A Unified Model for Interaction in 3D Environment

Julien Casarin, Dominique Bechmann, Marilyn Keller

A Unified Model for Interaction in 3D Environment

Julien Casarin
Strasbourg university
Gfi Informatique
Strasbourg, France
julien.casarin@gfi.fr

Dominique Bechmann
Strasbourg university
Strasbourg, France
bechmann@unistra.fr

Marilyn Keller
Gfi Informatique
Saint-Ouen, France
marilyn.keller@gfi.fr

Figure 1: We propose a new model for designing VR, AR and MR applications independently of any device. Our model allows the user to visualize a virtual world and to interact with it on any kind of device. These pictures shows a user interacting with the same virtual object on a desktop, on a touchscreen and on an Oculus Rift HMD.

ABSTRACT

The Virtual (VR), Augmented (AR) and Mixed Reality (MR) devices are currently evolving at a very fast pace. This rapid evolution affects significantly the maintainability and portability of the applications. In this paper, we present a model for designing VR, AR and MR applications independently of any device. To do this, we use degrees of freedom to define an abstraction layer between the tasks to be performed and the interaction device.

CCS CONCEPTS

• **Human-centered computing** → **Interaction design**; • **Computing methodologies** → *Graphics input devices*; Mixed / augmented reality; Virtual reality;

KEYWORDS

Model-Based Interactive System Development, Virtual Worlds, Input Techniques

ACM Reference format:
Julien Casarin, Dominique Bechmann, and Marilyn Keller. 2017. A Unified Model for Interaction in 3D Environment. In *Proceedings of VRST '17, Gothenburg, Sweden, November 8–10, 2017,* 7 pages.
https://doi.org/10.1145/3139131.3139140

VRST '17, November 8–10, 2017, Gothenburg, Sweden

ACM ISBN 978-1-4503-5548-3/17/11…$15.00
https://doi.org/10.1145/3139131.3139140

1 INTRODUCTION

The fast evolution of Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR) devices reduces significantly the lifetime of the applications running on those devices. On the other hand, there are major differences between devices (dimensions of the display, number of degrees of freedom of the interaction device). As a result, the portability of one application from one device to another often takes a long time and can be quite complicated. The existing approach consists in using game engines (Unity, Unreal Engine, ...) that enable the development of a single application for several types of devices. The combination of this approach with tools such as Middle VR [6] makes it possible to simplify the application development of VR, AR and MR by providing high-level tools. However, since the interaction has to be adapted to the specific features of each type of device, it is necessary to rework the application so that it supports a new technology. Thus, the appearance on the market of a new device leads to the need to update all the applications. We propose a new approach to increase significantly the portability of VR, AR and MR applications. This consists in considering devices as tools able to interact with any 3D environment. This approach requires the creation of a standard allowing the description of any environment for any device. If standards exist to describe objects of a 3D scene (OBJ, DAE, VRML, ...), there is no unified model so far, that allows to describe interactions of one or several users.

To create this model, we rely on the use of degrees of freedom to define an abstraction layer between the tasks to perform and the device. Manipulations, application control and selection are defined using a reduced set of interaction blocks. Those blocks are transmitted via a network to a device. Then the device interprets them and activates them according to the actions of the user.

After presenting the various interaction blocks that we used (section 3.3), we present our model implementation (section 4) , then we test it on two 3D environments samples with the three devices

shown Figure 1 (section 5). Finally, we show the contribution of this method in terms of portability of the applications and compare it to the existing applications (section 6).

2 STATE OF THE ART

2.1 3D Environments

In the remainder of this article, we call 3D environments any virtual, augmented and mixed environment that includes virtual objects positioned in a three-dimensional space. A **virtual environment** is a fully computer-generated three-dimensional environment [2]. Augmented Reality allows virtual environments to be superimposed on the real world. We call **augmented environment** an area of real space on which a virtual environment is superimposed. In Augmented Reality applications, the interaction level between real and virtual objects is highly variable. It can range from graphical positioning of virtual objects on a real image [1, 9] to a realistic inclusion of virtual objects in the real world [5].

