CREATING SOCIAL VIRTUAL REALITY IN CAMPUS ENVIRONMENTS

By

Daniel Bina Handojo

A capstone project submitted for
Graduation with University Honors

May 23, 2017

University Honors
University of California, Riverside

APPROVED

_____
Dr. Jiasi Chen
Department of Computer Science and Engineering

_____
Dr. Richard Cardullo, Howard H Hays Jr. Chair and Faculty Director, University Honors
Interim Vice Provost, Undergraduate Education

Abstract

This paper explains the programming details, development history, and planned features for VR'Tour, my campus tour application for UCR created in virtual reality. VR'Tour uses 360° videos and online features to emulate a real campus tour experience. You can join groups of people who are taking a tour and listen to a live tour guide talk about UCR. You can even ask the tour guide questions and talk with other people on the tour. After one year of development, a demo version of the application is complete and it features videos that simulate a walk from the campus store to the bell tower. Further details about the demo, including screenshots and pictures of people using it, are provided in this paper. VR'Tour's development had two key phases: handling 360° videos and programming the application. I mostly recorded videos in the beginning of the project with the help of several students from my research team. Once all of the videos were recorded, I focused on programming VR'Tour. I did this task on my own, but I worked with Dr. Jiasi Chen, my mentor for this project, and my teammates to design the features. There are many features we discussed that have not been implemented yet. For example, we want small informational notes to be displayed when people look at places of interest like the bookstore. I may continue developing this application in the future to complete these features and perhaps even publish the application for the public. Ideally, I may someday share a full version of VR'Tour with the UCR Campus Tours office.

Acknowledgment

Special thanks to Dr. Chen for agreeing to act as mentor for this project back in February last year. The commitment, resources, and time she gave were precious and invaluable. Thanks also to J. Pham for his initial part in starting the project. Thanks to C. Verdegan and M. Morelos for their programming efforts in Unity. J. Kaur, Z. Wang, and C. Yuan spent countless hours on synchronizing and stitching raw GoPro footage, which was amazing of them. Lastly, a very special thanks to Jesus, or rather God. His strength and patience brought this project from beginning to conclusion and made this project possible.

Table of Contents

Introduction

When I first considered working with virtual reality, I initially intended to create a social media application for Google Cardboard. The majority of free Google Cardboard applications on the Android Play Store are of very low quality, have unwieldy user interfaces, and do not have multiplayer functionality to enjoy with friends. My goal was to tackle this with an idea inspired from an xkcd comic that depicts the author trying to understand the size of clouds [1]. Users of the social media application would all be placed into a digital and scaled-down representation of the earth. The smaller size of the digital earth would give people the sensation of being giants. They would be able to see users in other countries over the ocean, for example. Virtual reality was, and still is, a very young medium and so no major social applications had been developed yet to my knowledge. Furthermore, I had enormous creative freedom for this project because of University Honors' multidisciplinary nature and the creative potential of virtual reality.

The idea of a campus tour application came from my mentor, Dr. Jiasi Chen. She too was excited about the recent rise of virtual reality, so she agreed immediately to work with me. Her proposal, which is included as an appendix, was to create a "software platform [for] easy content creation of VR tours, using the UCR campus tour as an example application". Tours would be composed of panoramic pictures, 360° videos, and audio commentary. The platform would contain a simple method to create navigable paths made of panoramic pictures in virtual reality. The campus tour application was meant to be a proof of concept and a secondary, equally important project. We actually discussed the campus tour more often in my meetings with Dr. Chen however than the platform.

We also discussed what my Capstone project would ultimately be. Initially, I had no intention of creating a campus tour application. It was discouraging to see how different my original idea was from Dr. Chen's idea. Furthermore, I found the idea of a virtual tour uninteresting because many virtual tours had been made for businesses already. We discussed our own ideas at length until we eventually began talking about the lack of multiplayer features in virtual reality. This led us to the idea of allowing multiple people to participate in a single tour, which became the essential idea for VR'Tour.

Intentions for VR'Tour

I wanted to create a virtual reality application for Google Cardboard that includes social functionality. As I previously mentioned, there were very few social applications in virtual reality when I first started this project. There are more in development today, but many of these are restricted to premium virtual reality devices such as Oculus Rift or HTC Vive [2, 3, 4]. The challenge of incorporating social functionality into a Google Cardboard application is creating a useable user interface [5]. Google Cardboard only has one button on its side for developers to work with and requires users to gaze at menu options within virtual reality. VR'Tour solves this by presenting the menu options to users before entering virtual reality.

My other goal was to make virtual tours more exciting. Most virtual tours [6, 7, 8], including the one offered by UCR's Campus Tours Offices [9], use panoramic pictures. These pictures are beautiful but they often lack people, making the area feel empty and lifeless. Another issue is that virtual tours are typically silent and only present

tour information in text. Harvard's virtual tour tackles this issue by including a video recording of a student tour guide with each picture.

