Interactive configurable virtual environment with Kinect navigation and interaction

João Pinto[1] jhpinto@ua.pt
Paulo Dias[1,2] paulo.dias@ua.pt
Sérgio Eliseu[3] s.eliseu@ua.pt
Beatriz Sousa Santos[1,2] bss@ua.pt

[1]DETI/UA - Department of Electronics, Telecommunications and Informatics
[2]IEETA - Institute of Electronics and Telematics Engineering of Aveiro
[3]iD+ / i2ADS - Faculty of Fine Arts, University of Porto

Abstract

As a solution to immersive virtual museum visits, we propose an extension upon the platform we previously developed for Setting-up Interactive Virtual Environments (pSIVE) that maintains all of the Virtual Environment creation features of the platform as well as content association (PDF, Video, Text), but also allows gesture-based interaction and one to one navigational input using skeleton tracking with a Kinect that is calibrated in the real world space where the user is located in. In order to validate our new navigation and interaction methods, a comparative study was performed, comparing our gesture-based interaction and positional tracking with a controller button navigation and interaction method.

Keywords

Virtual Environment, Navigation and Interaction, User Study, Kinect, Virtual Reality

1. Introduction

With the rising popularity of high-end Virtual Reality (VR) hardware in the entertainment market, immersive VR applications start to have wider expression due to cost reduction and software availability.

In 2013, as a response to the complexity of building a virtual environment, the platform for Setting up Interactive Virtual Environments (pSIVE) was developed as a master thesis [Souza 13] to not only allow an easy configuration of a virtual scene using a diversity of models, but also interact with the environment using non-conventional hardware such as trackers and head-mounted displays (HMDs).

Within that context, and as an expansion upon that project, we have added the possibility of exploring a fully immersive environment with one to one position mapping from the real world to the virtual world, as well as gesture-based interaction with menus and content within that virtual world detected by a Kinect. What this means is that the user can configure a virtual museum with custom content and navigate through it either hands-free by walking in a real empty room, as shown in figure 1, or standing still with a physical controller, while viewing the virtual museum through an HMD.

In this paper, we discuss the platform's architecture, its features, how it is configured and the ways in which the users can interact with the virtual content (menus and navigation).

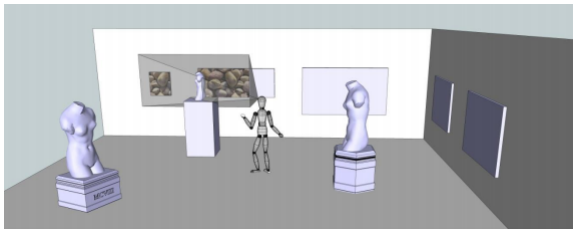


Figure 1: Virtual Museum concept

2. Related and Previous Work

One of the most commonly used types of display in Virtual Reality applications is the Cave Automatic Virtual Environments (CAVE) [Burdea 03], shown in figure 2. It consists of a room where the entire surface area is covered with high resolution projections that generate rapidly alternating images for each eye, allowing any number of users in the room to perceive the visual data. Stereoscopic shutter glasses are synchronized with the projectors in order for the user to only see the images for the correct eye, which enables depth perception. The projectors are placed out-

side of the room, and are controlled by one or more computers.

Figure 2: Multiple users in a CAVE [Craig 09]

Our work possesses some similarities to a CAVE system, in regards to the usage of the physical space for navigation. While in a cave the users are restricted to the empty room they are situated in, with the walls acting as a "screen" of sorts, our work is adaptable to any room, and utilizes an HMD as the user's window into the virtual world.

The Empty Museum [Hernandez 03] is perhaps the application in literature that most closely resembles ours, thematically and practically. It features a multi-user immersive walkable and wireless system where users can navigate a virtual environment with content such as audio, text, images and video. The system set up (figure 3) consists of a laptop computer in a backpack that renders the environment according to the user's per spective, connected to a system that captures wireless movement [Foxlin 98] and using virtual reality goggles.

Figure 3: Empty museum setup [Hernandez 03]

When compared to our system, The Empty Museum has the advantage of supporting multiple users at the same time, with 3D avatars within the virtual environment. Its interaction, however, is solely determined by the user's position, while we support button input for interaction, as well as gestures, allowing for more complex interactions. Their tracking system must also be mounted on the ceiling, while ours is easy to mount and easily expandable by setting up more kinects.

