48] is described, in which the user's attention is driven by visual feedback. The target applications are widely different. In fact, in [49], visual feedback is used in a movement analysis system supporting sport training with immersive VR and wearable technology. In [48], feedback is provided through wearable AR to make the user interact with a robot.

6.2 Building reconfigurable passive haptic interfaces on demand using off-the-shelf construction bricks

The possibility to touch and "feel" 3D assets in a virtual environment is a capability that can improve the user's sense of immersion and presence in MR experiences [194]. Interfaces designed with the aim of stimulating the human sense of touch are known as *haptic interfaces* (or *haptics*), and can be subdivided into *active* or *passive* depending on the underlying technology. Active haptics are based on computer-controlled actuators to exert forces on the user [333], whereas passive haptics are able to provide the users with feedback through their shape, weight or other inner physical attributes. In particular, authors in [298] defined passive haptics as "*interfaces [that] use energetically passive actuators which may only remove, store, or redirect kinetic energy within the system*".

Most of the solutions experimented so far in the literature leveraged either everyday objects or custom-made props (often with complex hardware) chosen or designed to mimic as much as possible the virtual objects. However, these solutions are characterized by poor flexibility and usability, which can prevent their applicability in professional scenarios.

To deal with the above issues, a new class of haptic devices was developed in [43], which can be built/assembled "on-demand" using off-the-shelf construction bricks (namely, the LEGO Mindstorms EV3 elements). As reported in this thesis, the possibility to assemble interfaces "on-demand" has been already investigated in the field of 3D character animation [173]. In that case, the use of a tangible prop mimicking the aspect of the virtual objects to be animated is proven to largely ease the animators' job. In the following, the approach based on reconfigurable tangible interfaces is applied to the creation of haptic devices which can be leveraged in many ways within the same application, and can be reassembled several times to create new interfaces tailored to various application's requirements.

For assessment purposes, the devised methodology was applied to assemble several passive haptics requested to carry out a number of tasks in a VR scenario reproducing an imaginary escape room.

6.2.1 Related work

Haptic devices can be classified as either active or passive depending on the technology used to implement the interface.

The Phantom [210] represents one of the first examples of the first category. As with several active haptics devices, it is a grounded machine able to apply a controlled force vector on the user's fingertip. Ungrounded solutions have been also developed, usually in the form of wearable devices such as the hand exoskeleton presented in [125]. The main advantage offered by active haptics regards their versatility, since they can be easily re-adapted to be used in different scenarios. However, they are generally found to be cumbersome to use because of their electro-mechanical complexity [1]. Moreover, it is quite difficult to use them for implementing functionalities different from those they have been designed for (like motion blocking for the exoskeleton in [125]).

Passive haptics generally present a low complexity, since no active computer-controlled actuator is normally involved. An example of wearable passive haptics is proposed in [1], where a device (named Elastic-Arm) uses a rubber band to connect the user's shoulder and wrist in order to generate a passive force feedback when the user attempts to extend the arm. Another interesting example is presented in [2], where an handheld device (called the Virtual Mitten) containing springs allows the user to grasp virtual objects by tightening and relaxing the hand. In [248], squeeze feedback is provided using a foam ball. In some cases, actuators are not used at all, making the passive haptics a "simply" physical – possibly approximated – proxy of the corresponding virtual object [194].

Considering this case, several solutions in the literature tried to establish the optimal degree of fidelity for these props: in particular, the work in [209] and [150] showed that, given the strong impact that visual feedback has on perception, there is no need for high fidelity in shape details, but simulating the weight of the represented object seems to enhance the experience only if the prop resembles the form of the real object. The main drawback of this approach is the lack of generality of props and, consequently, the need to make use of separate physical objects for each virtual object. An interesting solution to this problem was described in [60], where it was demonstrated how two physical devices – namely, a foldable prop and a suspended ball – could be reused several times in the same VR application to simulate different objects with the user being unaware of that.

The approach described in the following builds upon the above concepts, and integrates them in a methodology letting users construct a passive haptic interface using off-the-shelf components that can be reconfigured multiple times, thus

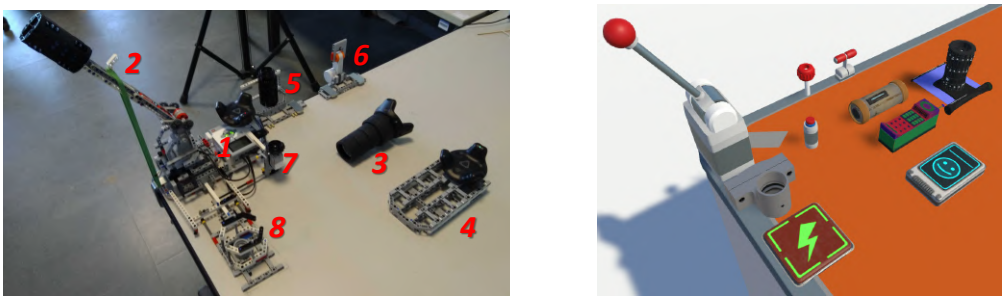
addressing both flexibility and ease of use limitations that characterize existing work.

6.2.2 System overview

As mentioned, in order to asses the proposed approach, a VR application presenting an imaginary escape room was developed. The scenario consists of a futuristic nuclear power plant, where users are requested to complete a set of tasks that allow them to achieve the final goal of the application, i.e., escape from an emergency situation. At the beginning, the lights of the virtual environment are turned off, and a telephone starts to ring in the dark. During the call, an imaginary colleague briefly describes the situation and provides the user with some hints to control the emergency. The procedure to deal with the emergency is based on three stages. In the first stage, settled in the auxiliary power room, the user is requested to restore the facility power through a lever that has to be identified in the dark by making use of the weak light emitted by the keypad of the phone and feeling its shape with the hand. In order to enter in the second stage, the user is asked to identify the device (i.e., a tablet) that activates the teleportation after a hand scanning. In the second stage, the user is teleported in the control room, and he or she has to secure the reactor room from radioactive risk sources by first activating and then manipulating a robotic arm with the aim to move boxes that contain dangerous material in a safety area. To enable the control of the robot, a battery must be plugged in a socket and oriented properly. Lastly, in the reactor room, the user has to search in the environment for some hidden clues which become visible when pointed by a UV beamer. Clues can be used to obtain a password needed to activate the procedure for inserting control rods in the reactor, stabilizing the nuclear reaction and closing the emergency. The scenario was developed with the Unity game engine.

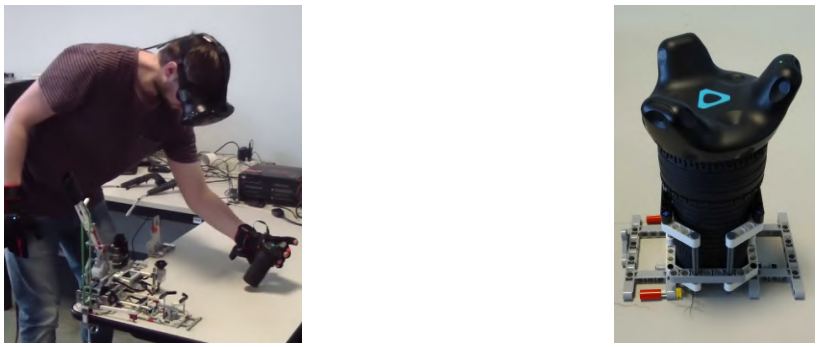
As illustrated in Fig. 6.1b, bricks, servomotors and sensors in the LEGO Mindstorms EV3 Core and Expansion sets were used to build a number of props. Each prop acts as a physical proxy for one or more of the virtual objects (Fig. 6.1b) the user is requested to interact with in order to solve the proposed challenges. The encoders of servomotors were used to gather rotation data, disregarding the potential active capabilities offered by the use of the motor. Sensors and servomotors were connected to a LEGO Intelligent Brick – the component labelled (1) in Fig. 6.1b) – which communicates information about the state of a button, the rotation of a knob, etc. to Unity over a WiFi connection in order to synchronize the real and the virtual scenarios.

During the experience, the user wears a HTC Vive VR headset (Fig. 6.2a), which provides him or her with both visual and audio feedback. In order to improve the sense of presence and the naturalness of interaction, a real-time virtual reconstruction of the user's hands was integrated in the virtual scenario. At first,



(a) (b)

Figure 6.1: Comparison between psychical props and corresponding virtual objects; (a) psychical props, and (b) virtual objects.



(a) (b)

Figure 6.2: Setup of the system; (a) a user interacting with the system, and (b) example of pros joined together.

hand tracking was based on the Leap Motion controller. However, with this technology, it was difficult to cope with hands' occlusions. Therefore, the Manus VR gloves were adopted to collect fingers' articulation data. Leap Motion controller was nonetheless leveraged to measure the actual size of the user's hands, in order to appropriately resize the geometry of the virtual hands. The position/orientation of the user's hands, as well as passive haptics, were tracked using HTC Vive trackers attached to the user's wrists and to physical assemblies.

To achieve the final goal of escaping from the room, the user has to use the above props to perform the tasks below (like in the solution proposed in [60], during the experience the user is unaware of reusing the same prop in more than one task).

- *Identify the attributes of objects*: assembled props allow the user to feel the differences among the objects he or she interacts with in terms, e.g., of weight, shape, surface, etc, since props have been designed to mimic their physical counterparts. For example, in the first stage, the user has to recognize the ringing phone represented by the cylinder assembly (3) in Fig. 6.1b based on

its haptic feedback and on the flashing keypad. Similarly, the user needs to find the shape of the lever (2) in Fig. 6.1b by moving his or her arm in the dark and feeling it in order to turn on the lights (Fig. 6.3a). The user is also requested to touch the tablet device (4) in Fig. 6.1b as well as its smooth surface when he or she has to place the hand on it for triggering teleportation (Fig. 6.3b).

- *Pulling / Pushing*: the lever (2) in Fig. 6.1b is leveraged in the first stage to activate the lights, and is reused later to manipulate the height of the robotic arm's gripper (Fig. 6.3c); the elastic mounted on the prop exerts a force that tries to keep the lever in the initial position.

- *Rotation*: the physical knob (5) in Fig. 6.1b is used twice, first to move left and right the robotic arm's gripper during the second stage, then to specify the password digits in the third stage ((Fig. 6.3d)). The smaller lever (with no return feedback) (6) in Fig. 6.1b is used to move the gripper forward and backward.

- *Pressing*: the pushable prop (7) in Fig. 6.1b is used as a button and, depending on the task being performed by the user, can be leveraged to either make the gripper of the robotic arm open or close, or to confirm the password digits being inserted.

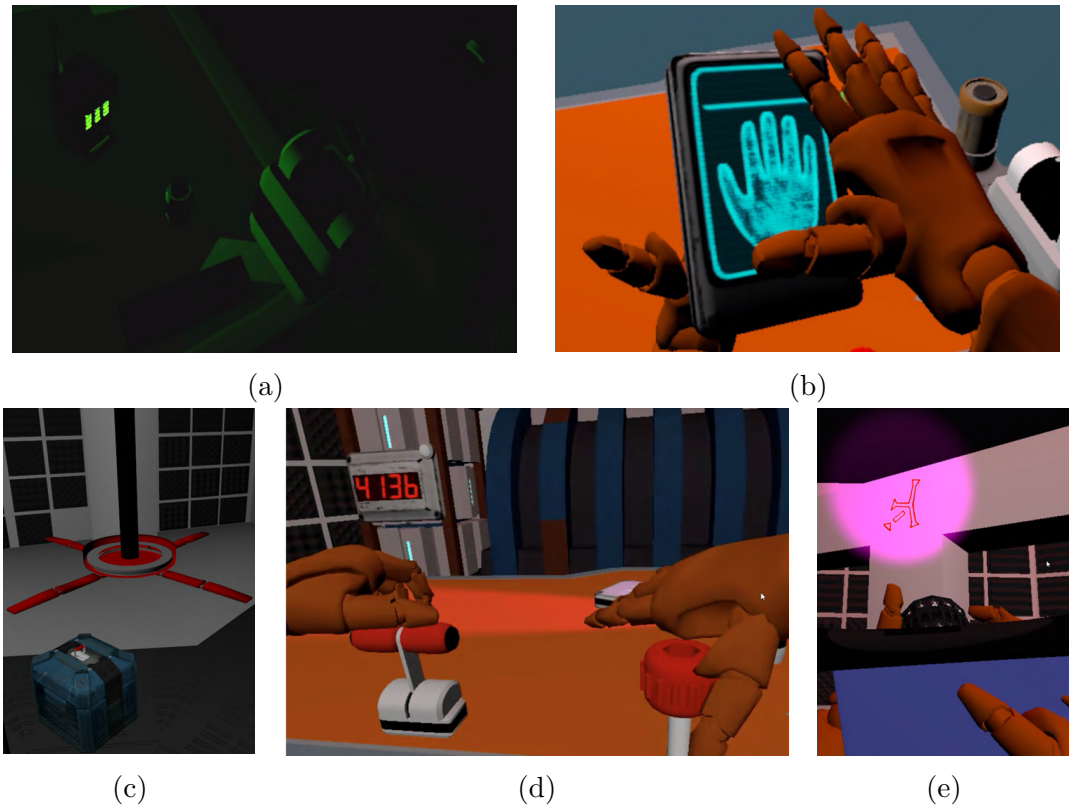
- *Insertion / Plug in*: the prop leveraged in the first stage as a phone (component labeled (8) in Fig. 6.1b) is reused in the second stage to mimic the battery that has to be plugged in a socket and rotated in order to power the robotic arm.

- *Join*: when the battery is plugged in the socket (as shown in Fig. 6.2b), it becomes a new, single object, i.e., the UV beamer, that can be used to find hidden clues in the third stage (Fig. 6.3d).

A video showing the tasks as well as the props is available for download[1]. Instructions for building the props presented above are also available[2]; in order to show the flexibility of the proposed solution, instructions for building, with the same bricks, other props that could be possibly used in different tasks are also provided.

[1]Video of the tasks: https://www.youtube.com/watch?v=tbruIRw9TJ4&feature=youtu.be

[2]Building instructions: http://vr.polito.it/papers/ieeevr2019/instructions/



(a) (b) (c) (d) (e)

Figure 6.3: Example of tasks; (a) identify the lever in the dark, (b) perform palm scanning, (c) control the robotic arm, (d) enter the password digits, and (e) find hidden cues.

6.2.3 Future developments

Future work could be focused on investigating the usability of this approach (both concerning the assembly and the use of the physical props) through user studies involving other scenarios and tasks. Moreover, mechanisms for the automatic generation of assembly instructions as well as the creation of further props supporting, e.g., pressure/squeeze feedback or dynamic changes in textures and materials could be studied. Lastly, the opportunity to use the considered technology to implement also active haptics and/or to support AR scenarios could be possibly explored.

6.3 A movement analysis system based on immersive VR for sport training

In the last decade, VR and its applications have became commonplace. However, only recently they started to largely take advantage of the incredible developments that are affecting related technology, from hardware for real-time graphics to consumer-grade headsets for immersive experiences, wearable interaction devices, motion tracking systems, etc.

Training is becoming one of the most common applications of VR, encompassing several areas of interest, such as healthcare [205], industry [308], military [37] and automotive, where training is delivered to several users, from professionals users such pilots or medical doctors [78, 157], to common people [177]. According to [24], the possibility of acquiring information multiple times from different channels improves the users' ability to learn and memorize contents.

In this perspective, VR can be finally used to improve the variety of educational delivery mechanisms, supporting them with new techniques especially in those fields where traditional approaches are supposed to be weak for learning purposes. In fact, the use of VR has been proved to make users active elements in the learning process and not just passive entities, since the immersion in a virtual experience lets them directly interact with the concepts to learn [24]. Concepts can be represented in a new meaningful way, not possible with traditional tools, and interaction further stimulates users by enhancing their motivation [86]. The adoption of VR makes it possible to evaluate learner performance in an easier way, since learning sessions can be configured, recorded and analyzed in the same environment [308].

In the training context, sport is a field where new technology-based solutions are more and more taking the place of traditional practices. This trend is confirmed by the great work reported in the literature investigating the use of VR for training in baseball [94], rugby [38], handball [30], golf [156], dance [80], etc.

In the following, attention will be devoted to tools dealing with analyzing and enhancing the mechanics of movements in order to improve performance and reduce the risk of injuries [77, 178]. To this purpose, a VR platform was developed to analyze and train sport players' movements. The platform, originally described in [49], focuses on basketball, and is meant to improve the technical gesture, by providing users with ad-hoc hints which are not available when traditional training methods are used. In particular, real-time visual feedback concerning body articulation during the execution of the gesture, as well as a quantitative assessment of the overall gesture at the end of the movement are provided. These data are measured by comparing the tracked movement with a previously recorded gesture performed by the same user that serves as a reference. The use of VR also allows the user to navigate the virtual environment, in order to observe the reference movement from different points of view. The reference movement is represented

together with the tracked one, by using an intuitive interaction method known as the "ghost metaphor" [325]. An object tracking module was also developed which makes it possible to represent the real object (e.g., the ball that has to be thrown in the basket) in the virtual environment.

A prototype implementation was developed using consumer hardware, in order to asses the effectiveness of the system through a preliminary evaluation. The prototype was evaluated by considering the training of movements that are common in basketball, within a small-scale (room-scale) setup. Given the promising results achieved with this configuration, a large-scale implementation will be developed in the future, by using high-end hardware. It is worth noting that the affordance of the hardware used so far makes the current prototype an interesting solution for home training scenarios, where the presence of a real trainer is not needed.

6.3.1 Related work

It is worth noting that the improvement of the gesture mechanics is a well-studied topic in the literature. As a matter of example, the system in [80] leveraged real-time tracking and a large screen to teach dance movements. In particular, the projected screen is used to display a video of an expert executing the movement to be learned, whereas a professional motion capture system developed by Vicon is able to accurately track the body of the user, who is facing the screen. The real-time positions of the user's tracked joints are overlapped with the video of the expert. Three different methods have been tested to provide users with feedback: full (16 joints tracked and compared with the expert model), reduced (4 joints) and no feedback. Results showed that reduced feedback is more effective than full feedback. The work in [53] presents a training system based on motion capture and VR that allows users to reproduce an ideal movement by following a virtual teacher projected on a wall screen. As in the previous work, user's movement is tracked using a motion capture system; however, in this case, reconstructed movements are shown in the virtual environment next to the virtual teacher. Virtual feedback helps the user to learn the movement without the real trainer. In particular, three types of feedback were adopted: immediate feedback (to indicate, with different colors, when the position of the body is accurate during the performance), a score report (that evaluates the overall performance) and a slow-motion replay. A similar system was presented in [16]. The system includes three parts: a VR-based motion-guiding interface, in which the ghost metaphor is used to show the movement the user has to replicate; a posture-oriented motion retargeting module, which manages the transformations to be applied in order to consider the difference in body size between the trainer and trainee; a motion evaluation and advice module, which provides feedbacks for making the user follow more closely the reference motion. The system was tested considering fencing training and a dance imitation game.

The target of the work seen so far is represented by sports characterized by a

motion profile trained with minimal interaction with the environment, generally defined as free motion. The system presented in [67] implements a basketball training tool that lets the users improve their skills with free throws on a simulated basketball court in VR. The developed system plays the role of the coach, intended as the subject responsible for transferring skills to the players and providing information about the effectiveness of the gesture performed. The system is comprised of a software module dealing with the graphics reconstruction/generation of the 3D elements belonging to the scene (e.g., the ball, the player's hand, the field and the visual feedback for the user), a motion capture system to track the user's body, and a component for examining the movement performed and calculating the ball trajectory. Experimental results revealed that the system was considered as engaging and stimulating for the purpose of training.

