6-8-2015

Using virtual reality for anxiety therapy

Bryce Mariano
Santa Clara University

Paul Thurston
Santa Clara University

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
DEPARTMENT OF WEB DESIGN AND ENGINEERING

Date: June 8, 2015

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY
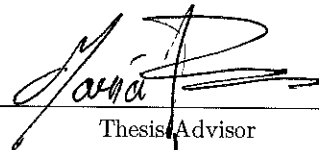
Bryce Mariano
Paul Thurston

ENTITLED

Using Virtual Reality for Anxiety Therapy

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREES
OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING
BACHELOR OF SCIENCE IN WEB DESIGN AND ENGINEERING

_____
Thesis Advisor

_____
Department Chair

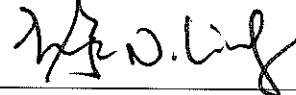# Using Virtual Reality for Anxiety Therapy

by

Bryce Mariano
Paul Thurston

Submitted in partial fulfillment of the requirements
for the degrees of
Bachelor of Science in Computer Science and Engineering
Bachelor of Science in Web Design and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 8, 2015

Using Virtual Reality for Anxiety Therapy

Bryce Mariano
Paul Thurston

Department of Computer Engineering
Department of Web Design and Engineering
Santa Clara University
June 8, 2015

ABSTRACT

Phobias, defined as a persistent and often irrational fear of an object or situation, are a very common type of anxiety disorder that can make it extremely difficult if not impossible for sufferers to interact with the world in a normal and healthy fashion. Traditionally therapists have used a concept known as systematic desensitization to help patients gain control of the emotional and physical reaction to their phobia. Systematic desensitization is employed using a type of treatment known as exposure therapy in which the patients are gradually made to think about and eventually face whatever triggers their anxiety until they are able to control their response during full exposure. While this treatment method can prove effective for some patients, we have identified two major problems with it. First, many patients are reluctant to undergo treatment and would rather ignore their phobia than be forced to face it. Second, depending on the type of phobia, treatment can quickly become expensive if the therapist has to travel with the patient outside of the office.

This paper outlines a system we developed that implements emerging virtual reality technologies as a tool for therapists to supplement the traditional treatments for anxiety disorders such as phobias. Our system uses the Oculus Rift, the Leap Motion, a mobile application, and a simulation running in the Unity Game Engine to create an immersive virtual experience for the patient while still giving the therapist the necessary control over the treatment. We believe that this system has the potential to provide therapists with a safer, more controlled, cheaper, and ultimately more effective way to expose their patients to their phobias.

Acknowledgements

Table of Contents

List of Figures

Chapter 1:   Introduction

1.1   Problem Statement

Phobias are defined in clinical psychology as persistent and often irrational fears of an object or situation [2]. Usually people suffering from a phobia will go to great lengths to avoid their fear, and these actions are typically disproportional to the actual danger posed. It is natural to feel afraid when in danger, as fear is an essential part of the "fight-or-flight" response. However, people suffering from a phobia will react strongly to a specific object or situation with often crippling anxiety attacks, which can lead to significant interference in their day-to-day lives. These anxiety attacks can be triggered by essentially anything that may remind sufferers of the object or situation they fear, sometimes making necessary daily activities such as working, driving, or even leaving their house nearly impossible.

There are two empirically validated treatments for phobias: exposure therapy and cognitive therapy. The first, exposure therapy, involves patients being gradually exposed to the feared object or situation while in a relaxed state. Therapists and patients work together to construct "anxiety hierarchies," a list beginning with the least anxiety-producing situations to the most anxiety-producing situations (e.g. from imagining a spider in the next room to placing an actual spider on the patient's lap). Patients will then undergo each situation in the anxiety hierarchy until they are able to control their emotional reaction during full exposure to their fear. The second type of therapy used for treating phobia patients is known as cognitive therapy. In cognitive therapy, therapists attempt to help patients understand and change how they think about their phobias. The goal is to identify thoughts and feelings of anger, guilt, or confusion and help patients accept that their phobias are not rational.

We have identified two problems with traditional exposure therapy that keep it from always being a successful treatment method; thus a more adaptable and effective solution is required. The main problems we found with traditional treatment methods are 1) the reluctance of patients to undergo treatment and 2) financial restrictions depending on the phobia. The first problem is that many patients simply cannot overcome the challenge of willingly being exposed to their phobia and as such would rather live with their disorder than get treatment that would require them to face their fear. The second problem with traditional exposure therapy is that while some phobias such as fear of spiders, snakes, or small spaces can be easily brought into a therapist's office, some require that the patient be brought to an external location for treatment. Phobias such as fear of heights, crowded places, or flying can

become very expensive to treat with each therapy session requiring the therapist and patient to leave the office. With these problems in mind we sought to create a tool that supplements and improves the effectiveness of the current therapeutic methods by offering a more approachable and cost effective form of treatment.

1.2 Related Work

The University of Southern California's Institute for Creative Technologies has also produced a project known simply as Bravemind [1] that similarly implements virtual reality technologies for medical benefit. The project is described on the University of Southern California's website as follows: "Bravemind is a clinical, interactive, virtual reality (VR) based exposure therapy tool being used to assess and treat post traumatic stress disorder (PTSD)." The team working on project Bravemind recognized the same issues with traditional exposure therapy; mainly, that it relies on the patients' willingness and ability to face their fear. This problem is made even worse in the case of treating post traumatic stress disorder, as it further requires that the patients be able to effectively imagine their traumatic experience since recreating it with traditional methods is not a viable option. As the team states, "this very tendency to avoid the cues and reminders of the trauma is one of the cardinal symptoms of PTSD." The team hopes to solve these problems using virtual reality technology just as we have with our project. The team identified the same potential for this type of treatment because it not only allows interactive and immersive environments to be created and tailored to patients' needs but also provides the therapists with the ability to control and document patient responses.