Brown University's VENLab [Tarr 02] is an immersive virtual reality space, with a $12m^2$ walkable area that uses an IS-900 head tracker to measure the user's position in real time, and an 80 degree field of view HMD to fully immerse the user in the virtual environment (figure 4).

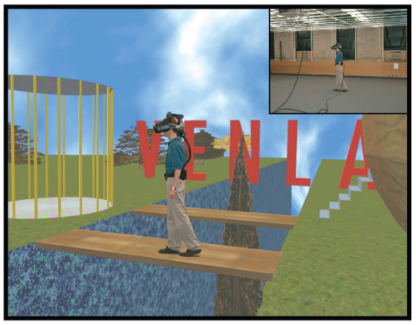


Figure 4: User walking in the VENLab space [Tarr 02]

The KidsRoom's (figure 5) [Bobick 99] aim is to explore computer vision technologies in an augmented reality interactive playspace for children. To do so, it recreates a child's bedroom composed of two real walls and two large video projection screens where images are projected from outside of the room, as well as ceiling mounted colored lights, speakers, cameras and a microphone, all controlled by a six-computer cluster. There are 4 cameras pointing at the room, one for tracking people, two for action recognition of people in the room, and one to provide a view of the room for spectators.

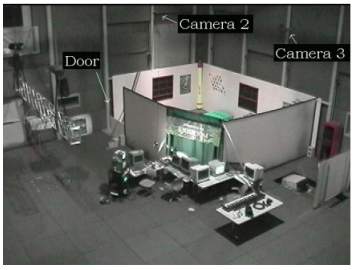


Figure 5: KidsRoom bedroom setup [Bobick 99]

pSIVE, a platform we previously developed, can be used to set up a virtual scene using a diversity of models, and those models can have a variety of content (PDF, Video, Text) attached to them. It uses OpenSceneGraph [Wang 10] as a graphical engine and VRJuggler [Bierbaum 01] as a middleware to interpret input from trackers and controllers, which are used to define orientation within the scene and interact/navigate, respectively.

Content can be accessed through a 2D linear menu that pops up when the user presses a button on his controller while looking at an object that has been configured with content.

3. Platform

Taking advantage of the previous work done in pSIVE, and still with its goals of easy configuration of the scene, support for content such as videos, PDF, text and images, and versatility of hardware in mind, we have expanded and up-

dated upon that platform. Support for the Oculus Rift[1] was added, as well as abandoning the VRJuggler library used by pSIVE, which we found hard to set-up and hasn't been updated since 2013, in favor of osgVRPN[2], a Virtual Reality Periphery Network (VRPN) [Taylor 01] library for osg that gives us lower level control of the input into our system in addition to being easier to configure.

VRPN is a library that provides a network-transparent interface between applications and physical devices used in VR systems using a client-server approach (the client being the application, and the server being the interpreter of input from the physical devices). It provides a layer of abstraction that classifies inputs into several categories (Analog, Button, Dial, ForceDevice, Sound, Text, and Tracker), which allows us to receive generic input from different devices.

With these tools and features in mind, we have designed a platform that lets the user experience a personalized virtual museum, complete with interactive content accessed through menus using gestures or a physical controller.

In the ideal use case scenario, the user carries a laptop connected to an Oculus Rift and running the client application on a backpack, while one or more computers run servers sending, to the client computer, the user's skeleton data from kinects, calibrated to the empty room the user is in.

Another use case scenario lets the user navigate using a physical controller, such as a WiiMote, removing the need to physically navigate in an empty room and carry a laptop on his/her back.

3.1. System Architecture

For the graphical engine, we have decided to continue using OpenSceneGraph (used in pSIVE), due to previous work, its active community, and the availability of VR libraries such as osgVRPN, and osgOculusViewer[3] (oculus rift support). Using a well-known game engine such as Unity[4] would have been a plausible alternative, if not for the pay-wall behind some features and the higher degree of control and flexibility that a more generic open source graphics engine such as OpenSceneGraph can provide.

The application's architecture and workflow is detailed in figure 6. It is configured with several XML files and receives input information from one or more VRPN Servers, which is then interpreted by the osgVRPN library. That information is then handled in one of two ways: interaction with menus or content (Menu Handler), or navigation (Camera Handler). The scene is then rendered for use with the Oculus Rift using the osgOculusViewer library.

