Tech Review for Noctilucent VR - Team 53

Taylor Fahlman, Joshua Bowen, Adam Puckette

November 14, 2016

Abstract

The goal is to create a virtual reality (VR) application that would allow users of most head-mounted virtual reality devices to view and manipulate point-cloud data. The solution will draw upon two existing codebases. First, the open source virtual reality platform (OSVR), which is a codebase designed with compatibility in mind. Second, the open source point-cloud visualization software CloudCompare, which is well-supported and handles multiple file formats. These two codebases will allow us to create modular and platform-independent software. This way, anyone with a virtual reality headset will be able to download and view point-cloud data without needing specialized hardware. In the solution, there will be a graphical user interface (GUI) which will allow the user to view and manipulate the point-cloud data from a first-person and third-person perspective.

1 Introduction

Taylor is in charge of the selection and implementation of the software component of the project. Regardless of the software used for viewing data, some framework needs to be used to integrate the Virtual Reality component. The three main frameworks in use today are the Open Source Virtual Reality framework, the Oculus Rift framework, and the HTC Vive framework. There also needs to be a framework for the UI. This is highly dependent on the software we select to view the data, but the benefits of each will still need to be explored.

Joshua is in charge of the selection of VR headset hardware, as well as hardware input methods. It is incredibly important that the hardware we use is effecient and coincides with the goals we set for our project. In this regard we must find a technology that meets our goals while is also isn't gonna break the bank of any potential users of our software. On top of the VR tech is also the interaction tech. It's incredibly important that the tech we use to interact with the VR enviroment is useful, easy to use with no sight, and isn't a money sink.

Adam is in charge of the selection of data analysis software the project will use. The point-cloud visualization software used is vitally important to the project. The software needs to have the performance to meet the demands imposed by VR, and must be compatable with multiple systems and file formats.

2 Virtual Reality Development Framework

2.1 Options

The three options for the VR development framework are Open Source Virtual Reality, the Oculus Rift SDK, and the HTC Vive SteamVR SDK.

2.2 Goals

The goal is for the framework is to enable as many users of VR as possible, as well as to work with the software we choose for the data analysis.

2.3 Criteria

- Cross-platform - Well supported - Supports software chosen - Extensible

2.4 OSVR

The Open Source Virtual Reality project has two major components: a headset (hardware) and a framework (software) to enable other software to use various headsets. Here, the focus in on the viability of the software component of OSVR. One big benefit to using the OSVR framework comes from the title; open source. This means that the source code of OSVR is open and free for anyone to use, change, and contribute to. Specifically for our project, this is helpful as there could arise some limitation or feature that is needed that other frameworks do not provide. In that case, writing our own code of top of OSVR, and possibly integrating it back into the core is easy to do. Another benefit is that it works with a variety of headsets, and it is supposedly easy to add support for new headsets. The biggest downside is that while there is an official headset for OSVR, it is still not the most robust framework. There are many improvements that can be made still, and the lower overall quality of this technology shows in demos of both the software and hardware. Specifically for the software, just getting a demo to work was complicated. The firmware also needed to be upgraded, which is a technical and long process, and there have been reports of this bricking various hardware devices. As well, it does not support, for example, the Oculus Rift as well as the Oculus SDK does. [1]

2.5 Oculus Rift SDK

The Oculus Rift SDK is a software development kit intended for the Oculus Rift headset. The main benefits of this software is that it is professionally made and supported, with a very high standard. It also is integratable with software such as the Unreal game engine, and the Unity game engine. It also has a large community to learn from and share with. However, there are several drawbacks. First of all, this is designed for gaming applications mostly. While most VR is as of today, there exists some limitations in the library for the kind of data analysis required of this project. The software that the Oculus already works with are not ones this project is considering, so this does not give the SDK an advantage per say. The largest drawback, however, is that this SDK is only targeted and provides official support for the Oculus Rift. Other headsets would be difficult or impossible to use

with the Oculus Rift SDK. Due to the constraints and scope of the Oculus Rift SDK, it does not seem like the best choice for the project. [2]

2.6 HTC Vive SteamVR SDK

The HTC Vive SteamVR SDK is also a software development kit, centered around the HTC Vive headset and SteamVR by Valve. The SDK is part of the larger HTC OpenSense SDK. The SDK is intended to be used with the HTC Vive headset, and with SteamVR, which is primarily focused at gaming. The SDK has many of the same advantages and disadvantages as the Oculus Rift SDK. It is professionally made and supported, has a relatively large community, and works out of the box with Steam, a very popular and widely used software distribution service. However, Steam almost exclusively distributes game software. While there are some utility and data analysis packages available, Steam and even more so SteamVR are focused on games. This again means that the library has some limitations for what the project is attempting. As well, it is aimed at only using the HTC Vive headset. Because of these constraints, this does not seem like the best choice for the project. [3]

2.7 Discussion

Based on the criteria, OSVR meets all of them. It is cross platform, both operating systmes, and various headsets. It also supports any software, and is infinitely extensible. The only criteria where the other two have an advantage is in well supported. This is not ot say that OSVR is not well supported, but it doesn't have the same level or quality of support the others do. However, it is reasonably well supported and beats the other two options in the other criteria. Therefore, this project with use OSVR.