While project Bravemind is very impressive and has the potential to help a lot of patients suffering from PTSD, we found a few aspects that we felt could be improved. The main problem that we identified with the project is that it is built using proprietary technology. The virtual reality head mounted display as well as the simulation itself were both created in house rather than using pre-existing and available technologies. This means that the process of expanding the project to the consumer market will be much more difficult because the hardware would need to be redesigned to be manufactured on a large scale. It would also be very difficult to adapt the simulation for different patients because it was built using proprietary systems that future developers would need access to. The proprietary nature of the Bravemind system also means that cost becomes a very real issue. If a clinical psychologist wanted to purchase this system for use in his or her office, he or she would most likely be looking at a price tag in the thousands. During the early design process for our project we tried to keep these shortcomings in mind and create a system that would be accessible, adaptable, and affordable for our clients. These considerations led us to use the Oculus Rift for the head mounted display and the Unity Game Engine to create the simulation as well as the mobile application. By using pre-existing hardware and software that is free to download and use we ensured that our clients can purchase our product for around three

hundred dollars and have full capability to adapt it to their needs in the future.

1.3   Objectives

In order to address the problems we identified with traditional treatment methods for phobias we developed a virtual reality simulation intended to treat those suffering from this widespread anxiety disorder. The goal was to create a simulation that employed exposure therapy to provide a more approachable and more cost effective form of psychotherapeutic treatment for phobias. In order to accomplish this we built our system using inexpensive hardware and free software that is available for anyone for future development. We developed the project in collaboration with professionals from the games industry and clinical psychologists who treat phobia patients. The end result is a Windows application in which patients wear an Oculus Rift virtual reality headset in order to experience their phobia in a safe and controlled environment. This simulation is in no way intended to replace therapists or even traditional psychotherapy treatments but instead as a tool to be used by the therapist to supplement the normal treatment cycle. In practice, the therapist would be able to choose from a collection of common phobias and find one most appropriate for the patient. We believe that our project will help fill a void in the treatment of phobias by giving therapists a safe, more controlled, cheaper, and ultimately more effective way to expose patients to their phobias.

Chapter 2:   Requirements

To ensure that we created a product that is useful and effective for our potential customers, we defined a set of requirements for our system before beginning the design portion of the project. We identified three types of requirements that we had to keep in mind while desiging our system. Functional requirements define what our product needed to do in order to meet our goals, non-functional requirements define the more abstract qualities our product should have, and finally design constraints define how the solution was implemented.

Functional Requirements

The system will do the following:

- Allow for the therapist to choose a virtual experience appropriate to the patient

- Allow for the therapist to customize the details of the chosen virtual experience to control the level of exposure that the patient will receive

- Provide an alternative experience that is designed to calm the patient

Non-Functional Requirements

The system will be the following:

- Intuitive to use

- Responsive

- Affordable

Design Constraints

The system must do the following:

- Be usable on either a Windows or Mac computer

Chapter 3:   Conceptual Model

Due to the fact that our system will be used by clinical psychologists and therapist we needed to ensure that it was intuitive, flexible, and ultimately easy for someone who is not necessarily technically savvy to use. In order to make our product as simple to use as possible we decided to create two main parts that work in conjunction to make the complete system. The first and most important part of the system is the simulation itself. The simulation runs using the Unity Game Engine and is displayed on an Oculus Rift head mounted display. We created a demo simulation as a proof of concept but in the future a library of simulations covering the most common phobias would be needed in the final product, each with varying degrees of exposure. For example, in order to treat a specific phobia such as fear of snakes the psychologist would start the patient in a simulation where the snake is very far away or in another room, behind a window. As the patient begins to become desensitized to their phobia the therapist can move to simulations with more immediate exposure to snakes such as having it appear very close to the patient.

In Figure 3.1 you can see the interface for choosing a simulation from a library of possible choices. Each simulation is designed for a specific anxiety disorder. Once a simulation has been selected and started, the therapist is shown a screen like the one shown in Figure 3.2. This screen is the main control interface used throughout the simulation, and allows the therapist to control the parameters of the simulation in real time. The simulation shown in the figure is one targeted at those with a fear of heights. In this simulation the patient sees themselves standing on the top of a tall building. Using the interface shown the therapist is able to change the height of the building depending on how the patient is responding to the treatment. Using these parameters the therapist is able to maintain control over the experience and gradually expose the patient to their fear of heights at an appropriate pace.
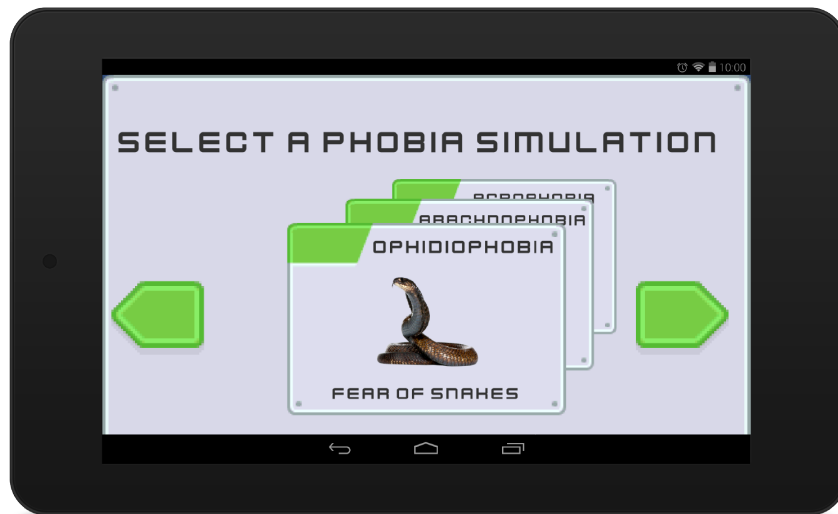
Figure 3.1: Simulation Selection screen: Shows how the therapist will choose from a library of simulations.
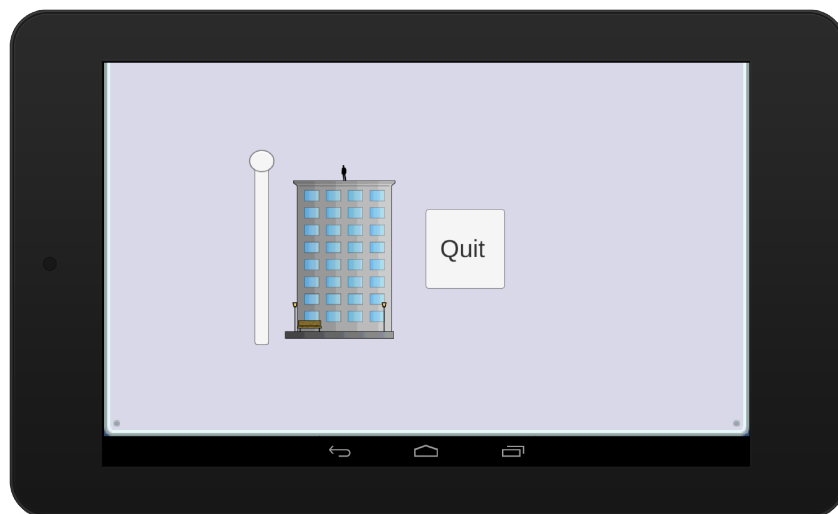
Figure 3.2: Simulation Control screen: Shows how the therapist will control the simulation

