Oregon State University
College of Engineering

CS Capstone Technology Review

November 21, 2017

VR Painting Application

Mike Bailey _____ _____
Signature *Date*

Prepared by

Group 66
PolyVox

Chris Bakkom _____ _____
Signature *Date*

Braxton Cuneo _____ _____
Signature *Date*

Richard Cunard _____ _____
Signature *Date*

Abstract

This document will look at an analyze solutions for the VR painting application project. This project has many components and will involve a variety of different tools for its completion. For my personal portion of the project I will be focusing on the user interface tool kits, the virtual reality head mounted displays and the tracking solutions for out objects. We will be using a single game engine for this project and many of them have their own ways of constructing a menu or user interface. The display to the user will be important to find out which fits our game engine and custom solution. For tracking the brush, user interaction and other objects we might want to place into our scene I will be looking over different ways of recording objects in a physical space.

CONTENTS

1 User Interface

This project will be using a game engine to help produce the final product. The game engines we have chosen, based on popularity, documentation, and VR integration, are Unity, Unreal and the CryEngine. Each of these has their own way of creating a user interface. Important things to keep in mind is how they can be interacted with. Since we are using virtual reality, it will be important that these UI's be flexible enough to use the input from a wide variety of sources. All three of the game engines referenced in this section are free to use for our type of project.

1.1 Unity

Unity uses a custom game object inheritance to define it's user interface models. The objects created can have custom visuals and interaction. Each button in the toolkit has an onClick even, that can give custom commands. There is also support for all kinds of screen manipulation and custom animations. This will be useful in provide support for tools that may not be as precise. Unity has full support for built in VR user interfaces. There are clear definitions for non-dietetic UI, that can overlay menus that do not exists in the world. This will be very useful when a heads-up display for editing functions.

This infrastructure provides some easy support for what we are trying to do. It would be the optimal choice for minimizing the time we spend on creating the user interface. With the open integration of VR interfaces, the Unity engine is my top choice for our UI.

1.2 Unreal

While the Unreal Engine doesn't have direct support of a UI, it does have a supported toolkit layer called UMG that provides the necessary functions. This is similar to a UI toolkit like Qt or MFC. This layer integrates with Unreal Engine while having its own separate code for customizing widgets, animation and events. There are tools for non-dietetic UI and in world space UI, but current this is still experimental in version 4.8. Epic games is releasing their Dragonfly Project that will show how these features can be used, however, the best we can hope for at the moment is community driven development on integrating these features.

Because of the short comings in Unreal's user interface it is probably not the best choice. It requires knowledge about this abstraction layer of UI and more work to do because it is not directly integrated into the engine. However, it is still a viable option for a user interface and could be used if the Unreal engine provides a benefit above the other engines in another way.

1.3 CryEngine

The CryEngine takes a similar approach to Unity in having all of its accessible features as close to the game engine as possible. UI elements are located in the game sdk files and all C++ source code is readily available. CryEngine has already solved some of the problems with the previous game engines by defining a UIEntity as a game object. This can be placed either in world space or non world space. This allows for custom UI interfaces that can be intractable as well as event driven in the world itself or outside of it in a 2D menu system.

While the CryEngine does have some great flexibility in its user interface, the learning curve for the CryEngine is steeper than most models. If we do decide to use the CryEngine we will have support for any kind of interface we want. The would be the optimal choice if we are trying to define a UI that is very unique and difficult to implement in the other engines

2 VR Head Mounted Displays

This section looks at the different specifications for the head mounted display for our user. When using and HMD, it is important to note the available compatiblity with our game engine as well as accuracy of the tracking solution and performance. We are trying to create an editing environment so being able to display fully rendered environments will need to be carefully considered when judging game engine integration. The project defines a user tests to be conducted at the end so there needs to be a realistically high standard of accuracy for this type of display. Price, although not a prime goal of this project, will also be a consideration.

2.1 HTC Vive

The HTC Vive is one of the most popular and highly rated VR headsets on the market, priced with the tracking and controllers at six hundred dollars. It has 2160 x 1200 resolution at 90hz with 110 degree field of view. The tracking technology includes a gyroscope, accelerometer, front facing camera and lighthouse tracking solution. This tracking provides a space of 15 x 15 feet. The PC requirements include a NVIDIA GeForce GTX 970 or a Radeon 480 or better, Intel i5-4590 or better and 4GB of RAM. Vive has been working to integrate with Valve's Steam platform to create libraries for development. It has supporting plugins for all three of the game engines listed above.

Because of the Vive's already integrated tracking, this is the most optimal solution for fast development. The plugins are readily available for every game engine, it has a great resolution and high refresh rate. What really puts it ahead of the Oculus Rift is the larger volume, which would help for a more immerse editing environment. This allows the user a greater amount of freedom.

2.2 Oculus Rift

The Oculus Rift is priced at five hundred dollars with it's controllers. It has a resolution fo 2160 x 1200 at 90hz and a 110 degree field of view. It's tracking solution includes an accelerometer, gyroscope, and the Constellation camera tracking system providing up to an 8 x 8 feet environment. It requires an NVIDIA GeForce GTX 960 or a Radeon RX 470 or better, an Intel i3-6100 or better and 8GB of RAM. Oculus has been working to create a VR standard with WebVR and has support for many titles for its platform. It also has software plugins for all of the game engines listed above.