3 UI Framework

The UI selection is dependent on the software that is selected to analysze the data. Still, the following analysis takes into account the pros and cons of the technologies independent of the software used, and will recommend one that is the best for the job, even if it does not match the software we select.

3.1 Options

The three options explored here for the UI framework will be Qt, an HTML5/CSS/Javascript combination, and GTK+.

3.2 Goals

The goal is to find an easy-to-use and effecient UI framework. The selection also needs to be able to work as an overlay in a virutal reality space, or be easily extensible if not.

3.3 Criteria

- Easy to use - Well supported - Can be used with VR.

3.4 Qt

Qt is a popular UI framework. It basic components comprise of a C++ API, a seperate library, Qt Quick, in a language called QML, and a full IDE for Qt development. It is designed to work with large C++ applications as a UI layer on top of logic of the software. The general idea is that the software is written from the ground up integrated with the C++ API, or built using C++ and then using Qt Quick or some other form of Qt on top of it. Qt has many benefits. Primarily, it is widely used, with a great deal of support, and a long history of use spanning about 20 years. It is very versatile and is considered a standard UI framework by many. In addition to the C++ API and the Qt Quick, it also has HTML5 support for web development. Another benefit is that its cross platform, ready for Windows, Mac OSX, iOS, Linux and Android, along with some other less used target platforms. The downsides of Qt stem from some of its benefits. It is a rather large and complex system, and can be heavy computationally. There is a lot going on in a Qt application, and in the case of working with a pre-existing codebase, it may be difficult to display the data without breaking another part. Also, while it is likely easy to integrate as an overlay with VR because it is based in C++, there is no known official VR overlay support. [4]

3.5 HTML5/CSS/Javascript

This combination of these web technologies is the standard of the modern web. This choice works well in the case we use a WebGL based software package. The benefits are versatility and near universal use. The shared online knowledge and resources for this choice are larger than arguably any other technology that could be considered here. The versatility of these technologies also mean that this selection is essentially unlimited in what it can do. There are many visualization tools built into Javascript in particular that would be useful. The main draw back is that with the size of the data this project will use, the limitations of these technologies would definitely show. There are known bottlenecks with complex graphics and these web techs. Also, there is no inherent support for overlay on a virtual reality HUD, so that will be an added challenge. And while combining these three technologies can be powerful, there is a possibility that with VR, the three will not go together well. [5]

3.6 GTK+

GTK+ is similar to Qt. It is a C/C++ based UI framework meant to integrate with pieces of software written in C++. It is part of the GNU Project, which means it is open source and free. It also is widely used and cross platform. However, there is no built in VR support. But this can be overcome with the fact that it is open source. There are not any significant differences from Qt other than the aformentioned fact that it is part of the GNU Project. [6]

3.7 Discussion

Assuming this section is completely independent, it looks as if GTK+ seems to be the best choice. It is relatively easy to use and decently supported. However, it beats the other two in terms of supporting VR. While not native, the fact the project is open source means that VR support can be added more easily than the other two. However, since the software selected uses Qt, and GTK+ and Qt are similar, our choice is Qt.

4 Virtual Reality Hardware

4.1 Options

The three options for VR hardware that I'll be looking at today are the Oculus DK2, Open Source Virtual Reality, and the HTC Vive.

4.2 Goals

Our goal with this review is to find a piece of hardware that isn't too bulky, has a good resolution, is comfortable to wear, and is affordable to the general public.

4.3 Criteria

As mentioned our goal is to find something that works well, is comfortable, and is affordable. A more strict criteria is something that is at least 720p ideally 1080p resolution. It should cost less than 500 Dollars, ideally even cheaper. Bulkiness, and comfort will not be a criteria as they aren't measurable, however if something has a pleathora of cables or appears actively uncomfortable that will be noted.

4.4 Evaluation

4.4.1 Oculus DK2

The DK2 is at this point an outdated piece of Oculus equipment however it is something we have access to and for development purposes would be sufficient. Current models of the Oculus Rift cost about 600 Dollars which is significantly over our ideal price range and might start becoming too expensive for the average user to want to invest in [7]. The website Rift Info indicates that the Oculus DK2 has a 1080p resolution as well as a 75Hz refresh rate [8]. Both of these meet our goals. Additionally Rift Info indicates that the DK2 weighs 440 grams or about a pound [8].