In [7], Milgram defines Mixed Reality as the continuum between reality and virtuality (Figure 2). Any environment including real elements and virtual elements can therefore be considered as a **mixed environment**.

In practice, environments are referred to as Mixed Reality environments when real objects and virtual objects interact (almost) as freely as if they would belong to the same physical space (subject to physical constraints such as gravity and collisions, occlusion, Bidirectional Data exchanges ...).

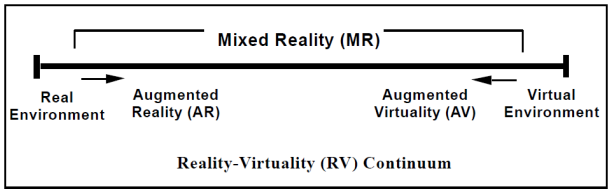

Figure 2: Milgram continuum 1995. [7]

2.2 Manipulation in 3D environment

In 1994, the work of R.J.K. Jacob & al. [3] showed that the device, whose control structure is the closest to the perceptual structure of a task, is the most suitable for performing this task. Therefore, it seems that the degrees of freedom composing the perceptual structure of a task can be used to describe it generically. In more recent work, M. Veit & al showed that users instinctively decompose some complex tasks in order to simplify them [10]. Moreover, this decomposition may vary during the task according to the desired speed of execution and/or the precision [11]. Finally, it appears that a separation of certain degrees of freedom in the design of the interaction leads to better performances of the user if this decomposition is consistant with how the user would simplify the task by himself [10, 12]. Therefore, we need to add to our generic tasks description, information on how the degrees of freedom are preferentially integrated or separated by the user.

3 OUR MODEL

The aim of our work is to restrict the porting of a set of 3D environments on a new device to the development of a single dedicated application.

For this purpose, we totally separate the development of the 3D environment from the user device. So the 3D environments developed with our model run on a separated computer or on a server. As these environments are described generically, it is possible to develop a dedicated application, for a given device, able to display all 3D environment. The portability the of 3D environments is clearly improved because the classic approach requires the development $\mathbf{n^*m}$ applications when our approach requires the development **n+m** applications (with **n** the number of devices and **m** the number of environments). In the same way, it is sufficient to develop a unique application (or client) to support a new device on all the environments and each new environment will be automatically supported by all the devices.

To achieve our model, we propose the use of an abstraction layer in order to design the interaction independently of the nature of the device. This abstraction layer consist in a set of interaction blocks common to all devices. The instantiation and the composition of these blocks makes it possible to script the interaction of the user with the environment.

Because of the separation between the 3D environments and the devices, the devices have to instantiate the interaction blocks transmitted by the environment in order to generate the user interface allowing the interaction with the environment.

3.1 Overall functioning

The overall functioning of our system is illustrated in Figure 3. A 3D scene is transmitted in real time to one or several devices through a communication network. A copy of the scene is then created and synchronized on each device. The same network allows the transmission of a set of basic interaction blocks to the device. These blocks constitute a set of standard objects allowing the description of the actions that the device may perform on the environment. Each device is able to perform a projection of these blocks on itself. This operation consists in generating dynamically the user interface making it possible to carry out the described actions with the device. Menus and parametrization tools are created on the fly to allow the setting of the environment and the selection of geometric tasks. The transmitted tasks are then projected on the degrees of freedom of the device. Finally, the interaction blocks also indicate which objects can be targeted and/or selected by the user. When implementing a client for a device, the interaction blocks projection is optimized according to the device properties (dimensions of the display, number of degrees of freedom of the interaction device, ...).

Once the projection on the device is completed, the interactions of the user with the device triggers the activation of some blocks. This activation is requested by the device through the network already used for the transmission of the scene and the interaction blocks. The environment reacts then by modifying the scene and/or the list of interactions given to the user. For example pressing a button could make an object of the scene bigger while an other could activate the rotation of an object and thus add the rotations along the x, y, and z axis to the interaction list

