Oregon State University
College of Engineering

CS Capstone Final Report

June 9, 2017

Virtual Reality Meditation App

Prepared for

Intel Corporation

Mike Premi _____ _____
Signature *Date*

Prepared by

Capstone Group 034

V.R.A.M.

Erik Watterson _____ _____
Signature *Date*

Makiah Merritt _____ _____
Signature *Date*

Padraig Gillen _____ _____
Signature *Date*

Abstract

The premise of this project is to develop a prototype of a Virtual Reality application that will guide users through several different guided meditation sessions. As part of a larger research study with a University of Washington (UW) Masters Capstone team, we will show that meditation within a virtual environment allows for a more holistic meditation experience. This will help individuals find solitude within a virtual scene that is both visually and auditorily captivating, while remaining in their everyday environments.

CONTENTS

1 Problem Statement 5

1.1 Problem Definition . . . 5
1.2 Proposed Solution . . . 5
1.3 Timeline . . . 5
1.4 Performance Evaluation . . . 6

2 Abbreviations, Acronymns, and Definitions 6

2.1 Acronyms . . . 6
2.2 Definitions . . . 6

3 Technology Review 8

3.1 Introduction . . . 8
3.2 Technologies . . . 8
3.3 Storage and Distribution of Project Files . . . 8
3.4 Unity Environment Lighting . . . 10
3.5 Head Mounted Displays (HMDs) . . . 12
3.6 Development Kits . . . 15
3.7 Unity Video Formats . . . 19
3.8 Unity Audio Formats . . . 21
3.9 Unity Scripting Languages . . . 23
3.10 GPU selection for VR . . . 24
3.11 Conclusion . . . 26

4 Project Design 27

4.1 Scope . . . 27
4.2 Purpose . . . 27
4.3 Intended Audience . . . 27
4.4 Conformance . . . 28
4.5 Conceptual Model for Software Design Descriptions . . . 28
4.5.1 Stakeholders and Stakeholder Concerns . . . 28
4.5.2 Design Views . . . 28
4.6 Design Descriptions . . . 29
4.7 Design Component: HTC Vive . . . 29
4.8 Design Component: Environment Lighting . . . 31
4.9 Design Component: Google Drive Housing for Project Files . . . 32
4.10 Design Component: Unity's Game Development Platform . . . 35
4.11 Design Component: .mp4 Video File Format . . . 37
4.12 Design Component: Unity Compressed Audio . . . 39
4.13 Design Component: Unity Scripting . . . 40

4.14 Discussion of Design Changes . . . 41

5 Software Requirments Specification 42
5.1 Purpose . . . 42
5.2 Scope . . . 42
5.3 Overview . . . 42
5.4 Team Requirements Description . . . 42
5.4.1 The Intel Stakeholder . . . 42
5.4.2 The UW Team . . . 43
5.4.3 The OSU Team (V.R.a.M.) . . . 43
5.5 Product Description . . . 43
5.5.1 Product Perspective . . . 43
5.5.2 Product Perspective Constraints . . . 44
5.5.3 Product Functions . . . 44
5.5.4 User Characteristics . . . 45
5.5.5 Constraints . . . 45
5.5.6 Hardware Limitations . . . 45
5.5.7 Interfaces to Other Applications . . . 45
5.5.8 Reliability Requirements . . . 45
5.5.9 Safety and Security Considerations . . . 46
5.5.10 Assumptions and Dependencies . . . 46
5.6 Addendums . . . 46

6 Gantt Charts 47

7 Expo Poster 49

8 Looking Back 50
8.1 What Did We Learn From All This? . . . 50
8.1.1 Makiah Merritt . . . 50
8.1.2 Erik Watterson . . . 50
8.1.3 Padraig Gillen . . . 52
8.2 Where Did We Learn About Our Tech . . . 52

9 Fall Blog Posts 53

10 Winter Blog Posts 57

11 Spring Blog Posts 65

12 Code Listings 72

13 Code Documentation 142

References 219

LISTINGS

1 Unity Scripting API: MovieTexture . . . 38
2 Game Controller . . . 72
3 JS OSC Receiver . . . 78
4 Meditation History Controls . . . 80
5 Menu System 2D . . . 87
6 Menu System VR . . . 96
7 Menu Trigger Pull . . . 104
8 Orb Controller . . . 105
9 Osc Test Sender . . . 112
10 Post Meditation Survey Controls . . . 113
11 Preference Loader . . . 118
12 Pre Meditation Survey Controls . . . 119
13 Scene Fader . . . 122
14 (Osc) Server . . . 125
15 Sky Master Interface . . . 126
16 Tree Object Manipulation . . . 138

1 Problem Statement

1.1 Problem Definition

Our stakeholder, Mike Premi, has a goal to create a Virtual Reality (VR) application that helps facilitate a meditative state. His hope is that by producing this application, he can offer a way to use immersion as a form of medication for individuals suffering from addiction and afflictions such as chronic pain or agitations. There is also a secondary assumption that this application can be used to be an entry point for individuals who are interested, but hesitant, in learning how to meditate. What Mike Premi would like to the Oregon State University (OSU) Capstone Team to achieve is a prototype that can used as a proof of concept, and as an initial platform of development for a professional software development company.

This project is a collaborative effort with several Intel employees (Marissa Powers, Mike Premi, and Lindsay Benjamin), a University of Washington (UW) Human Centered Design and Engineering Team of Masters Students (Ricki Mudd - Team Lead, Kira Cassels, Ryan Moore, and Paul Townsend), and ourselves, the OSU Capstone Team. Mike Premi will be in charge of the general process and keeping each group connected and communicating. Marissa Powers will be our project lead and technical consultant. Lindsay Benjamin will be our meditation expert and our contact at Intel. The UW team will be responsible for doing the user research and project efficacy analysis.

1.2 Proposed Solution

Our capstone team has been requested to complete three project tasks by our stakeholders. The first task is to create a user interface capable of housing an arbitrary amount of modules for the user to choose from. The second task is to work with the master's capstone team at the UW to iteratively construct, a low fidelity prototype that will be used to test the efficacy of a marketable end product. There will be two to three prototype iterations to test product efficacy. The last task is to use the research data that the UW team has gained from the second task, and while working alongside the UW team and Lindsay Benjamin, we are to construct a high fidelity prototype of a meditation experience.

1.3 Timeline

Based on the Computer Science Senior Capstone class curriculum [1], the following is the intended development timeline for these tasks:

- Document: October 28, 2016
- Technology review: November 4 2016
- Design document: November 23, 2016
- Fall progress report: Finals week, fall term
- Alpha level release (with demo): Week 6, winter term

1. http://eecs.oregonstate.edu/capstone/cs/

- Beta level release (with demo): Finals week, winter term
- Winter progress report: Finals week, winter term
- 1.0 level release: Monday prior to Expo
- Engineering Expo: Tentatively May 19, 2016
- Final report: Finals week, spring term

Officially, the projects ending date is Friday of finals week (June 16, 2017).

1.4 Performance Evaluation

On successful completion of our project, we will have, at minimum, a high fidelity prototype with validated value for a targeted end user and validated value for Intel as a company. Specifically, the high fidelity prototype will demonstrate a holistic meditation experience based on user research provided to us, accessible via a menu system developed by our team. Consistent collaboration amongst teams will be important and will be maintained throughout the project's duration to direct the development of the high fidelity prototype. We will submit our final deliverables for approval by our stakeholders according to the timeline presented above.

2 Abbreviations, Acronymns, and Definitions

2.1 Acronyms

HMD	Head Mounted Display
IP	Intellectual Property
NDA	Non-Disclosure Agreement
UX	User Experience
VR	Virtual Reality

2.2 Definitions

Application	Application refers to the package of meditative experiences and scenes in the prototype. Throughout this documentation the terms application and prototype will be used interchangeably.
Assets	A presentation of any item that can be used in the unity game or project [1].
Head Mounted Display	A device used by the user to view the virtual environment.