4.4.2 OSVR

Again I will be looking at a slightly outdated piece of hardware when it comes to the OSVR. The OSVR that we have on hand, and the one that I'll be looking at, is the OSVR Hacker Dev Kit 1.4. It has since been replaced with the Hacker Dev Kit 2, and the Dev Kit 2. Unlike with the Oculus DK2, however, the OSVR DK1.4 is still available for purchase. According to Sensics, the OSVR DK1.4 has a 1080p resolution and a refresh rate of 60Hz

[9]. The OSVR website has the cost of this device as being 300 Dollars [10]. OSVR also states that it should only need a GTX660 or AMD equivilant GPU to run [10].

4.4.3 HTC VIVE

Unlike with Oculus and OSVR, the HTC VIVE does not have an early developers version of the equipment available for purchase. The current model of the HTC VIVE listed on the website is 800 Dollars [11]. The Vive has a 2160 by 1200 resolution and a refresh rate of 90Hz [11]. While seemingly better than the other options this is about on par with production models of the other VR headsets [10][7].

4.5 Discussion

In general all of the systems are very comprable, with refresh rates, and resolutions being very similair. The HTC VIVE would appear to have superior resolution and refresh rate, but this is because it's a public model. As I mentioned about the more recent OSVR and the public release Oculus both have 2160 by 1200 resolution and higher refreshrates than I listed [7][10]. So the main difference comes down to pricing. Because the DK2 is no longer available you have to look at the public release version of the oculus, which is noticbly more expensive than the OSVR but also has better stuff. The price is double that of the OSVR DK1.4. Both, however, are still cheaper than the VIVE, however the VIVE does come with added peripheral hardware. The average user of our software is honestly not going to need anything better than the OSVR provides, at a price that's way more reasonable to the average consumer. If you were however looking for something a little more powerful, either the VIVE or Oculus would be superior. Which of those two are the better option come down to whether you are interested in the peripheral or not, which likely won't have any added bonus to our software.

4.6 Decision

For the purpose of our project the OSVR will be more than sufficient. It's refresh rate and resolution make it sufficient for it's needs. It's cheap cost also means it'll be accesible to people who are looking to use VR purely for this purpose.

5 Peripheral Hardware

5.1 Options

The three options for Interaction hardware that I'll be looking at are the Wiimote, the Leapmotion, and HTC vive controller.

5.2 Goals

Our goal with this review is to find a piece of hardware that is responsive, and has a wide range of usable functions. The idea hardware is something that will allow the user to easily navigate the virtual space without feeling like the device isn't doing what they want, or is difficult to use.

5.3 Criteria

In accordance with our goal something that meets our requirements is as follows. Costs less than 100 Dollars. Allows for free range of movement. Has a number of easy to identify buttons that can be discerned without removing the headwear.

5.4 Evaluation

5.4.1 Wiimote

According to Wiibrew, a website documenting the hardware and software tech used inside of the wii home console, the wiimote is a blue tooth device [12]. Essentially this means that the device should easily hook up to a computer through bluetooth. The issue comes that you'll need to manually install wiimote drivers otherwise the wiimote won't be much use [12]. The step of installing the wiidrivers likely isn't difficult, but it is an extra step that could confuse the process. A wiimote has 11 buttons that can be used for inputs, and then additionally a sync button and a power button which likely won't be usable [12]. The wiimote also has a built in accelerometer that tracks the position and angle of the wiimote [12]. Additionally, there is a built in Infered Camera that can pick up the position of the wiimote [12]. A wiimote with wii motion plus costs 20 Dollars on Amazon.

5.4.2 Leap Motion

The Leapmotion is a piece of technology in which the user's hand movements are translated into a digital environment. The technology itself is still in development so it's not perfect just yet. Pros of using Leap Motion would be that it doesn't require the user to hold a device while in VR. A considerable downside is a lack of buttons which means all actions would need to be performed with gestures. While it could mean actions like zomming in and selecting things are easy it seems like navigating a ui or more complex tasks wouldn't be. While a fun piece of technology I can't see it being very practical for our needs. A leap motion controller sells for 80 Dollars [13].

5.4.3 VIVE Controller

As far as pure tech power the VIVE controller is probably the best option. The Controller comes not only with it's 24 sensors for tracking, but a trackpad for even more precise controlling [11]. The VIVE also has a number buttons with dual stage feedback allowing for more precise input [11]. If we are looking for something that's going to get the job done and done very well the VIVE Controller seems like a great option. The main limitatioin with the VIVE is that it requires purchasing the whole package of the HTC VIVE which is a whopping 800 Dollars [11].