VR'Tour improves the virtual tour experience with 360° videos and voice chat. Multiple people who have tried the demo ask if the videos are live, demonstrating that using videos over panoramic pictures bring a sense of life to the campus. The voice chat feature adds an element of interactivity that Harvard's solution lacked by allowing people to speak with their tour guide during a virtual tour. It also adds a social element to the virtual tour experience by allowing tour participants to speak with each other.

In this paper, I discuss all of the features included in the demo for VR'Tour, along with my development process and research challenges. Before I discuss the demo's specifications, I first want to state that the demo is a prototype for VR'Tour. I created it as a proof of concept for the sake of the Capstone project and my research. You can look at the Unity project on my GitHub page here [10].

General Specifications for the Demo of VR'Tour

Demo Features and Libraries

The demo for VR'Tour is a functional campus tour application that includes the essential features planned for the full version. It allows you to create new tours, join tours, and use voice chat to communicate. The creator of the tour can change the video that everybody in a tour sees, taking them from location to location. Also, as you watch the 360° video, you can see where other people viewing the video are looking. This allows you and your friends to concentrate on individual items of interest in the tour to discuss them. The demo consists of six videos recorded at several points between the

campus store and the bell tower. You may view some screenshots of these videos in Figs. 1-2.

Figure 1—Demo screenshot of campus store

Figure 2—Demo screenshot of UCR path

All of the demo's online capabilities are provided by Photon, a "real-time multiplayer game development framework" [11]. This includes voice chat, which I will discuss later, and multiplayer networking. Photon defines two constructs for multiplayer that I use in the demo, lobbies and rooms. A lobby is a list of rooms that users can access before selecting a room. It is a useful construct for organizing and separating rooms, but the demo only uses one lobby, the default one. A room is a grouping of players who can interact with each other. Rooms are given a unique name when they are created so that

players may discover and join them. In the demo, every tour occurs in a room and is hosted by the room's creator.

Virtual reality functionality is provided by Google Cardboard's software development kit (SDK). One key feature is *stereo rendering*, where the phone screen is divided into two display windows, one for each eye [12]. This can be seen in Fig. 1. Another integral feature is *head tracking*, which adjusts where you look in a 360° video according to the rotation of your Android device [12]. The SDK achieves this by retrieving data from sensors on your smartphone, such as the gyroscope and accelerometer [13].

The demo's 360° video playback feature is provided by Easy Movie Texture, a video texture asset developed by JaeYunLee [14]. Unity defines a texture as an "image file" [15]. Textures are most commonly applied onto materials, which are assets that "control the appearance of a GameObject", to give it a "textured surface" at a low computational cost [16]. To play a video on the surface of a GameObject, such as a sphere or cube, you need to use a texture that loads the frames of the video. Unity's built-in MovieTexture class does not support Android or iOS at this time [17], which is why I rely on a third party asset. EasyMovieTexture is applied to the material of a sphere in order to play videos on its internal surface. When you use the demo, you are placed inside the sphere at its center to view videos in 360°.

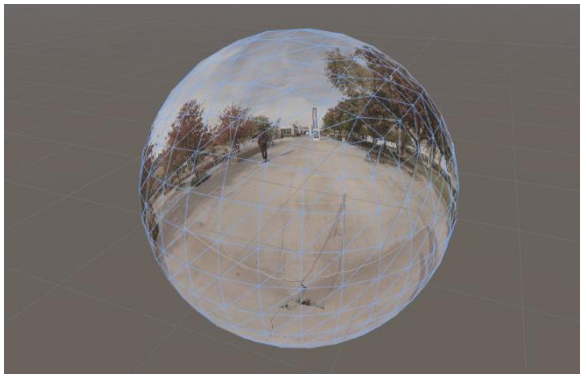

Figure 3—A digital sphere, displayed in Unity's editor, that is using EasyMovieTexture as its texture to play a video on its surface.

Creating and Joining Rooms

The demo first begins with a menu that allows you to create or join a room. There is a sprite image that indicates if you are connected to the lobby. You see a red x when you are disconnected and a green check mark when you are connected. Connecting to the lobby is an automatic process that occurs when the demo first starts. The demo must be connected to the lobby in order to create or join rooms. If it fails to connect, you can resolve this by restarting the demo to try again.

If you choose to create a room, you will be instructed to assign a name to your new room. The name you pick must be unique relative to the names of the other rooms in the lobby. After you submit your room name, Photon attempts to create a new room. If it succeeds, you will see a list of buttons labeled with “Bourns” and a number from 28 to 33, such as “Bourns28”. These correspond to the six videos available in the demo. The room creator is in charge of determining which videos everybody watches, so I refer to the list of buttons as the *tour guide menu*. I include a diagram of this whole process in Fig. 4.