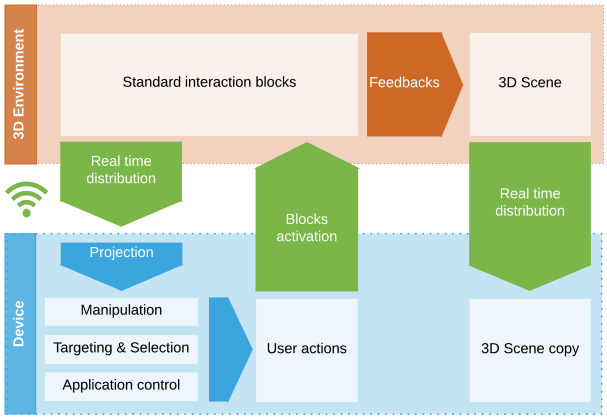

Figure 3: Main model

3.2 Basic interaction blocks

We define the interaction blocks as a finite set of instantiable and parametrizable objects (Figure 5). These objects describe the interactions with a 3D environment. Each VR, AR or MR device must be capable of interpreting and activating all of these blocks. The real-time evolution of the blocks allocated to the users, according to their actions, defines the interactive scenario associated with the 3D environment. We have chosen our interaction blocks according to the following criteria. First of all, the blocks must be interpretable by any type of device. Then, the composition and the parametrization of the blocks must make it possible to design any type of interaction with a 3D environment. It must also be possible to give information on the perceptual structure of the tasks so as to guide the projection on the device. Finally, the number of blocks must remain limited to facilitate the development of the interpreter for a given device.

3.2.1 Manipulation. The existing approach consists in directly linking the degrees of freedom of a 3D object to the degrees of freedom of an input device, while our approach consists in associating a generic motion to a task in a 3D space. This movement, described with interaction blocks that we name **Dof** and **Dof Integration**, is projected by the device on its own degrees of freedom. Figure 4 shows how the translation of a cube on a table can be achieved by composing a Dof Integration block with two Dof blocks and how these blocks can be projected on a gamepad.

The Dof object defines a generic movement with one degree of freedom. This degree of freedom can be either a translation or a rotation around an axis. The motion described is either continuous or discrete. This degree of freedom is defined relative to a reference object in the scene. If no reference object is defined, the user space is used as the default reference.

The Dof Integration object allows to combine several degrees of freedom. For example, this block can be used to define a translation not along an axis (1 Dof) but in a plane (2 Dofs) like in Figure 4. Thus, the interaction block Dof Integration allows to define a movement with several degrees of freedom. The device on which these interaction blocks will be instantiated must then integrate these degrees of freedom if its structure allows it. If the device is not able to integrate the set of degrees of freedom as defined by the Dof Integration block, they remain separated. In this case, a menu is generated to let the user select alternately the separated degrees of freedom.

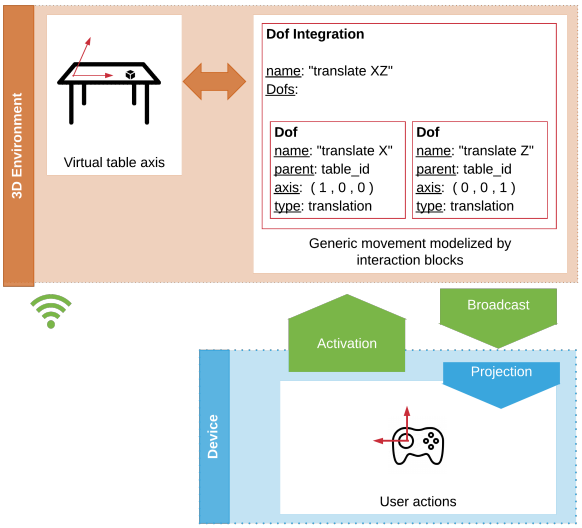

Figure 4: Sample of interaction blocs usage. *Two Dof blocks and a Dof Integration block are associated in order to describe the planar translation of a virtual cube on a table with a two degrees of freedom generic movement. These blocks are transmitted to a device equipped with a joystick. This device interprets the interaction blocks and projects them on the joystick by associating the described generic movement with a two degrees of freedom stick. This association allows to activate the Dof Interaction bricks when moving the stick and thus move the cube.*

3.2.2 Selection & Targeting. The way to target or to select a 3D object differs greatly from one device to another. Indeed, even if the use of ray casting is the most widespread technique, other techniques exist. On the other hand, the way of manipulating the casted ray totally depends on the structure of the device. Therefore, it seems inappropriate to constraint how the targeting and the selection are carried out by the device, so we introduce two basic interactions to indicate that a 3D object can be targeted or selected.