Chapter 4: Use Cases

In this section we will describe the various use cases that users will complete while using our virtual reality simulation. A use case illustrates the relationship between an Actor and the main actions he/she will perform on the system. The two Actors of our system will be 1) a phobia patient using the head mounted display during a therapy session and 2) the clinical psychologist administering the treatment to the patient. Figure 4.1 illustrates the various use cases that both the patient and therapist will go through while using our system.
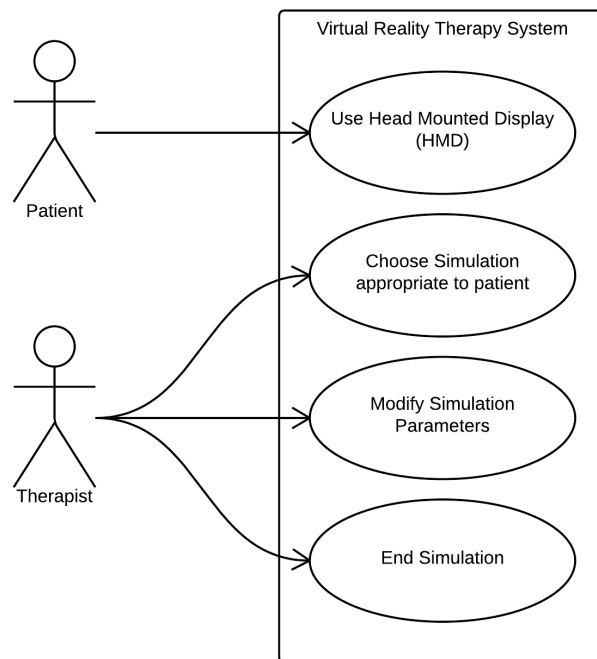
Figure 4.1: Use Case Diagram: Shows the actions that the different users might perform while using the system

Here is a more detailed description of the various use cases:

Undergo Phobia Treatment

Goal: Participate in therapy session by using Head Mounted Display (HMD)

Actor: Patient

Pre-condition: Patient has decided to get treatment for his/her phobia

Post-condition: Patient has been given exposure therapy treatment using the simulation

Scenario:

1. Wear the head mounted display

2. Experience the simulation chosen by the therapist

3. Face his/her phobia

4. Become systematically desensitized

Exceptions:

1. Control application loses connection to simulation

2. Patient becomes overly distressed and asks the therapist to end the treatment

Choose Phobia Simulation

Goal: Load a simulation that is appropriate to the patients phobia

Actor: Therapist

Pre-condition: Therapist is about to begin treatment session with patient

Post-condition: Therapist has begun simulation and treatment process

Scenario:

1. Use mobile application to choose simulation most appropriate to patient being treated

Exceptions:

1. Control application loses connection to simulation

2. Cannot find appropriate simulation for patient in library

Modify Current Simulation

Goal: Modify the simulation parameters as patient moves through the different stages of exposure therapy treatment

Actor: Therapist

Pre-condition: Therapist has begun simulation and treatment process

Post-condition: Patient has completed multiple stages of exposure therapy in the simulation

Scenario:

1. Use mobile application to modify simulation parameters appropriately according to the patient's reaction to the simulation

Exceptions:

1. Control application loses connection to simulation

End Simulation

Goal: End the simulation

Actor: Therapist

Pre-condition: Patient has completed multiple stages of exposure therapy in the simulation

Post-condition: Patient has completed therapy session

Scenario:

1. Use mobile application to end simulation

Exceptions:

1. Control application loses connection to simulation

Chapter 5:   System Design

5.1   Design Architecture

Our system design consists of two major components. The first is the VR simulation that the patient
will experience. He or she will do so by using a Head Mounted Display (HMD), which is connected to
a PC or Mac running the simulation. The second component is the control interface that the therapist
will use to control the simulation. This interface will run on a mobile device. The two components
communicate with each other via a data stream network. Figure 5.1 shows this relationship with both
components sending and receiving data.
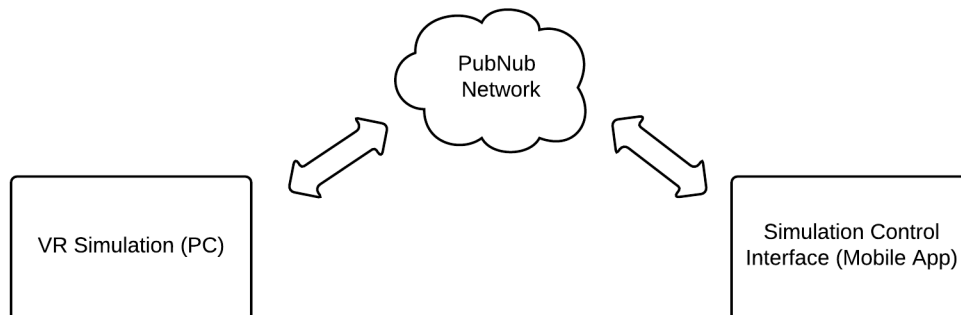
Figure 5.1: Entity Relationship Diagram: Shows the communication between the VR simulation and the
command interface on a mobile device.