Intel Stakeholder Team	Staff from Intel overseeing the progression and coordination of this project including both VRAM and UW Teams: Mike Premi (Project Founder), Marissa Powers (Project Lead), and Lindsay Benjamin.
Intellectual Property	Intellectual property (IP) refers to creations of the mind, such as inventions; literary and artistic works; designs; and symbols, names and images used in commerce.
Lead Time	The time between notification of a task and the period it's expected to be delivered.
Meditative Experience	A virtual reality scene where the user is intended to participate with the environment to help transition into and facilitate a meditation session.
Non-Disclosure Agreement	A written agreement between client and service provider that outlines what should and should not be talked about outside of the project's stakeholders.
Scene	It is a container for unity to store in use assets. The term Scene can in some cases can be synonymous with the word environment [1].
State	A single variant of an orientation and/or functionality of an object.[1].
Guide	While inside of an experience users will be guided through meditative states by cue's and an in VR entity.
Transition	During a meditative experience users will be guided through various stages of meditation - initial relaxation/stretching, experience, awareness. Transitions will be used to gently guide users through each of these meditative states.
Unity	A game development platform[2].
User Experience	User experience (UX) focuses on having a deep understanding of users, what they need, what they value, their abilities, and also their limitations
UW Team	A team comprising of the following University of Washington HDCE masters students: Ricki Mudd (Team Lead), Kira Cassels, Ryan Moore, and Paul Townsend.
Virtual Reality	Virtual reality is the term used to describe a three-dimensional, computer generated environment which can be explored and interacted with by a person
VRAM Team	A team comprising of the following CS Capstone students from Oregon State University: Erik Watterson(Team Lead), Padraig Gillen, and Makiah Merritt.

3 Technology Review

3.1 Introduction

The following is the review of the technology pertinent to the completion of the senior capstone project Meditation in Virtual Reality. In accordance with the senior capstone class guidelines, each member of the capstone group will be in charge of 3 different technologies and will write their researched expertise within this document section. Erik Watterson will be in charge of evaluating software for storing and distributing project builds, indoor lighting methods within a Unity application, and HMD (Head Mounted Display) selection to be used for this project. Makiah Meritt will be in charge of evaluating industry Game Development Kits and Video Formats as assets. Padraig Gillen will be in charge of evaluating audio formats in Unity, scripting languages in Unity, and cutting edge graphics cards capable of handling virtual reality applications.

3.2 Technologies

3.3 Storage and Distribution of Project Files

Technology Options

3.3.0.1 GitHub: Github is a free to use project repository solution popularly provided by, and used by, the Linux Foundation. It's popularized by its integration with Unix and Linux operating systems, while also being a social media-esk website. The website allows you to interact with your project submissions as though your blogging and commenting on your own web space within the Github community. There is an option for advanced accounts, such as educational or business accounts, to be inward facing and allow for user, or groups, privacy for their projects. The interface for Github is done either by a terminal, desktop application, or by the website, but all of which require a working understanding of revision control software in order to be used. That being said, Github is known for its implementation of revision control, being one of the world's best tools for that use. [3]

3.3.0.2 Google Drive: Google Drive is a free to use virtual drive. The storage it provides works like any other storage device, however the way it's accessed is traditionally via the Google Drive website. There are other means as well, such as cell phone and desktop applications. In interfacing with Google Drive, whether it be by desktop, phone, or web, so long as you have a working understanding of manipulating files within a UI on your system there should be only a minor learning curve. Google Drive also works in pair with other Google file services such as Google Docs. Any file stored on a virtual drive that corresponds with a Google service can be opened with that service directly from any of the three application being used. Unlike Github, all files start by being private. Other individuals can be added to a file or folder to which they can then be able to view or edit them, depending on the setting you choose. The last thing to note is that all Google Drive accounts are directly tied to an overarching Google account. So if an individual has a Gmail account, they also have a Google Drive account. [4]

3.3.0.3 Dropbox: Drop Box is a free to use or subscription based virtual drive. In a general sense, Drop Box stores files just like Google drive will. Allowing the ability to store any type of file required by the user. It has the same type of connectable applications as Google Drive, Web, desktop, and phone, and can also support user log in with a Google

Account. The tiered subscription levels that Drop Box offers are as follows: The free tier, which allows you 2 gigabytes of storage space and the use of the Drop Box applications. The Pro tier, which allows for 1 terabyte of storage space, 30 days of unlimited file recovers, basic view only sharable permissions. The Business tier, which offers everything in the Pro tier along with an unlimited storage cap, more in depth sharing permissions such as account transfer tools, collaboration tools such as manger run folder groups, and administration tools such as folder syncing tools and activity logging. The Enterprise tier, which offers everything in the Pro tier along with more willingness from the Drop Box support team to help the customer, such as advanced training lessons for admins and end users. [5]

Goals for Use In Design

Throughout this term, there will be a need to store our project files externally so that our stakeholders have access to them.

Evaluation Criteria

Since virtual reality applications tend to be fairly large in size, the technology we use will need a high data cap. We'll also need to guarantee a solution where gaining access to the service is isn't complicated since we can't assume all of our stakeholders are technical individuals or have the time to learn a new system. The service will need to respect user privacy and at least have an option to not be outwardly facing. Preferably, the technology won't have a low download speed cap. Lastly, It would also be preferable if the service was free.

Comparison of Options

Table 1: Comparison of Storage and Distribution Options

Criteria	**Github** [3]	**Google Drive** [4]	**Drop Box** [5]
File Size Cap	100MB	5TB	2GB
Ease of Access/Use	Hard learning curve	Easy learning curve	Easy to moderate learning curve
Privacy / Security	Option to change to in-ward facing (private)	Inherently private	Inherently private
Download Speed Limits	None specified	None specified	75% of bandwidth upload speed
Pay Wall	Free	Free	Free and tiered levels

Discussion of Options

The first issue that is apparent is Github's 100MB file size limit. This will cause issues with the VR builds that this group will be working on, as our current build is already 314MB's. This means that Github isn't even an option to choose. Beyond that, we became concerned about the 2GB of space that would be offered from Drop Box. As we can't guarantee that our development folder will be that small once we acquire a full set of assets for the project. Other than that, we had no real complaints about the features of the three solutions looked at.

Final Decision

The selected external storage solution we will be going with is Google Drive. This is mainly due to the ease of use and familiarity among the stakeholders, since all of our stakeholder and our capstone team have owned a Google account prior to this current project. Secondly, everyone involved in a document section listed within a virtual drive has the ability to view or edit it with little to no resistance from the service, something that cannot be said for Drop Box.

3.4 Unity Environment Lighting

Technology Options

3.4.0.1 Realtime Lighting: Realtime lighting is a way of using a shading algorithm such that it's easy to manipulate by the developer within the Unity Toolkit. To apply this method, a developer can use the directional, spot, or point light objects within a Unity scene. For each Realtime Lighting object it applies a geometric algorithm to the entire scene to determine the shading value of the pixel on the screen. These algorithms are updated once every second since this approach requires dynamic lighting changes. [6] [7]

3.4.0.2 Baked Global Illumination Lighting: Within 3D animation, the term "Baked" refers to an object having a static quality. In terms of lighting, this means that the shading will stay consistent and can be pre-processed by the system, instead of needing to have the system dynamically re-render the lighting. The way that the Unity toolkit developers decided to approach having a static lighting mechanism was to apply the lighting, or what they call a Lightmap, within the scene just like a material, texture, or mesh would be applied. [6] [7]

3.4.0.3 Precomputed Realtime GI Lighting: Just like with Baked Global Illumination, to create a precomputed lighting scheme means that the algorithm to create the lighting for the area needs to be calculated and stored before the scene is running. Since Real Time GI Lighting is based on the idea that the way the light interacts with the 3D world could change at any moment, the implementation of Precomputed Realtime GI Lighting needed to follow this same philosophy. To do this, the sections of the scene are broken down into clusters and each of those clusters have an individual lighting scheme applied to them. Since the precomputed approach requires retrieving data stored in memory, the clusters are small enough that manipulating them in real time isn't as taxing to the system. [6]

Goals for Use In Design

Our group will need to learn how to light indoor areas within our virtual reality scenes. Though it may be plausible to use multiple lighting techniques at once, we feel that we should be familiar with all methods but proficient in the best known method for indoor lighting.