Most of the above work leveraged large display or projected walls as output systems. However, these technologies could significantly decrease the sense of immersion [53] and present perception issues related to the virtual environment that could be particularly critical, e.g., regarding the estimation of distances [66]. To investigate the impact of these issues, the work in [66] leveraged a CAVE system in combination with three interaction paradigms targeted to basketball free-throw training. The three paradigms rely on different user perspectives and types of feedback: first-person perspective (1PP), third-person perspective (3PP) and 3PP with visual guidance represented as a sequence of ellipses placed along the ideal trajectory of the ball in a perfect throw. A Vicon system was used to track both the player's movement, and the ball. The main outcome of this work was that the modality with visual guidance performed better than the other ones since it helped the trainee to release the ball in a position that was closer to the optimal one, since it allowed to reduce the underestimation of distances in virtual environments related to the use of large-screen displays.

In [156], immersive VR is applied to evaluate the usefulness of golf training in a non-real-time system. Using this system, golfers have the possibility to compare their recorded golf swings with an ideal movement recreated by combining a variety of expert users. The two gestures (user's and ideal) can be observed within an immersive environment through a VR headset. The system also highlights the joints for which the difference from the ideal model is the highest, providing suggestions on how the golfer should improve his or her gesture. Besides the lack of real-time feedback, another limitation is represented by the very high-speed 3D motion capture system, which is very accurate but requires expert assistance for operation. These aspects make the system very expensive and not suitable for all settings.

Another system based on a VR headset for learning movements is presented in [63]. The work evaluates new training modalities for Tai Chi which leverage a virtual trainer and visual cues into an immersive environment. Although five training modalities were studied, outcomes did not show a clear benefit in the use of VR for this sport, probably due to the large system delay (>170 ms). However,

prior results suggest that, in order to be helpful for training, VR systems must guarantee simulation realism and interactivity.

By moving from the above considerations, the design of an immersive VR training system based on a head-mounted display will be presented, which provides different types of visual feedback for both real-time and off-line assessment [49]. This information can be used by trainees to improve a specific technical gesture that includes the interaction with a ball without the need of any external assistance.

6.3.2 Proposed system

The elements that are included in the overall architecture (which is shown in Fig. 6.4) consist of I/O devices, a Middleware layer and an Application layer. In the following, each layer will be described in detail.

I/O devices

This layer holds the set of hardware components that compose the VR system. The devised implementation relies on consumer hardware. In particular, a Microsoft Kinect 2.0 RGB-D camera provides two types of image frames, i.e., color and depth frames, which can be used by the Ball Tracking Module (detailed in Section 6.3.2) to track the position of the ball and represent it in the virtual environment. The Perception Neuron motion capture suit and the HTC Vive wireless trackers collect tracking data that let the system reconstruct the body of the player in the virtual environment. The head-mounted display of the HTC Vive system provides visual feedbacks to the user who is immersed in the virtual environment. Lastly, data gathered by a HTC Vive controller are leveraged during system calibration.

Middleware

This layer includes the software components that are used to convert the data gathered by the input devices into useful information for the application layer. In particular, an algorithm developed with the OpenCV library is able to track the position of the ball using the information in the color and depth frames delivered by the Kinect through its SDK. Steam VR and Neuron Axis acquisition and streaming software are used to access the tracking data of the HTC Vive controller/trackers and the Perception Neuron suit, respectively.

Application

This layer is made up of two blocks: the Ball Tracking Module and the Virtual Environment. The former block implements a three-dimensional object tracking algorithm written in C# which is based on the combination (fusion) of color and depth frames information. Specifically, the algorithm first tries to recognize the

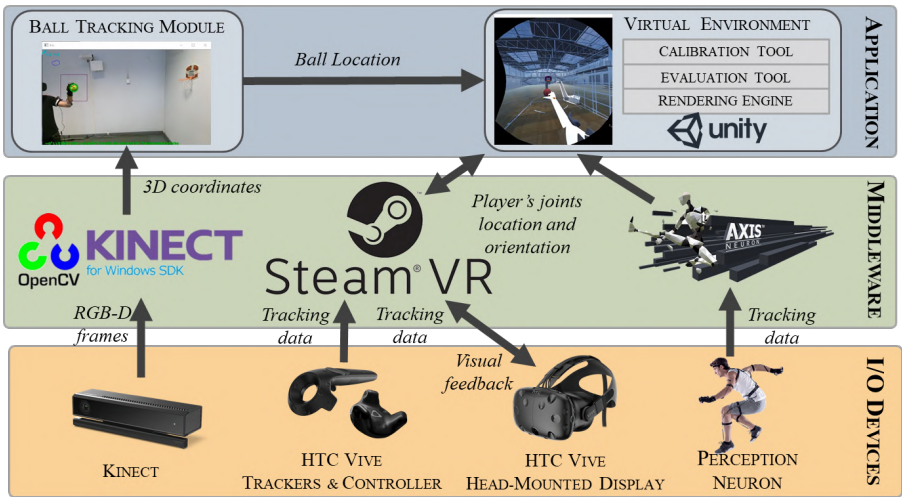
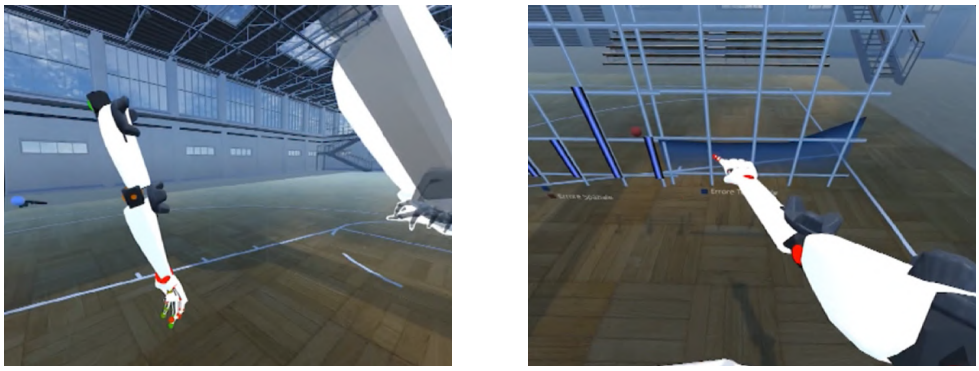


Figure 6.4: Overall system architecture [49].

2D region of the image that contains the ball through color information. To obtain valid 2D coordinates, it is necessary that the ball moves slowly, since its movements cannot be tracked due to blur or to the lack of color frames when the lighting of the environment is not adequate (since the Kinect device lowers the acquisition frame rate in low-light conditions). Hence, this approach based on color frames is used only for tracking slow ball's movements. If 2D coordinates are valid, they are mapped into the 3D coordinate system of the Kinect, called the CameraSpace, by leveraging the CoordinateMapper class of the Kinect SDK. Depth frames are considered to track high-speed ball movements, by searching in the frame those pixels that present a significant difference in terms of depth with respect to the background.

The Virtual Environment module is composed by three components implemented as scripts in the Unity game engine: the Rendering Engine, the Evaluation Tool, and the Calibration Tool. The Rendering Engine is in charge of managing the user's interactions within the 3D virtual environment and of generating visual feedback. The first type of feedback displayed by the system is shown in Fig. 6.5a. It presents, on the left side, the real-time skeleton of the user's arm as reconstructed by the input devices, whereas the movement to be trained is shown through the ghost metaphor on the right side. The color of joints displayed in the real-time skeleton, ranging from green to red, represents the current distance from the corresponding joint in the ghost arm.

The Evaluation Tool compares two time-series that represent the movement of the player and the gesture to be learned using the DTW algorithm. The cost for aligning the two series computed by the DTW algorithm can help the user to fix timing in gesture execution (the higher is the cost, the bigger was the time mismatch). Moreover, once the two sequences have been aligned, it is possible to



(a) (b)

Figure 6.5: Visual feedbacks provided by the VR system.

compute the Euclidean distance between joint angles. This information can help the user to adjust the orientation of joints (minimizing the above distance) while performing the movement. These two measures represent the second type of feedback provided by the system, which is displayed in two separate plots (Fig. 6.5b), thus allowing the user to evaluate his or her performance and the progress during training directly within the virtual environment.

The Calibration Tool was developed with the aim to find the rigid transformation (rotation and translation) between the HTC Vive and the Kinect coordinate systems in order to register the two systems. The calibration is needed to place the ball (which is tracked by the Kinect), in the correct position with respect to the user's skeleton joints (whose position is given in HTC Vive coordinates). The problem is solved by first placing the ball close to the HTC Vive controller (tracked by the HTC Vive base stations) and creating two datasets containing corresponding points in the two coordinate systems. Then, the optimal rotation matrix between the two systems is computed using Singular Value Decomposition (SVD). Finally, the optimal translation vector is computed by considering the rotation matrix and the centroids of the two datasets.

6.3.3 Case study

As previously mentioned, to asses the effectiveness of the devised VR system, a laboratory setup was created and used for testing the training of a technical gesture related to basketball. Specifically, a user study was carried out by asking 18 participants (selected among students at Politecnico di Torino in Turin, Italy) to evaluate the usability of the VR system and compare it with a traditional approach for basketball training. A professional basketball coach who served as technical manager of the Cus Auxilium Torino's youth sector was involved in order to design the experimental study and define the training procedure to be compared with

the proposed system. In the following, this latter method will be referred to as "Projected", whereas the proposed one will be named "Virtual". The Projected approach consists of showing a video of a user performing the gesture to be learned on a wall screen. The participant is requested to follow the observed movement as close as possible facing the wall. To track the user's arm within the virtual environment, two HTC Vive trackers were placed on his or her upper arm and elbow. Moreover, a subset of the Neuron Perception motion capture suit was leveraged to track user's hand and fingers, thus improving the sense of immersion (though finger joints were ignored in the computation of the metrics, since they were subject to drifts due to the nature of the sensors). The overall equipment worn by the user as well as the setup used for the experiment is shown in Fig. 6.6. Given the consumer-grade hardware selected for the prototype implementation and the limited space available in the laboratory, a small-scale (room-scale) setup was realized by considering a ball that is 10 cm in diameter and a small basket. The procedure of the tests included the following steps:

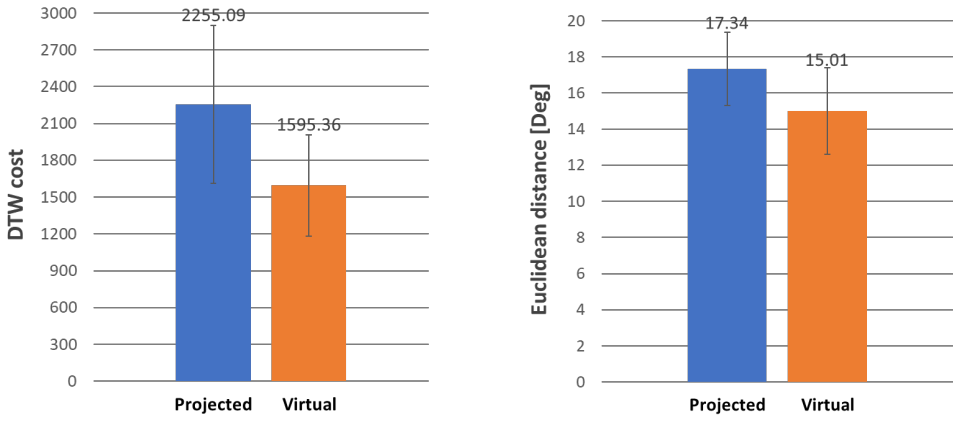
- brief presentation of the task to be performed;
- registration of the movement (a throw of the ball) to be replicated;
- a familiarization phase in which the user was invited to get acquainted with both the training systems;
- execution of two series of twenty throws with the Virtual and Projected system (the order defining the system to start with was randomly selected balancing the number of users who started with a given system);
- filling in of a subjective questionnaire on user experience.

The same metrics calculated by the Evaluation Tool were used as an objective way to compare the two approaches by identifying the system capable to let the users reduce the amount of errors in the learned movement.

6.3.4 Results

Results in term of DTW cost and joint angles distance are reported in Fig. 6.7. Statistical significance was studied through paired samples Student's t-tests ($p < 0.05$). According to Fig. 6.7a, the DTW cost metric achieved lower values ($t(17) = -2.81$, $p = 0.01$, $d = -0.49$) for the Virtual system ($M = 1535.38$, $SD = 1062.13$) than for the Projected one ($M = 2223.73$, $SD = 1659.04$). Also considering the Euclidean distance (Fig. 6.7b), it can be noticed that similar results were obtained (Virtual: $M = 14.93°$, $SD = 5.18$; Project: $M = 17.40°$, $SD = 4.38$; $t(17) = -2.66$, $d = -0.51$). These results indicate that, on average, participants were more precise

Figure 6.6: System configuration for the experimental evaluation.



(a) (b)

Figure 6.7: Objective results

both in terms of gesture timing and joints orientation when performed the twenty trials with the Virtual system.

The questionnaire completed at the end of the experiment was organized in two sections. The first section focused only on the Virtual system and was based on the work in [153]. It was aimed at evaluating features corresponding to six categories, namely functionality, user input, system output, simulation fidelity, sense of immersion/presence and overall system usability. Questions in each category were expressed in the form of statements to be evaluated on a five-point Likert scale from 1 (strong disagreement) to 5 (strong agreement). For each category, the users were invited to express also their overall evaluation on a scale from 1 (very unsatisfactory) to 5 (very satisfactory). In the second section, the two systems

were compared with respect to training effectiveness and usefulness. To this aim, the Kirkpatrick's model[3] and a part of the questionnaire in [198] were used. This section contained multiple-choice questions evaluated on a scale from 1 (absolutely no) to 5 (absolutely yes). Finally, the users were invited to express their overall preference for either the Virtual or the Projected system and provide motivations for their choice.

Overall, 16 out of 18 participants preferred the Virtual systems. Motivations that led to this clear preference can be found in answers provided in the previous sections. In particular, Table 6.1 reports average scores for the questions regarding the first section. Participants expressed a high appreciation for the system, since each category received, on average, a score greater than 4. In particular, with respect to functionality, the users found that the features provided by the system were easy to access and not ambiguous. Results regarding user input show that system response was acceptable and input devices were adequate for the interaction, as users felt to have the right level of control over the system at all times. With respect to statements related to system output, users found the resolution and the field of view of the display suitable for the task. They believed that information was represented in a meaningful way, and they were comfortable in using the system. The participants perceived that the simulation behaved in a very usual manner, they were aroused with the way they could interact with the simulation and did not feel disoriented. Concerning the sense of immersion/presence, the users stated that they had the feeling of being part of the virtual environment and it was always clear where they were. Lastly, users enjoyed working with the system, they found it easy to learn and noticed a real benefit in that kind of interaction with machines.

The average scores achieved in the second section are reported in Table 6.2. In this case, the Virtual and Projected systems were compared. By focusing on statistically significant results (indicated by the * symbol), it can be noticed that participation and interaction with the training system was boosted by the use of VR. Users perceived the Virtual system as more knowledgeable about the training topics, probably because the visual feedback made the training session easier to attend. The usefulness of the virtual training session is also confirmed by the last four questions, for which the Virtual system got significantly higher scores compared to the Projected one. Comments reported by the users to motivate their preferences are summarized in the following:

- users felt that VR is able to make the training task more stimulating;
- visual feedback provided by the Virtual system were considered as helpful for the purpose of learning;

[3]Kirkpatrick's model: http://www.nwlink.com/~donclark/hrd/isd/kirkpatrick.html

Table 6.1: Results for the first section of the questionnaire.

Categories	*M*	*SD*
Functionality	4.33	0.81
User input	4.38	0.67
System output	4.50	0.60
Simulation fidelity	4.30	0.59
Sense of presence	4.44	0.49
System usability	4.60	0.47

Table 6.2: Results for the second section of the questionnaire.

Statemets	**Virt.**	**Proj.**	**p-value**
The objectives of the training were clearly defined	4.88	4.55	0.11
Participation and interaction were encouraged*	4.50	3.38	<0.01
Content was organized and easy to follow*	4.72	4.11	0.03
The materials distributed were helpful	4.61	4.27	0.05
The training system was knowledgeable about the training topics*	4.55	4.00	0.02
The training system was well prepared	4.44	4.11	0.13
The training system objectives were met	4.44	4.05	0.06
The time allotted for the training was sufficient	4.44	4.33	0.33
The meeting room and facilities were adequate and comfortable	4.33	4.16	0.33
Do you feel that the training was worth your time?	4.22	3.66	0.06
Do you think that the training was successful?*	4.50	3.50	<0.01
Did the training session accommodate your personal learning style?*	4.50	3.38	<0.01
Would you be able to teach your new skill to other people?*	4.16	3.38	<0.01
Are you aware that you changed the way you make that specific movement?*	3.94	3.22	<0.01

- the use of the ghost metaphor gave the possibility to easily understand the (temporal and spatial) characteristics of the movement to be learned;
- a great advantage of the Virtual system was found in the opportunity to observe the reference movement from different viewpoints.

6.3.5 Future developments

Future work can be devoted to investigate the applicability of the proposed system in a large-scale (real-scale) configuration, by considering also gestures relevant for other sports. As previously mentioned, to keep down the costs of the prototype implementation (and indirectly making the system suitable for home settings), consumer-grade hardware was used. In the future, alternative technologies could be tested, especially for body tracking, to address issues introduced by the drifts of the motion capture suit and reconstruct the entire user's body in the virtual environment.

6.4 Interaction feedback for robot tele-operation with wearable AR

Today, the improvements achieved in the field of robotics make supporting human activities with robotic systems an ordinary practice in various application fields, ranging from the inspection of industrial plants [211], to assistance in home-care settings [146], search & rescue in dangerous environments [83], etc. In many of the above situations, the supervision of a human operator is still needed, especially when robots operate in critical situations [297].

In these contexts, HRI can be used to combine the humans' reasoning skills with robots capabilities in implementing a tele-operated system. During tele-operation, effective user interfaces should support the operator to perform the assigned task in the most effective way, without failures and damages to the robot [114]. Moreover, intuitive interaction modalities are needed to allow the operator to focus only on the goals of the task to be carried out and not on the complexity of the interaction with the robot, since complexity could affect his or her cognitive involvement with a negative impact on the overall performance [87].

Negative effects brought by unintuitive interfaces are more evident when the system involves the collaboration of multiple robots. Maneuvering multiple robots at the same time can be helpful in many scenarios [284], as it gives the opportunity to overcome the possible limitations of a single robot by leveraging the functionalities of another robot of the team [34, 284, 297].

According to [45], the factors that can influence the execution of a tele-operation task are the input method(s) and the design of the GUI. Considering the first factor, various approaches for implementing efficient input methods have been examined in the literature, ranging from conventional user interfaces (e.g., mouse, keyboard, gamepad, etc.) to NUIs (e.g., based on speech recognition, hand and body gestures, etc). The second factor regards the system's ability to provide proper feedback to the operator about the actual state of the robot and the conditions of the surrounding environment. It is worth saying that, especially when NUIs are considered as

input method, it is also important to provide the operator with feedback concerning the interaction (e.g., if gesture recognition techniques are used, feedback should be provided on the physical space in which the body or the hands of the user are being tracked).