5.2   Technologies Used

We used the following technologies for our product:

- Hardware:

  - Oculus Rift Development Kit 2: The Oculus Rift is a high quality head mounted display
    (HMD) which projects different images to each eye in order to create a 3D effect. The
    display uses a 75Hz low-persistence OLED display which lowers the time that pixels remain
    on the screen before they are updated. The combination of a high frame rate and small pixel

latency lowers the nauseating effects that people typically experience when using a HMD. The Oculus is also able to track head orientation using a combination of gyroscopic sensors and accelerometers. Additionally an infrared LED array is embedded in the Oculus's casing which can be tracked using the positional camera, which is then able to use this data to calculate the Oculus's position. The combination of all of these pieces allows the Oculus to accurately track the position and rotation of the user's head in addition to projecting a 3D image to the user without creating a sense of nausea.

- Leap Motion Controller: The Leap is an extremely small device that is designed to track a user's hands and fingers to a high degree of accuracy. It uses 2 cameras and 3 infrared LEDs to identify objects around the sensor. The images produced by these two cameras are then processed to figure out the position of the user's hands and fingers.

- Software:

  - Unity 3D: The Unity game engine greatly simplified the creation of the simulation. The engine has many tools such as graphics rendering, lighting and physics that streamline the process of creating the simulations. Unity allowed us to easily position 3D assets in the virtual space that the patients would see themselves in. Additionally Unity allows us to add our own C# code which allows us to control the simulation in real time.

  - PubNub: PubNub is a networking backend that allowed us to easily create a data stream between the VR simulation and the mobile app. Their system uses a publisher subscriber model where one user can publish data and any other users that are subscribed to that user will receive that data instantly.

  - JavaScript Object Notation (JSON): JSON is a human readable text format that is used to transfer data between applications. Because JSON is simply text it is language independent, meaning that any programming language can use JSON. In fact JSON is commonly used when communication between two different languages is needed. For our project we used SimpleJSON to parse and construct JSON strings in C#.

5.3 Design Rationale

We have chosen these technologies for a number of reasons. Because we want the system to be as affordable as possible we chose the Oculus Rift which is only $350, compared to similar devices which in the past have cost thousands. Also, because OculusVR was recently acquired by Facebook in a multi-billion dollar deal we can be confident that the company will have the resources to continue development of the Oculus and will continue to be an industry leader in virtual reality industry for years to come.

In addition to tracking head position with the Oculus we wanted to be able to track the position of the user's hands and fingers and create a virtual representation of those in the simulation. Because our hands are often in our field of vision, adding virtual arms and hands that correspond to the patient's actual arms will add an extra level of immersion to the patient experience, which in turn will increase the effectiveness of the therapy session. Our choice to use the Leap Motion Controller also falls under our desire to be affordable. The Leap is only $80 and yet is able to provide more accurate data about hand and finger positioning than other body tracking products like the more expensive Microsoft Kinect.

On the software side of things we chose to use the Unity game engine to help develop our system. Unity already implements many low-level systems needed for a 3D simulation such as graphics rendering, lighting and physics. Because we did not need to implement these systems ourselves we were able to focus our attention on developing the simulation itself and fulfilling our system requirements. Once again Unity fits in with our desire to create an affordable system because the free version of the engine supports Oculus integration.

Finally the last technology that we chose to use is PubNub which provided the networking backend for our system. PubNub allowed us to keep a constant data stream open between the VR Simulation and the control app, which allowed changes that are made on the app to update the simulation in real time. PubNub also has an SDK that supports Unity, which allowed us to smoothly integrate the two systems. Also, while the PubNub service is not free, it does have a tiered pricing model that scales as usage increases, so it also fits with our affordability requirement.

5.4   Implementation

Having defined our goals for the project and deciding on what technologies we wanted to use, we then needed to implement the system. Even though we focused on creating one phobia simulation for our demonstration, it was important that our system be easily expandable so that other phobia simulations could be easily added.

First we developed a flexible method of communicating data between the control interface and the simulation. By using JSON objects which are easily converted to text, developers can define how to interpret the JSON strings in phobia specific code, while the common code only needs to treat it as generic text. Listing 5.4 shows a typical JSON string, this one in particular being sent to the fear of heights simulation in order to change the height of the building. The first attribute is common to all JSON objects used in our product, and defines whether the instruction is meant to change scenes or modify the current one. The next attribute is specific to a modify scene command and states what scene is to be modified. This is used to verify that the instruction is being processed by the correct scene. Finally the Action attribute contains a list of data all of which defines how the scene is to be modified. In this example the action is interpreted as meaning that it will change the height of the building, and

uses a normalized value between 0 and 1 to figure out what height the building needs to be set to.

JSON Example

```
{

    "CommandType":"Modify",

    "SceneName":"Heights",

    "Action":{

      "Name":"ChangeHeight",

      "Type":"Slider",

      "Value":0.76

    }

}
```

Next we developed the simulation itself in Unity. In Unity environments are defined as Scenes in which 3D models, sound and code all reside. These scenes can be thought of like a level in a game. We have divided the simulation application into multiple scenes which we can easily switch between, with one scene acting as a standby scene with another scene for each phobia.

In order to make it easier for more phobia scenes to be added to the product, we used an Object Oriented design approach to allow for the code that we wrote to be as reusable as possible. Figure 5.2 shows the classes that we used to control the simulation and how they relate to each other.

Oculus

SceneController

-sceneName:string

+virtual PerformAction(name:string, data:float):void

HeightsSceneController

-building:Transform
-minHeight:float
-maxHeight:float
-targetHeight:float
-startHeight:float
-startTime:float

+override PerformAction(name:string, data:float):void
+SetHeightScale(scale:float):void

TherapyController

+Instance:static TherapyController
-localController:SceneController
-messageQueue:Queue<string>

+AddToQueue(msg:string):void
+ChangeScene(jsonnode:JSONNode):void
+ModifyScene(jsonNode:JSONNode):void
-ProcessMessage(jsonStr:string):IEnumerator

MessageReceiver

-pubnub:Pubnub
-publishKey:string
-subscribeKey:string
-channel:string

-OnReceiveMessage(msg:string):void

Control Interface

SceneChanger

-messenger:MessageController

+SendSceneCHangeMessage(sceneName:string)

HeightsController

-messenger:MessageController
-heightsSlider:Slider
-heightUpdateInterval:float
-currentHeight:float

+OnHeightChange():void

PubNub

MessageSender

-pubnub:Pubnub
-publishKey:string
-subscribeKey:string
-channel:string

-SendMessage(msg:string)

Figure 5.2: UML Class Diagram: Describes the class organization used in both the simulation and control interface.