Evaluation Criteria

We believe that a frame rate loss will be unavoidable, however that loss should be minimal. The lighting should illuminate the entire area, leaving only low contrast shadowing. The lighting technique applied shouldn't conflict with any other game object within the scene and cause the scene to stall or crash. A single halt within the program due to lighting will break the immersion for the user and must be avoided at all cost.

Comparison of Options

Table 2: Comparison of Unity Environment Lighting Options

Criteria	**Realtime Lighting** [6] [7] [6] [7]	**Baked GI Lighting** [6] [7]	**Precomputed Realtime GI Lighting** [6]
Framerate loss	Moderate	Minimal	Minimal to Moderate
Illumination	Illuminates single objects and not the surrounding scene.	Illuminates surrounding scene with a color bleed of the illuminated color.	Illuminates surrounding scene with a vague color bleed of the illuminated color.
Asset Conflicts	Other realtime computed objects within the unity scene.	Other precomputed objects within the scene.	Other precomputed objects within the scene.
Chance To Stall Program	High if there are conflicts.	Moderate if there are conflicts.	Moderate if there are conflicts

Discussion of Options

The issue we were largely concerned with was whether the baked or precomputed lighting would look fake within the scenery. What we found was that though you could tell the shading was applied to the object itself, it wasn't

inherently obvious unless we payed attention to it. We were actually surprised in how the Precomputed Real Time GI Lighting worked so much like the Baked Global GI Lighting. The ambient lighting looks very similar Baked Global GI Lighting however it's clear that it looks a bit differently once you reorient the direction of the HMD. Compared to the Realtime Lighting however, neither Baked GI Lighting or Precomputed Realtime GI Lighting look as splendid. In terms of computation and conflicts that could arise during development of the prototype, we believe that the Realtime Lighting would put a lot of strain on a system if there was a situation where there were a plethora of lighting sources within the area. For example, if we were to create a mass of candles within a scene to light the area, the system would be hard pressed to do the individual calculations on each of the candles.

Final Decision

Based on what we can derive from the lighting techniques, we feel that Precomputed Gi Lighting would be the most efficient way to approach most problems inherently. Given that it won't be an end all solution, we feel that it at least allows us to use memory stores instead of increase our computational complexity to achieve an adequate lighting option within our scene. This is rather important since most of the development issues we have seen while developing occur while in realtime as the scene is trying to manipulate itself.

3.5 Head Mounted Displays (HMDs)

Technology Options

3.5.0.1 Oculus Rift: Oculus Rift was the first consumer targeted VR HMD to be shown to the world. Developed by a man named Palmer Luckey, then later picked up by Facebook, it was designed to give a better user experience to the end user. At the time when Palmer was developing the initial Oculus Rift, Palmer realized that the other HMD's that were out there in the world were high cost, poorly constructed, display devices that were only used for research and testing. Since then, the Oculus has evolved quite a bit. Currently, it comes package with a Constellation Tracking Camera, which is used to track in 360 degrees around camera to locate the HMD. A item sold separately is the Oculus Touch which are two controllers that are to be used in conjunction with the HMD while in a virtual environment, allowing for easy hand tracking and environment manipulation. The headset itself is comprised of 2 OLED displays with the combined resolution of 2160 x 1200 and a refresh rate of 90hz. The Oculus also has two semi adjustable earpieces attached to the sides of the HMD for audio needs. [8] [9] [10] [11]

3.5.0.2 HTC Vive: The HTC Vive was a joint venture between Valve and HTC. In the beginning, both companies contemplated entering into the VR world and had labs dedicated to developing with augmented and virtual reality hardware. Around this same time, the Oculus Rift peeked its head out and Valve offered to help the Oculus team to some tracking features within the Rift. At some point, communication broke down between the two companies and Valve turned elsewhere to venture into the VR market. A short while later, HTC and Valve met and quickly hammered out a deal for the HTC Vive. In its current form, the Vive comes stock with the HMD, two controllers, a pair of earbuds that plug into a port in the back of the HMD, and two Lighthouse Tracking Sensors. These Lighthouse Tracking Sensors are used to check the distance from one sensor, to the vive, to the other sensor, in order to figure out where the user is

located in real space. The Headset itself is markedly just like the oculus in terms of video display, being comprised of 2 OLED displays with the combined resolution of 2160 x 1200 and a refresh rate of 90hz. [9] [10] [12] [13]

3.5.0.3 Google Cardboard / Samsung Gear VR: Google Cardboard and Samsung Gear VR are both mobile phone VR systems where the HMD is intricately made to holster a smartphone. This means that the power of the VR experience is derived from the smartphone system that's attempting to be used with the holsters. For the Samsung Gear VR, its compatible with any phone running Android 4.4 and higher and has at least 1.5GB of RAM. Conversely, Google Cardboard requires only Android version 4.1 and higher, or iOS 8.0 and higher. Because the Samsung Gear VR is marketing itself as a cutting edge VR system, it shows to reason as to why you're only able to run Android 4.4 and higher (Galaxy Note 4, Galaxy S6, and Galaxy S6 Edge are the only systems that run android 4.4) with the Gear VR holster. With that said, the cost of entry is substantially different between the two kits. Google Cardboard will run the consumer roughly $20 USD, whereas Samsung Gear VR will run the consumer roughly $200 USD. [14] [15] [16] [17] [18]

Goals for Use In Design

It is our goal to use an HMD in conjunction with our virtual reality development kit to create a virtual reality guided meditation prototype. This prototype will be used to test the efficacy of whether or not it's applicable to join meditation with VR. To best evaluate the prototype, it's crucial that we choose the most optimal HMD to be used for testing.

Evaluation Criteria

As the user could be wearing the HMD for 30 minutes or more, it's important that the HMD fits comfortably on the user's head so the HMD should come with adjustable features that fit the individual's head. The second criteria we are looking to evaluate is the video resolution quality, as we want to make sure that the video quality doesn't break immersion. Lastly, we want to make sure that the tracking within the virtual space is seamless, meaning there are no sudden movements based on the HMD reorienting itself in space.

Comparison of Options

Table 3: Comparison of Head Mounted Display (HMD) Options

Criteria	**Oculus Rift** [8] [9] [10] [11]	**HTC Vive** [9] [10] [12] [13]	**Google Cardboard** [14] [16] [17] [18]	**Samsung Gear VR** [15] [16] [18]
Comfort Adjustment Features	Adjustable straps for length of head. A knob to adjust the screen distances within the HMD. Foam faceplate.	Adjustable straps for length and width of head. A knob to adjust the screen distances within the HMD. Foam faceplate. No plastic sits on the face and head.	None	Single adjustable strap around head to change tightness.A knob to adjust the screen distances within the HMD. Foam faceplate. No plastic sits on the face and head.
Video Resolution Quality	2160 x1200	2160 x1200	Variable	Variable
Tracking Methods	Two point confirmation on HMD and controller locations through Lighthouse sensors. Gyroscope within the HMD.	A single 360 degree constellation sensor to track the headset and controllers. A gyroscope with the HMD.	Based on gyroscope within the phone, no secondary tracking.	Based on gyroscope within the phone, no secondary tracking.

Discussion of Options

Given that there are too many variables to consider for mobile app development, we don't consider a Google Cardboard and Samsung Gear VR to be valid HMDs for this project. On top of that, since the only tracking method for the cell phone based VR is the cell phone itself, there is nothing to guarantee stability if the cell phone's gyroscope runs into an issue. As for the Oculus Rift, we don't necessarily like the idea that the main tracking system is based on a single sensor. This could potentially cause issues if you were to turn around and face away from the censor. The comfort related inquiry into the Oculus Rift also seemed a bit dubious, as the plastic side bars that attach to the head straps aren't very forgiving to the width of a head. Comparatively, the HTC Vive has complete customization of head straps. Allowing for length and width of a head.

Final Decision

Our belief is that the HTC Vive will be the best selected HMD for our project. This is primarily due to the dual Lighthouse Sensors which give a better chance of stabilizing the headset. It can be very easy to become disoriented in VR, and as a tool designed to relax the user we want to limit any chance the user has to becoming disrupted. Beyond that, we feel the HTC Vive's hardware has good comfort control and in most everything else it's comparable to the Oculus Rift.