In order to provide a fully immersive VR experience with skeleton tracked navigation in mind, the application needed to be designed to be easily expandable in terms of area covered by the Kinect[5] sensor(s). As such, we

[1]https://www.oculus.com/
[2]https://github.com/VirtualMe/osgvrpn
[3]https://github.com/bjornblissing/osgoculusviewer
[4]http://www.unity3d.com/
[5]https://www.microsoft.com/en-us/Kinectforwindows/

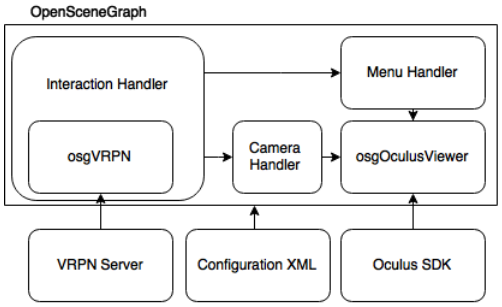


Figure 6: Application Architecture

have decided to use a PC based client-server architecture in which the client is responsible for all of the rendering of the virtual world and handling of the Head Mounted Display (HMD) orientation tracking, and the VRPN servers are used to communicate the user's skeleton information (positional data of head, hands and gripping gestures) to the client using one or several Microsoft Kinect devices collecting skeleton data with the Kinect SDK 1.8[6]. This arquitecture is shown in figure 7.

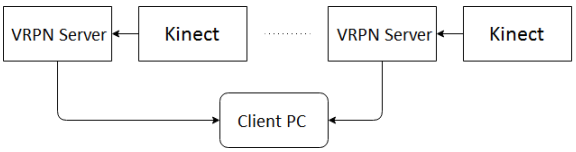


Figure 7: Client Server Architecture

osgVRPN is used in our client application with two purposes:

- Receiving information from any trackers supported by a standard VRPN server (WiiMote, Razer Hydra, Keyboard, etc.).
- Receiving information from our custom VRPN server with integrated Kinect support, something that was not available in pSIVE.

3.2. Platform Configuration

The platform, composed by the client which renders the scene and tracks the user's head orientation, and the server(s) which track(s) the user's body or read input from a physical controller, are configured through 3 XML files.

- Config.xml configures the scene, its objects and the object's contents with configurable scale, position and rotation within the scene.
- Kinect.xml configures the custom VRPN Kinect servers to listen to, using their server name and IP address.
- Controls.xml configures the navigation input buttons, allowing configuration of directional input and menu activation input.

[6]https://www.microsoft.com/en-us/download/details.aspx?id=40278

3.2.1. Scene configuration

The Config.xml file consists of a list of models, their physical attributes (size, rotation, location), and their available content.

- Filename: Path to the model.
- Label: Model description that is shown when an object is interactive.
- Rotate/Translate: User defined object rotations and translations in all axis.
- Context: Whether the object is interactive or not.

In that same file, each model has a list of available content with the following properties:

- Type: From 0 to 3, Text, PDF, Video and Image data types, respectively.
- Label: Menu entry label.
- userContent: In case of the Text data type, the text to display. In all other data types, it is the path to the content to be shown.

3.2.2. Server Configuration

In order to utilize the same library that is used for the button input (osgVRPN), a custom VRPN server with 12 channels and 3 buttons was created to support the information gathered by the Microsoft Kinect sensor. These channels are, in order, the left hand's x,y,z position, the right hand x,y,z position, the head's x,y,z position and the Kinect x,y,z position within the virtual scene.

The buttons are used to convey three binary inputs:

- Left hand grip.
- Right hand grip.
- Skeleton detected.

The left and right hand grip inputs are detected using the Kinect 1.8 SDK, and are used in most of the hands-free interactions. The skeleton detection is necessary in order for the client application to gather information from the Kinects that are detecting the user among the many Kinect servers the client application can be communicating with at any given time.

All positional information is conveyed after being transformed into virtual world coordinates using the calibration method described in section 4.

4. Kinect Calibration

The servers need to be configured with a transformation matrix in order to transform the Kinect coordinates (distance in meters from the Kinect camera) into our virtual world coordinates.