The following is a description of how each class is used on both the Oculus side and the control interface side of the system.

- Oculus

  – `MessageReceiver`: This class is in charge of interfacing with the PubNub SDK and receiving messages. It owns a PubNub object and gives it function callbacks to be triggered when an event happens, most importantly the receipt of a message. This message is passed on to the `TherapyController` object as a string.

14

– `TherapyController`: This class is in charge of interpreting the JSON message. Because there only needs to be one master controller for the entire simulation this object is implemented as a singleton, meaning that there can only be one object of the class in existance at any one time. This is to prevent multiple therapy controllers from being created as scenes are opened and closed which could cause errors.

This controller parses the JSON string into a JSON Node using SimpleJSON which converts the string into usable data formats like ints, floats and booleans. Using the 'Command-Type' attribute the controller decides whether it needs to change scenes or modify the current one. If it decides that the scene does need to be modified then it passes the rest of the JSON data to the SceneController residing in the current scene.

– `SceneController`: The `SceneController` is a base class that is meant to be inherited from when a phobia scene is developed. It contains a virtual function `PerformAction` that is called by the `TherapyController` object. Because of polymorphism, we can create as many classes as we like that inherit from `SceneController`, and the `TherapyController` does not need to be changed; it simply needs to call the `PerformAction` function. The `SceneController` object also has an attribute that identifies the scene that it is meant to control. This allows for a redundancy check to be made to make sure that the scene the JSON data is meant for is the same as the current scene, preventing other messages from interfering with the scene.

– `HeightsSceneController`: The `HeightsSceneController` inherits from `SceneController` and is specific to the phobia simulation used in our demonstration, the fear of heights. This controller reads the rest of the data from the JSON object and modifies the scene accordingly, in this case changing the height of the building.

• Control Interface

– `MessageSender`: Like the `MessageController` on the Oculus side, this object is in charge of interfacing with the PubNub SDK. However this controller is instead in charge of publishing messages instead of receiving them.

– `SceneChanger`: This class is associated with the phobia selection interface. When a phobia is selected then a `SceneChanger` object will convert this decision into a JSON string to be sent and then pass that information to the `MessageSender`.

– `HeightsController`: This class is associated with the interface meant to control the fear of heights simulation. Whenever the user makes a change on the app the `HeightsController` object is notified and then converts this information into a JSON string which it passes to the `MessageSender`.

Figure 5.3 shows the sequence of events that occur when a JSON message is received on the simulation side of the product. The JSON message would be a modification instruction similar to the one shown in Listing 5.4. The message starts out being added to a queue to be processed by the `TherapyController`. A queue is used to make sure that messages are processed in order in case multiple messages are received in close proximity to each other. The `TherapyController` then converts the message and decides the appropriate response, in this case modifying the scene. The `TherapyController` will then find the `SceneController` in the current scene and pass along the instruction by calling `PerformAction`.
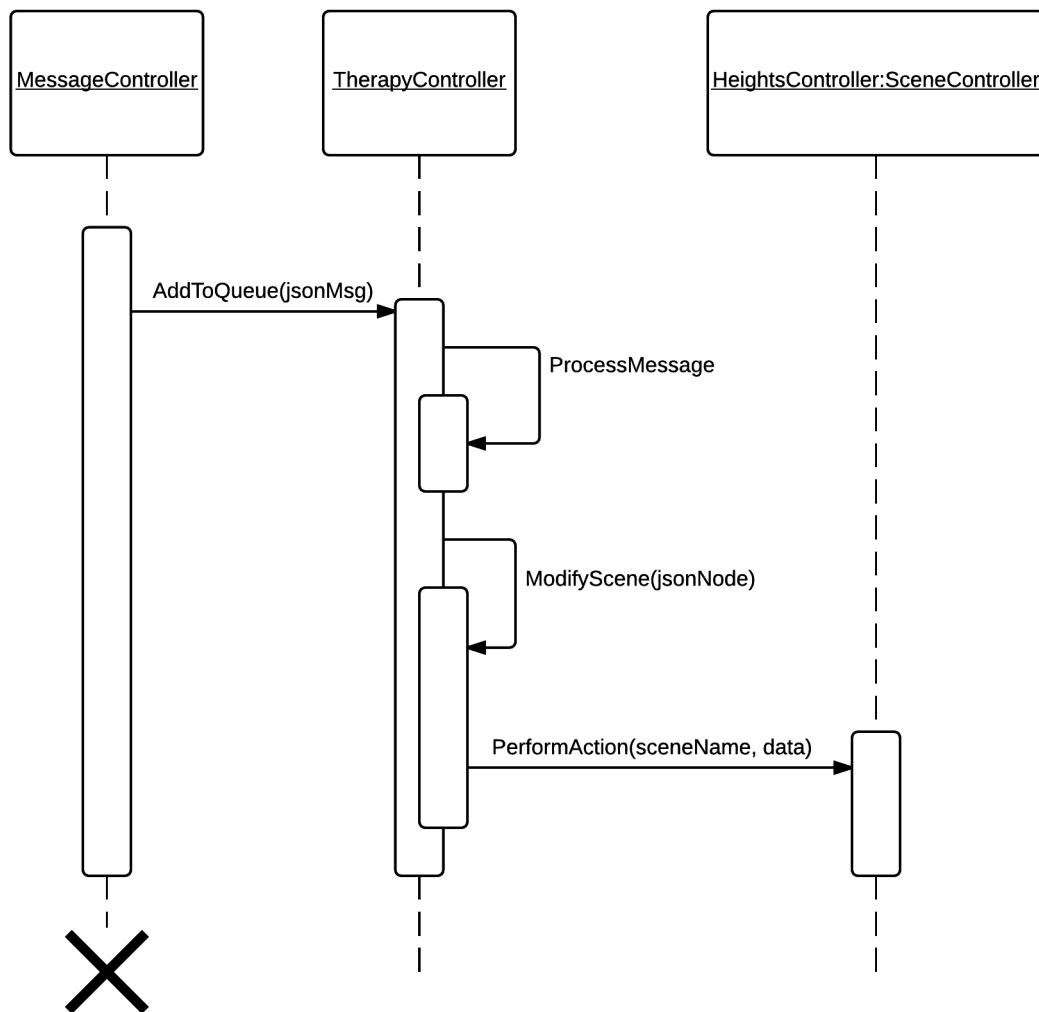

Figure 5.3: UML Sequence Diagram: Shows the sequence of events when a scene is modified.