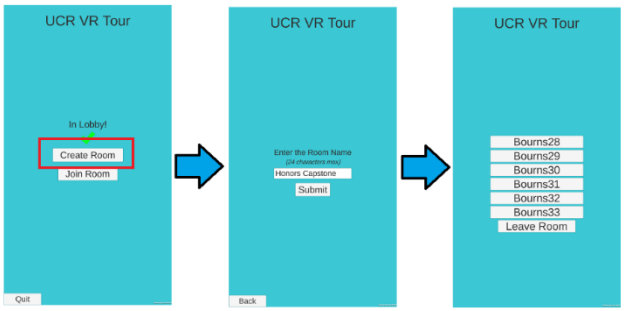

Figure 4—How to create a room in the VR'Tour demo

If you choose to join a room, a list of available rooms will be displayed. You can join the room of your choice by pressing it from the list. A button labeled “Start Tour” will appear if Photon successfully adds you to the room. Note that this end result differs from what occurs when you create a room. After you start the tour, the demo application switches to stereo rendering to let you view 360° videos of the UCR campus through a virtual reality headset. The process for joining a room is depicted in Fig. 5.

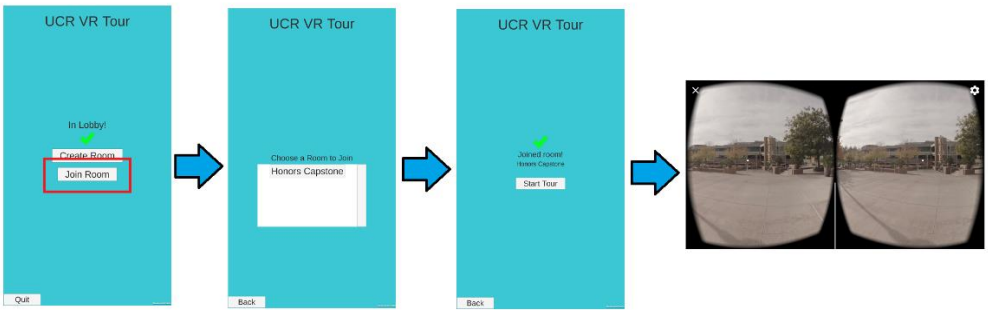

Figure 5—Example of joining a room in the VR'Tour demo

The framework established here for joining and creating rooms is meant to mimic aspects of a real-life campus tour. A tour guide hosts a tour by creating a room and is given the tour guide menu to take participants from location to location. People who wish to take a tour can follow a tour guide by joining a room. There are two limitations to this framework that are worth noting however.

The first is that, for the sake of the demo, participants in a tour are only able to view the video that the tour guide selects. There is no way for a tour participant to explore independently. The second limitation, which can be seen in Fig. 4, is that there is no way to view a selected video from the tour guide menu. The application is designed in such a way that the menu is in front of the video. This typically is not a problem when I present the demo at events because I use my computer to run the application in Unity. Unity has a "scene" window that lets you inspect every object in the digital environment while the demo is running. By using the scene window, I can view the video behind the tour guide menu.

Networking Foundations for the Tour

The virtual reality tour begins after you create or join a room. However, I should first explain the general architecture of a networked Unity application. It will be fundamental to understanding later features like avatars and voice chat.

In a networked application, each user in a room has his or her own instance of every present digital object [18]. For example, let us suppose that you and I are on a tour together using two separate smartphones. Both of us have a sphere that we view 360° videos on, but they are not the same sphere. They are two separate instances, one on your device and one on mine. Thus they do not necessarily have the same properties even though we are on the same tour.

Figure 6—There are two instances of objects 1 and 2, one for each user in a room [19].

In Unity, you can synchronize properties such as position, height, or rotation among every instance of an object if they all have a NetworkView [20, 21]. A NetworkView is a class that sends messages about its associated object's properties to every other instance of its object. Continuing from our previous example, say my sphere increases in size by 50%. If both of our spheres have a NetworkView, then your sphere will receive a message about my sphere's change in size and grow accordingly. However, the NetworkView has to be set to observe the object's *scale* property before it can send the message.

NetworkViews also allow you make remote procedure calls (RPCs) on multiple devices all at once. An RPC is a function that can be called on remote machines, such as the devices of other people in a room [22]. Calling RPCs on multiple devices is very similar to synchronizing properties. The process first starts by calling an RPC through an object with a NetworkView. The NetworkView sends a message about the RPC to the object's other instances, which then execute the RPC's code. This is very useful for when you want something to occur for every user in a room, such as giving everyone a room-wide notification message.

The demo uses Photon's networking library, not Unity's, but the concepts described above still apply. Photon intentionally designed its library to resemble Unity's and many of its classes are nearly exact parallels [23]. For example, the Photon equivalent of a NetworkView is PhotonView [24]. The reason I use Photon's library over Unity's built-in library is to use Photon's voice chat feature.