To this purpose, AR-based approaches are considered since solutions based on this technology capable to provide a visual feedback concerning both the robot's conditions and the working space of the input device proved to be effective in improving the user experience and enhancing the operator's spatial awareness [45]. The advantages brought by AR are confirmed by the growing number of solutions experimenting with this technology in the considered domain [97, 119, 247, 260, 269].

Considering the above observations, the work originally reported in [48] proposed the design and the development of a user interface based on hand gestures and wearable AR to tele-operate a robot team that includes a rover and a robotic arm. Interface design built upon a previous work ([45]) where AR technology was implemented on a desktop computer screen. The input method selected to manage the various robot's functionalities relies on a hand tracking system based on the Leap Motion controller. AR is used to visually represent the working space in which the operator can move his or her hand in order to tele-operate robots.

With respect to [45], in the system presented in [48] AR contents are displayed on a wearable video-see through device, which lets the operator visualize augmented information superimposed on his or her own hand. The basic idea is that affordances offered by this visualization can help the operator to have a clearer understanding of how interface works and a higher awareness of the space he or she is acting into. To assess the effectiveness of the proposed design, a comparison of the two interfaces was carried out through a user study.

6.4.1 Related work

In the literature, a large number of user interfaces for robot tele-operation have been presented that leverage heterogeneous technologies based on, e.g., electrooculography signal analysis and haptic manipulators [304], brain-machine interfaces [128] or teddy bear-based robotic interfaces [279].

Despite the richness of alternatives, joysticks and gamepads are by far the most common interfaces used in HRI applications, since they are rather cheap, easy to design, and can provide accurate control. However, these interfaces could be difficult to use, especially by inexperienced users as well as when the robots have many functionalities or a large number of DOFs to control [119]. Specifically, the need to operate at the same time on numerous buttons and levers could make the mapping between the user input and resulting robot movement not intuitive. In order to address this drawback, a number of solutions in the literature experimented with body and hand tracking using vision-based techniques, wearable inertial sensors,

etc. These approaches to interaction with machines, often referred to as NUIs, are able to improve different aspects of the HRI in tele-operation scenarios. For example, NUIs can reduce the operator's training time and the cognitive effort, enhancing situation awareness [182].

For example, in [224], an industrial robot is manipulated by means of a wearable system comprised of a number of accelerometers mounted on the operator's arm. In [3], a first-person view- and body motion-based control method for a rover is described. The interface is aimed to improve the sense of presence and spatial understanding by directly mapping the user's movements (captured by sensors on the operator's hand) into the robot's actions. In [160], a remote robotic manipulator is controlled using an interface that combines surface-electromyography with inertial measurements. In [326], a depth camera is used to track human body joints and objects to be grasped. One of the operator's hand articulates the seven DOFs of the robotic arm, whereas the pose of the other hand manages the gripper. In this case, the Microsoft Kinect V2 is adopted to track the operator.

It is worth observing that some of the solutions based on NUIs could be affected by a limited working area and by occlusion phenomena. Furthermore, since the physical connection between the user and the system is generally removed, feedback about, e.g., the space he or she is moving his or her body or hands into needs to be provided (for instance by means of additional information conveyed on a screen).

This issue was addressed, for example, in [34]. This work presented a NUI-based solution to control a robot team (composed by a rover, a robotic arm, and a drone) through the Leap Motion controller. The interface displays on different viewports the 2D views (front and top) that represent the working volume of the hand gesture driven controller (together with the video provided by the camera mounted on the robots). Although a NUI-based solution was adopted, the repeated gaze switching among different viewports caused by the decoupling of the information displayed increased the complexity of the interface. For this reason, approaches based on AR were considered.

AR technology has been widely adopted to let the tele-operation systems improve the perception of the real word with digital contents that can be used to provide effective feedback. For example, in [328], it is described how AR can improve the operator's situational awareness and reduce the cognitive load since the user can be immersed in a helpful representation of the remote site. In [260], authors confirmed through a user study that AR can support inexperienced users to drive mobile robots by leveraging intuitive tele-operation interfaces. The AR interface presented in [247] leverages hand tracking and gesture recognition to support the articulation of a robotic arm. The interface provides the operator with an exocentric vision of the robot (slightly behind and above it). In [328], AR is used to carry out remote maintenance tasks with robots. A wearable immersive display is responsible for recreating the maintenance environment by showing physical and

virtual objects in the operator's workspace. A handheld manipulator lets the operator control a robotic arm, whereas inverse kinematics solvers and motion planning algorithms are applied to visualize areas of the environment that can be reached by the robot. A new solution for tele-operating seven DOF manipulators in rescue missions based on MR is presented in [196]. The MR interface executed on a handheld device combines the virtual representation of the robotic arm, obtained through its embedded sensors, with camera images to help the operator make decisions in real-time on the commands to be issued.

As anticipated, based on the above review, a wearable AR-based interface for the control of a robot team is developed by specifically building on the results of [45] and leveraging AR technology in a different way with the goal to provide the operator with an improved visualization of the working space.

In [45], desktop AR is used to implement a hand gesture-based interface for the control of a robot team including a rover and a robotic arm. Hand tracking and hand gesture recognition are achieved by using the Leap Motion controller. The rover can be tele-operated by moving the hand in a number of control boxes defined in the Leap Motion controller's working volume. Robotic arm is controlled by mapping the operator's hand on the arm's end-effector. The position and orientation of operator's hand as well as control boxes are shown on a computer screen as AR contents, overlapped to the video stream received by the camera mounted in the robot.

With respect to [45], the interface proposed in [48] makes use of video see-through AR on a head-mounted display to let the user visualize, at the same time, his or her hand and augmented contents. By leveraging the presence of the real hand in the field of view and its affordances, several changes were introduced in the way AR contents are displayed. In particular, the video stream provided by the remote camera was linked to the position of the tracked hand. The control box-based interaction paradigm which was proved already to be significantly effective was not altered intentionally, in order to specifically focus on the impact that see-through visualization can have on the perception of the environment and of operations performed.

6.4.2 Tele-operation system and proposed interface

The blocks that constitute the overall architecture of the tele-operated system, and their relationship, are represented in Fig. 6.8. In the following, blocks will be described in detail.

I/O devices

The Leap Motion controller allows the user to interact with the system by leveraging a hand tracking-based interface. The device sends to the Controller manager

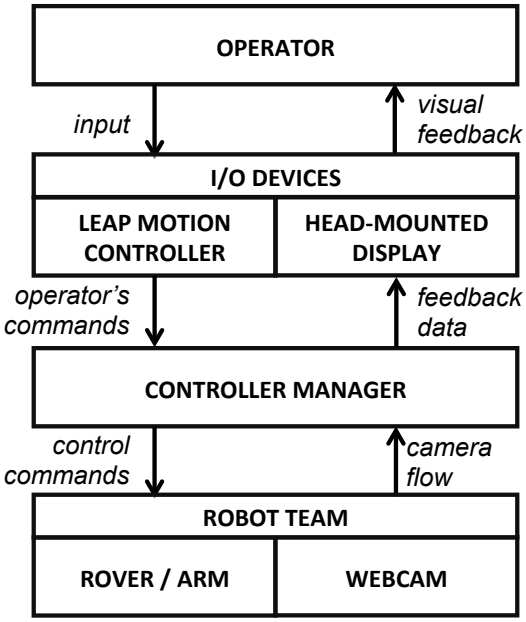


Figure 6.8: Conceptual architecture of the tele-operation system [48].

the real-time position, orientation and status (open/closed) of the operator's hand. The use of the Leap Motion controller defines a working volume that is shaped as an inverted pyramid centered on the device and expands upward with a field of view that is 150° wide and 120° deep on each side, and a range which extends from approximately 2.5 to 60 cm above the device's plane.

As in [45], the working volume of the Leap Motion controller is arranged in several interactive regions, that can be used to issue different commands to the robot (more details about regions' position and shape are provided in the following).

A video-see-through HMD (based on a Samsung Galaxy S6 smartphone and a Samsung Gear VR device) is used to make the operator visualize AR contents representing the size and shape of the working volume, the currently active interactive regions, as well as the video streamed by the robot's camera. Virtual contents are displayed on top of the Leap Motion controller in their real world position. Video stream, in turn, is attached to the operator's hand, in order to bring it where the operator's is expected to point his or her gaze.

Controller manager

This block is in charge of converting the operator's input (e.g., moving of the tracked hand forward, left, etc.) provided by the Leap Motion controller into suitable commands for the active robot. For example, if the rover is the active robot, the position of the operator's hand is converted into a value of velocity that is sent to the robot. Furthermore, the block transmits the hand tracking information

and the video stream to the HMD, where augmented contents enabling effective interaction are visualized.

The interaction schema relies on the state diagram illustrated in Fig. 6.9. Names of the states are expressed using capital letters. The transitions between two states are enabled by specific gestures (e.g., roll the hand, move the hand to the extreme left/right, etc.). The output shown to the operator on the HMD is represented in the state box.

Initially, the system is in the *IDLE* state. In this phase, the two robots are in a rest position and only the hand's position and orientation are tracked. The video stream of the remote camera is attached to the red point, which indicates the position of the operator's hand.

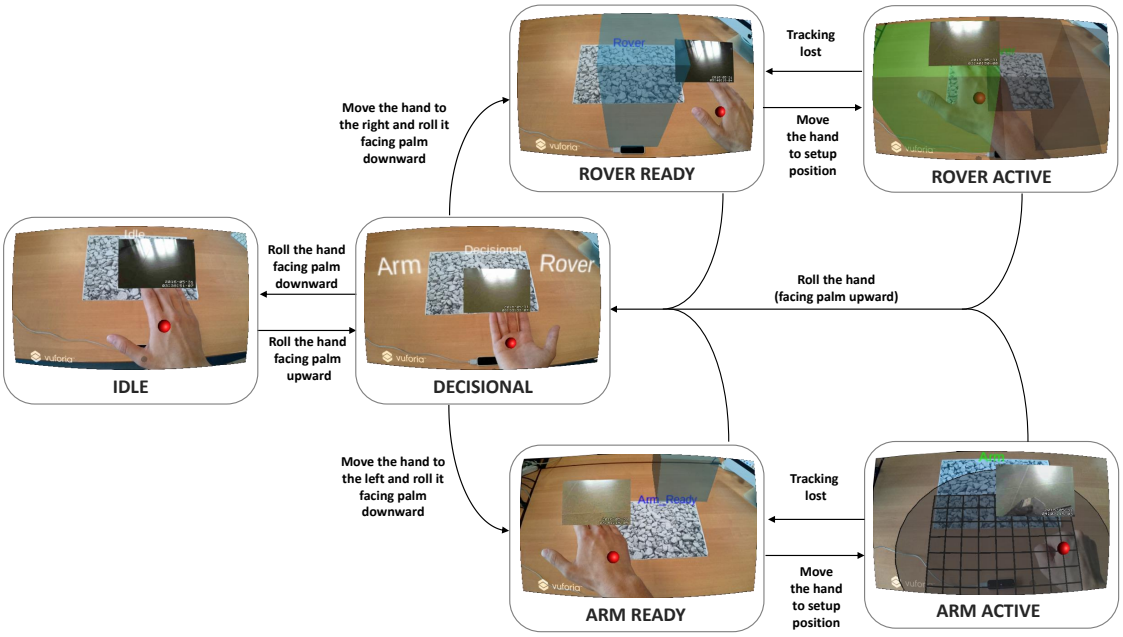


Figure 6.9: State diagram of the interface for the control of the rover and the robotic arm and screenshots showing the states of the AR interface.

When the operator rolls the hand facing the palm upward, the system moves into the *DECISIONAL* state. This state is designed to let the operator chose which robot to activate. The user interface is divided into two areas labeled with the name of the robot that can be activated (rover to the right, robotic arm to the left). If the user moves the hand to the left/right reaching one of the two sides and rotates the palm downward, the system moves to the dedicated state for controlling a specific robot (*ROVER READY* and *ARM READY*, respectively). At the beginning, interaction is not yet enabled for safety reasons. The interface presents a region in the working volume colored blue (different for the two robots). This region represents the safe position to be reached by the hand in order to

start the interaction with the given robot. This solution forces the operator to assume an initial pose preventing unwanted commands. When the hand reaches the safe region, the interaction with the selected robot starts, and continues until the transition to a different state occurs or when tracking is lost.

If the operator selects the rover (*ROVER ACTIVE*), he or she can drive it by moving the hand inside the region represented by four boxes in a cross configuration. Each box represents a different command that can be issued to the rover. For example, when the hand enters the box to the left, as shown in Fig. 6.9, the rover turns left. When the hand enters the farthest box, the rover moves forward. The region selected is colored green to provide the user with visual feedback about the command issued. The size of the interactive regions dynamically changes depending on the height reached by the operator's hand in order to consider the shape of the Leap Motion controller's tracking space. The value controlling the speed and the direction of the rover depends on the distance between the hand and the vertical axis of the Leap Motion controller.

When the control of the robotic arm is enabled (*ARM ACTIVE*), it can be manipulated through an inverse kinematics solver, which lets the operator move the robot's end-effector in all the directions by simply moving his or her hand within the tracking space in order to articulate all the joints of the robotic arm. An ellipsoid indicates the boundaries of the working volume at any given height as shown in Fig. 6.9. Boundaries become larger by moving the hand upward following the vertical axis of the Leap Motion controller, and vice versa. A grid on the bottom shows the position of the floor with respect to the end-effector.

Additional visual feedback information are associated with other relevant aspects of the interaction, such as transitions between states or tracking losses.

The interaction schema described above was defined in [45] through the evaluation of different alternatives with a trial and error process. For example, the orientation of the tracked hand can be directly used to manage the rover direction. However, despite its intuitiveness, this approach proved to be less accurate than the finally adopted approach. Another decision that was made regarded the position and shape of the boxes that identify the control commands for the rover. Differently than in a configuration where the entire interaction area is used, the cross configuration that was ultimately implemented optimizes the separation between different commands. Furthermore, this configuration provided higher flexibility compared to alternatives based on multiple control boxes, each designed to set a specific velocity. Regarding the control of the robotic arm, forward kinematics was experimented as well. However, inverse kinematics was chosen, since tele-operation proved to be faster with it.

Robot team

The robot team considered the work included a rover and a robotic arm, that are illustrated in Fig. 6.10.

The rover was built by assembling the Lynxmotion Aluminum 4WDI Robot Kit[4]. The operator can specify two control parameters, namely speed and direction, in order to drive it.

The robotic arm was constructed by assembling the Lynxmotion AL5D kit[5]. It is a manipulator with five DOFs that includes a base, a shoulder, an elbow, and two wrists joints. In addition, a servo motor is used to open and close a gripper mounted at the end of the kinematic chain. The hand's position tracked by the Leap Motion controller is leveraged to determine the angles of the first three joints (base, shoulder, elbow) through an inverse kinematics solver that maps the spatial coordinates of the hand's palm to the end-effector of the robotic arm. The orientation of the two wrist joints is controlled by means of the forward kinematics; specifically, the roll and pitch of the operator's hand are used to set the angles of the two joints. Gripper status (open/close) is controlled by opening/closing the palm.

Rover and arm are connected to the network through an Arduino board. The remote environment in which the robot is moving can be visualized through a webcam (Logitech C525) mounted on the robotic arm, which streams the video to the remote operator on a separate network connection managed by a Raspberry Pi.

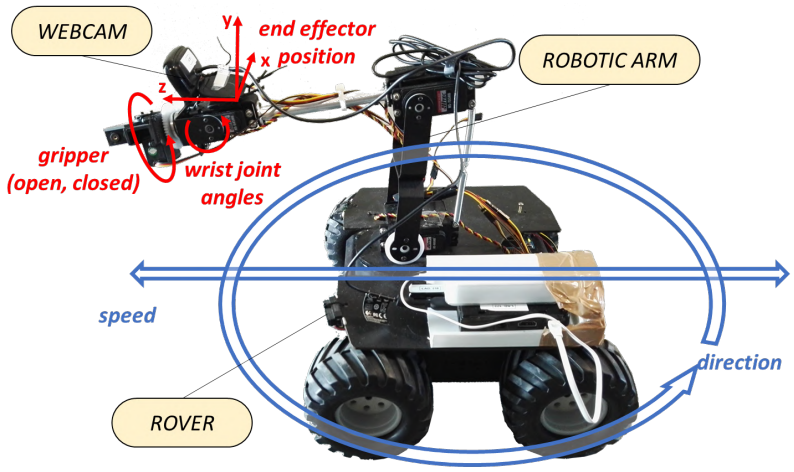


Figure 6.10: Robot team controlled using the proposed interface.

[4]Lynxmotion Aluminum 4WDI Robot Kit: http://www.lynxmotion.com/p-603-aluminum-4wd1-rover-kit.aspx

[5]Lynxmotion AL5D: http://www.lynxmotion.com/c-130-al5d.aspx

6.4.3 Experimental evaluation

In order to evaluate the effectiveness of the proposed interface, a user study was carried out by asking 15 participants (selected among students at Politecnico di Torino in Turin, Italy) to perform a *reach-and-pick* task. The task included two sub-tasks, each involving a different robot of the team.

In the first sub-task, participants were asked to tele-operate the rover making it follow a pre-defined path created by positioning physical elements on the ground (styrofoam boxes to define boundaries, paper tape to draw median strip, plastic ramp, etc.) as shown in Fig. 6.11. The sub-task was considered as completed when the rover reached a specific position marked by a colored rectangle at the end of the ramp. This position is used as the starting configuration for the second sub-task. In this sub-task, participants were requested to tele-operate the arm and explore the environment, in order to find two colored objects located to the left and to the right. Afterwards, they were invited to use the gripper to pick up the first object found (this way, experience was comparable for all the participants). The second sub-task was considered as completed as when the object was grabbed with the gripper.

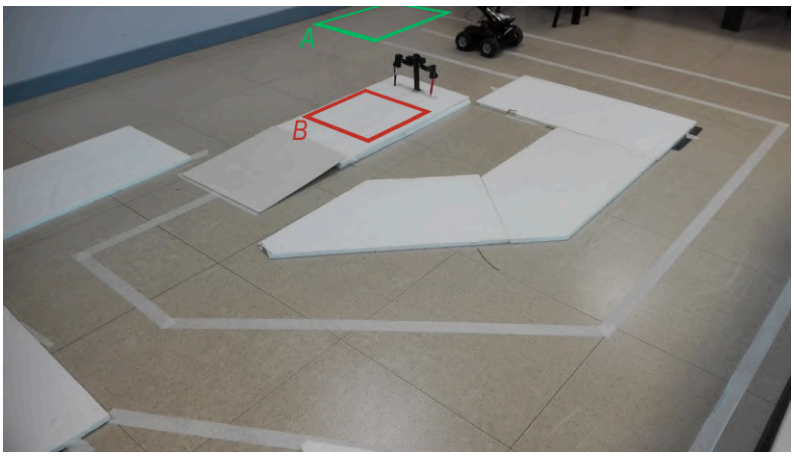
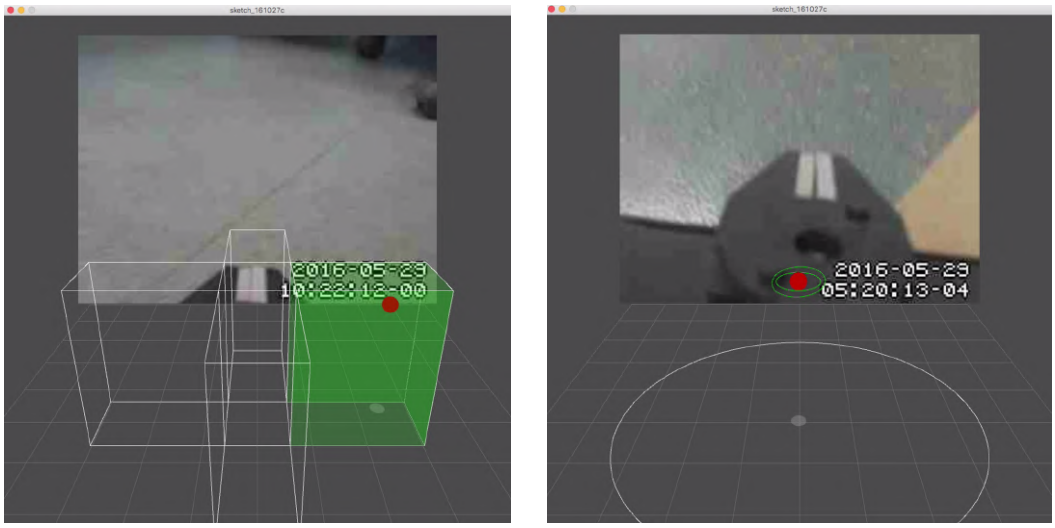


Figure 6.11: Configuration of the environment for the experimental task.

