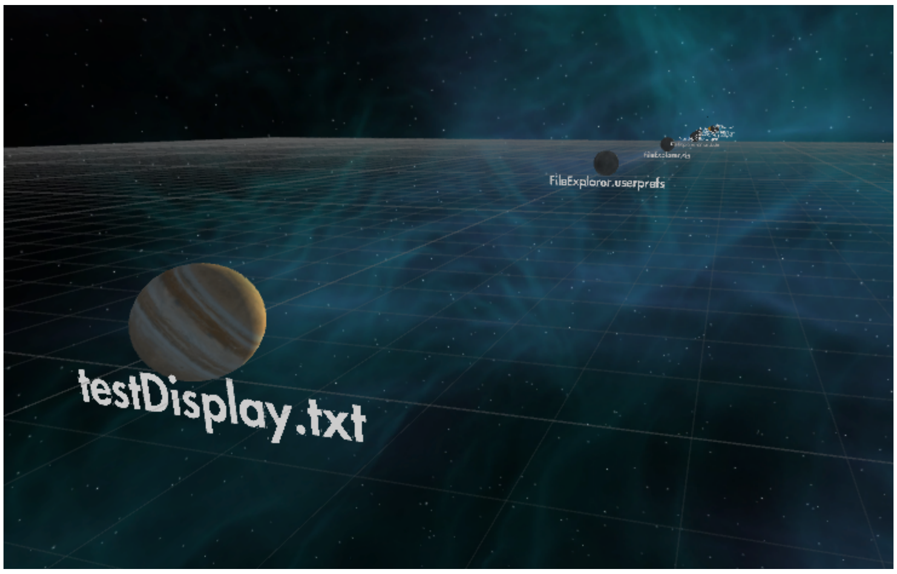
EPFL CS-444 Virtual Reality

File Explorer in Outer Space - FEOS

Group 3: Florian Junker, Sidney Bovet

May 11, 2015



1 Project presentation & User experience

This project aimed to build a 3D file explorer to have an entertaining yet useful VR experience. The base idea is to have all folders represented by solar systems. All files within that folder are planets, whose texture changes according to the file type itself. Figure 1.1 show a general example of the left eye of the user's view.

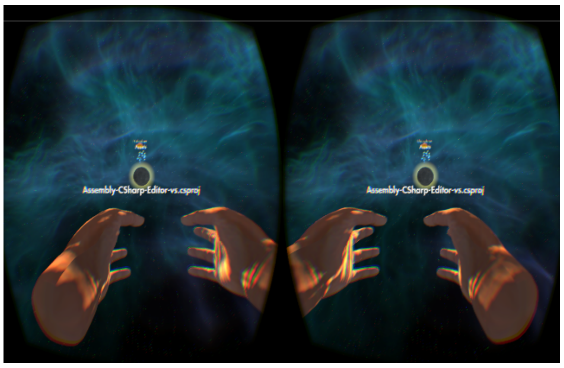

Assembly-CSharp-Editor-vs.csproj
Assembly-CSharp-Editor-vs.csproj

Figure 1.1: A general screenshot of the file explorer's scenario and GUI.

Immersion in the VE is provided by two devices: a HMD and a Razer Hydra. The former provides a 360-degree field of regard as opposed to other webcam-based head tracking solutions, while the latter gives 6 DOF, occlusion-resistant hand tracking. By coupling the Razor hydra with virtual hands animated according to some trigger buttons pressed, one can feel quite a good sense of Agency given some training. For instance, the user can grab the air using both bottommost triggers to pull and push herself along the axis on which the planets appear. An asteroid belt is used to show the user that the end of the folder has been reached.

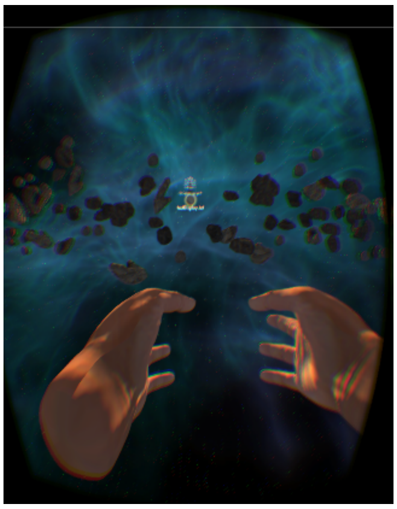


Figure 1.2: The asteroid belt at the end of the folder.