Avatars and Relevant Tour Features

When the virtual reality tour begins, you automatically instantiate a digital avatar object. An instance of your avatar is generated for every person in the room. This object, for the sake of the demo, is a simple blue sphere with visors. Avatars are placed in the center of the sphere that plays 360° videos, where you are placed also. In Unity, you cannot see objects that are in the same position as you, so you do not see avatars during the demo. However, they enable us to have certain features because they all have PhotonViews.

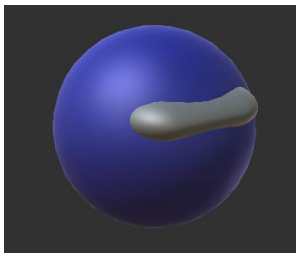

Figure 7—The avatar that is instantiated for every person in a room

One such feature, that lets you see where other people are looking, relies on synchronizing an avatar's rotation property. Your generated avatar rotates according to the rotation of your smartphone so that it faces the direction that you face. Likewise, everyone else's avatar rotates according to the rotation of their smartphone. All of these changes are synchronized between instances. If you were able to see the avatars, you would be able to tell where everyone is looking in the tour. However, as I mentioned before, every avatar is placed in a blind spot.

To overcome this, I placed a blue square object in front of every avatar that moves around to follow their corresponding avatar's rotation. As you can see in Fig. 8, the blue square is visible to the participants in a room so they can see where other people are

looking. The one that corresponds to your avatar is invisible to you so that it does not obstruct your view. To save network bandwidth, the blue square does not have a PhotonView. It is instantiated as a local object and its position is determined by the rotation of its corresponding avatar rather than by a synchronized position property.

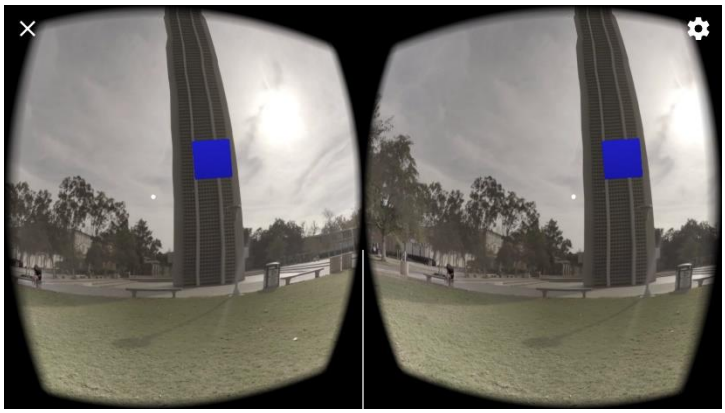

Figure 8—The blue square indicates where other people in the tour are looking.

Another feature that relies on PhotonView is the tour guide menu, which makes use of RPCs. Every person in a room has an avatar, including the tour guide. The tour guide menu uses the PhotonView attached to the tour guide's avatar to send RPCs to every other device in the room. The tour guide menu uses an RPC that changes the video that is currently playing in the tour. This is how the tour guide controls which video is currently playing in the tour.

Four out of the five classes needed to use Photon's voice chat feature, including PhotonView, are attached to avatars [25]. This includes PhotonVoiceRecorder, PhotonVoiceSpeaker, AudioSource, and PhotonView. PhotonVoiceRecorder records audio from the person who owns the avatar it is attached to. PhotonVoiceSpeaker receives remote audio data from other people in the room and plays it on your device.

AudioSource is a Unity class that handles playing sound clips. PhotonView sends and receives audio data between instances of objects, similar to how it synchronizes properties.

The fifth class, PhotonVoiceSettings, handles settings for Photon's voice chat feature [25]. The main two settings that affect the demo are AutoConnect and AutoTransmit. Photon separates multiplayer interaction and voice chat into two room types. You enter the first type of room when you first start the demo. You enter the second type of room, a "voice room", automatically and simultaneously if AutoConnect is turned on. Once you join a voice room, your device immediately starts recording audio if AutoTransmit is turned on. In the demo, both of these settings are turned on so that the voice chat feature works without any user interaction.

The Virtual Reality Tour

The demo's tour consists of six videos recorded on the path from the bookstore to the bell tower. The first video in the tour, named Bourns28, begins playing automatically after you join a room. The other videos are played when they are selected from the tour guide menu. When a video completes, EasyMovieTexture is set to loop them to maintain a sense of liveness in the tour. Each video has a resolution of 3840 by 1920, nearly 4K [26], to avoid pixelation from being played in 360° on a sphere. I store the videos in local memory and their combined size is 702 MB.

In Bourns28, if you look at the bookstore sign, you will see a red circle and red square containing information about the bookstore (See Fig. 9). This object is called an *annotation*. For VR'Tour, I want to create a number of annotations for key locations in

UCR. In the demo however, I only have one annotation at the bookstore as a proof of concept.