Chapter 6:   Test Plan

Unit Testing

The unit testing aspect consisted of assessing the individual components of our project to ensure that they functioned as intended independently of one another. Mainly, the simulation we created and the mobile control application had to open and run smoothly before any testing could be done on the communication and interfacing between these two parts. We performed both black box and white box testing on the system components. White box testing assumes knowledge of the code, while black box testing does not. Both were used to ensure that the components functioned properly and without errors.

Integration Testing

Integration testing was done once the separate components were brought together to form the full system. For our project integration testing was to make sure that the mobile application communicated consistently with the simulation. This part of our testing process was crucial because it was imperative that the clinical psychologist using our system on a patient has complete control over the simulation itself. If the mobile application loses connection with the simulation and can no longer send commands, then the therapist administering the therapy would have no way to change the parameters of the simulation and, more importantly, stop the simulation if the patient becomes overwhelmed.

Acceptance Testing

The acceptance aspect consisted of demonstrating our project to clinical psychologists and therapists to ensure that the simulation therapy we created is acceptable to those in industry. The most important aspect we needed to consider when determining the validity of our project was whether or not our tool actually improves the effectiveness of the exposure therapy. Since conducting clinical trials was far outside the scope of our project we are relying on the opinions of the therapists themselves to test the successfulness of the virtual reality simulation.

Another important part of showing our project to clinical psychologists was seeing how easily they could start and manage the simulation. Since clinical psychologists will need to administer the exposure therapy to their patients the control application needed to be intuitive to them. We

could only consider our project fully successful if the industry professionals who will actually be using our product can use it easily and efficiently as intended.

Multiple times throughout the project we met with Dr. Kieran Sullivan, chair of the Psychology Department at SCU, in order to get this kind of informed feedback on the system. She informed us of the importance of ensuring the therapist had total control over the simulation at all times which led us to create the mobile application. After our demo simulation was fully functional we demonstrated it to her and confirmed the validity of our system as a possible therapeutic tool.

Chapter 7:   Societal Issues

7.1   Ethical

We believe that our project was a good use of our skills in engineering and video game design for the betterment of others. Video games have traditionally been designed and used strictly for entertainment and have only recently shown their potential for use in other industries such as healthcare and education. After identifying the issues we found with traditional treatment methods for phobias we felt confident that we could use our skills to create a system that could provide a better alternative and ultimately improve the quality of life of those suffering from a potentially crippling anxiety disorder. However, recognizing that we are not experts in psychology, we had to be careful to avoid causing more problems than we solved. The biggest potential issue we saw was creating a complicated system that therapists might be apprehensive to use in their practice. Meeting with and getting feedback from a licensed psychologist allowed us to assess how usable and intuitive our product was to our clients. It is also important to us to emphasize again that our product is in no way meant to replace therapists but rather to supplement traditional treatment methods.

7.2   Social

The social impact of our project is arguably the most important societal issue to consider. Since our project is essentially a healthcare tool intended for use by clinical psychologist to supplement the treatment of phobia patients, it can have far-reaching social impacts. Phobias are an extremely common form of anxiety disorder and can have crippling social effects of those who suffer from them. We believe that by providing a more approachable, affordable, and effective form of phobia treatment our project has the potential to help many people overcome their fears and allow them to interact with the world in a healthy manner.

7.3   Political

We do not feel as though our project has any significant political impact because it does not require us to take into consideration the will of the general public or elected representatives since it is not a public project.

7.4   Economic

The economic considerations we made during the design phase of our project were mainly focused around the affordability of our system. In order for our product to be an attractive option for therapists in treating their patients with phobias it needed to be not only more effective but also more fiscally sensible. Ensuring that our finished product was affordable was also important in order to make it more available and approachable to a wider range of patients. Using our system means that the therapists would no longer need to travel outside the office to treat certain patients which could potentially lower cost of entry for the patient.

7.5   Health and Safety

It is vitally important to consider the potential impacts on health and safety that our project could have as our system is essentially a tool meant to improve existing therapeutic methods. Meeting with a licensed therapist led to the inclusion of the mobile application that gives the therapist administering treatment full control over the simulation and most importantly the ability to terminate the simulation if the patient becomes overwhelmed. In the future a relaxation simulation could also be created that would actually help the patient regain control of their emotions rather than simply terminating the whole simulation. These considerations were made in the hopes that the resulting product would provide a superior treatment option for phobias and have real potential to improve people's mental health.

7.6   Manufacturability

We do not feel as though our project has any significant impact on manufacturability. Again, as we used existing hardware we did not have to make these kinds of considerations during development since we only produced software. Clients looking to use the system would just need to purchase the appropriate hardware and then download our simulation software for use.

7.7   Sustainability

The concept of sustainability in relation to our project has to be considered in two ways. The more narrow definition of sustainability refers to the ability of the product to continue to be viable and useful for a reasonable amount of time. In this way we feel as though our product is very sustainable. Our decision to use pre-existing hardware that will continue to be developed by Oculus as well as software that is free and available to future developers means that our product could easily be expanded and improved on by future teams. The broader definition of sustainability refers to a product's impact on the surrounding community or region. We do not believe our project has any significant impact the surrounding community or region because we only produced software.

7.8 Environmental Impact

We do not feel as though our project has any significant environmental impact. Again, since we only produced software we did not have to make these kinds of considerations during development.

7.9 Usability

We feel as though our project is successful in its usability. By using pre-existing hardware and software with well-established support communities and information online we have ensured that future development on the project will be as straightforward as possible. We have also done all we could to make the system itself easy to use for our clients. Since our product is intended to be used by clinical psychologists and licensed therapists who will not necessarily be technically knowledgeable we had to make sure setting up and using the simulation was as intuitive and straightforward as possible. The development of the mobile application was inspired by this need for usability as we felt it was a great way to give the therapist the ability to control the simulation with technology he or she is already familiar with.