3.6 Development Kits

Technology Options

3.6.0.1 Unity: Unity is a game development platform for 2D and 3D games. With broad platform support and vast community expertise, the Unity platform accommodates to both novice and expert developers. C# and JavaScript are used to create the various animations within a world space.

3.6.0.2 Unreal Engine: Unreal Engine is an industry leading game development platform. Providing the ability to create both 2D and 3D games within a single environment it is a powerful suite commonly used for console and desktop games. C++ and Blueprints (a flowchart like scripting process) are the two primary languages implemented by Unreal to give developers access to their world spaces.

3.6.0.3 CryENGINE: Also a major game development platform, CryENGINE is a feature packed game development platform aimed at providing high end visual effects and immersive environments.

Goals for Use In Design

We will be using a Game Development kit heavily throughout our project because our goal is to build interactive environments in a the Virtual Reality space for users. Through the development kit our end goal will be the generation and manipulation of the virtual reality environment, the definition and interpretation of player controls, and the implementation of circumaural audio to complete the immersive experience. As such we will expect the kit to provide access to present controls and properties to manipulate each of these features or for the manual definition of interfaces.

Evaluation Criteria

Cost

1) What is the total cost to use the product for the duration of this project?

Compatibility

1) Which head mounted display devices are supported by the development suite?

Plugins and Assets

1) 3D Models - What types of (3D) models can be imported and used within the development suite?

2) Internal Marketplace - Is there a community market place from which assets (textures, materials, objects, sounds, effects, etc.) can be obtained?

Development

1) Learning Curve - What is the difficulty newcomers will face adjusting to the development suite?
2) Documentation - How useful is the standard documentation for the development suite?
3) Available Programming Languages - What are the primary programming languages used by the development suite to control objects, create effects, and generate actions?

Comparison of Options

Table 4: Comparison of Development Kits Options

Criteria	**Unity** [19] [20] [21]	**Unreal Engine** [22] [23]	**CryENGINE** [24] [25]
Cost	$0.00	$0.00	$0.00
HMD Compatibility	Oculus Rift Samsung Gear VR Playstation VR Microsoft HoloLens Steam VR HTC Vive Google Daydream	Samsung Gear VR Google Daydream Oculus Rift SteamVR HTC Vive	HTC Vive Oculus Rift Open Source VR
Plugins and Assets			

Continued on next page

Table 4 – *Continued from previous page*

Criteria	**Unity**	**Unreal Engine**	**CryENGINE**
Models	Because Unity lacks a full fledged 3d modeling system, it can import the following formats (as well as more proprietary formats): • .FBX • .OBJ • .MAX • .Blend • .dae • .3DS • .dxf	Primarily .FBX and .obj, other types are likely supported but aren't specifically outlined in documentation.	Yes See: http://docs.cryengine.com/display/SDKDOC2/Art+Asset+File+Types
Community Marketplace	Yes See: https://www.assetstore.unity3d.com/en/	Yes See: https://www.unrealengine.com/marketplace	Yes See: https://www.cryengine.com/marketplace
Development (Subjective Opinion)			
Learning Curve (for beginners)	Low - Moderate	Moderate - High	Moderate - High

Continued on next page

Table 4 – *Continued from previous page*

Criteria	Unity	Unreal Engine	CryENGINE
Documentation	Great, well laid out and references to documentation are interlaced with application features. Documentation is also available with installation (allowing it to be accessed offline). Documentation is split between an interface manual and a scripting API. See: https://docs.unity3d.com/Manual/index.html	Present and useful for most information, but can be cryptic depending on the desired type of information. No offline documentation. See: https://docs.unrealengine.com/latest/INT/	Similar experience to Unreal. See: http://docs.cryengine.com/display/SDKDOC1/Home
Programming Languages	C# and JavaScript for most interactions. ShaderLab is Unities proprietary shader language.	C++ and Blueprint.	C++ and Lua primarily. Scripts may be written with C#.

Discussion of Options

The three development kits outlined above are all common kits in today's Game Development market. All three support Virtual Reality Devices and appeal to new users with free versions. Furthermore, although all three can be used to develop for desktop platforms it is evident that Unity is the least mature in this arena, spreading its roots widely in the mobile audience whilst Unreal Engine and CryENGINE are heavy hitters in the desktop and console development arenas. The learning curve of each tool is expressed by this understanding as well with Unreal and CryENGINE appearing to possess a higher learning curve than Unity. Lastly, each development kit supports importing models and other assets.

Final Decision

Based upon the relatively low learning curve and openly accessible documentation our project will rely upon Unity. Although, Unreal Engine and CryENGINE naturally yield themselves to higher quality content (speaking namely of graphics and physics effects) these features would most likely fall to the wayside as we maintain a focal point on the

feasibility of Meditation within VR. Additionally, Unity provides the greatest ability to branch to new platforms and possesses a low system requirements for development and running Unity applications.

References: [26][27][28]

3.7 Unity Video Formats

Technology Options

Note: *container* formats encapsulate audio, video, and subtitles as a single file.

3.7.0.1 .mp4: MP4 is a video format created based largely upon the MOV file format. MP4's are widely accepted because of its universal nature and low size.

3.7.0.2 .avi: AVI, short for Audio Video Interleave, is a container developed by Microsoft for its media player application [29]. According to the File.org site, AVI files have less compression than its MPEG or MOV competitors but is limited to 2 GB in size [30].

3.7.0.3 .mov: The MOV file format is a codec developed by Apple for their Quicktime media player. Because of its proprietary origins MOV wasn't as widely adopted by non-Mac users and led to the creation of alternative non-proprietary formats. MOV also heavily features the H.264 because of its high video quality and

Goals for Use In Design

Currently, one of the experiences we would like to test is the use of 360 video to capture a serene environment. Within this environment we would allow users to practice meditation, whether guided or unguided. To realize this task we would implement high quality video that is also of sufficient frame rates so users don't experience any adverse effects of viewing within an HMD. At the time of this writing, Unity supports video through animated textures called Movie Textures. Supported formats are those playable by Apple's QuickTime: .mov, .mpg, .mpeg, .mp4, .avi, .asf; from this list we'll examine .mp4, .avi, and .mov.

Evaluation Criteria

1) Device Compatibility
   - The device limitations, such as Mac vs. PC or iOS vs Android, that inhibit the formats adoption to new platforms.
2) Video Compression
   - The video codec used for encoding video media within the given format.
3) Audio Compression
   - The audio codec used for encoding audio media within the given format.

4) Subtitle Support

- Whether or not subtitles can be used with this video format.

5) File Size

- Relative file size once a video has been encoded.

Comparison of Options

Table 5: Comparison of Unity Video Format Options

Criteria	.mp4	.avi	.mov
Device Compatibility	Universal [29]	Universal	All Apple and Modern Windows (PC) devices * Older PC devices require codec extensions.
Video Compression	MPEG-4 or H.264	[2]	Primarily H.264
Audio Compression	AAC or AC3	[3]	Primarily AAC
Subtitle Support	With plugin [31]	With plugin [31]	Native [31]
File Size	Small - Medium [32]	Large [32] with 2 GB max [30]	Large [32]

Discussion of Options

Of the discussed file types mov and avi file types are "considered the most flexible and compatible video file formats" [32]. Additionally, mov and avi files are low compression whereas mp4 has high compression. The differences in compression creates variations in video quality in exchange for size and access speed [33]. MOV and MP4 videos can

2. Although MPEG-4 was mentioned to be common, detailed info could not be found. Speculatively, as with MOV most popular video codecs be usable.

3. Info could not be found. Speculatively, as with MOV most popular audio codecs are usable.

be of similar quality depending on which video codec is used during their creation, should one use the same code for MOV as that of MP4 the gains of MOV are minimized.

