Chris Baumler
ME/HCI 580
Final Project Report

Project Objective:

The project goal is to create a unique virtual reality experience allowing users to visualize folders on their hard drives. Users should be presented with an immersive 3D virtual environment that represents folders as physical objects with which they can interact. Possible interactions should include actions such as deleting folders, moving one folder inside another folder, and opening a folder to see the subfolders within it. The interface should seem natural and intuitive for the environment.

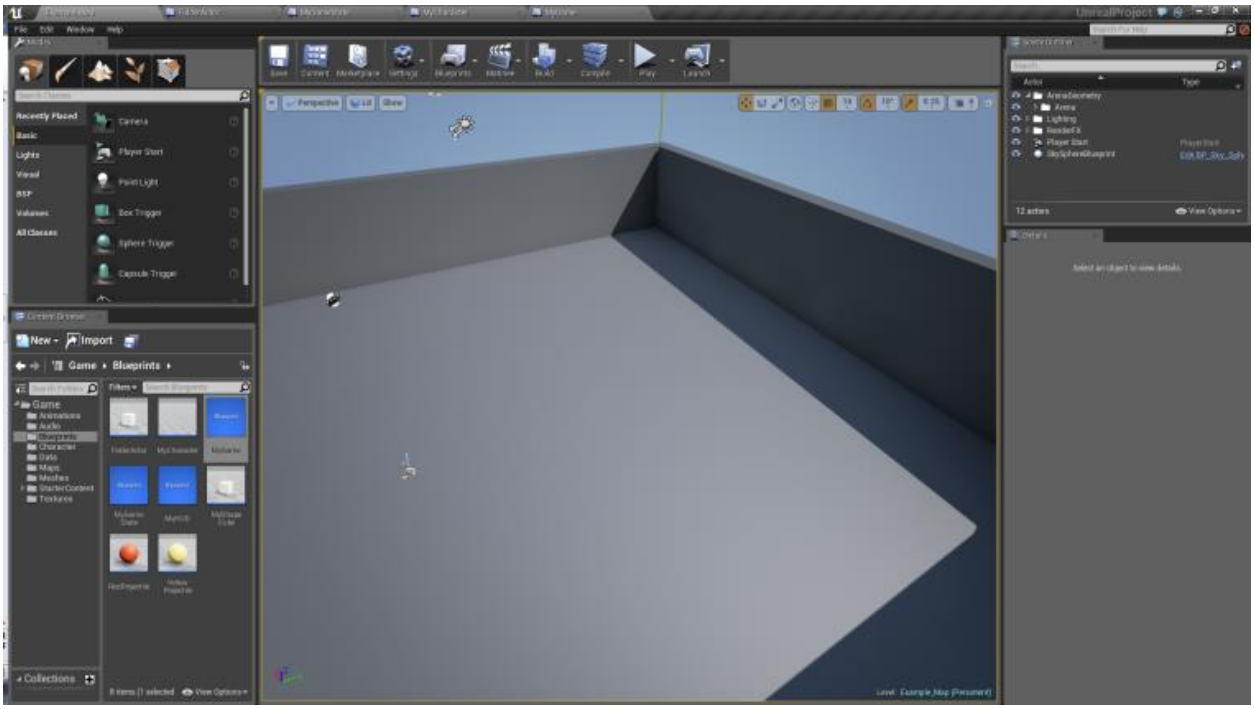
Hardware/Software:

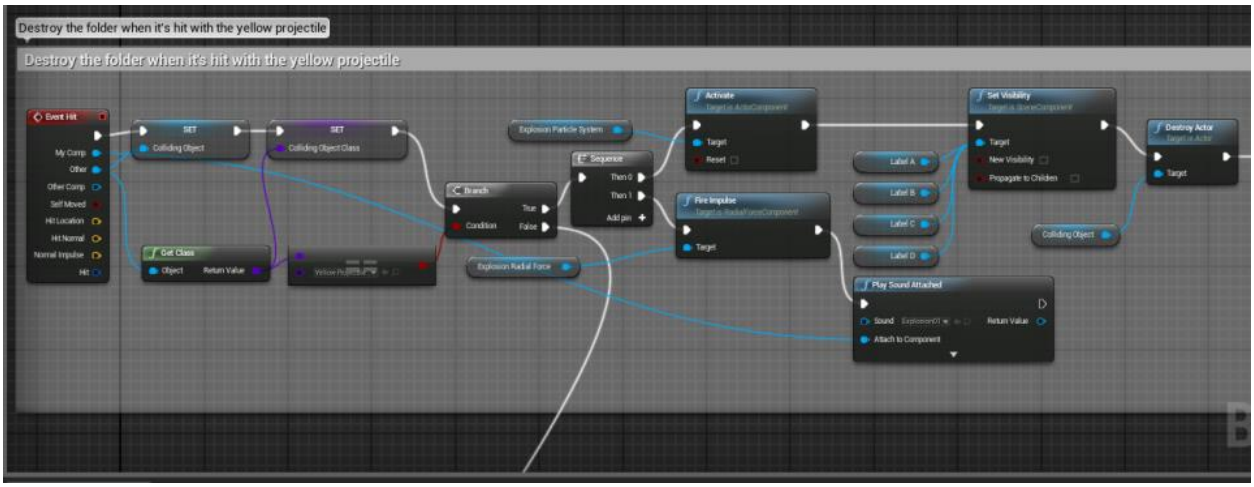
The hardware used for the project is the Oculus Rift Development Kit 2. This device is a head-mounted display (HMD) developed by Oculus VR. The HMD connects to a computer via HDMI to receive video input. The generated video input must be split-screen stereo and must be distorted inversely to the distortion caused by the HMD's lenses so that the lenses reverse the effect. This processing is accomplished with the help of the Oculus SDK. Within the HMD itself are the lenses which enhance the field of view for the device and also an electronic display that displays a different image to each of the user's eyes. Presenting the user with a different image for each eye results in the perception of a stereoscopic 3D image. A final component of the HMD is an infrared positional tracking system. This includes infrared sensors placed in the front of the HMD and a camera mounted a few feet from the user.