Visualization ToolKit (VTK) [Schroeder 98] was used due to previously developed work, familiarity, and easy access to its Iterative Closest Point (ICP) algorithm, to develop a program that grabs a 3D cloud from a Kinect sensor (shown in figure 8) and allows the user to roughly position it within a 3D model of the room (designed using SketchUp) that the Kinect is in, using the keyboard to translate and rotate the frame (roughly positioned frame shown in figure 9).

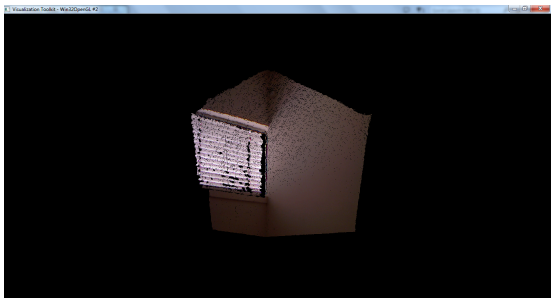


Figure 8: Captured 3D cloud

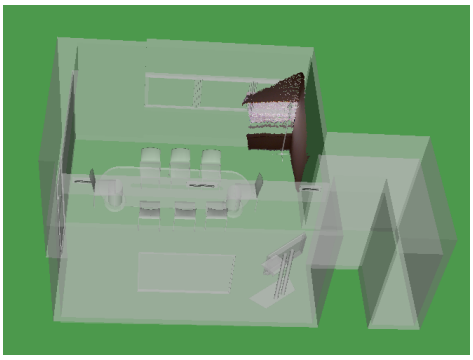


Figure 9: Manually overlaid 3D cloud

In order to fine tune the position of the frame we use the ICP algorithm [Besl 92] present in VTK, and the frame is matched as closely as possible to the 3D model (the outcome of this algorithm is shown in figure 10). The output of the ICP algorithm is a transformation matrix that we combine with the user manual positioning that when applied to the data collected from the Kinect successfully converts points from one coordinate system to the other.

That 4x4 matrix is finally exported from the calibration program as a *.mtx file, ready to be read as a command line parameter by the server that is collecting information from the Kinect from which the matrix was calculated.

5. Interaction

When developing a VR application, and with the advent of more advanced HMDs such as the Oculus Rift, VIVE and Morpheus, some of the traditional approaches to user interfaces and the display of information have to be reconsidered and at times re-purposed for use in a VR environment.

We took advantage of the use of an HMD with orientation tracking in order to utilize a gaze directed method of se-

Figure 10: 3D cloud after ICP algorithm

lecting objects within the virtual scene using a ray-tracing technique. This method was tested and compared against using a hand held selection tool with orientation and position, described as a ”laser pointer selection” method in [Souza 14], and was proven to be both faster and less prone to error regardless of distance to the selected object.

Interactive objects display a configurable label in the center of the screen when looked at, which indicates that upon activation a menu containing the object's content will pop up in front of the user.

Input in our system can be done in one of two ways:

- Gestures using a Microsoft Kinect (SDK 1.8).
- Button input using any kind of controller device supported by VRPN.

5.1. Navigation

There are two ways to navigate the virtual environment, with different use cases in mind:

- Skeleton tracking based navigation using a custom VRPN server with a connected Kinect.
- Physical controller based navigation that does not require the user to move in the physical space.

Navigation in this skeleton tracking mode is done by tracking the user's head with a Kinect and positioning the camera correctly within the scene using the tracking data, and as such does not require navigational ”input” per se.

It is worth noting that while the user's position in the scene is being tracked by the Kinect sensor, the HMD does not know the user's correct initial head orientation. To solve this issue, we require that the user initially performs a calibration gesture (grip with his/her right fist) while looking at the closest real-world Kinect sensor in order to offset the HMD orientation to look at the representation of that Kinect in the virtual world.

Navigational input using a physical controller is gaze-directed and reads button information from a generic VRPN server as input to our application. In order to keep the number of buttons needed to a minimum, the button to exit a menu is the ”right” button. For that same reason (the movement buttons are also used to navigate the menu system), navigation in this mode is locked when the user has a menu or content open, while in the gesture-based mode the user can move freely while browsing menus or experiencing content. Table 1 shows button and gesture mapping for navigation actions.