Figure 9—Annotation of the UCR bookstore

To summarize, the virtual tour component of the demo mainly consists of 360° videos of UCR and voice chat. Participants can discuss the videos they watch together and ask the tour guide questions about UCR. They can also point at specific locations in the tour through the feature that lets you see where other people are looking. The tour guide takes participants from location to location and can share interesting facts about UCR with them. The demo gives students a way to explore UCR together in an immersive tour experience. Its 360° high-resolution videos give the campus a sense of life and voice chat makes the tour more interesting by turning it into a social experience.

Process for Creating 360° Videos

For recording, my research team used six GoPro Hero 4 cameras, inserted into a Freedom 360° GoPro mount. The GoPro mount is shaped like a cube and each of its sides has a slot where a GoPro is inserted. We also used a GoPro remote that connects via wi-fi to our six cameras. The GoPro remote enables us to start and stop recording from all six

cameras simultaneously. The recording process was straightforward, but processing the videos to use them in virtual reality was complicated. We first needed to *synchronize* the six videos by audio, then *stitch* them together into a single video.

Synchronizing Large Sets of Videos Efficiently

Our research team needed to find an efficient way to synchronize videos because we recorded a lot of videos for the tour. When we record videos, the GoPros do not start or finish recording at the exact same moment even though we have a GoPro remote. If we play all six videos at once, we will hear things like words repeating and noises overlapping. Synchronizing the videos involves shifting when they start and end so that we hear a single clear recording if we play all of them at once.

We initially synchronized videos manually using VSDC Free Video Editor, which was extremely time-consuming. This process involved comparing the audio of two videos at a time and adjusting their start times by units of milliseconds. Each 360° video has six GoPro videos, so we were doing this process five times per 360° video. Being off by a few milliseconds on any one of the six videos could ruin the stitching process, so we had to be very thorough with each video.

Dr. Chen advised us to use Adobe Premiere Pro to automatically synchronize all of our videos at once. This dramatically sped up our work because Adobe Premiere can synchronize multiple videos at once in less than 10 seconds. Once we complete the synchronization, we need to process and save the synchronized files from Adobe Premiere. There is a backlog feature however in Adobe Media Encoder that lets us process these videos in the background. This allowed us to synchronize large sets of videos at a time without waiting for videos to process.

Stitching Videos Together and Stitching Issues

Once the videos were synchronized, we stitched them together using Kolor Autopano Video Pro. Autopano Video automatically stitches video files based on algorithms that attempt to determine how videos should blend together to make a complete scene. This can be likened to completing a jigsaw puzzle, where each puzzle piece is one of our videos. We can make minor adjustments to the scene, like straightening out horizon lines, and try to improve the video blending before saving the result as our 360° video.

Autopano Video is usually successful at blending, but there were cases in our research when it failed to stitch videos together properly. This was caused by the location we recorded at. Indoor places with walls that were a single color or with a lot of desks caused the algorithm to fail, producing images like what you see in Fig. #. In less severe cases, simply rerunning the blending process could sometimes fix the issue. In the most severe case, we

Autopano Video is usually successful at blending, but there were cases in our research when it failed to stitch videos together properly. This was caused by the location we recorded at. Indoor places with walls that were a single color or with a lot of desks caused the algorithm to fail, producing images like what you see in Fig. 10. In less severe cases, simply rerunning the blending process typically fixed the issue. In the most severe case, we used a template that defined where each camera's view should be placed in the final scene. In other words, the template defined where our six "puzzle pieces" should go since Autopano Video could not determine this automatically. We obtained this template from videos we finished that were successfully stitched together.

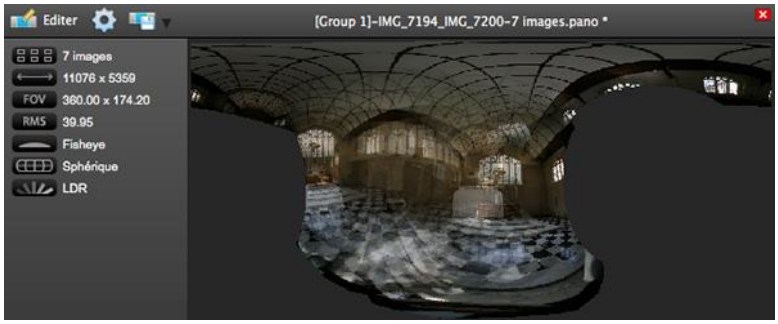

Figure 10—A case where Autopano Video fails to blend 7 images together into a single panoramic picture [27].

Now we had a complete scene, but videos were still in the wrong place in the scene. The issue was that we had the correct locations for each video, thanks to our template, but not the correct mapping of videos to locations. For example, a video of the floor could be placed where the ceiling should be. We were able to fix this manually by swapping videos until they were mapped to the correct locations. From there, we only needed to do minor blending adjustments to finish stitching the 360° video.