7.10 Lifelong Learning

We believe that our project has contributed to our learning outside of our school experience because it led us to approach real world problems and consult with industry professionals. Meeting with professionals with backgrounds in computer engineering, video game design, and psychology meant that we had to consider multiple perspectives and try to balance a arrange of suggestions to make the most well rounded and compelling system possible. Completing a project of this scope for the first time also presented challenges with regards to planning and coordination that we had not faced before. Having to stay organized and on task over such a long period of time on a single project was a learning experience in its own regard.

7.11 Compassion

At the very beginning of the senior design project when we were still considering what to do the idea of compassion played an important role in leading us to design the virtual reality treatment system. Both of us are very passionate about the video game industry and hope to make careers as video game designers after graduation. We thought about how we could take our skills in video game design and apply them to a project that would not just be entertainment but actually help people. Once we started researching more we realized that phobias are a very real and widespread problem and that by developing a tool that would improve traditional methods of treatment we could help people suffering from these potentially crippling anxiety disorders.

Chapter 8: Conclusions

8.1 Summary

The problem that we set out to solve with our project was what we view to be a current lack of approachable, affordable, and effective methods for treating patients suffering from phobias. The concept at the heart of exposure therapy, systematic desensitization, led us to the idea to build a virtual reality experience that would give therapists the power to treat a wider range of patients suffering from phobias right in their office. We used a combination of cutting-edge virtual reality technologies and a mobile application to create a prototype demo as a proof of concept. The demo shows what a simulation to treat patients with a phobia of heights would look like. We used the Oculus Rift head mounted display, Leap Motion controller, Unity Game Engine, PubNub network, and a tablet to build the system.

8.2 What We Learned

Since both members of our team were engineers, the biggest hurdle for us was learning about the psychological aspect of our project. However thanks to the help that we received from Dr. Sullivan we were able to learn about phobias and how to treat them. This knowledge is what helped us define our project and design it to be as effective as possible.

From a technical perspective we also learned about how to develop virtual reality applications. We learned about how to bring multiple pieces of technology together for a single product. Learning how to build on others work through SDK's and API's was incredibly useful and has prepared us to work in industry.

8.3 Future Work

There are multiple ways in which we feel the project could be improved and developed further in the future. Most importantly in order for this project to become a usable tool for therapists the library of simulations would need to be expanded to include the most common phobias. We also believe that it would be useful to have the therapist be able to monitor how the patient is responding to the treatment. This could include the therapist seeing a patients heartbeat or some other form of biometric feedback on the control interface. Finally we believe that the visual quality of the simulations could be improved using more advanced rendering techniques. This would need to be balanced with further optimizing the

simulation so that it still performs well on less powerful hardware. All of these improvements would improve this product and hopefully lead to it eventually being used in a clinical setting.

Chapter 9:   References

[1] Albert Rizzo, JoAnn Difede, Barbara O. Rothbaum, J. Martin Daughtry, Greg Reger. "Virtual Reality as a Tool for Delivering PTSD Exposure Therapy" *Post-Traumatic Stress Disorder: Future Directions in Prevention, Diagnosis, and Treatment.* Springer, 2013.

[2] Whitbourne, Susan Krauss., Richard P. Halgin. *Abnormal Psychology: Clinical Perspectives on Psychological Disorders.* New York, NY: McGraw-Hill Higher Education, 2013