The **Hover** object tells the device if a virtual object can be targeted by the user. This block can be activated to indicate the beginning and the end of the targeting and makes it possible to create a visual feedback showing to the user that he can interact with the object.

The **Pick** object allows to associate an event to the selection of a virtual object by the user.

3.2.3 Application control. The application control gathers all the interfaces available to the user to set up the application and/or select tools: Menus, buttons, text input, ...Since the 3D environment is created separately from any device, the application control

have to be described to the devices with some specific interactions blocks. We propose to describe, in a generic way, the means of parametrization by two interaction blocks:

The **Action** object allows the launch of an event. This object can have a list of parameters as argument. An Action without parameters typically corresponds to a button in the final interface. An action with parameters corresponds to an input form.

The **Direct Input** object allows to modify in real time a parameter of the application or a 3D object (numerical value, options selection, text field, ...).

If the two previously described blocks are sufficient to describe all the elements necessary for the application control, they do not always make it possible to generate an optimal user interface. Indeed, the arrangement and the grouping of the similar elements of the interface are essential for ergonomics. We introduce then a last interaction block called **Interaction List** which guides the creation of menus and submenus during the generation of the user interface. The Interaction List block is used to gather several interaction blocks that have the same semantic value or close uses. This aims to cluster elements which must eventually be grouped in the user interface.

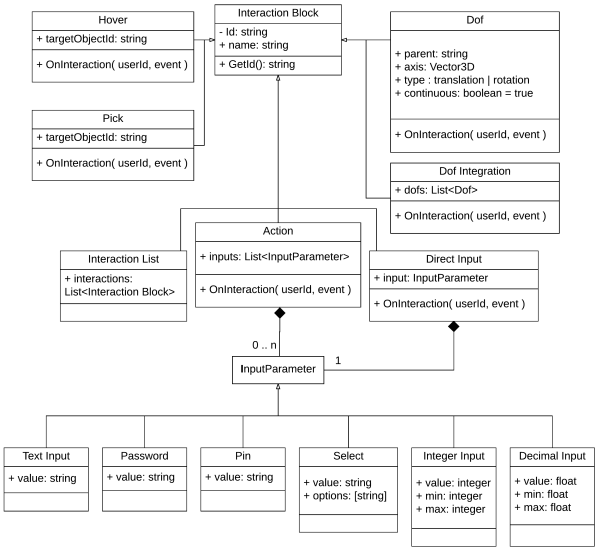

Figure 5: Basic interaction blocks (UML)

4 IMPLEMENTATION

4.1 Shared Virtual Environment Platform (SVEP)

To validate our model, we implemented a Web platform able to host simultaneously multiple 3D environments. This platform was developed with the Node.js technology which allows us to quickly create 3D scenes using the Three.js library. A 3D environment hosted on the platform is made of a script, jointly defining a 3D scene and interaction blocks, but also resource files corresponding to 3D models and textures. Finally, Socket.io was used to create a WebSocket allowing real-time communication between a 3D environment and the visualization devices. All the technological choices for the platform were made to validate the model quickly. A future use of game engines such as Unity or Unreal Engine is considered in order to enrich the developed 3D environments and increase performances. The global architecture of our implementation is presented in Figure 6.