Developing in Virtual Reality

Initially, I had very little understanding of how to develop applications for Google Cardboard. Much of my confidence in my ability to succeed came from my year-long internship in which I developed an Android application for the company Morning Sign Out. My intention was to develop a virtual reality application for Android because of my comfort and experience with the Android operating system. I also was capable of using Android Studio, a development environment program that developers use to create applications. Because of this, when I first started reading Google's instructions for how to develop applications for Google Cardboard, I focused solely on working with Android Studio.

The problem with using Android Studio for virtual reality is that the learning curve is very steep. Virtual reality relies heavily on the field of computer graphics. Computer graphics is responsible for generating and displaying three-dimensional objects and environments, which it does through applying many mathematical and physical concepts. My previous Android programming experiences never covered computer graphics or matrix mathematics. I also had not taken a computer graphics course up to that point. It turns out that, to create digital objects and environments, it requires a lot of verbosely written code that involve matrices. Thus, I quickly learned that I had no idea how to begin programming for this application.

Even if I had understood computer graphics concepts, another problem was that the math that describes three-dimensional objects was far too "low-level" for me to finish VR'Tour in a timely manner. It takes a lot of code to define and manipulate simple objects at the level of flat planes and cubes. To describe something as complex as, say, a human, it can take thousands of triangles and consequently thousands of coordinates. This would quickly become unmanageable. I only had a year to create VR'Tour, so there was no way that I could use Android Studio to develop for virtual reality.

Google's Cardboard demo treasure hunt project, which you can examine in [28], is an excellent example of these two issues. In the application, your task is to search for a cube that spawns in a random location around you. The only other digital object in the application's "world", or digital environment, besides the cube is a coordinate grid that represents the floor. This alone uses around 300 lines of vector and matrix values, which are stored in WorldLayoutData.java [29]. The difficult learning curve can be seen in the other file, TreasureHuntActivity.java. Google's application uses OpenGL for generating

graphics. Just to clarify, OpenGL is a "software interface to graphics hardware" [30]; it works with graphics hardware on a hardware level, but it is still programmable. The problem is that OpenGL has its own terminology and set of functions, which you have to be familiar with in order to use. Simple tasks like placing a cube in the world have a series of steps that have to occur in the right order for you to succeed. For this reason, it was impractical to use Android Studio to develop VR'Tour.

To overcome these problems, I accepted a suggestion from Mark de Ruyter, a member of UCR's Association of Computing Machinery (ACM) chapter, and used the Unity game engine software instead of Android Studio to develop the application. Mark advised me to use Unity over OpenGL because I wanted to create a fairly complex application in a short period of time and learning OpenGL would take too long. Unity contains a graphical user interface (GUI) that drastically simplifies the process of working with digital objects and environments. For example, if I want to have one object face another object, instead of calculating a rotation vector, I can rotate the object by clicking and dragging my mouse. The GUI is much friendlier to new developers and Unity offers a wide variety of tutorials. Unity enabled me to develop VR'Tour within a year through its powerful features and interface.

Conclusion

The demo for VR'Tour proved to be a functional social virtual reality application and an enjoyable virtual tour. I presented the demo at Electrical and Computer Engineering (ECE) Day and the meeting for the CSE Board of Advisors. Both times, I received positive reviews about it from professors and event participants. They were

impressed by the videos and the voice chat. Some offered suggestions for the future, such as using ambient noise for tour videos or submitting the demo to the UCR YouTube page.

My mentor and I have many more features planned for the full version of VR'Tour. We want to give the tour guide more ways to manage the room, such as volume controls for participants in the room or kicking people out. We also want to create more features for tour participants. The blue square that shows where other people are looking is supposed to be a profile image and a name. Participants would be able to use a "raise hand" gesture to ask the tour guide questions or emotes to express to others how they feel. The most important feature to implement however is a full tour of the UCR campus.

To quickly create a full tour of UCR, it would be helpful to use Dr. Chen's original proposal idea to create a platform that simplifies the tour creation process. The video recording process can be left unchanged, but synchronization and stitching should be automated. There needs to be a framework for adding videos into an existing tour without editing the original VR'Tour program. There should also be a fast method for creating new annotations.

One interesting problem my mentor and I spent a fair amount of time on was how the application should access tour videos. I decided early on that due to the number of videos in a tour, I could not have them stored in local memory. The videos would need to be downloaded over the internet, but because of their size and resolution, this proved to be difficult. We considered three main options, which I discuss below.

The most rudimentary option was to ask the user to download the videos before taking a tour. This would create long waiting periods, which I wanted to avoid to make

the tour more appealing. A second option was to download and play the video simultaneously with EasyMovieTexture. After trying this, I discovered that playing videos over 5 seconds using our near 4K resolution caused significant lagging. Experimenting with video file size, length, and resolution showed that EasyMovieTexture can handle longer videos if they are around 20 MB, which lower resolutions help to achieve. I wanted to avoid this however because of pixelation.