Action	Gesture	Button
Navigate forward	Walk	Up key
Navigate backward	Walk	Down key
Navigate left	Walk	Left key
Navigate right	Walk	Right key

Table 1: Navigation gesture/button mapping

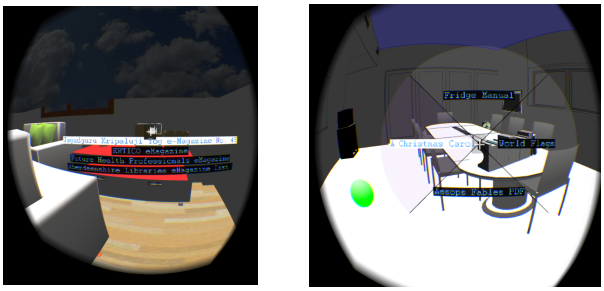
5.2. Menus

Maintaining the standard used in pSIVE, we have implemented two styles of adapted 2D menus in our application. Adapted 2D menus are a representation of standard 2 dimensional menus rendered on 3D geometry within the scene, and are the most prolific group of 3D system control techniques [Bowman 04], and can be present in multiple forms (linear, radial, etc.).

Things like the position, size and shape of menus have to be taken into special consideration due to specifications such as the fish-eye lens distortion in some HMD's, and particularly in the one used in our implementation, the Oculus Rift, which makes text hard to read when close to the edges of the screen. With those constraints in mind, we made the decision to place the menus and content centered on the user's field of view, where his/her gaze is focused, as well as avoiding the use of thin text as advised in the Oculus best practises guide [VR 15].

The application supports two different types of adapted 2D menus, illustrated in figure 11, namely linear and radial menus.

Linear menus are typically displayed when using a physical controller to interact with the scene, and are a modified version of the linear menus used in pSIVE due to an incompatibility with the previous implementation and the render distortion needed in order to use the Oculus Rift.



(a) Linear menu (Left eye) (b) Radial menu (Left eye)

Figure 11: Menu types

When interacting using gestures, radial menus are displayed. We have implemented this format for our gesture-based interaction due to faster selection times and lower error rates [Callahan 88] [Chertoff 09] when compared to linear menus.

Table 2 shows button and gesture mapping for actions related to menu interaction.

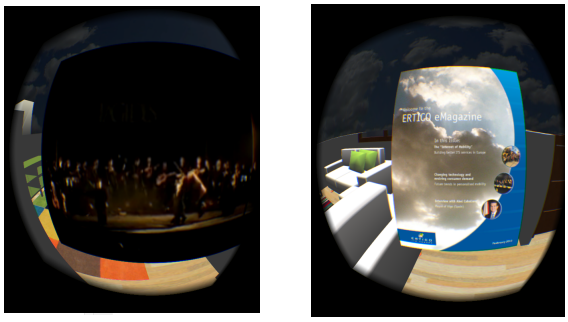
Action	Gesture	Button
Browse Menu	Move right hand	Up/Down key
Activate Option	Grip right hand	Activation key
Exit Menu	Grip both hands	Right key

Table 2: Menu browsing gesture/button mapping

5.3. Content

Content, such as images, videos (figure 12a) or PDF files (figure 12b) are rendered on a quadrangular plane and placed in front of the user. When the content is active, if pertinent, certain interactions such as browsing the PDF pages are possible.

Table 3 shows the button and gesture mapping for interaction with content.



(a) Video content (Left eye) (b) PDF content (Left eye)

Figure 12: Content rendered on a quadrangular plane

Action	Gesture	Button
Previous page	Grip left hand	Left key
Next page	Grip right hand	Right key
Exit content	Grip both hands	Down key

Table 3: Content browsing gesture/button mapping

6. Demo Environment

For the demonstration and testing environment, we modelled a meeting room in SketchUp after a real meeting room in our department, and added several objects that are not present in the real room. Each of those objects was then configured to contain various text, video, pdf and image contents.

In order to test the Kinect-based navigation method, we set up and calibrated two Kinects in the room (set up seen on figure 13, user interaction seen on figure 14), one running on each laptop.