In the following, the two-subtasks are referred to as *rover* and *arm*. Each participant was requested to perform the sub-tasks by using both the desktop interface presented in [45] (later labeled *DI*) and the wearable interface (*WI*). The two screenshots in Fig. 6.12 and Fig. 6.12 show the DI when the controlled robots are the rover and the arm, respectively. In order to reduce learning effects, the interface to start with was selected in a random way.

Videos showing the execution of the two sub-tasks with both the DI[6] and the

[6]Video for the DI: https://drive.google.com/open?id=0B27BuRM-44ZhanN5Ujk2LUZuLWM



(a) Interface for controlling the rover

(b) Interface for controlling the robotic arm

Figure 6.12: Screenshots showing the desktop interface developed in [45].

WI[7] are available for download.

Participants were given time to familiarize with robot control, by letting them interact with both the rover and the arm using the two interfaces. During this time, they were introduced to the functioning of the Leap Motion controller and gestures supported. Furthermore, they were instructed to interpret the information provided by the two interfaces, and they were invited to get accustomed with tele-operation by controlling robot's parts with the sole support of the video streamed by the on-board camera.

In order to evaluate the performance obtained by the two interfaces, both objective and subjective evaluations were considered. Concerning objective evaluation, the time needed to execute each sub-task was measured. Furthermore, the number of control commands issued during the execution of the task was collected. For the rover sub-task, control commands were easy to distinguish, since they were represented by the number of time the operator's hand entered a control box. For the arm sub-task, given the continuous nature of the inverse kinematics-based control mechanism used, a metric based on the number of trials was adopted, measuring the number of times a participant closed the gripper to grab the object (possibly missing it).

The objective evaluation was accompanied by a subjective evaluation based

[7]Video for the WI: https://drive.google.com/open?id=0B27BuRM-44ZhSmlrbUhlaFo2SGc

on feedback collected through a questionnaire completed at the end of the experiment by each participant. The questionnaire included five sections, encompassing a rather broad set of questions adapted from different analysis tools in order to reach a comprehensive and multi-faceted understanding of participants' experience. Questions were expressed in the form of statements to be evaluated (for each sub-task and for each interface) on a five-point Likert scale from 0 (strong disagreement) to 4 (strong agreement).

The first section aimed to investigate the usability of the two interfaces based on attributes defined by Nielsen in [228]. Thus, five questions were asked to evaluate learnability, efficiency, memorability, (recovery from) errors and satisfaction.

The second and third sections were defined based on the questionnaire proposed in [267], which evaluates the usability of handheld AR devices and applications. In particular, 10 questions investigating the comprehensibility (level of understanding of the information presented) and four questions focusing on manipulability (ease of handling) of the interface were included in these two sections.

The fourth section included eight questions on robot control based on the evaluation approach proposed in a work with a similar aim [119].

The last section included nine questions based on usability heuristics defined in [98, 271].

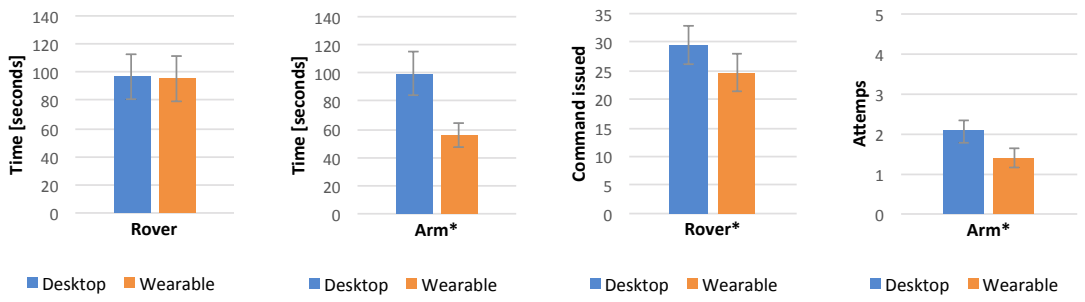
Participants were also invited to express their preference for the DI or the WI for each sub-task.

6.4.4 Results

Objective results obtained in terms of completion time and control commands are reported in Fig. 6.13. Bars height represents the average value (lower is better). Standard deviation is also reported through error bars. The * symbol is used to indicate when differences were statistically significant. Statistical significance was analyzed by running paired samples Student's t-tests ($p < 0.05$).

As shown in Fig. 6.13a, the difference in completion time for the rover sub-task with the two interfaces was not significant (DI: $M = 1$ m 37 s, $SD = 33$ s; WI: $M =$ 1 min 35 s, $SD = 33$ s). However, for the arm sub-task (Fig. 6.13b), participants were significantly faster ($t(14) = -3.88$, $p < 0.01$, $d = -1.68$) with the WI ($M = 1$ min 40 s, $SD = 33$ s) than with the DI ($M = 56$ s, $SD = 17$ s).

Regarding control commands (Fig. 6.13c and Fig. 6.13d), it can be noticed that the number of commands issued for the rover sub-task was significantly lower ($t(14) = -2.42$, $p = 0.03$, $d = -0.69$) with the WI ($M = 24.67$, $SD = 6.76$) than with the DI ($M = 29.40$, $SD = 6.90$). Moreover, participants, in the arm sub-task, made a number of attempts significantly lower ($t(14) = -4.18$, $p < 0.01$, $d = -1.21$) with the WI ($M = 1.40$, $SD = 0.51$) than with the DI ($M = 2.07$, $SD = 0.59$).



(a) (b) (c) (d)

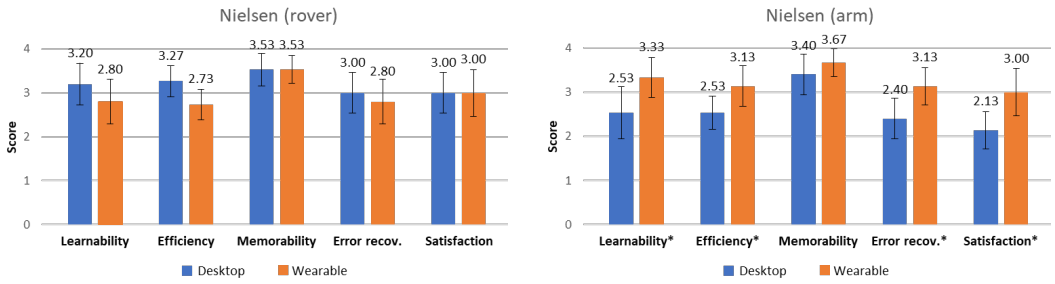
Figure 6.13: Objective results in terms of completion time and control command for the two sub-tasks (performed by tele-operating the rover and the robotic arm) by using the DI and the WI.

Results concerning Nielsen's attributes are depicted in Fig. 6.14. Results collected in the remaining sections of the questionnaire are reported in Table 6.3– 6.6, in order to present also the actual formulation for each statement. Statistical significance of data collected was studied by using the same approach adopted for the objective evaluation. Statistically significant results are marked with a * symbol.

According to Fig. 6.14a, results achieved for the rover sub-task were not statistically significant. When the arm sub-task is analyzed, performance with the WI appears to be significantly better than with the DI for all the attributes (except for memorability). More specifically, significant differences were found for the learnability (DI: $M = 2.53$, $SD = 1.19$ s; WI: $M = 3.33$, $SD = 0.90$ s; $t(14) = -3.29$, $p < 0.01$, $d = -0.76$), efficiency (DI: $M = 2.53$, $SD = 0.74$ s; WI: $M = 3.13$, $SD = 0.92$ s; $t(14) = -2.80$, $p = 0.01$, $d = -0.72$), errors (DI: $M = 2.40$, $SD = 0.91$ s; WI: $M = 3.13$, $SD = 0.83$ s; $t(14) = -2.95$, $p = 0.01$, $d = -0.84$), and satisfaction (DI: $M = 2.13$, $SD = 0.83$ s; WI: $M = 3.00$, $SD = 1.07$ s; $t(14) = -2.69$, $p = 0.01$, $d = -0.90$). The above difference can be appreciated in Fig. 6.14b.

Regarding comprehensibility (Table 6.3), an interesting result (which is statistically significant) is that, for the arm sub-task, the DI required a higher mental effort (concentration) than the WI ($t(14) = 2.25$, $p = 0.04$, $d = 0.74$). This result could be linked to the benefits brought by affordances associated with the visualization of augmented contents on the operator's hand that is possible in the WI. No significant differences were found between the two interfaces concerning appropriateness, readability, meaningfulness and responsiveness of information displayed.

With respect to interface manipulability (Table 6.4), results confirm the outcomes already observed using Nielsen's attributes. In particular, statistical significance was verified only for the question concerning easiness of issuing robot control commands ($t(14) = -2.32$, $p = 0.03$, $d = -0.71$) and easiness of the interface ($t(14) = -2.36$, $p = 0.03$, $d = -0.62$) for the arm sub-task. Participants found it



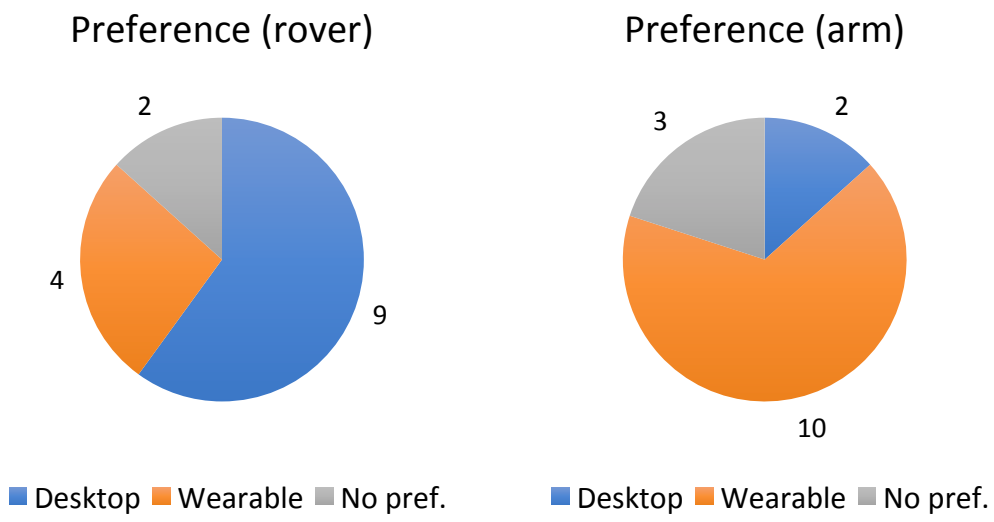
(a) (b)

Figure 6.14: Subjective results concerning Nielsen's attributes of usability for the two sub-tasks using the DI and the WI.

easier to issue commands with the WI rather than with the DI. Moreover, the WI was found to be simpler and less difficult to use than the DI.

Interesting findings can be observed in the fourth section regarding robot control (Table 6.5), where results in terms of users' preferences depend on the sub-task considered. In fact, for the rover-task, participants found it easier to drive the robot with the DI than with the WI ($t(14) = 2.20$, $p = 0.04$, $d = 0.68$), and stated that they would like to control the robot that way ($t(14) = 2.57$, $p = 0.02$, $d = 0.73$). Conversely, opposite results are obtained when considering the arm sub-task. More specifically, participants found it more difficult to control the robotic arm with the DI than with the WI ($t(14) = -3.21$, $p < 0.01$, $d = -0.80$), and reported that they would like to control the robot in a different way ($t(14) = -2.86$, $p = 0.01$, $d = -0.69$). Furthermore, for the arm sub-task, statistically significant results were obtained also for statements regarding confidence in robot control ($t(14) = -4.02$, $p < 0.01$, $d = -0.96$), consistency ($t(14) = -3.57$, $p < 0.01$, $d = -0.64$) and interface suitability for the specific operation ($t(14) = -2.47$, $p = 0.02$, $d = -0.59$), confirming the improved performance of the WI for the control of the robotic arm.

These opposite results can also be observed for some statements in the last section of the questionnaire (Table 6.6). In fact, when driving the rover, participants felt less confused with the DI than with the WI ($t(14) = 2.36$, $p = 0.03$, $d = 0.68$), whereas for the arm sub-task the opposite rating was observed ($t(14) = -2.20$, $p = 0.04$, $d = -0.568$). This section provides other interesting outcomes. First, it indicates that, in the rover sub-task, participants found the layout for the visualization of 3D contents more appropriate with the DI than with the WI ($t(14) = 2.32$, $p = 0.03$, $d = 0.75$). This is probably due to the fact that, sometimes, the control regions in the interface are hidden by the window showing the video streamed by the robot's camera. This limitation was far less critical when operating with the robotic arm. Second, as expected, participants felt that their eyes were much more tired after having operated with the WI than with the DI in both the rover ($t(14) = -4.00$, $p < 0.01$, $d = -1.53$) and the arm sub-task ($t(14) = -4.00$,



(a) (b)

Figure 6.15: Preference expressed by participants for a specific interface in the execution of the two sub-tasks.

$p < 0.01$, $d = -1.53$).

Finally, helpful indications can be obtained by considering preferences expressed by the participants concerning the use of a specific interface to carry out the two sub-tasks. As shown in Fig. 6.15, participants clearly preferred the DI for teleoperating the rover, whereas, when the robotic arm needed to be controlled, preference was, conversely, almost completely for the WI.

6.4.5 Future developments

Given the promising results obtained, future work could focus on further investigating the operators' preference for a specific interface by taking into account a wider set of tasks to be performed. Moreover, alternative ways to issue control commands as well as to visualize the working volume, interaction regions, etc. could be evaluated. Finally, the applicability of AR to deliver effective interaction feedback with other input mechanisms could be studied.

6.5 Concluding remarks

This chapter described solutions aimed at providing users with helpful feedback that can support their activities in different domains. More specifically, in [43], the creation of reconfigurable passive haptics based on toy construction bricks available

Table 6.3: Statements used in the subjective evaluation (second section of the questionnaire).

	Rover		**Arm**	
Comprehensibility of information	**DI**	**WI**	**DI**	**WI**
I think that interaction requires a lot of mental effort and concentration	1.40 (0.80)	1.67 (1.01)	2.00* (1.03)	1.20* (1.05)
I thought the amount of information displayed on screen was appropriate	3.40 (0.88)	3.07 (1.00)	3.27 (0.85)	3.33 (0.79)
I thought that the information displayed on screen was difficult to read and interpret	0.67 (0.94)	0.80 (0.75)	0.73 (0.77)	0.87 (0.81)
I felt that the information display was responding fast enough	2.67 (1.35)	2.67 (1.40)	2.40 (1.31)	2.67 (1.53)
I thought that the information displayed on screen was confusing	1.00 (1.21)	1.20 (1.17)	1.27 (1.18)	0.93 (1.06)
I thought that the meaning of information displayed on screen was self-explanatory	3.00 (1.26)	3.00 (1.15)	2.80 (1.33)	3.07 (1.18)
I thought the words and symbols on screen were easy to read	3.80 (0.40)	3.60 (0.61)	3.73 (0.44)	3.67 (0.47)
I felt that the information displayed on the screen was flickering too much	0.73 (1.12)	1.07 (1.24)	0.87 (1.20)	1.00 (1.10)
I thought that the information displayed on screen was consistent	3.27 (1.06)	3.27 (0.85)	3.20 (1.11)	3.20 (0.83)

Table 6.4: Statements used in the subjective evaluation (third section of the questionnaire).

	Rover		**Arm**	
Manipulability of the interface	**DI**	**WI**	**DI**	**WI**
I thought that interaction requires a lot of body muscle effort	0.87 (1.20)	0.87 (1.20)	0.93 (1.18)	0.87 (1.15)
I felt that using the application was comfortable for my arm and hands	2.93 (1.12)	2.93 (1.12)	2.73 (1.18)	2.73 (1.18)
I found it easy to issue commands through the interface	3.13 (0.62)	2.87 (0.72)	2.60* (1.02)	3.26* (0.77)
I thought the operation of the interface is simple and uncomplicated	3.27 (0.93)	3.33 (0.70)	2.73* (1.12)	3.33* (0.70)

on the market was presented. The use of this approach to implement passive haptics promises to be more flexible compared to other solutions presented in the literature. The capabilities of the proposed approach were evaluated by using reconfigurable

Table 6.5: Statements used in the subjective evaluation (fourth section of the questionnaire).

	Rover		Arm	
Robot control	**DI**	**WI**	**DI**	**WI**
It was easy to control the robot this way	3.20* (0.83)	2.60* (0.88)	2.40* (0.80)	3.13* (0.96)
I controlled the robot with confidence	3.13 (0.88)	2.87 (0.81)	2.27* (0.77)	3.13* (0.96)
I would like to control the robot this way	3.20* (0.83)	2.40* (1.25)	2.40* (1.02)	3.20* (1.22)
A lot of training is necessary for using this interface	0.47 (0.50)	0.53 (0.50)	0.80 (0.75)	0.67 (0.60)
Many people could control the robot easily using this interface	3.20 (0.98)	3.00 (0.89)	2.73 (1.00)	2.93 (1.18)
I could control the robot as I expected	3.27 (0.93)	2.87 (0.81)	2.40* (1.08)	3.07* (0.93)
The task was very difficult to execute	0.80 (1.05)	1.13 (1.09)	1.20 (1.05)	1.00 (1.10)
The interface is suitable to control the robot and execute the task	3.33 (0.79)	2.87 (1.09)	2.93* (0.93)	3.47* (0.81)

passive haptics in the execution of a number of tasks in an immersive environment.

In [49], a VR-based training system supporting users in the training of sport gestures is presented. The system does not require the presence of a real trainer, since the use of real-time visual feedback allows the user to self-evaluate his or her performance both during and after the execution of the gesture directly within the VR environment. Moreover, an object tracking module was developed to enable physical interaction with other objects (e.g., a ball) during the training session. Results achieved with a small-scale laboratory setup on a basketball-related movement revealed the possible advantages for improving both gesture timing and spatial positioning brought by the proposed system. It is worth noting that, although the main focus of the system proposed was biomechanical training, the introduction of other virtual assets in the environment (like different levels of sound noise, moving audience, changing lights, etc.) could enable its adoption also for psychological training.

Finally, in [48], a wearable AR-based interface aimed to support an operator during hand tracking-based tele-operation of remote robot teams including a rover equipped with a robotic arm has been proposed. AR is used to provide the operator with a visual feedback about the position of his or her hand in the working space in order to make it easier to interact with the robot. The proposed interface has been

Table 6.6: Statements used in the subjective evaluation (fifth section of the questionnaire).