The last option, which Dr. Chen suggested, was to use MPEG DASH. MPEG DASH is an adaptive streaming format that segments large video files into smaller segments for faster playback over the internet [31]. Neither EasyMovieTexture nor any other third party asset I could find supported streaming though. For the sake of completing a prototype, I chose to store the videos in local storage in the demo for smooth playback and high resolution. In the future, it would be beneficial to research a streaming solution for Unity so that we can store the videos on a server and keep using high resolution videos.

VR'Tour is far from complete, but its demo is a substantial prototype that shows VR'Tour's potential. Combining my goal to create a social virtual reality application with Dr. Chen's idea of a campus tour application was well worth the effort. I may continue to work on the prototype over the summer with Dr. Chen to add additional features and include more UCR locations. If possible, it would be great to offer this application to the UCR Campus Tours Offices as a way of providing tours to out of state or international students.

APPENDIX

R'Welcome: Virtual Reality Campus Tour

R'Welcome: Virtual Reality Campus Tour

Jiasi Chen
Department of Computer Science and Engineering

Introduction: Virtual reality (VR) enables users to explore virtual environments, such as campuses, museums, and real estate. Major companies (Sony, Facebook/Oculus, HTC) are releasing powerful VR hardware to consumers in 2016. However, the price point of such devices are out of reach for casual consumers (e.g. $1600 for an Oculus Rift package, $750 for a Playstation VR package). Instead, mobile device manufacturers (Google, Samsung, LG) have developed a cheaper alternative: smartphone-based VR headsets ($20), which use the smartphone as the VR display.

Proposal: Our goal is to develop a software platform to enable easy content creation of VR tours, using the UCR campus tour as an example application. We imagine prospective undergraduate students being issued a Google Cardboard, who then use their existing smartphones to tour the UCR campus. We call this the "R'Welcome" app. Students will be able to virtually navigate the campus, with audio commentary provided at points of interest. The commentary can also be customized to the student's intended major and extracurricular interests. For example, departments can provide additional commentary to prospective majors, and student clubs can advertise their presence on campus.

Figure 1: Virtual reality tour of UCR campus will incorporate important landmarks and be navigable by prospective students, analogous to a "3D Google StreetView."

This project develops both the underlying content creation framework and the standard UCR tour. We will develop (a) methods to capture panoramic images and stitch them together into virtual navigable world; (b) methods to trigger playback of audio commentary based on user's current gaze; and (c) UCR-specific video and audio content. Currently, we have purchased a smartphone (Samsung Galaxy S6) and several smartphone VR viewers (Google Cardboard, Samsung Gear VR). We have also engaged with the UCR Office of Research and Economic Development, and will borrow their camera capture system (Freedom360 F360 Mount, 6x GoPro HERO4 Black) to capture the 360-degree content.

Related Work: 360-degree YouTube videos (e.g. [1]) are confined to a single pre-recorded commentary and lacks user navigation. VR museum tours (e.g. [2]) allow users to select interesting objects to hear additional commentary, but the underlying content creation framework is custom-built and not commercially available. Google Expeditions [3] enables virtual "field trips" for K-12, but both participation and content creation are currently invite-only.

References

[1] The White House, "360 holiday tour at the white house." https://www.youtube.com/watch?v=98U2jdk8OGI.

[2] Royal Academy of Arts, "Ai Weiwei 360." https://www.royalacademy.org.uk/exhibition/ai-weiwei-360.

[3] Google, "Expeditions pioneer program." https://www.google.com/edu/expeditions/.

Bibliography

[1] R. Munroe. *Depth Perception* [Online]. Available: https://xkcd.com/941/.

[2] R. Franklin, (2017, April 18). *Facebook Spaces: A New Way To Connect With Friends In VR*," Facebook [Online]. Available: https://newsroom.fb.com/news/2017/04/facebook-spaces/. [May 22, 2017].

[3] *AltspaceVR Inc | Be there, together.*, AltspaceVR, Inc. [Online]. Available: https://altvr.com/. [May 22, 2017].

[4] *Werewolves Within*, Ubisoft, [Online]. Available: https://www.ubisoft.com/en-US/game/werewolves-within/. [May 22, 2017].

[5] B. Lang (2017, January 16). *VR Interface Design Contest with $10,000 in Cash Prizes Launched by Purple Pill*, Road To VR [Online]. Available: http://www.roadtovr.com/purple-pill-vr-interface-design-contest-10000-prizes/. [May 22, 2017].

[6] Kolor. *Panotour Gallery*, [Online]. Available: http://www.kolor.com/panotour/panotour-gallery/. [Accessed 17 May 2017].