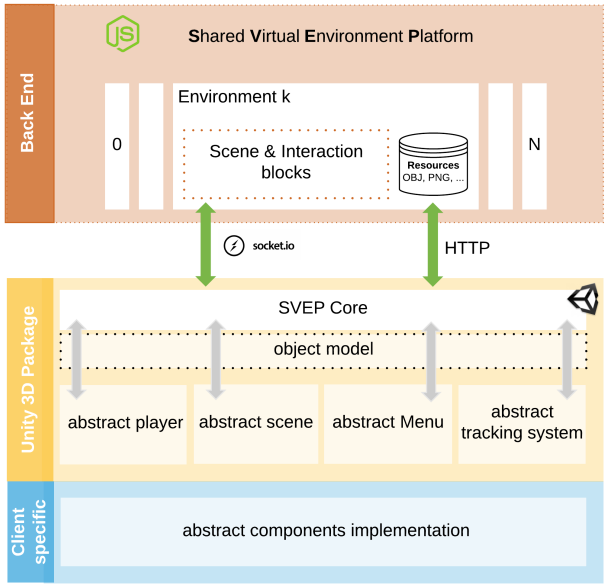

Figure 6: SVEP

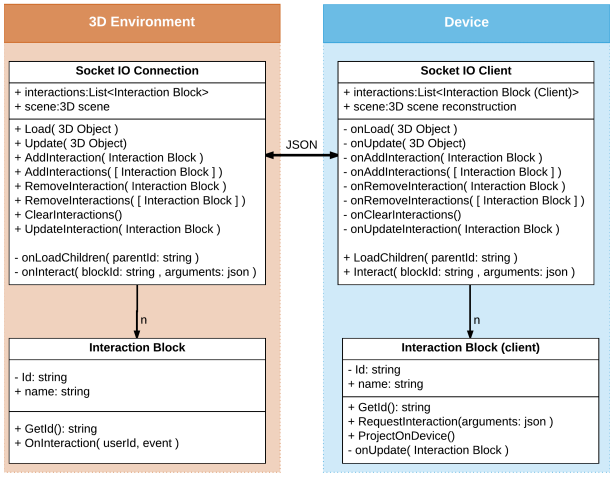

Figure 7: Communication between a SVEP Client and a 3D environment

4.2 SVEP Unity 3D client

In order to speed up the implementation of a client for a given device, we have developed a package for the Unity 3D game engine. This package makes it possible to disregard the network connection with the SVEP platform during the development of a client. Moreover, it allows to pool the architecture of all clients using Unity (Figure 6). A SVEP client is made of 5 modules:

4.2.1 SVEP Core. The SVEP Core module is in charge of the communication between the clients and the hosted 3D environments (Figure 7). It also deserializes messages sent by SVEP into standard objects and it controls the other modules.

4.2.2 Scene. The Scene module dynamically loads the virtual scene of the 3D environment and synchronizes it in real time. The Scene module must extend the Abstract class Scene of the Unity package. The use of an abstract class makes it possible to restrict the developments for a given client to the implementation of a loading strategy adapted to the device capabilities. Several devices can therefore use the same implementation of this module if they have similar performances (RAM, graphic memory, quality of the network connection).

4.2.3 Player. The Player module is responsible for the dynamic projection of the interaction blocks on the device. If an abstract class of the Unity package allows to pool the interfaces with the SEVP Client module, it is necessary to specifically implement the projection of different interaction blocks on the device.

4.2.4 Menu. The Menu module allows the user to navigate between the 3D environments available through the SVEP platform.

4.2.5 Tracking System. The Tracking System module (or TS) is only implemented for the Augmented and Mixed Reality devices. It is responsible for the dynamic positioning of virtual objects based on the real objects of the environment (markers, GPS position ...).

5 RESULTS

5.1 Tested devices

In order to evaluate the portability of 3D environments developed according to our model, we implemented three clients (Figure 1) for the SVEP platform and could verify that the development of a single application allows the portage of all the 3D environments on a new device. The main stages of the implementation are summarized below and in Table 1.

5.1.1 Scene. Since the computing power and the graphic capabilities are similar, all the tested devices share the same implementation of the Scene module.

Table 1: Tested devices

Device	1	2	3
Display	Smart TV	Pc touchscreen	Oculus rift HMD
Navigation	Mouse & keyboard	ARToolKit	Oculus rift IR tracking
Tracking System	.	ARToolKit & Camera	.
Interaction System	Mouse & keyboard	Touchscreen	Oculus rift remote
Max Integrated Dofs	2	2	1+1
UI	2D overlay	2D overlay	2D in world position

5.1.2 Tracking System. Only the device 2 is capable of displaying a 3D environment in Augmented Reality. It is therefore the only device implementing the Tracking System module. The module makes it possible like in [4] to perform, in real time, the tracking with markers using the ARToolKit.