	Rover		**Arm**	
Other heuristics	**DI**	**WI**	**DI**	**WI**
I felt that my arm or hand became tired after using the interface	1.20 (1.22)	1.20 (1.33)	1.27 (1.24)	1.13 (1.26)
I felt that my eyes became tired after using the interface	0.27* (0.57)	1.87* (1.31)	0.27* (0.57)	1.87* (1.31)
Layout for the visualization of 3D contents is visually pleasant	3.13 (0.96)	2.60 (1.14)	3.00 (0.89)	2.93 (0.93)
Layout for the visualization of 3D contents is efficient	3.47* (0.81)	2.80* (0.91)	3.13 (0.81)	2.93 (0.77)
The system provided me with a proper feedback about what I was working on	3.07 (0.93)	2.93 (1.06)	2.67 (1.14)	3.00 (0.97)
I was not confused or lost while performing the task	3.40* (0.71)	2.80* (0.98)	2.80* (1.22)	3.40* (0.80)
I was not requested to memorize things unnecessarily	3.87 (0.34)	3.93 (0.25)	3.80 (0.54)	3.93 (0.25)
The first-time experience with the interface is encouraging	3.33 (0.79)	3.27 (1.06)	2.93 (1.12)	3.33 (0.87)
It was easy to find the desired control options at any time	3.40 (0.95)	3.27 (1.00)	3.13 (1.20)	3.33 (0.94)

compared with a similar solution in which AR contents are displayed on the screen of a desktop computer. Objective observations suggested that when users have to explore the surrounding environment and pick up objects using the robotic arm, the wearable interface is considered better than the desktop interface for what it concerns both time and number of attempts required. When driving the rover, the wearable interface appeared to be capable only to reduce the number of commands issued. Subjective results clearly showed a larger appreciation for the wearable interface to control the robotic arm. Regarding rover operation, it was not possible to identify a clear advantage in the use of one of the two interfaces. In fact, if on the one hand, the lower number of commands to be issued could make the wearable interface more attractive for driving the rover, on the other hand, eye fatigue could make the desktop interface better than the other one.

Chapter 7

Conclusions

This document presented research that has been carried out within the duration of my PhD studies to design and develop interfaces that can support users in the creation and use of computer graphics assets in different application domains, ranging from 3D animation to sport training, public exhibitions management, interaction with computers and robots, etc. Attention was focused on aspects pertaining to human-centered design to account for the fact that ever more users with different skills and backgrounds are addressing the computer graphics field, each with his or her own needs and requirements. Besides involving them in the design steps, which led to the development of solutions able to satisfy the requirements set by the specific application being considered, the different user categories were also involved in experimental evaluations in order to assess the effectiveness of such solutions and to collect feedback for further improvements.

Content presented in this document has been structured following the stages that characterize a computer graphics pipeline, since the challenges related to the computer graphics field concern the whole process, from the creation to the utilization of graphics assets. By moving from this consideration, Chapters 2–4 focused on the generation of computer graphic assets. In particular, the field of computer animation was studied extensively in the first two chapters, by considering both issues regarding user input and system output.

With respect to the user input, Chapter 2 investigated the use of non-traditional interfaces based on reconfigurable tangible props and/or voice commands, as well as alternative interaction methods based on animator's performance and NLP for generating virtual character animations.

Experimental results showed that the system based on TUIs and NUIs presented in [173] was considered a significantly more intuitive way for animating virtual characters compared to traditional approaches based on M&K. Besides the improved intuitiveness, these alternative interfaces proved to make the animation of virtual characters a less time-consuming operation compared to conventional approaches. An interesting outcome of the above work is represented by the fact that the use of

the proposed interfaces was shown to make novice users achieve results comparable to those obtained by expert users with traditional methods, thus confirming the capability of the investigated interfaces in leveling skills among users. Although encouraging results were obtained, the improvements brought by TUIs and NUIs were paid with a reduced accuracy and an increased physical effort.

Results also confirmed the usability, intuitiveness, and effectiveness of the system based on the use of animator's performance and NLP in [172], especially for specific contexts like fast animation prototyping or interactive storytelling. However, limitations in terms of flexibility of animations that could be generated through the devised system could prevent its adoption in scenarios in which animators are more interested in fine-grained control of the produced output rather than in interface intuitiveness or simplicity.

Regarding system output, the limited dimensionality of traditional solutions shifted the attention of the research community on alternative methods based on VR, since this technology enables interactions with 3D objects by means of interfaces that are natively 3D, thus improving the users' sense of presence, as well as their creativity and productivity. In this scenario, immersive VR was specifically considered in Chapter 3, and its usage was studied in the context of a traditional pipeline for creating animations. Differently than in majority of work found in the literature, the VR systems explored in the chapter were not meant to replace traditional graphics suites, but rather to improve them by proposing alternative interaction methods. This focus is confirmed by the integration of these VR systems into a well know animation suite. This way, multiple users are allowed to operate simultaneously on the same animation with both the traditional and the VR modality.

More specifically, in the first step of the above investigation, the advantages brought by the use of VR technology for the execution of five representative animation tasks were analyzed through a comprehensive user study originally published in [171]. The results revealed that VR allows both expert and novice users to create animations faster than with the traditional approach. Moreover, the users expressed a higher appreciation for the VR-based interface than for the native, M&K-based one. The VR technology was judged as more usable and stimulating than conventional interaction methods, being perceived as easier to learn and to use. Benefits were more evident when analyzing the results and preferences achieved/expressed by novice users, whereas performance of expert users was almost comparable.

By considering the encouraging results achieved with the above work, a drill-down in the specific field of virtual character animation was performed, by bringing to VR all the functionalities required to carry out required tasks, and then testing their effectiveness though a further user study. In particular, the rigging, skinning and posing steps were considered. Experimental results reported in [51] confirmed the fact that advantages gained with the use of VR that had been already observed for general-purpose animation can be extended also to generating virtual character

animations, since the improvements in terms of completion time, intuitiveness and usability with respect to traditional approaches were preserved.

Afterwards, the attention was shifted to the character posing step, by proposing new interfaces based on reconfigurable tangible props [47] and 3D sketches to manipulate the skeleton of virtual characters into immersive virtual environments. Although some of the activities reported in the last sections of Chapter 3 are still in progress, it can be observed that, compared to solutions based on 2D visualization, the devised techniques could help the users to reduce the time needed for articulating the characters, by also letting them achieve better results in terms of accuracy. As already observed in Chapter 2, the use of TUIs confirmed to be capable of reducing the impact of previous experience, letting users with limited skills to achieve results comparable to those of skilled users. The VR-based approach was found to be characterized by a lower mental effort and higher usability; the possible drawback is represented by a slightly higher physical effort, which may represent a possible issue for long-lasting animation sessions.

Chapter 4 reported solutions that were developed by considering new technologies, like VR and AR, as well as automatic methods leveraging intelligent computing techniques to make users who are usually not familiar with computer graphics tools to produce more generic graphics assets.

In particular, in the first part of the chapter, unskilled users, e.g., storytellers, screenwriters, etc. were considered, and an automatic system to support them in the generation of a full 3D scene from a 2D image was proposed. The basic idea, originally reported in [5], was also to provide users with an intuitive tool allowing them to explore into an immersive environment the scene that was automatically generated and apply changes to its layout by means of 3D interactions. The promising results revealed the capabilities of the devised approach to recreate a scene similar to the input provided, as well as to enhance the user's understanding of the objects' layout, making the tool suitable for applications like fast prototyping.

In the second part of the chapter, specific use cases were considered regarding constructive art and sport training. Even these fields, that are experiencing the diffusion of methods based on new technologies to ease the development of involved tasks, are characterized by users, like artists and coaches, who probably are not familiar with the generation of graphics assets. For the first use case (based on work reported in [46] and [14]), the graphics assets to be created/manipulated are represented by artworks. Two alternative AR-based systems were proposed, allowing users to manipulate and configure, through intuitive interfaces based on tangible props and tag tools, the components to be used for assembling the artworks. In the second use case (which was addressed in [50] and [6]), the devised system allows coaches to reconstruct basketball tactics and realistic players' movements, which can then be visualized within a VR environment. Although extensive experimental evaluations have still to be carried out in order to assess their actual effectiveness, comments and feedback provided by the users who tested the proposed solutions

were, in general, very positive.

After having discussed graphics assets creation, Chapter 5 addressed aspects regarding access to and interaction with them. To this aim, the domain of interactive graphics applications targeted to public exhibitions was specifically taken into account. In particular, discussion focused on the design of a tool created by leveraging principles of EUD to allow users with limited programming skills to build 3D interactive applications with a VPL [4]. The results achieved through the two user studies carried out by involving both users with 3D graphics and programming skills, and possible target end-users confirmed the advantages of the proposed tool compared to both a traditional graphics suite and a previous VPL-based tool in the literature. Besides improved usability, advantages concerning time requested to perform tasks of interest (which was shorter with the proposed tool than the other ones), number of errors made and building components needed (which were found to be lower with the devised tool).

Lastly, Chapter 6 tackled the last stage of graphics pipeline, focusing on the use of created assets into specific application domains. In particular, the attention was posed on HMI aspects, investigating the possibility of employing graphics assets, and innovative interfaces, to provide users with improved feedback on aspects regarding the operations being performed. More specifically, the use of reconfigurable TUIs as passive haptic feedback sources was investigated with the aim to enable more natural and realistic interactions with the components of a VR immersive application [43]. The higher flexibility of the proposed approach with respect to other solutions in the literature was demonstrated by making us of the devised passive haptics for the execution of a number of tasks of an escape room game. Moreover, two other systems originally reported in [48] and [49] were analyzed to observe how visual feedback can support users into two different areas, namely sport training and HRI.

In particular, the work in [49] explored the use of a VR training system to support self-evaluation of user performance within an immersive environment. The system is able to provide the users with real-time visual feedback that can be used to improve the execution of technical gestures. Experimental results obtained by considering a basketball-related gesture showed improvements brought by the proposed system with respect to a traditional training method both in terms of timing and spatial positioning.

In [48], a wearable AR interface was described that provides a robot operator with visual feedback that can make more intuitive and effective the control of a robotic team with a hand tracking-based tele-operation system. Experimental results revealed that the wearable AR interface achieved better performance (in terms of task duration, number of commands issued and users' satisfaction) when tasks requiring the robots to explore the surrounding environment or pick objects are considered.

In conclusion, the research presented in this document suggested that the use of

innovative interfaces (reconfigurable TUIs, NUIs, etc.), and emerging technologies (such as VR and AR) brings several advantages to the domains of interest in terms of both objective aspects (like the time needed to carry out the assigned task) and subjective aspects (like users' satisfaction and system usability). Moreover, one of the factors that characterizes most of the solutions presented in this document is the improved intuitiveness they can offer compared to existing alternatives. This aspect allows users with different needs and skills to directly approach fields which were hardly accessible for them because of the significant complexity of traditional interfaces.

These advantages were found not only for what it concerns applications targeted to the generation of graphics contents (like animated characters and virtual scenes), but also in those contexts in which graphics assets are used, like, e.g., interactive applications for public exhibits, HRI, and training.

Thus, a confirmation was obtained of the fact that benefits coming from the adoption of the above interfaces and technologies in common application scenarios reflect on the whole computer graphics pipeline, not only on specific stages.

Future researches in this domain can take advantage of the results achieved in this thesis to provide different user categories with new ways to generate and interact with graphic assets, which were proved to be more efficient in terms of time, performance, usability, and intuitiveness.

Glossary

Acronyms / Abbreviations

$1PP$ First-Person Perspective

$2D$ Two-Dimensional

$2D$ Two-Dimensional

$3D$ Three-Dimensional

$3D$ Three-Dimensional

$3DUI$ 3D User Interface

$3PP$ Third-Person Perspective

ACM Association for Computing Machinery

API Application Program Interface

AR Augmented Reality

BGE Blender Game Engine

CAD Computer-Aided Design

$CAVE$ Cave Automatic Virtual Environment

CG Computer Graphics

CSV Comma-Separated Values

DOF Degree Of Freedom

DTW Dynamic Time Warping

EUD End-User Development

EUP End-User Programming

FBX Filmbox

$FIBA$ International Basketball Federation

FK Forward Kinematics

GUI Graphical User Interface

HCI Human-Computer Interface

HMD Head-Mounted Display

HMI Human-Machine Interaction

HOT Hold your Own Tools for AR-based constructive art

HRI Human-Robot Interaction

IK Inverse Kinematics

IoT Internet of Things

ISO International Organization for Standardization

$JSON$ Java Script Object Notation

KNN K-Nearest Neighbors

LE Leap Embedder

LOA Line Of Action

$LXFML$ LEGO Digital Designer XML

$M\&K$ Mouse and Keyboard

$MFCC$ Mel-Frequency Cepstral Coefficient

MVC Model View Controller

$NASA-TLX$ NASA Task Load Index

NBA National Basketball Association

NLA NonLinear Animation

NLP Natural Language Processing

NUI Natural User Interface

P/O Position or Orientation

$P\&O$ Position and Orientation

PDA Personal Digital Assistant

PSO Particle Swarm Optimization

PUN Photon Unity Networking

RF Random Forests

$RGB-D$ Red Green Blue Depth

RMS Root Mean Square

SDK Software Development Kit

SUS System Usability Scale

SVD Singular Value Decomposition

SVM Support-Vector Machines

$T4T$ Tangible interface 4 Tuning 3D object manipulation tools

TUI Tangible User Interface

USB Universal Serial Bus

VPL Visual Programming Language

VR Virtual Reality

$VRML$ Virtual Reality Modeling Language

VSE Visual Scene Editor

$WebGL$ Web Graphics Library

$Wi-Fi$ Wireless Fidelity

$WIMP$ Windows-Icons-Menus-Pointer

WPF Windows Presentation Foundation

$X3D$ Extensible Three-Dimensional

XML Extensible Markup Language

Bibliography

[1] Merwan Achibet et al. “Elastic-Arm: Human-scale passive haptic feedback for augmenting interaction and perc. in virtual environments”. In: *Proceedings of the IEEE VR*. 2015, pp. 63–68.

[2] Merwan Achibet et al. “The Virtual Mitten: A novel interaction paradigm for visuo-haptic manipulation of objects using grip force”. In: *Proceedings of the IEEE 3DUI*. 2014, pp. 59–66.

[3] Jonggil Ahn, Euijai Ahn, and Gerard J Kim. “Robot Mediated Tele-presence through Body Motion Based Control”. In: *Proceedings of the International Conference on Systems, Man, and Cybernetics*. 2015, pp. 463–468.

[4] Cannavò Alberto et al. “A visual editing tool supporting the production of 3D interactive graphics assets for public exhibitions”. In: *International Journal of Human–Computer Studies* 141.1 (2020), p. 102450.

[5] Cannavò Alberto et al. “Automatic generation of affective 3D virtual environments from 2D images”. In: *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. 2020, pp. 113–124.

[6] Cannavò Alberto et al. “Automatic recognition of sport events from spatio-temporal data: an application for virtual reality-based training in basketball”. In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. 2019, pp. 310–316.

[7] Jane Alexander, Jake Barton, and Caroline Goeser. “Transforming the art museum experience: Gallery One”. In: *Proceedings of Museums and the Web*. 2013.

[8] Roberto Andreoli et al. “A framework to design, develop, and evaluate immersive and collaborative serious games in cultural heritage”. In: *Journal on Computing and Cultural Heritage* 11.1 (2018), p. 4.

[9] Michelle Annett et al. “The pen is mightier: understanding stylus behaviour while inking on tablets”. In: *Proceedings of Graphics Interface 2014*. 2014, pp. 193–200.

[10] Carmelo Ardito et al. “From smart objects to smart experiences: An end-user development approach”. In: *International Journal of Human-Computer Studies* 114.1 (2018), pp. 51–68.

[11] Dennis Arsenault and Anthony Whitehead. “Wearable sensor networks for motion capture”. In: *Proceedings of the 7th International Conference on Intelligent Technologies for Interactive Entertainment*. 2015, pp. 158–167.

[12] Vikas G Ashok, Song Feng, and Yejin Choi. “Success with style: Using writing style to predict the success of novels”. In: *Proceedings of the conference on empirical methods in natural language processing*. 2013, pp. 1753–1764.

[13] Halim Cagri Ates, Alexander Fiannaca, and Eelke Folmer. “Immersive simulation of visual impairments using a wearable see-through display”. In: *Proceedings of the 9th ACM International Conference on Tangible, Embedded, and Embodied Interaction*. 2015, pp. 225–228.

[14] Giuseppe Attanasio et al. “HOT: Hold your own tools for AR-based constructive art”. In: *Proceedings of the 12th Symposium on 3D User Interfaces*. 2017, pp. 256–257.

[15] Daniel Avrahami, Jacob O Wobbrock, and Shahram Izadi. “Portico: tangible interaction on and around a tablet”. In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 2011, pp. 347–356.

[16] Seongmin Baek, Seungyong Lee, and Gerard J Kim. “Motion retargeting and evaluation for VR-based training of free motions”. In: *The Visual Computer* 19.4 (2003), pp. 222–242.

[17] RM Baños et al. “Sense of presence in emotional virtual environments”. In: *Presence* 1.1 (2004), pp. 156–159.

[18] Ilya Baran and Jovan Popović. “Automatic rigging and animation of 3d characters”. In: *ACM Transactions on Graphics* 26.3 (2007), 72–es.

[19] Simone Barbieri et al. “Enhancing Character Posing by a Sketch-based Interaction”. In: *Proceedings of the SIGGRAPH 2016 Posters*. 2016, 56:1–56:2.

[20] Connelly Barnes et al. “Video puppetry: A performative interface for cutout animation”. In: *ACM SIGGRAPH Asia 2008 papers*. 2008, pp. 1–9.

[21] Barbara Rita Barricelli et al. “End-user development, end-user programming and end-user software engineering: A systematic mapping study”. In: *Journal of Systems and Software* 149.1 (2019), pp. 101–137.

[22] Rüdiger Beimler, Gerd Bruder, and Frank Steinicke. “Smurvebox: a smart multi-user real-time virtual environment for generating character animations”. In: *Proceedings of the Virtual Reality International Conference: Laval Virtual*. 2013, pp. 1–7.

[23] Mafkereseb Kassahun Bekele et al. “A survey of augmented, virtual, and mixed reality for cultural heritage”. In: *Journal on Computing and Cultural Heritage* 11.2 (2018), p. 7.

[24] John T Bell and Scott H Fogler. “The investigation and application of virtual reality as an educational tool”. In: *Proceedings of the American Society for Engineering Education Annual Conference*. 1995, pp. 1718–1728.

[25] Julien Cohen Bengio, Kaori Ogino, and Barnaby Robson. “A holistic approach to asset quality and efficiency”. In: *ACM SIGGRAPH 2018 Talks*. 2018, pp. 1–2.

[26] Bruna Berford et al. “Building an animation pipeline for VR stories”. In: *Proceedings of the ACM SIGGRAPH 2017 Talks*. 2017, pp. 1–2.

[27] Donald J Berndt and James Clifford. “Using dynamic time warping to find patterns in time series”. In: *Proceedings of the KDD workshop*. 1994, pp. 359–370.

[28] Emmanuel Bernier et al. “The ICEA plug-in for virtual reality, immersive creation and edition of animation”. In: *Proceedings of the 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems*. 2012, pp. 36–42.