Chapter 10:   Appendix: Source Code

```csharp
using UnityEngine;
using System.Collections;

public class SceneController : MonoBehaviour {

    [SerializeField]
    private string _sceneName;

    public string SceneName {
        get { return _sceneName; }
        set { _sceneName = value; }
    }

    public virtual void PerformAction(string name, float data) {
        Debug.Log("Perform Action");
    }
}
```

```csharp
using UnityEngine;
using System.Collections;

public class HeightsSceneController : SceneController {

    [SerializeField]
    private Transform _building;

    [SerializeField]
    private float _minHeight = 0.0f;

    [SerializeField]
    private float _maxHeight = 40;

    private Transform Building {
        get { return _building; }
        set { _building = value; }
    }

    private float MinHeight {
        get { return _minHeight; }
        set { _minHeight = value; }
    }

    private float MaxHeight {
        get { return _maxHeight; }
        set { _maxHeight = value; }
    }
```

```
    private float TargetHeight { get; set; }
    private float StartHeight { get; set; }
    private float StartTime { get; set; }

    void Awake(){
        StartHeight = TargetHeight = Building.position.y;
    }

    public override void PerformAction(string name, float data) {
        if (name == "ChangeHeight") {
            SetHeightScale(data);
        }
    }

    public void SetHeightScale(float scale) {
        StartTime = Time.time;
        StartHeight = Building.position.y;
        TargetHeight = (MaxHeight * scale) + (MinHeight * (1.0f - scale));
    }

    void Update() {
        float t = (Time.time - StartTime)/0.25f;

        if (t < 1) {
            float h = Mathf.Lerp(StartHeight, TargetHeight, t);
            Vector3 position = Building.position;
            position.y = h;
            Building.position = position;
        }
    }
}
```

```
using System;
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using SimpleJSON;
using Debug = UnityEngine.Debug;

public class TherapyController : MonoBehaviour {

    public static TherapyController Instance {get; set;}

    private SceneController LocalController { get; set; }
    private Queue<string> MessageQueue { get; set; }

    void Awake(){
        if(Instance != null){
            Debug.Log("Controller already exists");
            Destroy(gameObject);
            return;
        }

        Instance = this;
        DontDestroyOnLoad(gameObject);
```

```
        MessageQueue = new Queue<string>();
    }

    void Update() {
        if (MessageQueue.Count > 0) {
            string msg = MessageQueue.Dequeue();
            StartCoroutine(ProcessMessage(msg));
        }

        if(Input.GetKeyDown(KeyCode.Escape)){
            Application.Quit();
        }
    }

    public void AddToQueue(string msg) {
        MessageQueue.Enqueue(msg);
    }

    private IEnumerator ProcessMessage(string jsonStr) {
        var jsonNode = JSONNode.Parse(jsonStr);
        Debug.Log(String.Format("JSON: {0}", jsonStr));
        string commandType = jsonNode["CommandType"];
        if (commandType == "Modify") {
            ModifyScene(jsonNode);
        } else if (commandType == "ChangeScene") {
            ChangeScene(jsonNode);
        } else {
            Debug.LogError(commandType + " is not a valid command type");
        }
        yield return null;
    }

    public void ChangeScene(JSONNode jsonNode) {
        string targetScene = jsonNode["SceneName"];
        Application.LoadLevel(targetScene);
        Debug.Log("Scene Changed");
    }

    public void ModifyScene(JSONNode jsonNode) {
        Debug.Log("Modify Scene");
        GameObject go = GameObject.FindGameObjectWithTag("SceneController");
        LocalController = go.GetComponent<SceneController>();

        if (LocalController == null) {
            Debug.LogError("Scene Controller not found");
            return;
        }

        string sceneName = jsonNode["SceneName"];

        if (sceneName == LocalController.SceneName) {
            Debug.Log("Send Action");
            LocalController.PerformAction(jsonNode["Action"]["Name"], jsonNode["Action"]
        } else {
            Debug.LogError("JSON Scene Name (" + sceneName + ") does not match local sce
        }
    }
```

```
}
```

```
using UnityEngine;
using System.Collections;
using System.Security.Cryptography.X509Certificates;
using System.Text.RegularExpressions;
using System.Threading;
using PubNubMessaging.Core;
using SimpleJSON;

public class MessageReceiver : MonoBehaviour {

    private Pubnub pubnub;

    private string publishKey, subscribeKey, channel;

    void Awake() {
        pubnub = new Pubnub(publishKey = "demo", subscribeKey = "demo");
        pubnub.Subscribe<string>(channel = "mychannel", DisplaySubscribeReturnMessage, I
    }

    void Update() {
        if (Input.GetKeyDown("p")) {
            JSONNode node = JSONNode.Parse("{\"CommandType\":\"ChangeScene\", \"SceneNam
            pubnub.Publish<string>(channel, node.ToString(), DisplayReturnMessage, Displ
        }
    }

    private void DisplaySubscribeReturnMessage(string msg) {
        JSONNode jsonNode = JSONNode.Parse(msg);
        string jsonMsg = jsonNode[0].ToString();
        jsonMsg = jsonMsg.TrimStart(' ', '\"');
        jsonMsg = jsonMsg.TrimEnd(' ', '\"');
        TherapyController.Instance.AddToQueue(jsonMsg);
        Display(jsonMsg);
    }

    private void DisplayReturnMessage(string msg) {
        Display(msg);
    }

    private void DisplaySubscribeConnectStatusMessage(string msg) {
        Display(msg);
    }

    private void DisplayErrorMessage(PubnubClientError error) {
        Display(error.StatusCode.ToString());
    }

    private void Display(string txt) {
        Debug.Log(string.Format("CALLBACK LOG: {0}", txt));
    }
}
```

```
using System;
using UnityEngine;
```

```
using System.Collections;
using PubNubMessaging.Core;

public class MessageSender : MonoBehaviour {

    private Pubnub pubnub;

    private string publishKey = "demo";
    private string subscribeKey = "demo";
    private string channel = "mychannel";

    void Awake() {
        pubnub = new Pubnub(publishKey, subscribeKey);
    }

    public void SendMessage(string msg) {
        pubnub.Publish<string>(channel, msg, DisplayReturnMessage, DisplayErrorMessage);
    }

    private void DisplayReturnMessage(string msg) {
        Display(msg);
    }

    private void DisplayErrorMessage(PubnubClientError error) {}

    private void Display(string msg) {
        Debug.Log(String.Format("CALLBACK LOG: {0}", msg));
    }
}
```

```
using UnityEngine;
using System.Collections;
using SimpleJSON;

public class SceneChanger : MonoBehaviour {

    string jsonInputStr = "{
                            \"CommandType\":\"ChangeScene\",
                            \"SceneName\":\"Heights\"
                          }";

    MessageController Messager { get; set; }

    void Awake(){
        Messager = GetComponent<MessageController>();
    }

    public void SendSceneChangeMessage(string sceneName){
        JSONNode node = JSON.Parse(jsonInputStr);
        node["SceneName"] = sceneName;
        Messager.SendMessage(node.ToString());
    }
}
```

```
using UnityEngine;
using UnityEngine.UI;
```

```csharp
using System.Collections;
using SimpleJSON;

public class HeightsController : MonoBehaviour {

    [SerializeField] private Slider _heightSlider;
    [SerializeField] private float _heightUpdateInterval = 0.2f;

    MessageController Messenger { get; set; }

    private float HeightUpdateInterval {
        get { return _heightUpdateInterval; }
    }

    private float LastUpdateTime { get; set; }
    private float CurrentHeightValue { get; set; }
    private bool SendUpdate { get; set; }

    string sliderCommandFormat = "{
                                    \"CommandType\":\"Modify\",
                                    \"SceneName\":\"Heights\",
                                    \"Action\":{
                                        \"Name\":\"ChangeHeight\",
                                        \"Type\" : \"Slider\",
                                        \"Value\" : 1.0
                                    }
                                  }";
    JSONNode sliderCommand;

    public Slider HeightSlider {
        get { return _heightSlider; }
    }

    void Awake() {
        SendUpdate = false;
        Messenger = GetComponent<MessageController>();
        sliderCommand = JSON.Parse(sliderCommandFormat);
    }

    void Update() {
        if (Time.time > LastUpdateTime + HeightUpdateInterval) {
            if (SendUpdate) {
                sliderCommand["Action"]["Value"].AsFloat = CurrentHeightValue;
                Messenger.SendMessage(sliderCommand.ToString());
                SendUpdate = false;
            }
        }
    }

    public void OnHeightsChange() {
        if(HeightSlider == null){
            Debug.LogError("HeightSlider not set");
            return;
        }
        float value = HeightSlider.value;
        if (!Mathf.Approximately(CurrentHeightValue, value)) {
            CurrentHeightValue = HeightSlider.value;
            SendUpdate = true;
```

```
        }
    }
}
```