5.1.3 Menu & Application control. For devices 1 and 2, the user interface grouping the Menu module and the application control generated by the Player module is a traditional 2D interface superimposed to the 3D scene. However, the Oculus Rift (device 3) does not allow this type of interface, so a three-dimensional interface has been designed for this device.

5.2 Environment 1 : Virtual object manipulation & feedbacks

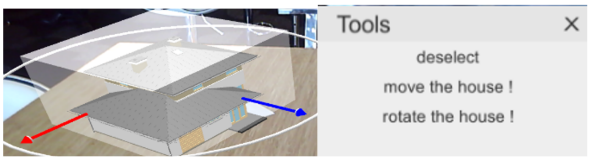

Figure 8: Integration of the degrees of freedom

In order to evaluate our model, we developed a 3D environment allowing a user to manipulate a virtual object (Figure 8) and published it on the SVEP platform. Our objective was to verify that our model allows the manipulation of a virtual object by any device, that a reaction to the manipulations of the user can be dynamically generated, and finally, that it is possible to update in real time the interactions available to the user so that we can script his or her exchanges with the environment.

Figure 9: Separation of the degrees of freedom

5.2.1 Interaction scenario. The user has two way of interacting proposed in this scenario. The manipulation mode allows manipulation of degrees of freedom of the object which can thus be positioned by translation in the XY plan and oriented by a rotation of Z axis. An action also allows to switch into a configuration mode. This second mode gives to the user a set of interactions allowing him to configure the degrees of freedom of the manipulation mode. Figure 10 shows the menu generated by device 1 (Table 1) for configuration mode. The configuration makes it possible to choose if manipulations are continuous or discrete. In the case of a discrete rotation, the rotation pitch is configurable. The translations along the X and Y axes can be integrated (Figure 8) or separated (Figure 9). Finally, selecting the object results in returning to the manipulation mode.

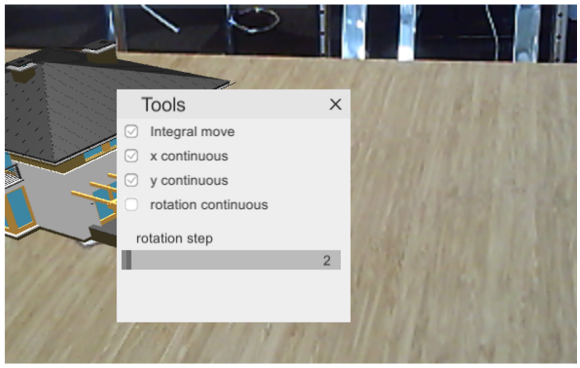

Figure 10: Manipulation configuration

5.2.2 Feedbacks. Since the device has no knowledge of either the semantic nature of manipulations, or the manipulated object, it cannot generate itself visual feedbacks during the interaction to indicate to the user the awareness of his interaction and the possible movements. This return must be generated directly by the environment upon activation of the interaction blocks. Figure 11 summarizes the different feedbacks generated by the 3D environment. When targeting, a transparent selection box is displayed around the object. If the object is selected, arrows and circles are displayed to indicate possible translations or rotations. Finally, the arrows and circles displayed during the selection change colors when activating interaction blocks that they represent.