[29] Mikhail Bessmeltsev, Nicholas Vining, and Alla Sheffer. “Gesture3D: Posing 3D characters via gesture drawings”. In: *ACM Transactions on Graphics* 35.6 (2016), 165:1–165:13.

[30] Benoit Bideau et al. “Using virtual reality to analyze sports performance”. In: *IEEE Computer Graphics and Applications* 30.2 (2009), pp. 14–21.

[31] Aude Billard et al. “Survey: Robot programming by demonstration”. In: *Handbook of robotics* 59.1 (2008), pp. 1371–1394.

[32] Mark Billinghurst, Hirokazu Kato, Ivan Poupyrev, et al. “Tangible augmented reality”. In: *ACM SIGGRAPH ASIA* 7.2 (2008), pp. 1–11.

[33] Mark Billinghurst, Tham Piumsomboon, and Huidong Bai. “Hands in space: Gesture interaction with augmented-reality interfaces”. In: *IEEE Computer Graphics and Applications* 34.1 (2014), pp. 77–80.

[34] Stefano Bonaiuto et al. “Tele-operation of robot teams: a comparison of gamepad-, mobile device and hand tracking-based user interfaces”. In: *Proceedings of the 41st Annual Computer Software and Applications Conference*. 2017, pp. 555–560.

[35] Miguel Borges et al. “HTC Vive: analysis and accuracy improvement”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2018, pp. 2610–2615.

[36] Péter Borosán et al. “Rigmesh: Automatic rigging for part-based shape modeling and deformation”. In: *ACM Transactions on Graphics* 31.6 (2012), pp. 1–9.

[37] Stéphane Bouchard et al. “Virtual reality and the training of military personnel to cope with acute stressors”. In: *Advanced Computational Intelligence Paradigms in Healthcare 6. Virtual Reality in Psychotherapy, Rehabilitation, and Assessment*. 2011, pp. 109–128.

[38] Sébastien Brault et al. “Detecting deceptive movement in 1 vs. 1 based on global body displacement of a rugby player”. In: *International Journal of Virtual Reality* 8.4 (2009), pp. 31–36.

[39] Brian Broll et al. “A visual programming environment for learning distributed programming”. In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 2017, pp. 81–86.

[40] John Brooke. “SUS–A quick and dirty usability scale”. In: *Usability Evaluation in Industry* 189.194 (1996), pp. 4–7.

[41] Margaret M Burnett and Christopher Scaffidi. *The encyclopedia of human-computer interaction*. The Interaction Design Foundation, 2014.

[42] Andres Bustillo et al. “A flexible platform for the creation of 3D semi-immersive environments to teach Cultural Heritage”. In: *Digital Applications in Archaeology and Cultural Heritage* 2.4 (2015), pp. 248–259.

[43] Davide Calandra et al. “Building reconfigurable passive haptic interfaces on demand using off-the-shelf construction bricks”. In: *Proceedings of the 26th Conference on Virtual Reality and 3D User Interfaces (VR)*. 2019, pp. 1403–1404.

[44] Murilo C Camargo, Rodolfo M Barros, and Vanessa T O Barros. “Visual design checklist for graphical user interface (GUI) evaluation”. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 2018, pp. 670–672.

[45] Laura Cancedda et al. “Mixed reality-based user interaction feedback for a hand-controlled interface targeted to robot teleoperation”. In: *Proceedings of the 4th International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. 2017, pp. 447–463.

[46] Alberto Cannavo et al. “T4T: Tangible interface for tuning 3D object manipulation tools”. In: *Proceedings of the 12th Symposium on 3D User Interfaces*. 2017, pp. 266–267.

[47] Alberto Cannavò and Fabrizio Lamberti. “A virtual character posing system based on reconfigurable tangible user interfaces and immersive virtual reality”. In: *Proceedings of the Conference on Smart Tools and Applications in Computer Graphics*. 2017, pp. 53–62.

[48] Alberto Cannavò and Fabrizio Lamberti. "User interaction feedback in a hand-controlled interface for robot team tele-operation using wearable augmented reality". In: *Proceedings of the Conference on Smart Tools and Applications in Computer Graphics*. 2017, pp. 53–62.

[49] Alberto Cannavò et al. "A movement analysis system based on immersive virtual reality and wearable technology for sport training". In: *Proceedings of the 4th International Conference on Virtual Reality*. 2018, pp. 26–31.

[50] Alberto Cannavò et al. "A participative system for tactics analysis in sport training based on immersive virtual reality". In: *Proceedings of the 4rd Workshop on Everyday Virtual Reality*. 2018, pp. 1–4.

[51] Alberto Cannavò et al. "Immersive Virtual Reality-based interfaces for character animation". In: *IEEE Access* 7.1 (2019), pp. 125463–125480.

[52] Brendan Cassidy et al. "A virtual reality platform for analyzing remote archaeological sites". In: *Interacting with Computers* 31.2 (2019), pp. 167–176.

[53] Jacky C P Chan et al. "A virtual reality dance training system using motion capture technology". In: *IEEE Transactions on Learning Technologies* 4.2 (2010), pp. 187–195.

[54] Angel Chang, Manolis Savva, and Christopher Manning. "Interactive learning of spatial knowledge for text to 3D scene generation". In: *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*. 2014, pp. 14–21.

[55] Angel Chang et al. "Text to 3d scene generation with rich lexical grounding". In: *arXiv preprint arXiv:1505.06289* (2015).

[56] Angel X Chang et al. "SceneSeer: 3D scene design with natural language". In: *arXiv preprint arXiv:1703.00050* (2017).

[57] Min-Wen Chao et al. "Human motion retrieval from hand-drawn sketch". In: *IEEE Transactions on Visualization and Computer Graphics* 18.5 (2011), pp. 729–740.

[58] Danqi Chen and Christopher D Manning. "A fast and accurate dependency parser using neural networks". In: *Proceedings of the conference on empirical methods in natural language processing*. 2014, pp. 740–750.

[59] Jiawen Chen, Shahram Izadi, and Andrew Fitzgibbon. "KinÊtre: animating the world with the human body". In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 2012, pp. 435–444.

[60] Lung-Pan Cheng et al. "iTurk: Turning passive haptics into active haptics by making users reconfigure props in Virtual Reality". In: *Proceedings of the ACM CHI*. 2018, 89:1–89:10.

[61] Byungkuk Choi et al. “SketchiMo: Sketch-based motion editing for articulated characters”. In: *ACM Transactions on Graphics* 35.4 (2016), 146:1–146:12.

[62] Chris Christou et al. “A versatile large-scale multimodal VR system for cultural heritage visualization”. In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. 2006, pp. 133–140.

[63] Philo Tan Chua et al. “Training for physical tasks in virtual environments: Tai Chi”. In: *Proceedings of the conference Virtual Reality*. 2003, pp. 87–94.

[64] Haili Chui and Anand Rangarajan. “A new point matching algorithm for non-rigid registration”. In: *Computer Vision and Image Understanding* 89.2-3 (2003), pp. 114–141.

[65] Loic Ciccone, Martin Guay, and Robert Sumner. “Flow Curves: An intuitive iterface for coherent scene deformation”. In: *Computer Graphics Forum* 35.7 (2016), pp. 247–256.

[66] Alexandra Covaci, Anne-Hélène Olivier, and Franck Multon. “Third person view and guidance for more natural motor behaviour in immersive basketball playing”. In: *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. 2014, pp. 55–64.

[67] Alexandra Covaci et al. “A virtual reality simulator for basketball free-throw skills development”. In: *Proceedings of the Doctoral Conference on Computing, Electrical and Industrial Systems*. 2012, pp. 105–112.

[68] Bob Coyne and Richard Sproat. “WordsEye: an automatic text-to-scene conversion system”. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 487–496.

[69] G Crippa, Valentina Rognoli, and Marinella Levi. “Materials and emotions, a study on the relations between materials and emotions in industrial products”. In: *Proceedings of the 8th International Conference on Design & Emotion: Out of control*. 2012, pp. 1–9.

[70] Tiziana D’Orazio et al. “Soccer player activity recognition by a multivariate features integration”. In: *Proceedings of the 7th International Conference on Advanced Video and Signal Based Surveillance*. 2010, pp. 32–39.

[71] Jose Danado and Fabio Paternò. “Puzzle: A visual-based environment for end user development in touch-based mobile phones”. In: *Proceedings of the International Conference on Human-Centred Software Engineering*. 2012, pp. 199–216.

[72] Nicholas Davis et al. “Creativity support for novice digital filmmaking”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2013, pp. 651–660.

[73] Francesco De Pace et al. “A comparison between two different approaches for a collaborative mixed-virtual environment in industrial maintenance”. In: *Frontiers in Robotics and AI* 6.1 (2019), p. 18.

[74] Michael F Deering. “HoloSketch: A virtual reality sketching/animation tool”. In: *ACM Transactions on Computer-Human Interaction* 2.3 (1995), pp. 220–238.

[75] Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. “Empowering end users to customize their smart environments: Model, composition paradigms, and domain-specific tools”. In: *ACM Transactions on Computer-Human Interaction* 24.2 (2017), pp. 1–52.

[76] Paul C DiLorenzo. “Premo: DreamWorks animation’s new approach to animation”. In: *IEEE Computer Graphics and Applications* 35.4 (2015), pp. 14–21.

[77] Lindsay J DiStefano et al. “Integrated injury prevention program improves balance and vertical jump height in children”. In: *The Journal of Strength & Conditioning Research* 24.2 (2010), pp. 332–342.

[78] Kai-Uwe Doer, Jens Schiefel, and W Kubbat. *Virtual cockpit simulation for pilot training*. Tech. rep. Darmstadt University Institute for flight mechanics and control, 2001.

[79] Mira Dontcheva, Gary Yngve, and Zoran Popović. “Layered acting for character animation”. In: *ACM SIGGRAPH 2003 Papers*. 2003, pp. 409–416.

[80] Daniel L Eaves et al. “The short-term effects of real-time virtual reality feedback on motor learning in dance”. In: *Presence: Teleoperators and Virtual Environments* 20.1 (2011), pp. 62–77.

[81] Manabu Eitsuka and Masahito Hirakawa. “Authoring animations of virtual objects in augmented reality-based 3d space”. In: *Proceedings of the 2nd IIAI International Conference on Advanced Applied Informatics*. 2013, pp. 256–261.

[82] Thomas O Ellis, John F Heafner, and WL Sibley. *The GRAIL language and operations*. Tech. rep. RAND Corp. Santa Monica, CA, 1969.

[83] Milan Erdelj and Enrico Natalizio. “UAV-assisted disaster management: Applications and open issues”. In: *Proceedings of the International Conference on Computing, Networking and Communications*. 2016, pp. 1–5.

[84] Chris Esposito, W Bradford Paley, and JueyChong Ong. “Of mice and monkeys: a specialized input device for virtual body animation”. In: *Proceedings of the symposium on Interactive 3D graphics*. 1995, 109–ff.

[85] Isaac Esteban, Judith Dijk, and Frans C.A. Groen. "From images to 3D models made easy". In: *Proceedings of the 19th ACM International Conference on Multimedia*. 2011, pp. 695–698.

[86] Anke Eyck et al. "Effect of a virtual coach on athletes' motivation". In: *Proceedings of the International Conference on Persuasive Technology*. 2006, pp. 158–161.

[87] Sarah Faisal, Paul Cairns, and Brock Craft. "Infovis experience enhancement through mediated interaction". In: (2005), pp. 1–7.

[88] Luca Fascione et al. "Manuka: A batch-shading architecture for spectral path tracing in movie production". In: *ACM Transactions on Graphics* 37.3 (2018), p. 31.

[89] Sidney Fels et al. "Swimming across the Pacific: A VR swimming interface". In: *IEEE Computer Graphics and Applications* 25.1 (2005), pp. 24–31.

[90] Andreas Fender, Jörg Müller, and David Lindlbauer. "Creature teacher: A performance-based animation system for creating cyclic movements". In: *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*. 2015, pp. 113–122.

[91] Andrew Feng et al. "Fast, automatic character animation pipelines". In: *Computer Animation and Virtual Worlds* 25.1 (2014), pp. 3–16.

[92] Tien-Chieng Feng et al. "Motion capture data retrieval using an artist's doll". In: *Proceedings of the 19th International Conference on Pattern Recognition*. 2008, pp. 1–4.

[93] Adso Fernández-Baena and David Miralles. "Avatar: Tangible interaction and augmented reality in character animation". In: *Proceedings of the Workshop: Interacting with Smart Objects*. 2014, p. 28.

[94] Philip W Fink, Patrick S Foo, and William H Warren. "Catching fly balls in virtual reality: A critical test of the outfielder problem". In: *Journal of vision* 9.13 (2009), pp. 14–14.

[95] Allan Fowler, Teale Fristce, and Matthew MacLauren. "Kodu game lab: A programming environment". In: *The Computer Games Journal* 1.1 (2012), pp. 17–28.

[96] Rita Francese, Michele Risi, and Genoveffa Tortora. "Iconic languages: Towards end-user programming of mobile applications". In: *Journal of Visual Languages & Computing* 38.1 (2017), pp. 1–8.

[97] Jared A Frank and Vikram Kapila. "Towards teleoperation-based interactive learning of robot kinematics using a mobile augmented reality interface on a tablet". In: *Proceedings of the Control Conference*. 2016, pp. 385–392.

[98] Nathan Gale, Pejman Mirza-Babaei, and Isabel Pedersen. "Heuristic guidelines for playful wearable augmented reality applications". In: *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. 2015, pp. 529–534.

[99] Valentina Gatteschi et al. "Semantics-based intelligent human-computer interaction". In: *IEEE Intelligent Systems* 31.4 (2015), pp. 11–21.

[100] Paul Gault, Judith Masthoff, and Graham Johnson. "DiCER: A distributed consumer experience research method for use in public spaces". In: *International Journal of Human-Computer Studies* 81.1 (2015), pp. 49–71.

[101] Giuseppe Ghiani, Fabio Paternò, and Lucio D Spano. "Cicero designer: An environment for end-user development of multi-device museum guides". In: *Proceedings of the International Symposium on End User Development*. 2009, pp. 265–274.

[102] Giuseppe Ghiani et al. "Personalization of context-dependent applications through trigger-action rules". In: *ACM Transactions on Computer-Human Interaction* 24.2 (2017), pp. 1–33.

[103] Jeff Gipson et al. "VR story production on Disney animation's" cycles"". In: *ACM SIGGRAPH 2018 Talks*. 2018, pp. 1–2.

[104] Oliver Glauser et al. "Rig animation with a tangible and modular input device". In: *ACM Transactions on Graphics* 35.4 (2016), pp. 1–11.

[105] Steven Gold et al. "New algorithms for 2D and 3D point matching: Pose estimation and correspondence". In: *Pattern Recognition* 31.8 (1998), pp. 1019–1031.

[106] Satoshi Gondo et al. "Soccer tactics analysis supporting system displaying the player's actions in virtual space". In: *Proceedings of the 18th International Conference on Computer Supported Cooperative Work in Design*. 2014, pp. 581–586.

[107] Philip Gorinski and Mirella Lapata. "Movie script summarization as graph-based scene extraction". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1066–1076.

[108] Alexandros Gouvatsos et al. "Posing 3D Models from Drawings". In: *Computer in Entertainment* 15.2 (2017), 2:1–2:14.

[109] Stefan Gradl et al. "Virtual and augmented reality in sports: an overview and acceptance study". In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. 2016, pp. 885–888.

[110] Walters Graham. "The Story of Waldo". In: *Course Notes: 3D Character Animation by Computer* 1.1 (1989), pp. 6–79.

[111] Michael Gregory and Dan Seddon. "Creating photoreal creatures that audiences can connect with". In: *ACM SIGGRAPH 2019 Talks*. 2019, pp. 1–2.

[112] Thomas von der Grün et al. "A real-time tracking system for football match and training analysis". In: *Microelectronic systems*. 2011, pp. 199–212.

[113] Martin Guay, Marie-Paule Cani, and Rémi Ronfard. "The line of action: An intuitive interface for expressive character posing". In: *ACM Transactions on Graphics* 32.6 (2013), pp. 1–8.

[114] Cheng Guo and Ehud Sharlin. "Exploring the use of tangible user interfaces for human-robot interaction: a comparative study". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2008, pp. 121–130.

[115] Fabian Hahn et al. "Sketch abstractions for character posing". In: *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. 2015, pp. 185–191.

[116] Miller C Handler. *Digital Storytelling. A Creator's Guide to Interactive Entertainment*. Burlington: Focal Press, 2004.

[117] Daniel Harley et al. "Tangible vr: Diegetic tangible objects for virtual reality narratives". In: *Proceedings of the Conference on Designing Interactive Systems*. 2017, pp. 1253–1263.

[118] Sandra G Hart and Lowell E Staveland. "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research". In: *Advances in Psychology*. 1988, pp. 139–183.

[119] Sunao Hashimoto et al. "Touchme: An augmented reality based remote robot manipulation". In: *Proceedings of the 21st International Conference on Artificial Reality and Telexistence*. 2011.

[120] Robert Held et al. "3D puppetry: a kinect-based interface for 3D animation". In: *Proceedings of the 25th Symposium on User Interface Software and Technology*. 2012, pp. 423–434.

[121] Mark Henne et al. "The making of toy story [computer animation]". In: *Proceedings of the 41st Computer Society International Conference: Technologies for the Information Superhighway*. 1996, pp. 463–468.

[122] Rorik Henrikson et al. "Multi-device storyboards for cinematic narratives in VR". In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 2016, pp. 787–796.

[123] Heiko Herrmann and Emiliano Pastorelli. “Virtual reality visualization for photogrammetric 3D reconstructions of cultural heritage”. In: *Proceedings of the International Conference on Augmented and Virtual Reality*. 2014, pp. 283–295.

[124] Jr Hill and S Francis. *Computer graphics using open gl*. Pearson Education, 2008.

[125] Ronan Hinchet et al. “DextrES: Wearable haptic feedback for grasping in VR via a thin form-factor electrostatic brake”. In: *Proceedings of the ACM UIST*. 2018, pp. 901–912.

[126] Thomas Hofmann and Joachim M Buhmann. “Pairwise data clustering by deterministic annealing”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.1 (1997), pp. 1–14.

[127] Johan F Hoorn, Elly A Konijn, and Gerrit C Van der Veer. “Virtual Reality: Do not augment realism, augment relevance”. In: *Upgrade-Human-Computer Interaction: Overcoming Barriers* 4.1 (2003), pp. 18–26.

[128] Enrique Hortal et al. “SVM–based brain–machine interface for controlling a robot arm through four mental tasks”. In: *Neurocomputing* 151.1 (2015), pp. 116–121.

[129] Jiawei Huang et al. “Visualizing natural environments from data in virtual reality: combining realism and uncertainty”. In: *Proceedings of the 26th Conference on Virtual Reality and 3D User Interfaces (VR)*. 2019, pp. 1485–1488.

[130] Yazhou Huang, Lloyd Churches, and Brendan Reilly. “A case study on virtual reality American football training”. In: *Proceedings of the Virtual Reality International Conference*. 2015, pp. 1–5.

[131] Yazhou Huang et al. “Modeling gaze behavior for virtual demonstrators”. In: *Proceedings of the International Workshop on Intelligent Virtual Agents*. 2011, pp. 155–161.

[132] Hugo C Huurdeman and Chiara Piccoli. “"More than just a Picture"–The importance of context in search user interfaces for three-dimensional content”. In: *Proceedings of the Conference on Human Information Interaction and Retrieval*. 2020, pp. 338–342.