Per Unity's documentation for its Movie Texture we must consider the limitations of playable formats and a final conversion that occurs during the import process. Beginning with format limitations Unity, as noted, respects anything playable by Apples Quicktime beyond this for maximum compatibility the video format must utilize H.264 Baseline Profile Level 3 [34] and MPEG-4 Part 2 for iOS devices followed by H.263, H.264 AVC, and MPEG4 SP for Android devices. Each file format being presented supports the use of the mentioned codecs. It must be noted that the nature of AVI, MOV, and MP4 are fundamentally different. AVI and MOV are video containers what can use files of type H.264, MPEG-4, AAC, etc. whilst MP4 is a video file created using H.264 or MPEG-4 (for its video) and AAC (for its audio). The import process for Unity's Movie Texture automatically converts the chosen file type is automatically converted into the Ogg Theora file format; an important to our final video format decision because the conversion will impact video quality. Ogg Theora, like AVI and MOV, is a container format created by the Xiph organization. Furthermore, "Ogg is a stream oriented container...[a] major design difference over other container formats" [35]. Likely, Unity features this video format because of its open source nature and has built in functionality that can take advantage of the formats bit-stream.

Final Decision

Although high video quality is important for production releases, our prototypes don't require the use of high quality videos. Additionally, obtaining 360 videos for testing has proven slightly more difficult than anticipated. As such we would anticipate the difficulty of obtaining high quality 360 videos to be substantially higher, unless they were provided directly from our sponsors. These factors lead us to recommend the use of .mp4 files. Their lightweight file size, minimal quality loss, and universality means they are easy to obtain and embed within our builds.

It seems appropriate to note here that the best file choice would be .ogg as those can be migrated into Unity without hassle. As it currently stands, Unity requires Apples Quicktime to be installed. Once installed developers must load Unity as an administrator in order to import videos, so it can make use of Quicktimes Quicktime Tools.

References: [36][37][38][39][40][41][42][43][44]

3.8 Unity Audio Formats

Technology Options

3.8.0.1 **Native**: The audio file is stored uncompressed on the hard drive, in a format like WAV or AIFF. This will require a lot of storage space and time spent reading disk. No decoding needs to be done at run time, [45] which means a low CPU footprint.

3.8.0.2 **Compressed**: The audio data is compressed before storage, which means a small file size, but it requires the CPU to decompress before playing. This can happen in 3 ways: Decompress On Load, Compressed In Memory, or

Streaming [46]. Each method increasingly trades a lower memory footprint for a slower sound response. Since this app will not run on mobile platforms, the encoding will be Ogg Vorbis.

3.8.0.3 **Tracker Modules**: Tracker modules allow the developer to include only a few necessary sounds, and then programmatically tell the game when to play each one. It is very similar to MIDI, but with provided sounds for the sound card to play, thus avoiding a dependence on the sound card's implementation of a noise [47]. The main advantage is in creating chiptune like beats for games, as it uses PCM audio storage and is essentially the same quality as native without live repetition or modification.

Goals for Use In Design

In order to build an immersive VR experience, the audio must be seamless. As our primary user instructions are often delivered from audio tracks, the audio must be high fidelity. This means no timing issues, distortion, or fuzziness should be noticeable by the user. Furthermore, all transitions should be smooth, even if different audio tracks are playing simultaneously.

Evaluation Criteria

- Sound file attainability / creation effort
- Sound Quality
- File Size
- CPU Usage
- Live playback timing
- RAM Usage

Discussion of Options

Our project has some flexibility due to it being a series of prototypes. We can expect a high powered CPU to be present when running the application. Additionally we have no hard upper boundary on project file size, but would like to keep it as nimble as possible. This means we could use uncompressed audio if necessary without compromising the project's goals. However, it seems likely that compressed audio will have suitable audio quality, while also allowing for smaller executable and faster loading times from the hard disk. Since the CPU will not likely be a bottleneck, we can sacrifice some processing power to decoding the tracks. Finally, tracker modules will likely go completely unused in this project. Most audio tracks involve things like ambient noise, voice recordings, and sound effects. None of these will need pitch shifting or melodic arrangement to justify the increased developer time in creation.

Final Decision

This project will largely use compressed (Ogg Vorbis) audio tracks. With a lower compression ratio, we can maintain all noticeable audio fidelity. Additionally, this will keep file size low, which ensures the sound can quickly be played as the disk is read. Streaming decoding allows near instant results and takes advantage of the CPU power available to us. If we need to add short, repeated sound effects of some kind, we can use ADPCM compression with *decompress on load* instead to allow for zero latency, high quality audio playback.

References: [45][47][46]

3.9 Unity Scripting Languages

Languages supported in Unity 5

3.9.0.1 **C#**: A fairly well used object-oriented language, present in 80% of Unity projects [48]. It is strongly typed, and is the default option when creating a Unity script. It supports a syntax similar to C, and can be viewed as an addition of mandatory classes.

3.9.0.2 **UnityScript**: This is Unity's JavaScript implementation, and is sometimes simply referred to as JavaScript. It is used in roughly 20% of Unity projects, and has a reputation for being more beginner friendly. According to community built documentation, UnityScript differs from JavaScript in a few major ways: [49]

- Addition of classes, which can be assumed by filename
- Stricter syntax (must have semicolons, no variable assignment as an expression)
- No global variables, adds support for private variables

3.9.0.3 **Boo**: A .NET language with Python like syntax, Boo is an unofficially supported scripting language for Unity. It is statically compiled [50]. The official scripting documentation does not mention its existence [51], but the editor will still support any Boo scripts included in a project.

Goals for Use In Design

We would like to use a scripting language that is easy to develop in, but minimizes errors, both in-game and before compilation. This means it should be expressive enough to quickly develop a concept without a ton of necessary boilerplate. The language must also support a strong object-oriented model to work well with the Unity engine, as every game object already inherently has properties and methods.

Evaluation Criteria

- Documentation
- Ease of learning

- Community support

Comparison of Options

Table 6: Comparison of Scripting Languages

Criteria	**C#**	**UnityScript**	**Boo**
Language Typing	Strong	Dynamic	Static
Unity project usage	80.4%	18.9%	0.44%
Unity examples	Yes	Yes	No

Discussion of Options

Our development team has the most familiarity with C# from the beginning, and have found it relatively straight forward to develop with, as well as migrate in plugin scripts written with it. We are comfortable with the way it interacts with the Unity editor, and generally find it easy to work with. UnityScript would be fairly easy to pick up, and would be easy to port any existing code to. However, with Visual Studio's tab completion and ease of access to documentation, it doesn't seem that using UnityScript would simplify the development process very much. Both languages have a huge advantage over Boo, as they are supported in the official documentation and examples. Boo might be nice for quickly iterating through versions of our prototype, but will prove difficult to find resources on debugging VR issues.

Final Decision

We're going to develop with C#. It enjoys the highest community support, and has documentation beyond just Unity guides. With its object oriented model we can use very high level abstractions without losing access to the power of the underlying C. Stronger typing leaves less room for errors that might be present in UnityScript.

References: [48][49][50][51]

3.10 GPU selection for VR

Potential graphics cards

3.10.0.1 **NVIDIA Titan X Pascal**: NVIDIA introduces this card on their website with the statement: "We packed the most raw horsepower we possibly could into this GPU." This is their leading GPU, and represents an upper limit on the

stock power available from consumer graphics cards right now. It runs on NVIDIA's Pascal architecture, and includes 12 GB of memory at a speed of 10 GB/s [52].

3.10.0.2 **NVIDIA GeForce GTX 1080**: This is NVIDIA's flagship graphics card, and a standard choice for high end consumer builds. It also runs the Pascal architecture, and comes with 8 GB of memory at the same speed of 10 GB/s [53].

3.10.0.3 **AMD Radeon RX 480**: This card is a cheaper alternative to some of the NVIDIA offerings, while still offering up to 8 GB of memory. The main selling point comes from a very low base price tag of $199, though the 8 GB model will run more in the $240 range [54].