The software used for the project is Unreal Engine 4, a game engine developed by Epic Games. This engine has the Oculus SDK built into it, allowing for native support for the Oculus Rift HMD. The engine is written in C++, and program editing can be accomplished by coding or by using a graphical editor. The graphical editor is a powerful tool that allows users to create graphics content such as meshes, textures, and particle effects. It also provides a powerful visual programming system called Blueprints. This system allows program functionality to be created by placing visual components and connecting them together much like what is done in LabVIEW or Simulink. In addition to Blueprints, Unreal Engine 4 also has a built in physics engine which allows game objects to interact with the environment in a physically realistic manner. The major limitation of Unreal Engine is that functionality is geared toward game creation. Doing tasks not typically associated with a video game can prove much more difficult than game related tasks.

Final Product:

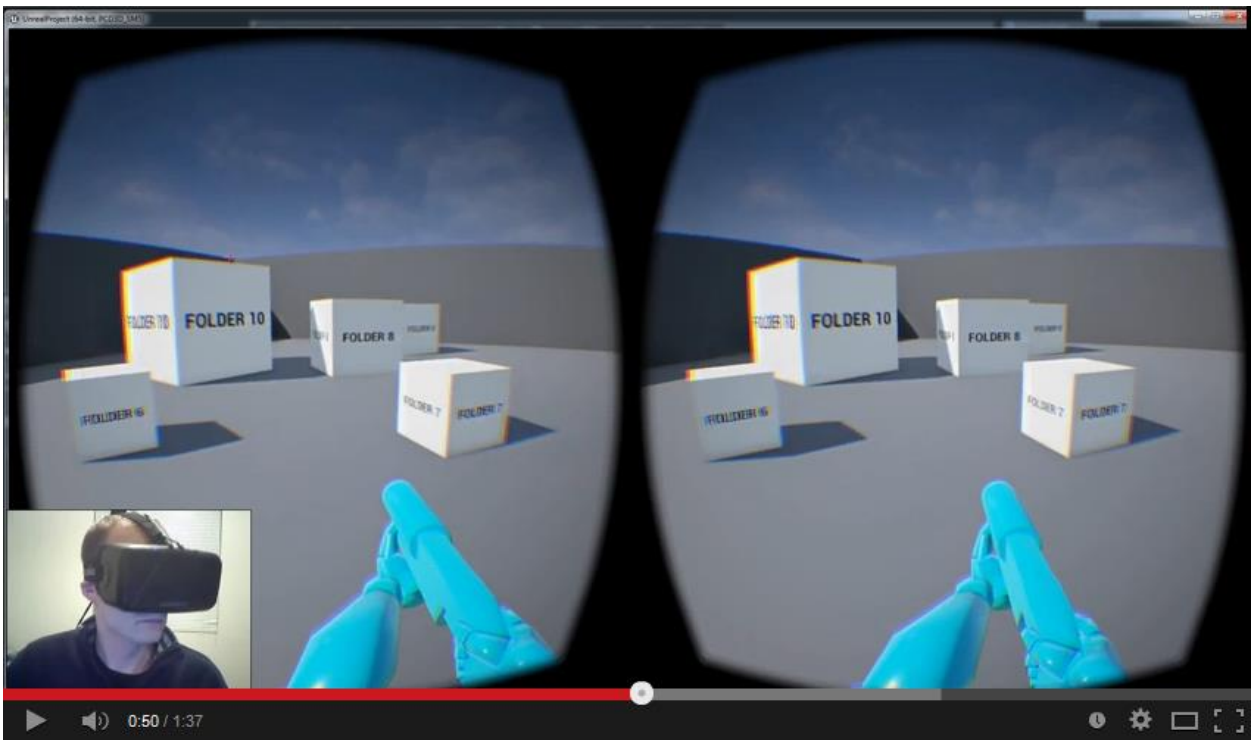
The functionality of the finished application is designed primarily using Blueprints. The exception to this is a custom data module that allows the program to read an external CSV file containing lists of folders. This component is programmed in C++ and is intended to allow the program to present actual folders from the user's hard drive. However, no program to automatically generate the CSV file is provided as part of this project. Focus has instead been placed on providing functions for user interaction. These interactions are designed using Blueprints. The following figures depict the Unreal Engine 4 graphical editing tool and an example Blueprint.