[133] Nazrita Ibrahim and Nazlena Mohamad Ali. “A conceptual framework for designing virtual heritage environment for cultural learning”. In: *Journal on Computing and Cultural Heritage* 11.2 (2018), p. 11.

[134] Takeo Igarashi. “Computer graphics for all”. In: *Communications of the ACM* 53.7 (2010), pp. 71–77.

[135] Takeo Igarashi, Tomer Moscovich, and John F Hughes. "Spatial keyframing for performance-driven animation". In: *ACM SIGGRAPH 2006 Courses*. 2006, 17–es.

[136] Amy Ingram, Xiaoyu Wang, and William Ribarsky. "Towards the establishment of a framework for intuitive multi-touch interaction design". In: *Proceedings of the International Working Conference on Advanced Visual Interfaces*. 2012, pp. 66–73.

[137] Andri Ioannidou, Alexander Repenning, and David C Webb. "AgentCubes: Incremental 3D end-user development". In: *Journal of Visual Languages & Computing* 20.4 (2009), pp. 236–251.

[138] Satoru Ishigaki et al. "Performance-based control interface for character animation". In: *ACM Transactions on Graphics* 28.3 (2009), pp. 1–8.

[139] Hiroshi Ishii and Brygg Ullmer. "Tangible bits: towards seamless interfaces between people, bits and atoms". In: *Proceedings of the Conference on Human factors in computing systems*. 1997, pp. 234–241.

[140] ISO. *ISO 9241-210:2019(en) Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. Tech. rep. International Organization for Standardization, 2019.

[141] Bret Jackson and Daniel F Keefe. "Lift-off: Using reference imagery and freehand sketching to create 3d models in vr". In: *IEEE Transactions on Visualization and Computer Graphics* 22.4 (2016), pp. 1442–1451.

[142] Alec Jacobson et al. "Tangible and modular input device for character articulation". In: *ACM Transactions on Graphics* 33.4 (2014), pp. 1–12.

[143] Jens Jebens et al. "Evolving complexity management on the LEGO Batman movie". In: *Proceedings of the ACM SIGGRAPH 2017 Talks*. 2017, p. 5.

[144] Myounghoon Jeon et al. "From rituals to magic: Interactive art and HCI of the past, present, and future". In: *International Journal of Human-Computer Studies* 131.1 (2019), pp. 108–119.

[145] Kyuman Jeong and Han-Jin Cho. "Photo quality enhancement by relocating subjects". In: *Cluster Computing* 19.2 (2016), pp. 939–948.

[146] Hairong Jiang et al. "3D joystick for robotic arm control by individuals with high level spinal cord injuries". In: *Proceedings of the International Conference on Rehabilitation Robotics*. 2013, pp. 1–5.

[147] Wenchao Jiang and Zhaozheng Yin. "Human activity recognition using wearable sensors by deep convolutional neural networks". In: *Proceedings of the 23rd ACM international conference on Multimedia*. 2015, pp. 1307–1310.

[148] Dongsik Jo and Gerard J Kim. "IoT+ AR: pervasive and augmented environments for "Digi-log" shopping experience". In: *Human-centric Computing and Information Sciences* 9.1 (2019), p. 1.

[149] Michael P Johnson et al. "Sympathetic interfaces: using a plush toy to direct synthetic characters". In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 1999, pp. 152–158.

[150] Monica L H Jones et al. "The use of physical props in motion capture studies". In: *SAE International Journal of Passenger Cars-Mechanical Systems* 1.2008-01-1928 (2008), pp. 1163–1171.

[151] Beate Jost et al. "Graphical programming environments for educational robots: Open Roberta-yet another one?" In: *Proceedings of the International Symposium on Multimedia*. 2014, pp. 381–386.

[152] Ken Kahn. "Metaphor Design. Case Study of an Animated Programming Environment". In: *Proceedings of the 1995 Computer Game Developer Conference*. 1995.

[153] Roy S Kalawsky. "VRUSE–A computerised diagnostic tool: for usability evaluation of virtual/synthetic environment systems". In: *Applied Ergonomics* 30.1 (1999), pp. 11–25.

[154] Hirokazu Kato et al. "Virtual object manipulation on a table-top AR environment". In: *Proceedings of the IEEE and ACM International Symposium on Augmented Reality*. 2000, pp. 111–119.

[155] So Kato and Hiroyuki Tominaga. "A Practical Lesson of Introductory Programming Exercise with Lego Robot Control and Game Project". In: *Proceedings of the Association for the Advancement of Computing in Education*. 2009, pp. 1747–1752.

[156] Philip Kelly et al. "A virtual coaching environment for improving golf swing technique". In: *Proceedings of the 2010 ACM workshop on Surreal media and virtual cloning*. 2010, pp. 51–56.

[157] Ali Al-Khalifah et al. "Using virtual reality for medical diagnosis, training and education". In: *International Journal on Disability and Human Development* 5.2 (2006), pp. 187–194.

[158] Preety Khatri. "3D Animation: Maya or Blender". In: *Global Sci-Tech* 10.1 (2018), pp. 40–47.

[159] Huhn Kim. "Effective organization of design guidelines reflecting designer's design strategies". In: *International Journal of Industrial Ergonomics* 40.6 (2010), pp. 669–688.

[160] Min Kyu Kim et al. “Implementation of real-time motion and force capturing system for tele-manipulation based on semg signals and imu motion data”. In: *Proceedings of the International Conference on Robotics and Automation*. 2014, pp. 5658–5664.

[161] Felix Kistler and Elisabeth André. “User-defined body gestures for an interactive storytelling scenario”. In: *Proceedings of the IFIP Conference on Human-Computer Interaction*. 2013, pp. 264–281.

[162] Felix Kistler, Birgit Endrass, and Elisabeth André. “Full body interaction with virtual characters in an interactive storytelling scenario”. In: *Proceedings of the International Conference on Intelligent Virtual Agents*. 2014, pp. 236–239.

[163] Frank Klawonn. *Introduction to computer graphics: Using Java 2D and 3D*. Springer Science & Business Media, 2012.

[164] Brian Knep et al. “Dinosaur input device”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995, pp. 304–309.

[165] Mykhaylo Kostandov et al. “Interactive layered character animation in immersive virtual environments”. In: *ACM SIGGRAPH Poster Compendium* 22.3 (2006), pp. 409–416.

[166] Uros Krcadinac et al. “Textual affect communication and evocation using abstract generative visuals”. In: *IEEE Transactions on Human-Machine Systems* 46.3 (2015), pp. 370–379.

[167] Ernst Kruijff et al. “Designed emotions: Challenges and potential methodologies for improving multisensory cues to enhance user engagement in immersive systems”. In: *The Visual Computer* 33.4 (2017), pp. 471–488.

[168] Harold W Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97.

[169] Ranjitha Kumar et al. “Bricolage: example-based retargeting for web design”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2011, pp. 2197–2206.

[170] Mikko Kytö et al. “Improving 3D character posing with a gestural interface”. In: *IEEE Computer Graphics and Applications* 37.1 (2015), pp. 70–78.

[171] Fabrizio Lamberti, Alberto Cannavò, and Paolo Montuschi. “Is immersive virtual reality the ultimate interface for 3D animators?” In: *IEEE Computer Magazine* 53.4 (2020), pp. 36–45.

[172] Fabrizio Lamberti et al. “A Multimodal interface for virtual character animation based on live performance and natural language processing”. In: *International Journal of Human–Computer Interaction* 35.18 (2019), pp. 1655–1671.

[173] Fabrizio Lamberti et al. “Virtual character animation based on affordable motion capture and reconfigurable tangible interfaces”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.5 (2017), pp. 1742–1755.

[174] Caroline Larboulette and Sylvie Gibet. “I am a tree: Embodiment using physically based animation driven by expressive descriptors of motion”. In: *Proceedings of the 3rd International Symposium on Movement and Computing*. 2016, pp. 1–8.

[175] Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. “Interactive control for physically-based animation”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 201–208.

[176] Gun A Lee, Gerard J Kim, and Mark Billinghurst. “Immersive authoring: What you experience is what you get (wyxiwyg)”. In: *Communications of the ACM* 48.7 (2005), pp. 76–81.

[177] Woon-Sung Lee, Jung-Ha Kim, and Jun-Hee Cho. “A driving simulator as a virtual reality tool”. In: *Proceedings of the International Conference on Robotics and Automation*. 1998, pp. 71–76.

[178] Adrian Lees, Jos Vanrenterghem, and Dirk De Clercq. “Understanding how an arm swing enhances performance in the vertical jump”. In: *Journal of Biomechanics* 37.12 (2004), pp. 1929–1940.

[179] Luis Leite. “Virtual marionette”. In: *Proceedings of the international conference on Intelligent User Interfaces*. 2012, pp. 363–366.

[180] Luis Leite and Veronica Orvalho. “Anim-actor: understanding interaction with digital puppetry using low-cost motion capture”. In: *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*. 2011, pp. 1–2.

[181] Nicola Leonardi et al. “Trigger-action programming for personalising humanoid robot behaviour”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–13.

[182] Steven J Levine, Shawn Schaffert, and Neal Checka. “Natural user interface for robot task assignment”. In: *Proceedings of RSS workshop on Human-Robot Collaboration for Industrial Manufacturing*. 2014, pp. 1–5.

[183] Canlin Li et al. “Automatic 3D scene generation based on Maya”. In: *Proceedings of the 10th International Conference on Computer-Aided Industrial Design & Conceptual Design*. 2009, pp. 981–985.

[184] Changjian Li et al. “Robust flow-guided neural prediction for sketch-based freeform surface modeling”. In: *Proceedings of the SIGGRAPH Asia 2018*. 2018, p. 238.

[185] Juncong Lin et al. “A sketching interface for sitting pose design in the virtual environment”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.11 (2012), pp. 1979–1991.

[186] Morten Lind and Amund Skavhaug. “Using the Blender Game Engine for real-time emulation of production devices”. In: *International Journal of Production Research* 50.22 (2012), pp. 6219–6235.

[187] Cheng-Li Liu. “A study of detecting and combating cybersickness with fuzzy control for the elderly within 3D virtual stores”. In: *International Journal of Human-Computer Studies* 72.12 (2014), pp. 796–804.

[188] Hugo Liu and Push Singh. “ConceptNet–A practical commonsense reasoning tool-kit”. In: *BT Technology Journal* 22.4 (2004), pp. 211–226.

[189] Jingbo Liu et al. “Two-finger gestures for 6DOF manipulation of 3D objects”. In: *Proceedings of the Computer Graphics Forum*. 2012, pp. 2047–2055.

[190] Karen C Liu and Victor B Zordan. “Natural user interface for physics-based character animation”. In: *Proceedings of the International Conference on Motion in Games*. 2011, pp. 1–14.

[191] Zhi-Qiang Liu and Ka-Ming Leung. “Script Visualization (ScriptVis): a smart system that makes writing fun”. In: *Proceedings of the International Conference on Machine Learning and Cybernetics*. 2003, pp. 2990–2995.

[192] Li Liwei. “Applications of computer virtual reality technology in modern sports”. In: *Proceedings of the 4th International Symposium on Information Science and Engineering*. 2012, pp. 358–361.

[193] David Lizcano et al. “Implementation of end-user development success factors in mashup development environments”. In: *Computer Standards & Interfaces* 47.1 (2016), pp. 1–18.

[194] Benjamin Lok et al. “Effects of handling real objects and self-avatar fidelity on cognitive task performance and sense of presence in virtual environments”. In: *Presence: Teleoperators & Virtual Environments* 12.6 (2003), pp. 615–628.

[195] Lian Loke and Toni Robertson. “Design representations of moving bodies for interactive, motion-sensing spaces”. In: *International Journal of Human-Computer Studies* 67.4 (2009), pp. 394–410.

[196] Percy W Lovon-Ramos et al. “Mixed reality applied to the teleoperation of a 7-DOF manipulator in rescue missions”. In: *Proceedings of the Robotics Symposium and 4th Brazilian Robotics Symposium*. 2016, pp. 299–304.

[197] Jiajie Lu et al. “A new framework for automatic 3D scene construction from text description”. In: *Proceedings of International Conference on Progress in Informatics and Computing*. 2010, pp. 964–968.

[198] Jason Lucas and Walid Thabet. "Implementation and evaluation of a VR task-based training tool for conveyor belt safety training". In: *Journal of Information Technology in Construction* 13.1 (2008), pp. 637–659.

[199] Barry Lunt et al. *Curriculum guidelines for undergraduate degree programs in information technology*. Tech. rep. Information Technology, 2008.

[200] JT Luo et al. "On-screen characters: Their design and influence on consumer trust". In: *Journal of Services Marketing* 20.2 (2006), pp. 112–124.

[201] Monica G Maceli. "Tools of the trade: A survey of technologies in end-user development literature". In: *Proceedings of the International Symposium on End User Development*. 2017, pp. 49–65.

[202] Matthew B MacLaurin. "The design of Kodu: A tiny visual programming language for children on the Xbox 360". In: *Proceedings of the ACM Sigplan Notices*. 2011, pp. 241–246.

[203] John Maloney et al. "The Scratch programming language and environment". In: *ACM Transactions on Computing Education* 10.4 (2010), p. 16.

[204] Christopher D Manning et al. "The Stanford CoreNLP natural language processing toolkit". In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.

[205] Fabrizia Mantovani et al. "Virtual reality training for health-care professionals". In: *CyberPsychology & Behavior* 6.4 (2003), pp. 389–395.

[206] Chen Mao, Sheng Feng Qin, and David Wright. "Sketch-Based Virtual Human Modelling and Animation". In: *Proceedings of the 8th International Symposium on Smart Graphics*. 2007, pp. 220–223.

[207] Michael R Marner, Ross T Smith, and Bruce H Thomas. "Mapping 2D input to 3D immersive spatial augmented reality". In: *Proceedings of the Symposium on 3D User Interfaces*. 2015, pp. 171–172.

[208] Stacy Marsella et al. "Virtual character performance from speech". In: *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2013, pp. 25–35.

[209] Néstor Andrés Arteaga Martin et al. "Passive haptic feedback for manual assembly simulation". In: *Procedia CIRP* 7.1 (2013), pp. 509–514.

[210] Thomas H Massie and Kenneth J Salisbury. "The Phantom haptic interface: A device for probing virtual objects". In: *Proc. ASME Haptic Int. for Virtual Env. and Teleop. Systems Symp.* 1994, pp. 295–300.

[211] Carlos Mateo et al. "Hammer: An Android based application for end-user industrial robot programming". In: *Proceedings of the International Conference on Mechatronic and Embedded Systems and Applications*. 2014, pp. 1–6.

[212] Fiona McDermott, Laura Maye, and Gabriela Avram. "Co-designing a collaborative platform with cultural heritage professionals". In: *Proceedings of the 8th Irish Human–Computer Interaction Conference*. 2014, pp. 18–24.

[213] Armand McQueen, Jenna Wiens, and John Guttag. "Automatically recognizing on-ball screens". In: *Proceedings of the MIT Sloan Sports Analytics Conference*. 2014.

[214] Alvise Memo and Pietro Zanuttigh. "Head-mounted gesture controlled interface for human-computer interaction". In: *Multimedia Tools and Applications* 77.1 (2018), pp. 27–53.

[215] Zeno Menestrina and Antonella De Angeli. "End-User Development for serious games". In: *New Perspectives in End-User Development*. 2017, pp. 359–383.

[216] Ronald A Metoyer, Lanyue Xu, and Madhusudhanan Srinivasan. "A Tangible Interface for High-Level Direction of Multiple Animated Characters". In: *Proceedings of the annual international conference Graphics Interface*. 2003, pp. 167–176.

[217] Prabhaker Mishra et al. "Descriptive statistics and normality tests for statistical data". In: *Annals of cardiac anaesthesia* 22.1 (2019), p. 67.

[218] Takashi Miyazaki et al. "Teaching materials using AR and VR for learning the usage of oscilloscope". In: *Proceedings of the International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. 2017, pp. 43–52.

[219] Seong-Eun Moon et al. "Assessing product design using photos and real products". In: *Proceedings of the CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2017, pp. 1100–1107.

[220] Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. "3DTouch and HOMER-S: intuitive manipulation techniques for one-handed handheld augmented reality". In: *Proceedings of the Virtual Reality International Conference: Laval Virtual*. 2013, pp. 1–10.

[221] Larissa Munishkina, Jennifer Parrish, and Marilyn A Walker. "Fully-automatic interactive story design from film scripts". In: *Proceedings of the International Conference on Interactive Digital Storytelling*. 2013, pp. 229–232.

[222] Hideyuki Nakanishi. "FreeWalk: A social interaction platform for group behaviour in a virtual space". In: *International Journal of Human-Computer Studies* 60.4 (2004), pp. 421–454.

[223] Asma Naz et al. "Emotional qualities of VR space". In: *Proceedings of the IEEE Virtual Reality*. 2017, pp. 3–11.

[224] Pedro Neto, J Norberto Pires, and A Paulo Moreira. "Accelerometer-based control of an industrial robotic arm". In: *Proceedings of the International Symposium on Robot and Human Interactive Communication*. 2009, pp. 1192–1197.

[225] Ta H D Nguyen et al. "Real-time 3D human capture system for mixed-reality art and entertainment". In: *IEEE Transactions on Visualization and Computer Graphics* 11.6 (2005), pp. 706–721.

[226] Diederick C Niehorster, Li Li, and Markus Lappe. "The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research". In: *i-Perception* 8.3 (2017), pp. 1–23.

[227] Jakob Nielsen. "10 usability heuristics for user interface design". In: *Nielsen Norman Group* (1995).

[228] Jakob Nielsen and Rolf Molich. "Heuristic evaluation of user interfaces". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1990, pp. 249–256.

[229] Daisuke Ninomiya, Kohji Miyazaki, and Ryohei Nakatsu. "Networked virtual marionette theater". In: *Proceedings of the International Conference on Technologies for E-Learning and Digital Entertainment*. 2008, pp. 619–627.

[230] Lasguido Nio et al. "Conversation dialog corpora from television and movie scripts". In: *Proceedings of the 17th Oriental Chapter of the International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques*. 2014, pp. 1–4.

[231] Naoki Numaguchi et al. "A puppet interface for retrieval of motion capture data". In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2011, pp. 157–166.

[232] Bruno Oliveira, Diego Azulay, and Paulo Carvalho. "GVRf and Blender: A Path for Android Apps and Games Development". In: *Proceedings of International Conference on Human-Computer Interaction*. 2019, pp. 329–337.

[233] Eric M Orendt, Myriel Fichtner, and Dominik Henrich. "Robot programming by non-experts: intuitiveness and robustness of one-shot robot programming". In: *Proceedings of the 25th IEEE International Symposium on Robot and Human Interactive Communication*. 2016, pp. 192–199.

[234] Noritaka Osawa and Kikuo Asai. "An immersive path editor for keyframe animation using hand direct manipulation and 3d gearbox widgets". In: *Proceedings of the 7th International Conference on Information Visualization*. 2003, pp. 524–529.

[235] Masaki Oshita. "Generating animation from natural language texts and semantic analysis for motion search and scheduling". In: *The Visual Computer* 26.5 (2010), pp. 339–352.

[236] Masaki Oshita. "Multi-touch interface for character motion control using model-based approach". In: *Proceedings of the International Conference on Cyberworlds*. 2013, pp. 330–337.

[237] Masaki Oshita, Yuta Senju, and Syun Morishige. "Character motion control interface with hand manipulation inspired by puppet mechanism". In: *Proceedings of the 12th International Conference on Virtual-Reality Continuum and Its Applications in Industry*. 2013, pp. 131–138.