Goals for use in design

- Support output to chosen HMD, as well as at least two monitors for development.
- Support high framerates in our app (target 90fps)
- Cost / availability
- Compatibility with HMD
- Performance

Comparison of Options

Table 7: Comparison of GPU Options

Criteria	**NVIDIA Titan X Pascal**	**NVIDIA GeForce GTX 1080**	**AMD Radeon RX 480**
Cost	$1200	$699	$199+
Core count	3584	2560	2304
Clock speed [base / boost] (MHz)	1417 / 1531	1607 / 1733	1120 / 1266
Memory	12 GB GDDR5X	8 GB GDDR5X	4 / 8 GB GDDR5
Memory bandwidth (GB/s)	480	320	224+

Continued on next page

Table 7 – *Continued from previous page*

Criteria	NVIDIA Titan X Pascal	NVIDIA GeForce GTX 1080	AMD Radeon RX 480
Power usage	250 W	180 W	150 W
DirectX 12 support	Yes	Yes	Yes

Discussion of options

All three of the considered GPUs are on the Vive recommended lists, and can be reasonably expected to work seamlessly with Windows. We can thus expect all 3 will support our game. We can reasonably expect the target businesses to afford a 1080, while the Titan X may have no substantially better return. The faster memory bandwidth will assist in easily reaching high frame rates, allowing us to experiment with more computationally expensive visual effects.

Final decision

As the stakeholder has provided a GTX 1080, we will use it for development. It seems to be doing fine with handling all the prototype's graphical load as we have developed. The increased expense of a Titan X does not seem to be justifiable at this time.

References: [52][53][54]

3.11 Conclusion

In this document section we have examined the use of various Storage and Distribution technologies, Lighting Techniques within Unity, current generation Head Mounted Displays, Game Development Kits, Video Formats, Audio Formats, and current generation GPUs that we could use throughout the duration of the Meditation in Virtual Reality project. Our final conclusions are as follows -

- **Storage Solution**: Google Drive, because of its familiarity among us as well as our fellow project collaborators.
- **Lighting Techniques**: Precomputed GI, because of its efficiency.
- **HMD Choice**: HTC Vive, because of dual sensors keeping a stabilized headset as well as the physical devices comfort during use.
- **Game Development Kit**: Unity, similarly to our selection for storage, our collaborators are familiar with Unity which will lead to better communication pertaining to deliverables. With Unities documentation and a diverse community, resources will also be abundant.

- **Video Format**: .mp4, because of its universal nature, decent quality, lightweight file size, and support within Unity.
- **Audio Format**: Compressed, because of the lower compression ratio we'll be able to keep file sizes low and provide quick playback time.
- **Unity scripting language**: C#, as it has the largest VR documentation, and allows us to easily work with community provided packages.
- **GPU**: Nvidia's GTX 1080 delivers the best performance for the price.

4 PROJECT DESIGN

4.1 Scope

This document section will cover design decisions pertaining to the creation of virtual reality environments, hereafter referenced as prototypes or experiences. Specifically this section will elaborate on the design decisions for the following key aspects of the virtual reality project:

- Project Management Specific
  - Storage of Prototype Builds
- Unity Specific
  - Environment Lighting
  - Audio Assets
  - Scripting In C#
- Hardware Specific
  - HTC Vive
  - Hardware Restraints

4.2 Purpose

The Virtual Reality in Meditation project aims to test the feasibility of meditation within virtual reality. Sponsored by Intel and in collaboration with masters students capstone team in the University of Washington HDCE program, prototypes will be used to examine the effect of meditation on individuals while in a virtual reality experience. This document section serves to outline the techniques and practices used in the development of prototypes used for said research.

4.3 Intended Audience

This document section will serve to inform the collaborators of this project about the design decisions the Oregon State University capstone team will progress with. As our collaborators come from various backgrounds this document section will be accessible to a general audience as any necessary knowledge will be presented within the immediate section. Ultimately, this section targets those with technical knowledge about game development possessing interest in virtual reality and/or meditation.

4.4 Conformance

This document section shall present design information and viewpoints pertaining to the creation of prototypes for this project.

4.5 Conceptual Model for Software Design Descriptions

4.5.1 Stakeholders and Stakeholder Concerns

Mike Premi Mike Premi the project founder, sponsor, and manager for the Meditation in VR project.

Marissa Powers Marissa Powers is the Meditation in VR project leader. She will be the main point of contact for the Oregon State Capstone Team to receive direction for prototype builds.

Lindsay Benjamin Lindsay Benjamin will be our storyboard expert and first point of contact for questions relating to the meditation experiences.

University of Washington Capstone Team This team is in charge of determining the efficacy of developing a product that combines meditation with VR. To do this, they will be conducting user testing on the prototype builds that the Oregon State Capstone team produces.

4.5.2 Design Views

The following are common views that will be discussed throughout this document section.

Users Users are the individuals who will partake in meditative experiences through developed prototypes. They do not need to posses any knowledge concerning the development of the project or knowledge concerning meditation practices. Users do however need to be open to meditation, as this will impact their ability to allow themselves to be fully immersed in an experience.

Developers Developers are classed as the individuals who will need detailed knowledge of the software tools and hardware used throughout this project. Specifically, developers will learn to posses an intimate knowledge of Unity and Unity's ecosystem; because this knowledge will be crucial for the development of accurate, quality, and enjoyable prototypes.

Collaborators Collaborators serve as a subsection of the developer viewpoint. Where developers must have a detailed understanding of the system as it pertains to the development of prototypes. Collaborators need only a high level knowledge such that they can explain their needs and desires accurately; knowing what is and isn't feasible within the Unity platform. However, collaborators also do not need to posses knowledge concerning

	the development of prototypes but instead might have crucial insights in other fields helping to guide the prototype design.
Unity Ecosystem	The Unity Ecosystem is a term used to describe the many features of Unity and its role as a development kit. As will be discussed in the Design Descriptions later, Unity contains almost all necessary tools for the development of prototypes. Additionally, Unity will use plugins to allow us to interface with hardware devices that in turn interact with users.
Software Design descriptions within the life cycle	Since this project is a research based project, we have tentative time frame for deliverables. As of this moment, our University of Washington Capstone Team stakeholders speculate that they would like to start user testing in early January with a low fidelity VR prototype, have a second testing period in the beginning of February with a medium fidelity VR prototype, and a final user testing period in late February with a high fidelity VR prototype. The development of the prototypes will follow a agile work flow model where we will be in contact with our team lead Marissa Powers on a weekly basis, to verify a currently active user story development. These user story deliverables will have variable time frames attached to them dependent on the input from the Intel Team and the University of Washington Capstone Team Stakeholders.

4.6 Design Descriptions

The following content outlines design choices for various components of the Meditation in Virtual Reality Application. Both hardware and software design aspects will be addressed to provide a fuller picture of the cooperation of components within the system.

4.7 Design Component: HTC Vive

Introduction

We will be discribe how the HTC Vive will be used to for the development of our application. This section has been authored by Erik Watterson.

Scope

In the section we will discuss what the HTC Vive is, how the HTC Vive will be used within the design of the software, the persons involved with the use of the HTC Vive within the design of the software, any design concerns of the stakeholders relating to HTC vive, a listing of the design viewpoints, the design views of those viewpoints, and the design rationale of using the HTC Vive while articulating any connections it may have with any other design subjects within this document section.

Description

The HTC Vive is an HMD that is used to give a single user a visual and auditory virtual experience. It comes stock with two lighthouse style laser sensors, two hand held controllers, and a pair of headphone earbuds that connect to a headphone port in the back of the HMD. [13]

Purpose within the Project

The HTC Vive will be used as the primary virtual reality medium for user testing within the Meditation in VR prototype. As such, we will be using the HTC Vive to help develop and test our prototype.

Stakeholders and Stakeholder Concerns

The pertinent stakeholders to using a HTC Vive are the user testing participants, the Oregon State University Capstone team developing the Meditation in VR prototype, and the University of Washington Capstone team in charge of user testing.

Design Viewpoints

The Design View used within this Design Subject is a Context View of a user using the HTC Vive.