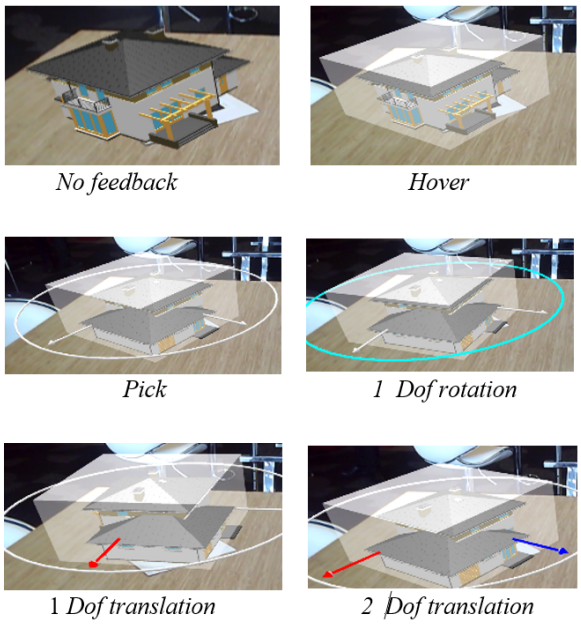

Figure 11: Feedbacks

5.3 Environment 2: Simultaneous manipulation of several objects

If the implementation of the environment presented in Section 5.2 allowed to validate the basic principles of our model, this single test is not sufficient to evaluate the relevance of the model for the scripting of complex interactions. We present below a complementary environment showing how the definition of a theoretical movement by interaction blocks makes it possible to script the simultaneous manipulation of several objects. With this test, we wanted to ensure that the Dof and Dof Integration interaction blocks are sufficient to create, not only virtual objects translations and rotations,but also complex manipulations like scaling, multiple selection or event triggering. The developed 3D environment shows that it is possible to associate such a complex manipulation to a generic translation or rotation move created with the Dof Integration and some Dof interaction blocks.

Figure 12: Sort the stack !

5.3.1 Interaction scenario. The environment is made of a stack of plates. The goal of the user is to sort the plates according to their size. The only authorized operation is to return the top of the stack (Figure 12).

5.3.2 One Dof selection. A block with one degree of freedom is used in association with a vertical movement. A movement from top to bottom or from bottom to top allows to add or remove a plate from the selection (Figure 13).

Using a discrete Dof allows to define the amplitude of the movement used to add or remove a plate from the selection. The possibility of selecting the plates in several goes allows the user to correct his selection.

5.3.3 Full task. A second discrete degree of freedom is used to trigger the reversal of the selection by associating it to an horizontal movement. The integration of these two Dof allows the user to select and flip the top of the stack in a single motion (Figure 14).

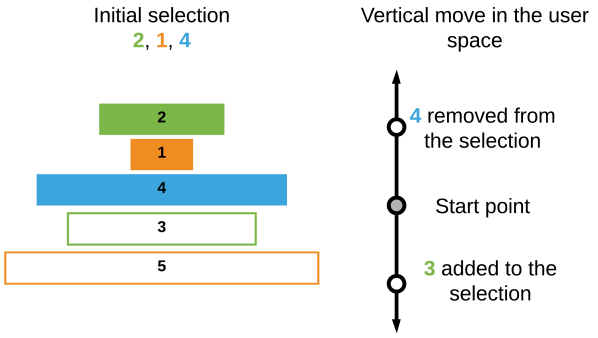

Figure 13: 1 Dof selection

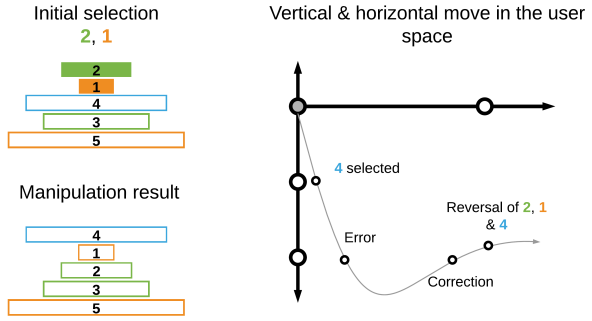

Figure 14: Selection & reversal integration

6 DISCUSSION

By construction, the portage on a new device of all the environments developed according to our model does not depend on the number of environments, but depends on the difficulty to implement the projection of interaction blocks on the device. The use of our model thus allows an increasing portability gain when the number of environments increases. On the basis of the tests presented in Section 5, it seems that the composition and the dynamic evolution of interaction blocks (Section 3.2) allow to script and describe the interactions with a 3D environment in a generic way. Moreover, it appears that this model naturally allows collaboration between different types of devices.

However, additional work is needed to establish good practices for scripting in a shared environment. Indeed, it is likely that visual feedbacks such as the one proposed in Figure 11, generate in some cases, cognitive overloads for the other users. These visual feedbacks could, on the other hand, be beneficial for solving known problems such as the join attention issue presented in [8].