Where the Oculus Rift does well is its lower machine requirements. A lot of the benefits are similar to the Vive, but it is slightly cheaper to produce a machine to run it. However, these are minor benefits and not the purpose of our project. It has a huge down fall in that it drastically cuts down the viewing environment. We would need to implement another tracking solutions to improve this, which undermines the Oculus benefits. We would only use this if we are really having trouble getting a high powered computer.

2.3 Gear VR

The gear VR is one of the cheapest solutions at 100 dollars, that is with out the galaxy note required to power it ranging from 200 to 500 dollars. It has a 96 degree field of view and the resolution depends on the phone running it, but the minim is 2560 x 1440 running at 60hz. The lower frequency and resolution might make computation thresholds easier to work with in our application. Since the gear VR has similar SDKs to the Oculus, all of the supported software so far is mostly compatible between the two. This might be important for creating a modular system for integrating furtherer editing tools. There is the down side of the limited tracking in which the system would have to piggy back off of another tracking solution.

Because of the limited VR tracking for the Gear VR, this is probably not the best solution. If we were to implement the Gear VR, we would have to implement another tracking solutions that makes up for the lack of tracking in the Gear VR. This is do to the lack of position data given back by the Gear system. If we wanted to track only by motion capture, then we could consider this because of its low price and accessibility.

3 Object Tracking

This section will look into the different ways this project can capture physical objects and represent them digitally. Our user will be able, at the minimum, interact with a user interface and create brush strokes in a 3D environment. These interactions are definable as operations take in the physical environment and then transfered to the editor. This section focuses on the established controllers provided by Oculus and HTC, motion capture and the DodecaPen presented by Oculus research.

3.1 Integrated VR Tracking

The most obvious solution would be the controllers that the Oculus Rift and HTC Vive already provide. These controllers are tracked by the lighthouse and constellation tracking solutions. They also have their own analog and digital buttons that relay input to their respective SKDs. This would be the simplest solution because it would require much less compatibility modifications for the game engines. Because the SDKs already have the plugins established with the game engines, the digital inputs can already be accessed fairly easily.

Because this tracking solution is already available to us if we choose the Vive or Oculus, this is the most optimal for fast development. The support for this kind of tracking is already integrated into the VR headset and game engines. This should be our choice, unless we decide that we want to track other kinds of objects in the volume. Even if we want to to support other tracking solutions for other objects, we can layer these kind of features on top of the integrated VR tracking solution.

3.2 Motion Capture

Mocap or motion capture, is a tracking solution that requires cameras and a higher demand for volume preparation. It also requires a high computation standard and would likely need it's own system. It is also one of the most costly solution as a high end mocap system can cost more that 100,000 dollars. Despite the overhead, motion capture makes up for it by being on of the most accurate tracking solutions on the market. There are lots of options for tracking active and passive markers placed on the object and even HMDs, providing better head tracking that the integrated solutions. It also has the benefit of being enormously flexibly, allow us to capture and track any physical object we want. This could open up a variety of different features not limited to just the brush.

This is one of the most optimal tracking solutions as far as accuracy and flexibility. However, it requires huge amount of set up, access to a volume and it is very costly. We should only use if we already have access to all of these requirements and have time to integrate this kind of system.

3.3 DodecaPen

The DodecaPen is a 6 dof tracking solution, research by the Media IC and System Lab at the National Taiwan University and the Vision and Learning Lab at the University of California at Merced. It uses geometric hexagon solid with QR

codes to identify each face on the end of a pen to get 6 degree of freedom tracking. There has been some very promising results, stating high accuracy with only using a web cam costing less than 100 dollars to capture the solid. This research is still very experimental and there isn't a marketable solution yet. So integrating this piece of technology, despite the calculations being completely open, would not be an easy feature. It would require image processing and custom code to interpret the data. This could probably be a project on its own, but it is one of the most promising tracking solutions for our type of application to date.

This is great tracking solution because of its low cost and accuracy. The only downside to this is that we would have to figure out how to integrate it ourselves. The Dodecapen is still experimental and although the solutions is completely open, there is no available software for us to use. We would have to figure out how to capture frames, do image processing and then do the calculations for six degree of freedom for this object. This should only be used as a last resort if for whatever reason every other tracking solution fails or we have ample time to integrate such a complicated system.

4 References

Interaction Components Unity Technologies -
https://docs.unity3d.com/Manual/UIInteractionComponents.html

User Interfaces for VR
https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr

UMG UI Designer
https://docs.unrealengine.com/latest/INT/Engine/UMG/index.html

Creating 3D Widgets
https://docs.unrealengine.com/latest/INT/Engine/UMG/HowTo/Create3DWidgets/index.html

User Interface - Technical Documentation
http://docs.cryengine.com/display/SDKDOC4/User+Interface

Oculus Rift vs. HTC Vive: Prices are lower, but our favorite remains the same
Digital Staff - https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/

DodecaPen: Accurate 6DoF Tracking of a Passive Stylus
https://research.fb.com/publications/dodecapen-accurate-6dof-tracking-of-a-passive-stylus/