Applicable Design Views

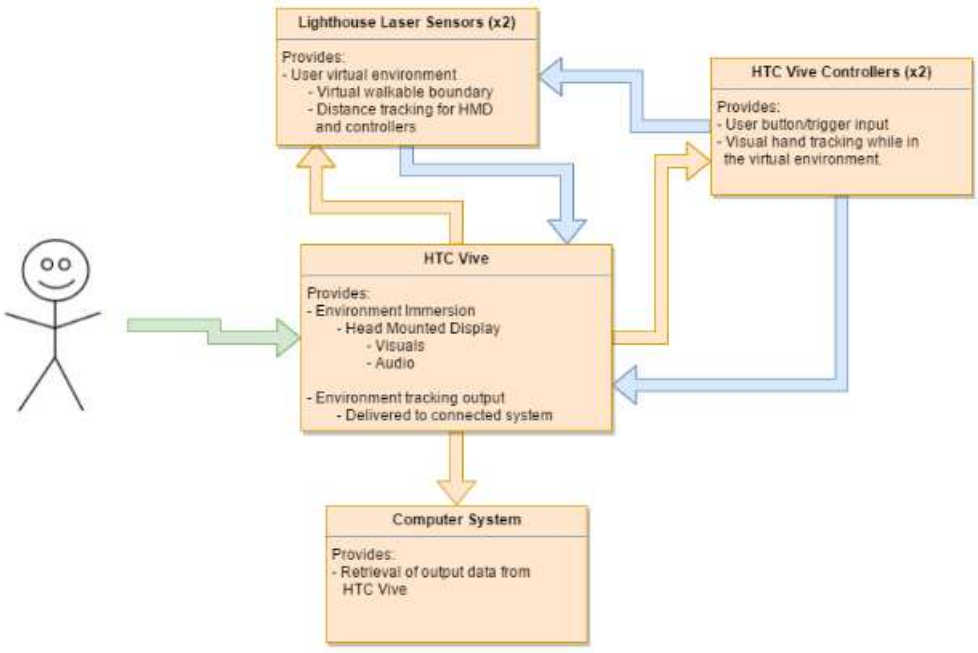


Figure 1: Context View diagram showcasing the data flow from one piece of hardware to another for the HTC Vive.

Within figure1, the green arrow represents user input, the orange arrows represent data transferred from the HTC Vive, and the blue arrows represent data transferred to the HTC vive.

Design Rationale

The method in which we will be using the HTC Vive hardware is the standard method promoted by the HTC Vive documentation. Specifically, we will be use the two lighthouse style laser sensors to track the HMD movement and the two controllers movement and activated inputs within the designated physical and virtual environment area. This tracked data is then transmitted back from the lighthouse sensors to the HMD which is cable linked to the computer system via an HDMI cable. [13]

For the hardware to integrate with another one of our Design Subjects, the Unity Developer Platform, the HTC Vive developers have provided Unity developers an asset package that connects with the Vive hardware API. Allowing event based applications based on the inputs from the HTC Vive hardware to be ran within Unity. [13] citeUnityInfo

4.8 Design Component: Environment Lighting

Introduction

We will be discribe how we will be designing the Environment Lighting within our Meditation in VR prototype. This section has been authored by Erik Watterson.

Scope

In the section, we will discuss what styles of Environment Lighting we will be using, how the Environment Lighting will be applied within the design of the software, the persons affected by the Environment Lighting within the design of the software, any design concerns of the stakeholders relating to Environment Lighting, a listing of the design viewpoints, the design views of those viewpoints, and the design rationale of using the Environment Lighting Methodology we chose while articulating any connections it may have with other design subjects discussed within this document section.

Description

Within our Meditation In VR application, it's important to implement lighting techniques that cast light which gives the environment a feeling of realism. For example, if we were to implement a lighting technique that was below par within a user test, the participant could clearly see pixelation within the shading, that could influence the outcome of the user research. In the worst case scenario it could even invalidate the user test with video tearing or the creation of digital artifacts. Another aspect within this design subject we must account for, that can very easily cause invalidation of a participant's test, is a loss of frames per second. It is an industry standard to have a VR application run at ninety or more frames per second. There is a strong correlation that as frame rates fall below that threshold, there is an increase in chance that the user will notice the frame rate loss and break immersion from the application. If the frame rate loss is high enough and not accounted for, it can become disorienting to some individuals, with the potential to induce headaches or nausea. This of course would portray an unethical treatment of a testing participant and must be avoided at all costs. [55] [56] [6]

Purpose within the Project

Though the prototype builds we make can be run without any directional lighting, in order to attribute realism to our Unity scenes we must outline how we can attribute shadows and cast light. Within an outdoor Unity scene, we will primarily use a real time directional lighting object within our scenes and secondarily use a precomputed realtime GI lighting method for when there is a need to introduce other lighting sources into the Unity scene. For indoor Environments, we will be Primarily using a Precomputer Realtime GI Lighting method for all light sources within the Unity scene. All light sources that will be using the Precomputed Realtime GI Lighting method will be static Unit objects. [6]

Stakeholders and Stakeholder Concerns

The pertinent stakeholders that will rely on good lighting development practices are any potential user testing participants, the Oregon State University Capstone team developing the Meditation in VR prototype, the University of Washington Capstone team in charge of user testing, and our Intel Stakeholder and storyboard expert Lindsay Benjamin.

Design Viewpoints

The Design View used within this Design Subject is a Logical View of the how to apply a directional lighting object, or use a pre-computed real-time GI lighting method within the scene.

Applicable Design Views

Design Rationale

The reason why a secondary lighting technique is so important is because if you apply multiples of the same style of lighting object within a single scene the algorithmic complexity of the scene increases at a rate that the not all computers can handle. If we apply lighting that is computed at compile time mixed with lighting that that is computed at realtime, we can help the computer delegate half of its workload more efficiently. By doing this, we can still keep the rich lighting that comes from realtime lighting and still have a lower algorithmic complexity that will affect video performance. [55] [56] [6]

4.9 Design Component: Google Drive Housing for Project Files

Introduction

We will be discribe how we will be using Google Drive within to store our various Meditation in VR prototype builds and related documentation. This section has been authored by Erik Watterson.

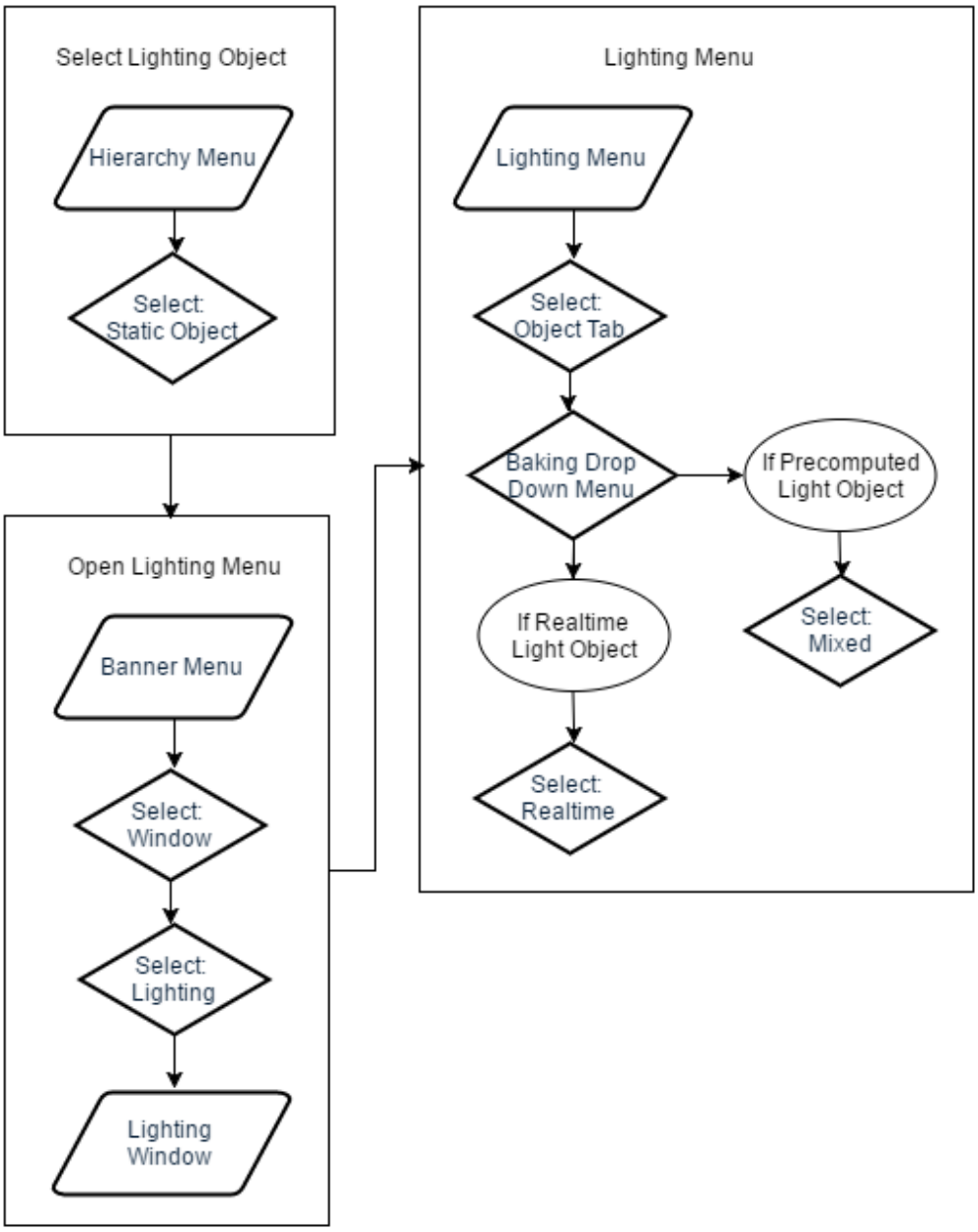


Figure 2: Logical View diagram showcasing how you can turn a static light object into one of the 2 specified lighting objects.