[7] Smithsonian Institution. *NMNH Virtual Tour: Smithsonian National Museum of Natural History* [Online]. Available: http://naturalhistory.si.edu/VT3/. [May 17, 2017].

[8] Harvard College. *Harvard Virtual Tour* [Online]. Available: https://college.harvard.edu/admissions/visit/virtual-tour. [May 17, 2017].

[9] University of California, Riverside. *Visit UCR | Campus Virtual Tours* [Online]. Available: http://admissions.ucr.edu/visit-ucr/index.html. [May 17, 2017].

[10] D. Handojo (2017, January 15). *UCR-UnityTour*, GitHub, Inc. [Online]. Available: https://github.com/Fire3galaxy/UCR-UnityTour. [May 21, 2017].

[11] Exit Games. *Introduction* [Online]. Available: https://doc-api.photonengine.com/en/pun/current/index.html. [May 10, 2017].

[12] Google. *Google VR SDK for Unity* [Online]. Available: https://developers.google.com/vr/unity/guide. [May 17, 2017].

[13] D. Kopitchinski and K.L. *Google Cardboard VR sensors*, StackOverflow [Online]. Available: http://stackoverflow.com/questions/26792526/google-cardboard-vr-sensors. [May 17, 2017].

[14] JaeYunLee, (2013, July 18). *Easy Movie Texture (Video Texture)*, Unity Technologies [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/10032. [May 12, 2017].

[15] Unity Technologies (2013). *Textures*. [Online]. Available: https://unity3d.com/learn/tutorials/topics/graphics/textures. [May 12, 2017].

[16] Unity Technologies, (2013). *Materials*, [Online]. Available: https://unity3d.com/learn/tutorials/topics/graphics/materials. [May 12, 2017].

[17] Unity Technologies, (2017, May). *MovieTexture*. [Online]. Available: https://docs.unity3d.com/Manual/class-MovieTexture.html. [May 12, 2017].

[18] Unity Technologies. *Networking Player Movement* [Online]. Available: https://unity3d.com/learn/tutorials/topics/multiplayer-networking/networking-player-movement?playlist=29690. [May 18, 2017].

[19] Unity Technologies. "6players.png", in *Networking Player Movement* [Online]. Available: https://unity3d.com/learn/tutorials/topics/multiplayer-networking/networking-player-movement?playlist=29690. [May 18, 2017].

[20] Unity Technologies. *Manual: Network View* [Online]. Available: https://docs.unity3d.com/Manual/class-NetworkView.html. [May 18, 2017].

[21] Unity Technologies. *Manual: State Synchronization Details (Legacy)* [Online]. Available: https://docs.unity3d.com/Manual/net-StateSynchronization.html. [May 18, 2017].

[22] Unity Technologies. *Manual: RPC Details (Legacy)* [Online]. Available: https://docs.unity3d.com/Manual/net-RPCDetails.html. [May 18, 2017].

[23] Exit Games. *Main Page* [Online]. Available: https://doc-api.photonengine.com/en/pun/current/index.html. [May 18, 2017].

[24] Exit Games. *PhotonView Class Reference* [Online]. Available: https://doc-api.photonengine.com/en/pun/current/class_photon_view.html. [May 18, 2017].

[25] Exit Games, "Photon Voice for PUN," [Online]. Available: https://doc.photonengine.com/en-us/voice/current/getting-started/voice-for-pun. [May 20, 2017].

[26] S. Jukic, (2016, April 5). *4K Resolution Guide - Compare 4k vs 1080p and Ultra HD (UHD) Resolution*, 4K. [Online]. Available: http://4k.com/resolution/. [May 21, 2017].

[27] Gtilloux (2012, April 25). "Ex 2 pano detected.jpg", in *Autopano - Common cases that don't stitch automatically - Repetitive patterns*, Kolor [Online]. Available: http://www.kolor.com/wiki-en/action/view/Autopano_-_Common_cases_that_don%27t_stitch_automatically_-_Repetitive_patterns. [May 22, 2017].

[28] cyisrrael (2016, June 10). *gvr-android-sdk/samples/sdk-treasurehunt at master*, Google VR [Online]. Available: https://github.com/googlevr/gvr-android-sdk/tree/master/samples/sdk-treasurehunt. [May 23, 2017].

[29] cyisrael (2016, June 10). *gvr-android-sdk/WorldLayoutData.java at master*, Google VR [Online]. Available: https://github.com/googlevr/gvr-android-sdk/blob/master/samples/sdk-treasurehunt/src/main/java/com/google/vr/sdk/samples/treasurehunt/WorldLayoutData.java. [May 23, 2017].

[30] M. Woo, J. Neider and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*, 1997 [Online]. Available: http://www.glprogramming.com/red/. [2016]

[31] J. Ozer (2011, November 22). *What is MPEG DASH?*, StreamingMedia.com [Online]. Available: http://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=79041. [May 22, 2017].