Figure 13: Room setup

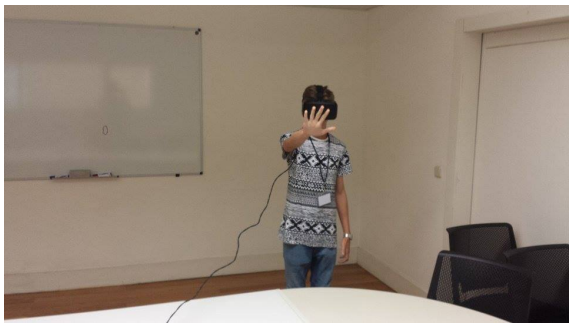


Figure 14: User performing gesture

7. Comparison of Navigation and Interaction Methods

In order to analyze and compare the two implemented navigation and interaction techniques (Kinect-based or controller-based), we ran an experiment with 12 volunteer participants from the Aveiro University's Summer Academy, which means that participants' range of ages is not very varied, ranging from 15 to 17 years.

7.1. Methodology

This experiment aimed to verify if the two methods of interaction and navigation are equally usable in our demonstration environment. Our experiment had two input variables, namely the two different navigation and interaction methods. In one method, the users walk in the room to navigate, and use gestures to interact with the scene. In the other method, the user utilizes a controller with buttons to navigate and interact. Both methods display radial menus. The output variables of our experiment are given in measures of time and number of mistakes made.

The experiment consisted of a simple test of navigation and interaction: The participants had to navigate from one end of the virtual room to the other, and interact with a plant pot located on a table (shown in figure 15). Upon interaction, a menu is shown, and the goal is for the users to select the "Flower" option available on the menu.

Participants were given an ID at the start of the experiment (12 users, IDs ranging from 0 to 11). Even numbered participants ran the experiment using the controller first, and odd numbered participants used the Kinect first, in order to attenuate possible bias due to learning effects. The participants were given no time to practice and learn the system beforehand. The participants were observed while performing the experiment and were asked to answer a questionnaire regarding their satisfaction, difficulties and preferred method. It is worth noting that while 12 participants ran the experiment, only 11 delivered the questionnaire.

Measurements were taken automatically during the experiment, and consisted of the time the participant took to get in front of the plant pot and activate the menu, the time the they took to select the correct option, as well as any incorrect options selected.

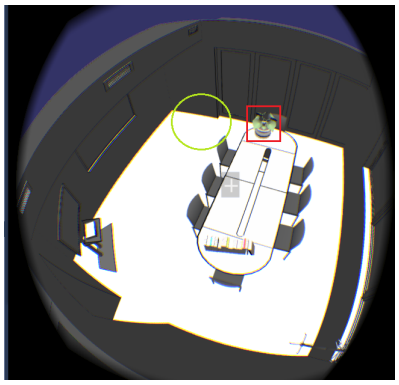


Figure 15: Experiment environment (interaction zone circled)

7.2. Results

Regarding the times measured, the differences in the time taken to reach the plant pot are negligible with the controller-based method averaging 35,25 seconds to reach the plant pot versus the 35,41 seconds of the Kinect-based method while the interaction was faster when using the controller-based method, which averaged 16,6 seconds to activate the correct option versus the 30,3s that the users took with the Kinect.

Users also made more mistakes with the Kinect-based method, with three participants out of twelve making one mistake in selecting the right option, while nobody selected the wrong option using the controller.

The questionnaire results are detailed in tables 4, 5 and 6. Tables 4 and 5 indicate participant answers to several parameters, represented as a median of each index in a scale of 1 (disagree) to 5 (agree), in regards to navigation and interaction, respectively. Table 6 details the number of participants that claim to prefer one method or the other, or have no preference.

While tables 4 and 5 seem to indicate little difference between the two options, the Kinect-based method seems to be predominantly preferred in both navigation and interaction, as shown in table 6.

	Kinect	Controller
Ease of positioning	4	4
Intuitiveness	4	4
Has Annoying features	2	2
Requires training	5	5
Satisfaction	4	4

Table 4: Questionnaire results regarding navigation

	Kinect	Controller
Ease of positioning	4	4
Intuitiveness	4	4
Has Annoying features	3	2
Requires training	5	3
Satisfaction	4	4

Table 5: Questionnaire results regarding interaction

	Navigation	Interaction
Kinect	7	6
Controller	1	3
Indifferent	3	2

Table 6: Questionnaire results regarding preferences (in number of participants)

With all this in mind, our results seem to show that even though the Kinect-based method for navigation and interaction is slower on both accounts, participants seem to prefer it, possibly due to the novelty factor.