7 CONCLUSIONS

The main objective of this article was to present a generic model of the interaction in 3D environment allowing to totally ignore the nature of the user device during the development of the environment. To do this, we have shown that the degrees of freedom can be used as an abstraction layer common to all devices to define manipulation of 3D objects. Finally, after presenting our implementation of the model, we have shown how it can be concretely used for the scripting in 3D environment.

Moreover, this model seems suitable for the creation of collaborative environments. The possibilities offered by facilitated collaboration between complementary devices open up new perspectives, particularly in the area of collaborative design. However, complementary work establishing good scripting practices with the model is to be foreseen before any rigorous evaluation of its contribution in the field.

REFERENCES

[1] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. 2001. Recent Advances in Augmented Reality. *IEEE Comput. Graph. Appl.* 21, 6 (Nov. 2001), 34–47. https://doi.org/10.1109/38.963459

[2] Steve Bryson. 1996. Virtual Reality in Scientific Visualization. *Commun. ACM* 39, 5 (May 1996), 62–71. https://doi.org/10.1145/229459.229467

[3] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and M. Preston Mullen, Jr. 1994. Integrality and Separability of Input Devices. *ACM Trans. Comput.-Hum. Interact.* 1, 1 (March 1994), 3–26. https://doi.org/10.1145/174630.174631

[4] H. Kato, K. Tachibana, M. Billinghurst, and M. Grafe. 2003. A registration method based on texture tracking using ARToolKit. In *2003 IEEE International Augmented Reality Toolkit Workshop*. 77–85. https://doi.org/10.1109/ART.2003.1320435

[5] Boriana Koleva, Holger Schnädelbach, Steve Benford, and Chris Greenhalgh. 2000. Traversable Interfaces Between Real and Virtual Worlds. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '00)*. ACM, New York, NY, USA, 233–240. https://doi.org/10.1145/332040.332437

[6] Sebastien Kuntz. 2015. MiddleVR a generic VR toolbox.. In *VR*, Tobias HÃűllerer, Victoria Interrante, Anatole LÃŀcuyer, and J. Edward Swan II (Eds.). IEEE Computer Society, 391–392. http://dblp.uni-trier.de/db/conf/vr/vr2015.html#Kuntz15

[7] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. 1995. Augmented reality: a class of displays on the reality-virtuality continuum. *Proc. SPIE* 2351, 282–292. https://doi.org/10.1117/12.197321

[8] Christian Schnier, Karola Pitsch, Angelika Dierker, and Thomas Hermann. 2011. Collaboration in Augmented Reality: How to establish coordination and joint attention?. In *Proceedings of the 12th European Conference on Computer Supported Cooperative Work (ECSCW 2011)*, Susanne Boedker, Niels Olof Bouvin, Wayne Lutters, Volker Wulf, and Luigina Ciolfi (Eds.). Springer-Verlag London, 405–416. https://doi.org/10.1007/978-0-85729-913-0_22

[9] J. Tumler, F. Doil, R. Mecke, G. Paul, M. Schenk, E. A. Pfister, A. Huckauf, I. Bockelmann, and A. Roggentin. 2008. Mobile Augmented Reality in industrial applications: Approaches for solution of user-related issues. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. 87–90. https://doi.org/10.1109/ISMAR.2008.4637330

[10] Manuel Veit, Antonio Capobianco, and Dominique Bechmann. 2009. Influence of Degrees of Freedom's Manipulation on Performances During Orientation Tasks in Virtual Reality Environments. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST '09)*. ACM, New York, NY, USA, 51–58. https://doi.org/10.1145/1643928.1643942

[11] Manuel Veit, Antonio Capobianco, and Dominique Bechmann. 2010. Dynamic Decomposition and Integration of Degrees of Freedom for 3-D Positioning. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology (VRST '10)*. ACM, New York, NY, USA, 131–134. https://doi.org/10.1145/1889863.1889891

[12] M Veit, A Capobianco, and D Bechmann. 2011. An Experimental Analysis of the Impact of Touch Screen Interaction Techniques for 3-D Positioning Tasks. In *Proceedings of the 2011 IEEE Virtual Reality Conference (VR '11)*. IEEE Computer Society, Washington, DC, USA, 75–82. https://doi.org/10.1109/VR.2011.5759440