[238] Ibrahim Ouahbi et al. "Learning basic programming concepts by creating games with Scratch programming environment". In: *Procedia Social and Behavioral Sciences* 191.1 (2015), pp. 1479–1482.

[239] Christos Ouzounis, Alex Kilias, and Christos Mousas. "Kernel projection of latent structures regression for facial animation retargeting". In: *arXiv preprint arXiv:1707.09629* (2017).

[240] Federica Pallavicini, Alessandro Pepe, and Maria E Minissi. "Gaming in Virtual Reality: What changes in terms of usability, emotional response and sense of presence compared to non-immersive video games?" In: *Simulation & Gaming* 50.2 (2019), pp. 136–159.

[241] JunJun Pan et al. "Automatic rigging for animation characters with 3D silhouette". In: *Computer Animation and Virtual Worlds* 20.2-3 (2009), pp. 121–131.

[242] Vinoth P S Pandian, Sarah Suleri, and Matthias Jarke. "Blu: What GUIs are made of". In: *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion*. 2020, pp. 81–82.

[243] Natapon Pantuwong. "A tangible interface for 3D character animation using augmented reality technology". In: *Proceedings of the 8th International Conference on Information Technology and Electrical Engineering*. 2016, pp. 1–6.

[244] Fabio Paternò. "End user development: Survey of an emerging field for empowering people". In: *ISRN Software Engineering* 2013.1 (2013), pp. 1–12.

[245] Randy Pausch et al. "Disney's Aladdin: first steps toward storytelling in virtual reality". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, pp. 193–203.

[246] Bryson R Payne, James F Lay, and Markus A Hitz. "Automatic 3D object reconstruction from a single image". In: *Proceedings of the ACM Southeast Regional Conference*. 2014, p. 31.

[247] Lorenzo Peppoloni et al. "Immersive ros-integrated framework for robot teleoperation". In: *Proceedings of the Symposium User Interfaces*. 2015, pp. 177–178.

[248] Adeline Pihuit, Paul G Kry, and Marie-Paule Cani. "Hands on virtual clay". In: *Proceedings of the International Conference on Shape Modeling and Applications*. 2008, pp. 267–268.

[249] Ana M[a] Pinto-Llorente et al. "Building, coding and programming 3D models via a visual programming environment". In: *Quality & Quantity* 52.6 (2018), pp. 2455–2468.

[250] Thammathip Piumsomboon et al. "Grasp-Shell vs gesture-speech: A comparison of direct and indirect natural interaction techniques in augmented reality". In: *Proceedings of the International Symposium on Mixed and Augmented Reality*. 2014, pp. 73–82.

[251] Jarkko Polvi et al. "Handheld guides in inspection tasks: Augmented reality versus picture". In: *IEEE Transactions on Visualization and Computer Graphics* 24.7 (2018), pp. 2118–2128.

[252] Aristidis Protopsaltis et al. "Scenario-based serious games repurposing". In: *Proceedings of the 29th ACM international conference on Design of communication*. 2011, pp. 37–44.

[253] Panagiotis Psomos and Maria Kordaki. "Analysis of educational digital storytelling environments: the use of the "dimension star" model". In: *Proceedings of the World Summit on Knowledge Society*. 2011, pp. 317–322.

[254] Hayes Solos Raffle, Amanda J Parkes, and Hiroshi Ishii. "Topobo: a constructive assembly system with kinetic memory". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2004, pp. 647–654.

[255] Hafizur Rahaman, Erik Champion, and Mafkereseb Bekele. "From photo to 3D to mixed reality: A complete workflow for cultural heritage visualisation and experience". In: *Digital Applications in Archaeology and Cultural Heritage* 13.1 (2019), e00102.

[256] Jun Rekimoto and Katashi Nagao. "The world through the computer: Computer augmented interaction with real world environments". In: *Proceedings of the 8th annual ACM symposium on User interface and software technology*. 1995, pp. 29–36.

[257] Helge Rhodin et al. "Interactive motion mapping for real-time character control". In: *Proceedings of the Computer Graphics Forum*. 2014, pp. 273–282.

[258] Keven Richly et al. "Recognizing compound events in spatio-temporal football data". In: *Proceedings of the International Conference on Internet of Things and Big Data*. 2016, pp. 27–35.

[259] Barbara Robertson. "Motion capture meets 3D animation". In: *On the Cutting Edge of Technology* 1.1 (1993), pp. 1–14.

[260] Francisco Javier Rodriguez Lera et al. "Augmented reality to improve teleoperation of mobile robots". In: *Proceedings of the Workshop of Physical Agents*. 2011, pp. 1–6.

[261] David F Rogers. *Computer graphics in engineering education*. Elsevier, 2016.

[262] Irene Rubino et al. "Integrating a location-based mobile game in the museum visit: Evaluating visitors' behaviour and learning". In: *Journal on Computing and Cultural Heritage* 8.3 (2015), p. 15.

[263] José-Manuel Sáez-López, Marcos Román-González, and Esteban Vázquez-Cano. "Visual programming languages integrated across the curriculum in elementary school: A two year case study using Scratch in five schools". In: *Computers & Education* 97.1 (2016), pp. 129–141.

[264] Francesca Samsel, Sebastian Klaassen, and David H Rogers. "Colormoves: Real-time interactive colormap construction for scientific visualization". In: *IEEE Computer Graphics and Applications* 38.1 (2018), pp. 20–29.

[265] Andrea Sanna et al. "Automatically mapping human skeletons onto virtual character armatures". In: *Proceedings of the International Conference on Intelligent Technologies for Interactive Entertainment*. 2013, pp. 80–89.

[266] Andrea Sanna et al. "Developing touch-less interfaces to interact with 3D contents in public exhibitions". In: *Proceedings of the International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. 2016, pp. 293–303.

[267] Marc E C Santos et al. "Toward standard usability questionnaires for handheld augmented reality". In: *IEEE Computer Graphics and Applications* 35.5 (2015), pp. 66–75.

[268] Markus Santoso. "Markerless augmented reality technology for real-space basketball simulation". In: *Proceedings of the International Conference on Consumer Electronics*. 2018, pp. 1–3.

[269] Markus Sauer, Florian Zeiger, and Klaus Schilling. "Mixed-reality user interface for mobile robot teleoperation in ad-hoc networks". In: *IFAC Proceedings Volumes* 43.23 (2010), pp. 77–82.

[270] Jeff Sauro and James R. Lewis. *Quantifying the User Experience*. Morgan Kaufmann, 2012.

[271] Satu E Schaeffer. "Usability evaluation for augmented reality". In: *Computer Science* 100.1 (2014).

[272] L Schafer. "Models for digital storytelling and interactive narratives". In: *Proceedings of the 4th international conference on Computational Semiotics for Games and New Media*. 2004, pp. 148–155.

[273] Dieter Schmalstieg and Andre Stork. "Unified patterns for realtime Interactive Simulation in Games and Digital Storytelling". In: *IEEE Computer Graphics and Applications* 39.1 (2019), pp. 100–106.

[274] Sebastian Schuster and Christopher D Manning. "Enhanced english universal dependencies: an improved representation for natural language understanding tasks". In: *Proceedings of the 10th International Conference on Language Resources and Evaluation*. 2016, pp. 2371–2378.

[275] Mikael Seidler. "Blender Eevee render engine in indie production: using Blender's Eevee render engine for art projects". In: *Media and Art* 1.1 (2018), pp. 1–43.

[276] Yeongho Seol, Carol O'Sullivan, and Jehee Lee. "Creature features: online motion puppetry for non-human characters". In: *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2013, pp. 213–221.

[277] Lee M Seversky and Lijun Yin. "Real-time automatic 3D scene generation from natural language voice and text descriptions". In: *Proceedings of the 14th ACM International Conference on Multimedia*. 2006, pp. 61–64.

[278] Hyunju Shim and Bo Gyeong Kang. "CAMEO-camera, audio and motion with emotion orchestration for immersive cinematography". In: *Proceedings of the International Conference on Advances in Computer Entertainment Technology*. 2008, pp. 115–118.

[279] Noriyoshi Shimizu et al. "A teddy-bear-based robotic user interface". In: *Computers in Entertainment* 4.3 (2006), 8–es.

[280] Hyun J Shin et al. "Computer puppetry: An importance-based approach". In: *ACM Transactions on Graphics* 20.2 (2001), pp. 67–94.

[281] Takaaki Shiratori et al. "Expressing animated performances through puppeteering". In: *Proceedings of the Symposium on 3D User Interfaces*. IEEE. 2013, pp. 59–66.

[282] Alexandre Carvalho Silva et al. "A strategy to present 2D information within a virtual reality application". In: *Proceedings of the 16th Symposium on Virtual and Augmented Reality*. IEEE. 2014, pp. 143–147.

[283] Richard Sinkhorn. "A relationship between arbitrary positive matrices and doubly stochastic matrices". In: *The Annals of Mathematical Statistics* 35.2 (1964), pp. 876–879.

[284] Marjorie Skubic et al. "Using a hand-drawn sketch to control a team of robots". In: *Autonomous Robots* 22.4 (2007), pp. 399–410.

[285] Meehae Song et al. "Digital heritage application as an edutainment tool". In: *Proceedings of the ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry*. 2004, pp. 163–167.

[286] Ruth M Sorrentino et al. "Virtual visualization: Preparation for the olympic games long-track speed skating". In: *International Journal of Computer Science in Sport* 4.1 (2005), p. 40.

[287] Michael Spahn and Volker Wulf. "End-user development of enterprise widgets". In: *Proceedings of the International Symposium on End User Development*. 2009, pp. 106–125.

[288] Anne Morgan Spalter. *The computer in the visual arts*. Addison-Wesley Longman Publishing Co., 1998.

[289] Misha Sra et al. "Auris: creating affective virtual spaces from music". In: *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. 2017, p. 26.

[290] Misha Sra et al. "Procedurally generated virtual reality from 3D reconstructed physical space". In: *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*. 2016, pp. 191–200.

[291] Sonia H Stephens. "A narrative approach to interactive information visualization in the digital humanities classroom". In: *Arts and Humanities in Higher Education* 18.4 (2019), pp. 416–429.

[292] Kathryn T Stolee and Teale Fristoe. "Expressing computer science concepts through Kodu game lab". In: *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. 2011, pp. 99–104.

[293] Andrew Stratton, Chris Bates, and Andy Dearden. "Quando: Enabling museum and art gallery practitioners to develop interactive digital exhibits". In: *Proceedings of the International Symposium on End User Development*. 2017, pp. 100–107.

[294] David J Sturman. "A brief history of motion capture for computer character animation". In: *Proceedings of the 21th Annual Conference on Computer Graphics and Interactive Techniques*. 1994, pp. 11–17.

[295] Evan A Suma, Sabarish Babu, and Larry F Hodges. "Comparison of travel techniques in a complex, multi-level 3D environment". In: *Proceedings of the IEEE Symposium on 3D User Interfaces*. 2007.

[296] Ivan E Sutherland. "Sketchpad - A man-machine graphical communication system". In: *Simulation* 2.5 (1964), R–3.

[297] Tsuyoshi Suzuki et al. "Cooperative formation among multiple mobile robot teleoperation in inspection task". In: *Proceedings of the 39th IEEE Conference on Decision and Control*. 2000, pp. 358–363.

[298] Davin Karl Swanson. "Implementation of arbitrary path constraints using dissipative passive haptic displays". PhD thesis. Georgia Institute of Technology, 2003.

[299] Scott Swearingen and Kyoung Lee Swearingen. "Creating virtual environments with 3D printing and photogrammetry". In: *SIGGRAPH ASIA 2016 Creating Virtual Environments*. 2016, pp. 1–4.

[300] Kenshi Takayama et al. "Sketch-based generation and editing of quad meshes". In: *ACM Transactions on Graphics* 32.4 (2013), pp. 1–8.

[301] Kosit Teachabarikiti, Thanarat H Chalidabhongse, and Arit Thammano. "Players tracking and ball detection for an automatic tennis video annotation". In: *Proceedings of the 11th International Conference on Control Automation Robotics & Vision*. 2010, pp. 2461–2494.

[302] Daniel Tetteroo et al. "End-user development in the Internet of Things era". In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 2015, pp. 2405–2408.

[303] Michael Thees et al. "Effects of augmented reality on learning and cognitive load in university physics laboratory courses". In: *Computers in Human Behavior* 108.1 (2020), p. 106316.

[304] Andres Ubeda, Eduardo Ianez, and Jose M Azorin. "An integrated electrooculography and desktop input bimodal interface to support robotic arm control". In: *IEEE Transactions on Human-Machine Systems* 43.3 (2013), pp. 338–342.

[305] Brygg Ullmer and Hiroshi Ishii. "Emerging frameworks for tangible user interfaces". In: *IBM systems journal* 39.3.4 (2000), pp. 915–931.

[306] Alvaro Uribe-Quevedo, Hernando Leon, and Byron Perez-Gutierrez. "Arm-like mechanism user interface for 3D animation". In: *Proceedings of the 13th International Conference on Control, Automation and Systems*. 2013, pp. 1463–1467.

[307] Pedro Valderas, Vicente Pelechano, and Oscar Pastor. "Towards an end-user development approach for web engineering methods". In: *Proceedings of the International Conference on Advanced Information Systems Engineering*. 2006, pp. 528–543.

[308] Etienne Van Wyk and Ruth De Villiers. "Virtual reality training applications for the mining industry". In: *Proceedings of the 6th international conference on computer graphics, virtual reality, visualisation and interaction in Africa*. 2009, pp. 53–63.

[309] Luisa Varriale and Domenico Tafuri. "Technology for soccer sport: The human side in the technical part". In: *Proceedings of the International Conference on Exploring Services Science*. 2016, pp. 263–276.

[310] Matthew Ventura et al. “Development of a video game that teaches the fundamentals of computer programming”. In: *Proceedings of the Annual IEEE Region 3 Technical, Professional, and Student Conference*. 2015, pp. 1–5.

[311] Chi T Vi et al. “Not just seeing, but also feeling art: Mid-air haptic experiences integrated in a multisensory art exhibition”. In: *International Journal of Human-Computer Studies* 108.1 (2017), pp. 1–14.

[312] Daniel Vogel, Paul Lubos, and Frank Steinicke. “AnimationVR-Interactive Controller-based Animating in Virtual Reality”. In: *Proceedings of the 1st Workshop on Animation in Virtual and Augmented Environments*. 2018, pp. 1–6.

[313] Georgios Vouzounaras, Petros Daras, and Michael G Strintzis. “Automatic generation of 3D outdoor and indoor building scenes from a single image”. In: *Multimedia Tools and Applications* 70.1 (2014), pp. 361–378.

[314] Shawn M Waldon et al. “SketchBio: A scientist’s 3D interface for molecular modeling and animation”. In: *BMC Bioinformatics* 15.1 (2014), p. 334.

[315] David Waller, Earl Hunt, and David Knapp. “The transfer of spatial knowledge in virtual environment training”. In: *Presence* 7.2 (1998), pp. 129–143.

[316] Ryoichi Watanabe et al. “The soul of ActiveCube: implementing a flexible, multimodal, three-dimensional spatial tangible interface”. In: *Computers in Entertainment* 2.4 (2004), pp. 15–15.

[317] David Weintrop et al. “Evaluating CoBlox: A comparative study of robotics programming environments for adult novices”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–12.

[318] Michael P Weller, Ellen Yi-Luen Do, and Mark D Gross. “Posey: instrumenting a poseable hub and strut construction toy”. In: *Proceedings of the 2nd international conference on Tangible and embedded interaction*. 2008, pp. 39–46.

[319] Ting-Sheng Weng, Meng-Hui Hsu, and Der-Ching Yang. “A study investigating the use of 3D computer animations of trigonometric functions to enhance spatial perception ability”. In: *International Journal of Information and Education Technology* 7.1 (2017), p. 23.

[320] Joe Willage. *Using VR to improve free throw percentage in the NBA*. 2017.

[321] Rafal Wojciechowski et al. “Building virtual and augmented reality museum exhibitions”. In: *Proceedings of the 9th International Conference on 3D Web technology*. 2004, pp. 135–144.

[322] Yirui Wu, Tong Lu, and Jiqiang Song. “A real-time mesh animation framework using kinect”. In: *Proceedings of the Pacific-Rim Conference on Multimedia*. 2013, pp. 245–256.

[323] Ken Xu, James Stewart, and Eugene Fiume. "Constraint-based automatic placement for scene composition". In: *Proceedings of Graphics Interface*. 2002, pp. 25–34.

[324] Katsu Yamane, Yuka Ariki, and Jessica Hodgins. "Animating non-humanoid characters with human motion data". In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2010, pp. 169–178.

[325] Ungyeon Yang. "Just follow me: a VR-based motion training system". In: *Proceedings of the Conference SIGGRAPH*. 2001.

[326] YuanRui Yang et al. "Real-time human-robot interaction in complex environment using kinect v2 image recognition". In: *Proceedings of the 7th International Conference on Cybernetics and Intelligent Systems and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. 2015, pp. 112–117.

[327] Patrick Ye and Timothy Baldwin. "Towards automatic animated storyboarding". In: *Proceedings of the 23rd AAAI conference on Artificial Intelligence*. 2008, pp. 578–583.

[328] AWW Yew, SK Ong, and AYC Nee. "Immersive Augmented Reality Environment for the Teleoperation of Maintenance Robots". In: *Procedia CIRP* 61.1 (2017), pp. 305–310.

[329] So-Yeon Yoon, Yun Jung Choi, and Hyunjoo Oh. "User attributes in processing 3D VR-enabled showroom: Gender, visual cognitive styles, and the sense of presence". In: *International Journal of Human-Computer Studies* 82.1 (2015), pp. 1–10.

[330] Wataru Yoshizaki et al. "An actuated physical puppet as an input device for controlling a digital manikin". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2011, pp. 637–646.

[331] Alan L Yuille and JJ Kosowsky. "Statistical physics algorithms that converge". In: *Neural computation* 6.3 (1994), pp. 341–356.

[332] Laura A Zager and George C Verghese. "Graph similarity scoring and matching". In: *Applied Mathematics Letters* 21.1 (2008), pp. 86–94.

[333] Andre Zenner and Antonio Krüger. "Shifty: A weight-shifting dynamic passive haptic proxy to enhance object perception in Virtual Reality". In: *IEEE Transactions on Visualization and Computer Graphics* 23.4 (2017), pp. 1285–1294.

[334] Yue Zhan and Michael Hsiao. "A Natural Language Programming Application for Lego Mindstorms EV3". In: *Proceedings of the International Conference on Artificial Intelligence and Virtual Reality*. 2018, pp. 191–192.

[335] Lin Zhang and Ling Wang. "VR-based basketball movement simulation". In: *Transactions on Edutainment* 5.1 (2011), pp. 240–250.

[336] Maliang Zheng and Daniel Kudenko. "Automated event recognition for football commentary generation". In: *International Journal of Gaming and Computer-Mediated Simulations* 2.4 (2010), pp. 67–84.

[337] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. "Movie review mining and summarization". In: *Proceedings of the 15th ACM international conference on Information and knowledge management*. 2006, pp. 43–50.

[338] Michael Zyda. "From visual simulation to virtual reality to games". In: *Computer* 38.9 (2005), pp. 25–32.

[339] RR Zylstra. "Normality Tests for small sample sizes". In: *Quality Engineering* 7.1 (1994), pp. 45–58.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.