6 Discussion

Of the three options, the VIVE Controller is by far the superior technology, it's feedback systems, precision, and input, are exactly what we need. The main issue of course rolls down on the fact that it is incredible expensive and therefore limits the people who could use our software. As our goal is to make a software as freely usable by people as possible it's counter productive to choose something as niche as the VIVE. This leaves us with the options of the Wiimote or the Leap Motion. I beleive that while the wiimote option requires a lot of extra work tog et working it provides a lot of what we need at an affordable price. The Leap Motion unforutnately just doesn't have the input and precision that we need even if it is more of a plug and play product.

7 Decision

For the purposes of our project, and with the goal of making our software as publically accessible as possible, the wiimote is the best option. However, if don't properly we can potentially design our software in such a way as to allow for VIVE Controller use by those to which that is accessible.

8 Point-Cloud Visualization Software

I (Adam Puckette) will be responsible for the point-cloud visualization software that we will use to render our point-cloud data. Our choice of software will depend on its ease of integration, openness, file format compatibility, and performance. This software is an essential component of the project, as we do not have the time or resources to write our own solution.

8.1 Options

CloudCompare is the software that was first recommended to us by our client, but we are considering two alternatives: LidarViewer and Potree Viewer. All are open-source projects that approach point-cloud visualization in very different ways. LidarViewer is the software that the current 3D TV solution uses. Unfortunately, there is no Windows version of this software. Finally, Potree Viewer is a WebGL-based project that can host the actual file remotely and uses a unique file format.

8.2 Goals

Our goals in this evaluation are as follows. One, to isolate potential points of failure in the proposed solutions. Two, to discover the most efficient solution to implement. Three, to ensure that the software used will meet overall project goals such as openness and portability. If this comparison helps us to decide which solution to use, then it will have accomplished its goals.

8.3 Criteria

The essential differences to be evaluated are as follows. One, supported Operating Systems. Ideally, the software should work on the Windows OS. Two, supported file formats. For greater ease of use and adaptability, the software should accept a wider variety of file formats as input. In short, adaptability is our main concern.

8.4 Evaluation

8.4.1 CloudCompare

CloudCompare is well-supported and widely used software that runs on Windows, Mac OS, and Linux [14]. It has a large number of built-in tools for evaluation and manipulation of point-cloud data, and accepts over 15 file types, including BIN, ASCII, PLY, OBJ, VTK, STL, E57, LAS, PCD, FBX, SHP, OFF, PTX, FLS/FWS, and DP. In addition, the software supports third-party plugins, something that we could use to create a VR mode.

8.4.2 PotreeViewer

Potree Viewer is a relatively recently developed project that uses WebGL to view point-cloud data [15]. It is Windows-only for the moment, and only works with a specialized file format called octree. This file format stores point-cloud data as a single sparse root node with a directory structure of child nodes beneath it. Potree Viewer has fewer built-in tools for point-cloud manipulation, but may perform better than CloudCompare on similar hardware.

8.4.3 LidarViewer

LidarViewer is the existing software used for the 3D TV setup. It has the advantage of already being designed for VR, but is hampered by being Linux-only [16]. In the absence of other options, LidarViewer would be acceptable.

8.5 Discussion

At first glance, CloudCompare is the obvious choice. Not only does it support Windows, Mac OS, and Linux, but it also accepts a minimum of 15 different file types. Our only concern is whether it can hold up to the rigorous standards required by VR headsets. Potree Viewer is an attractive alternative, but is slightly more limited in its current state. It is Windows-only, which is fine for our purposes, and can be set up to store files remotely on a server. It only accepts a single file format, called octree, which stores point-cloud data as a single sparse root node with a directory structure of child nodes beneath it. Potree Viewer would suit our needs, but may not be as adaptable as CloudCompare. Finally, we have LidarViewer. LidarViewer is Linux-only, and so would not be ideal for our project. However, it has the advantage of already having a VR framework (VRUI) to work with.

8.6 Decision

As CloudCompare is the most adaptable and well-documented point-cloud visualization solution available, we plan to use it for our project. Potree viewer may become a more viable option in the near future, but unless we encounter a good reason to switch to it, we will focus our efforts on CloudCompare.

References

[1] O. S. V. Reality, "Osvr."

[2] Facebook, "Oculus rift sdk."

[3] HTC, "Htc vive steamvr sdk."

[4] Qt, "Qt."

[5] Various, "The web standards model."

[6] G. Team, "Gtk+."

[7] O. V. LLC, "Oculus rift," 2016.

[8] R. Info, "Oculus rift specs - dk1 vs dk2 comparison."

[9] Sensics, "Osvr hdk hardware."

[10] O. S. V. Reality, "Osvr."

[11] H. Corporation, "Vive."

[12] Wiibrew, "Wiimote."

[13] L. M. Inc, "Leap motion."

[14] Various, "Cloudcompare readme."

[15] Various, "Github potree documentation."

[16] P. Gold and C. Bowles, "Lidar viewer manual."