Within Unreal Engine, objects that perform functions and interact in the virtual world are known as actors. The actors created in this application include a player actor which includes a camera view, providing the user with visual input of the virtual world. Additionally, the player actor receives input from the keyboard and mouse for performing actions in the environment. Similarly to a video game, the mouse is used to look around the environment, and the keyboard is used to move and jump. Looking around can also be accomplished using the head tracking present within the HMD. Pressing the left mouse button causes the player actor to spawn a yellow projectile. Pressing the right mouse button causes the player actor to spawn a red projectile. Pressing the ‘E’ key on the keyboard while looking at a folder, causes the folder to be selected and highlighted yellow. Subsequently, looking at another folder and pressing the ‘E’ key again, moves the selected folder within the second folder, causing it to grow in size.

In addition to the player actor, there is a heads up display actor for displaying an aiming reticle, a game state actor for controlling game state information, a red projectile actor, a yellow projectile actor, and a folder actor. The game state actor is responsible for reading the CSV file and spawning the initial set of folders present in the environment. The folder actor contains the functionality of the folders, which are represented as boxes in the environment. This functionality includes changing color and scale, creating text labels, blowing the box up when it overlaps with the yellow projectile, and spawning the subfolders within the folder when the box overlaps with the yellow projectile. The following figure shows the virtual environment created by this application.



Relationship to Course Content:

This project presents a virtual environment in accord with the definition provided in class. The application presents a 3D computer simulation that senses the user's position and actions via head tracking and user input through a keyboard and mouse. Immersion in the virtual world is accomplished through the use of stereoscopic visual feedback from the HMD as well as audio feedback.

The methods used for interaction in the application are similar to methods discussed in the class text. For example, selection is done essentially by pointing, or more specifically, ray-casting. To select a folder to destroy or open, the user turns to face the folder and fires a projectile at it. This is similar to pointing with a virtual ray where the projectile is the ray. The process of selecting a folder to move is similar. The main difference is that instead of using a projectile, an invisible ray is used. To provide feedback that the item is selected, its color is changed.

Another task described in the class text is travel. This application allows the user to perform the exploration task by freely moving around in the environment through the use of keyboard input. In this

manner, the general layout of the folders on the hard drive may be ascertained. Additionally, the application allows users to perform the search task, locating specific folders in order to perform actions on them. Maneuvering is mainly accomplished by turning with the mouse or the HMD's head tracker.

Summary:

Before choosing this project, I first attempted another project that involved integrating Oculus Rift support into an open source data visualization program called ANTz that used OpenGL for graphics. This proved to be a difficult task, because existing examples of OpenGL software working with the Oculus SDK were sparse, and there were several different flavors of OpenGL. I discovered that performing the task would require intimate knowledge of OpenGL which I didn't possess.

In pivoting away from my first project idea, I wanted to choose something I knew would be well supported. I chose Unreal Engine 4, because it had built in support for the Oculus Rift and the available documentation and tutorial material was excellent. The main challenges I faced using Unreal Engine 4 were a result of trying to do complex tasks inside the Blueprints visual programming system. The tool was very powerful, but it had a lot of limitations, especially when trying to do tasks not commonly performed by games. For example, I had a very hard time reading in a CSV input file.

If I were to do the project again, I would start out using Unreal Engine to have more time to focus on learning the tool and implementing features. I would also delve deeper into the programming elements of the engine to overcome the limitations of the Blueprints system. Some features I would have liked to implement include writing a program to auto-generate the CSV file with folders from the user's hard drive, being able to physically pick up folders and move them in addition to selecting them via ray-casting, and being able to get back folders that were opened - they currently just disappear. I'm happy with my Unreal Engine experience overall and the functionality I was able to implement. I'm also very happy with my Oculus Rift experience. Other than some difficulty getting the video signal from Unreal Engine to actually go to the HMD consistently, everything I did with the HMD went smoothly. The head tracking system worked almost flawlessly.

Through completing this project, I primarily learned how the Oculus Rift HMD works and how to use it as well as how to use Unreal Engine 4. I also gained practical experience with designing and implementing virtual environments and their interfaces. Overall, I would say the project was a success, and I enjoyed doing it.