8. Conclusions and future work

This paper presents an evolution and rework of pSIVE, with the goal of providing the tools to enable immersive museum visits in virtual reality. While pSIVE allowed for easy creation of Virtual Environments without knowledge of any programming languages, it lacked the tools for real time positional and gesture tracking with a Kinect, as well as the calibration tools necessary to tie the virtual world coordinates to real world positions. We have also implemented a new style of menu (radial), to accompany our implementation of gesture-based interaction.

The results of our experiment with participants seem to indicate a positive reaction, with the majority of users preferring to use the Kinect-based method of navigation and interaction within the virtual environment. Navigation wise, both methods seem to perform equally. When it comes to interaction, gestures seem to be fairly slower than button input. A proposed solution to make selection times faster is the implementation of a gaze directed method with gesture actuation selection.

Future work can involve multiple aspects, from implementing new and more efficient styles of menus and interaction methods, to finding a way of automating the calibration process. An expansion upon this project is already

in the works, with plans to allow users to create their own museum from a list of 3D models and using gestures to place them in the environment, in real time, as well as saving and sharing their custom museum with other users.

Acknowledgements

This work is supported by National Funds through FCT - Foundation for Science and Technology, in the context of the projects UID/CEC/00127/2013 and Incentivo/EEI/UI0127/2014.

References

[Besl 92] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.

[Bierbaum 01] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. Vr juggler: A virtual platform for virtual reality application development. In *Proceedings of the Virtual Reality 2001 Conference (VR'01)*, VR '01, pages 89–, Washington, DC, USA, 2001. IEEE Computer Society.

[Bobick 99] Aaron F. Bobick, Stephen S. Intille, James W. Davis, Freedom Baird, Claudio S. Pinhanez, Lee W. Campbell, Yuri A. Ivanov, Arjan Schütte, and Andrew Wilson. The KidsRoom: A Perceptually-Based Interactive and Immersive Story Environment, 1999.

[Bowman 04] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola, and Ivan Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[Burdea 03] G. Burdea and P. Coiffet. Virtual reality technology. In *Presence: Teleoperators & Virtual Environments*, volume 12, pages 663–664, 2003.

[Callahan 88] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88*, pages 95–100, New York, New York, USA, May 1988. ACM Press.

[Chertoff 09] Dustin B. Chertoff, Ross Byers, and Joseph J. LaViola. Poster: Evaluation of menu techniques using a 3D game input device. In *2009 IEEE Symposium on 3D User Interfaces*, pages 139–140. IEEE, 2009.

[Craig 09] Alan B. Craig, William R. Sherman, and Jeffrey D. Will. *Developing Virtual Reality Applications: Foundations of Effective Design*. Morgan Kaufmann, 2009.

[Foxlin 98] Eric Foxlin, Michael Harrington, and George Pfeifer. Constellation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pages 371–378, New York, New York, USA, July 1998. ACM Press.

[Hernandez 03] L. Hernandez, J. Taibo, A. Seoane, R. Lopez, and Rocio. Lopez. The empty museum. Multi-user interaction in an immersive and physically walkable VR space. In *Proceedings. 2003 International Conference on Cyberworlds*, pages 446–452. IEEE Comput. Soc, 2003.

[Schroeder 98] Will Schroeder, Kenneth M. Martin, and William E. Lorensen. *The Visualization Toolkit (2Nd Ed.): An Object-oriented Approach to 3D Graphics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[Souza 13] Danilo Souza. Platform for Setting up Interactive Virtual Environments, 2013.

[Souza 14] Danilo Souza, Paulo Dias, and Beatriz Sousa Santos. Choosing a selection technique for a virtual environment. In *Lecture Notes in Computer Science*, volume 8525 LNCS, pages 215–225. Springer Verlag, 2014.

[Tarr 02] Michael J Tarr and William H Warren. Virtual reality in behavioral neuroscience and beyond. *Nature neuroscience*, 5 Suppl:1089–92, November 2002.

[Taylor 01] Russell M. Taylor, II, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T. Helser. Vrpn: A device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, pages 55–61, New York, NY, USA, 2001. ACM.

[VR 15] Oculus VR. Best practices guide, January 2015. http://static.oculus.com/sdk-downloads/documents/Oculus_Best_Practices_Guide.pdf.

[Wang 10] Rui Wang and Xuelei Qian. *OpenSceneGraph 3.0: Beginner's Guide*. Packt Publishing, 2010.