Selection is achieved through regard: the last item that have crossed the centre of the screen if selected (i.e. simple click on a regular file explorer), and one can activate that selection (i.e. double-click) by pressing the topmost trigger and having a swift move towards the desired item.

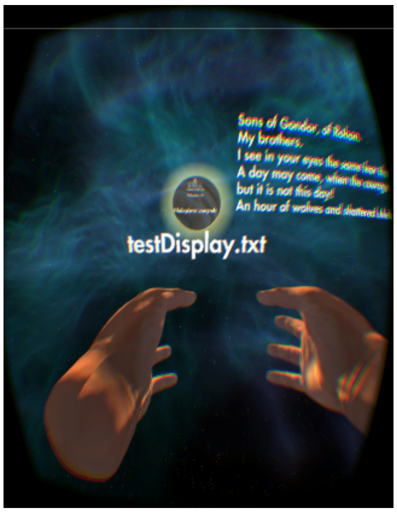

Sons of Gondor, of Rohan.
My brothers.
I see in your eyes the same fear th
A day may come, when the courag
but it is not this day!
An hour of wolves and shattered shie

Figure 1.3: How a text file is displayed in the VE.

Upon activation (i.e. double-click) a file will either show its content if it is supported (as shown in Figure 1.3 for a text file) or bump to show a negative feedback due to unsupported extension. A folder will trigger a solar system change and the user then can browse that folder.
- How the interaction loop works? What the user sees and how the application behaves with respect to his/her input.
- Relevant screenshot(s) (and/or photos showing the user playing) of the program running

2 Libraries, resources and algorithms used

The virtual environment was built using Unity.The project uses the Oculus integration plugin for displaying the VE and to achieve head tracking. A modified version of the Hydra framework provided by Razer is used for the tracking of the markers. Full use is made of the various C# libraries provided by Unity including utilities, math, random, etc.

The physic simulation and particle systems generation mechanisms are those of Unity. For instance the asteroid belt mentioned above is shattered by interacting with the user using Unity's built-in rigid bodies and mesh colliders.

3 Understand and compile the project

Since the project uses the Sixsense Razor Hydra as a marker tracking system, whose driver is only available for Windows, it is only compatible with a Windows PC. The Oculus and Hydra drivers must be installed on the computer. The version of the different components are the following:

Oculus VR Unity integration v0.5.0.1-beta

Sixsense Unity plugin v1.0.6

Oculus driver Windows - v0.5.0.1-beta

Razer Hydra driver Windows - v1.01

The whole project can be opened in Unity and compiled from there. The unity hierarchy is quite simple. There is one empty object called controller that contains scripts responsible for planets generation (PlanetController.cs) and SixenseInput.cs that allow us to use razer Hydra controllers. The later is provided by the unity integration and is not our product. The second game object is the representation of the player that is base on the OVRPlayerController from Oculus Unity integration. UserController a script attached to OVRPlayerController responsible for player movement. We added an empty virtual hands object as child of OVRPlayerController that contains itself RightHand and LeftHand. These three are responsible of everything that is in relation with the Razer Hydra or the hands animation.
Other game objects are generated by script at runtime. Typically Game Object that represent files and folders.

The most relevant C# files to look at in order to understand the project are:

`planetController.cs` Located on game object "controller". Generate planets and star clusters according to a given filesystem path. It is also responsible for the solar system movement and the file display

`UserController.cs` Located on game object "OVRPlayerController". Controls player movements.

`SixenseHand` Located on game object "OVRPlayerController/Hands/RightHand(LeftHand)". Animate the hand and detects controller movement corresponding to action.

`SicenseHandsController` Located on game object "OVRPlayerController/Hands". Handle some over movement that need both hands and correspond to action.

Once compiled, the `FileExplorerDirectToRift.exe` file can be launched to run the file explorer.

4 Discussion & Conclusion

During this project we learned a lot of thing regarding the field of VR itself and more generally concerning this kind of creative projects. Given the time this project was supposed to take we had to leave aside lots of improvements we discussed along the path, such as being able to launch executable applications, display images and videos, and so on. On the other hand we believe our code is ready to house these features since we made it in a clear and modulated fashion.
What was good about this project was not to have to stumble upon OpenGL issues and really being able to focus on the VR side of the project. On the other hand it was unclear how much time, effort and quality was expected, which has led us to quite a large workload.