Scope

In the section we will discuss the storage medium Google Drive, how Google Drive will be used as a storage medium, the persons affected by storing our project builds within a Google Drive, any design concerns of the stakeholders relating to Google Drive, a listing of the design viewpoints, the design views of those viewpoints, and the design rationale of using Google Drive while articulating any connections it may have with any other design subjects within this document section.

Description

This description of Google Drive is the same as the one found in our Tech Review document section. Google Drive is a free to use virtual drive. The storage it provides works like any other storage device, however the way it's accessed is traditionally via the Google Drive website. There are other means as well, such as cell phone and desktop applications. In order to interface with Google's applications, whether it be by desktop, phone, or web, so long as you have a working understanding of manipulating files within a UI on your system there should be only a minor learning curve. Google Drive also works in pair with other Google file services such as Google Docs. Any file stored on a virtual drive that corresponds with a Google service can be opened with that service directly from any of the three application being used. [57]

Purpose within the Project

Our intention by using Google Drive is to safely harbor the myriad of work relating to this project. This includes our Meditation in VR prototype builds, assets relating to those builds, documentation on those builds, and documentation relating to the project as a whole. A secondary feature of Google Drive is the ability to moderate who has access to the file located within the Google Drive.

Stakeholders and Stakeholder Concerns

The pertinent stakeholders to use Google Drive are the Oregon State University Capstone team developing the Meditation in VR prototype, the University of Washington Capstone team in charge of user testing, and our Intel Stakeholder group who will need to track development.

Design Viewpoints

The Design View used to evaluate Google Drive is a Dependency View of how the documents will be shared between the different stakeholders.

Applicable Design Views

Design Rationale

Google Drive was a primary choice due to the high single file upload size limit which will allow us to store large prototype builds. Since our stakeholder teams are located in Beaverton Oregon, and Seattle Washington, it's unfeasible to hand deliver a the various builds. Another aspect we had to keep in mind while choosing a storage medium was how that medium handled user security. Since our content is considered the intellectual property of the Intel Corporation, we needed to make sure that it couldn't be accessed without consent. [57]

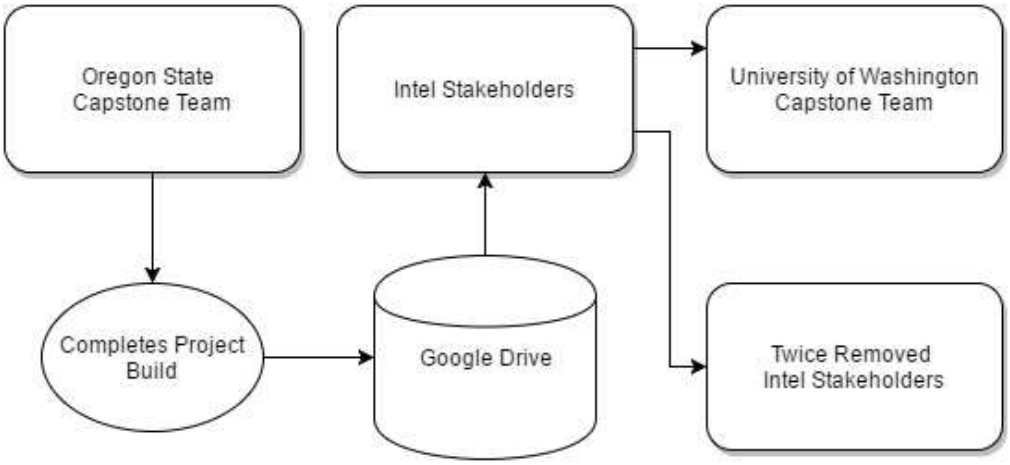


Figure 3: Dependency View diagram showcasing the a submission of a project build to the stakeholders

4.10 Design Component: Unity's Game Development Platform

Introduction

This section, written by Makiah Merritt, will address how we will use Unity as our development platform for creating prototypes.

Scope

Within the context of our project, Unity will serve as the foundation for everything we build.

Description

Unity is a game development platform for 2D and 3D games. The interfaces provided by Unity allow developers to create environments, called scenes, in which users will interact. In addition to the large scope of scene creation, Unity also comes packaged with many features, such as the MonoDevelop scripting editor, providing complete control over projects. Lastly, Unity serves as a resource manager for anything used within a project. As such developers are able to import assets as necessary or manipulate objects within a scene as desired.

Purpose within the Project

Within Unity we will develop our virtual reality environments through the use of built in functionality. Currently, Unity provides the ability to generate terrain maps, place and modify objects, and animate lifelike models. Control over any generated environments will also be a pivotal piece of our project. As a later section will outline, this project will implement C# scripts to grant us access to provide user interaction with developed prototypes. C# scripts will also be used to fine tune the environment to meet our development needs, such as audio manipulation, and to provide users with a responsive environment.

During the construction of environment prototypes we will import assets as necessary. Asset sources will either be community based (free and/or purchased) or generated in house by any of the projects collaborators.

Stakeholders and Stakeholder Concerns

This components stakeholders align with the projects stakeholders: Mike Premi, Marissa Powers, and Lindsay Benjamin. Additionally, Unity will be accessed by our partners at the University of Washington for their research goals. Each of these individuals has expressed hopes that Unity will be a good platform to build upon due to its accessibility to new users as well as its vast community of experts.

As the goal of this project is to develop a high fidelity prototype that can be used to validate a market segment for meditation within virtual reality, the primary concern of our stakeholders is to receive a prototype that immerses users into relaxing virtual reality environment. If this project is unable to deliver an immersive experience the goal of validating market segment cannot be realized.

Design Viewpoints

The deployment of Unity within our project will touch all design viewpoints outlined in the IEEE1016-2009 documents table 1. Primarily we will focus on context, composition, dependency, structure, interface, and resources.

The concerns associated with these viewpoints align with an end goal expressed by our stakeholders that our project yield itself to future development. Guided in part by this, it is desirable that the composition of prototypes be of a modular nature allowing components, interfaces, and resources to be shared throughout the entire application through a common interface. The next concern yields itself to the resources that will be loaded into Unity throughout the project. Anything implemented must add to the immersive experience of the applications end users. The interface developed also yields itself to end user concerns as it must be intuitive and enriching for users while inside a prototypes environment. Lastly, there is a large inter-dependency between Unity and the other tools used throughout this project. Success relies upon each entity being able to interact with each other without users being aware.

Applicable Design Views

Figure 4 below serves to illustrate how Unity will interact with high level entities of our project. Users will receive and provide input through the HTC Vive. Transferred information must tunnel through the Steam Application which in turn connects to Unity through its SteamVR scripting library. Once inside of unity the scenes will be modified as desired and re-transmitted back to the user through this same pipeline.

Design Rationale

Unity will be implemented in this manner because it provides an single interface for developers and users. Development will primarily be performed within the Unity ecosystem using the standard interfaces provided. All necessary assets will be stored within a project folder which unity creates and organized according to the hierarchy that we assign. Because

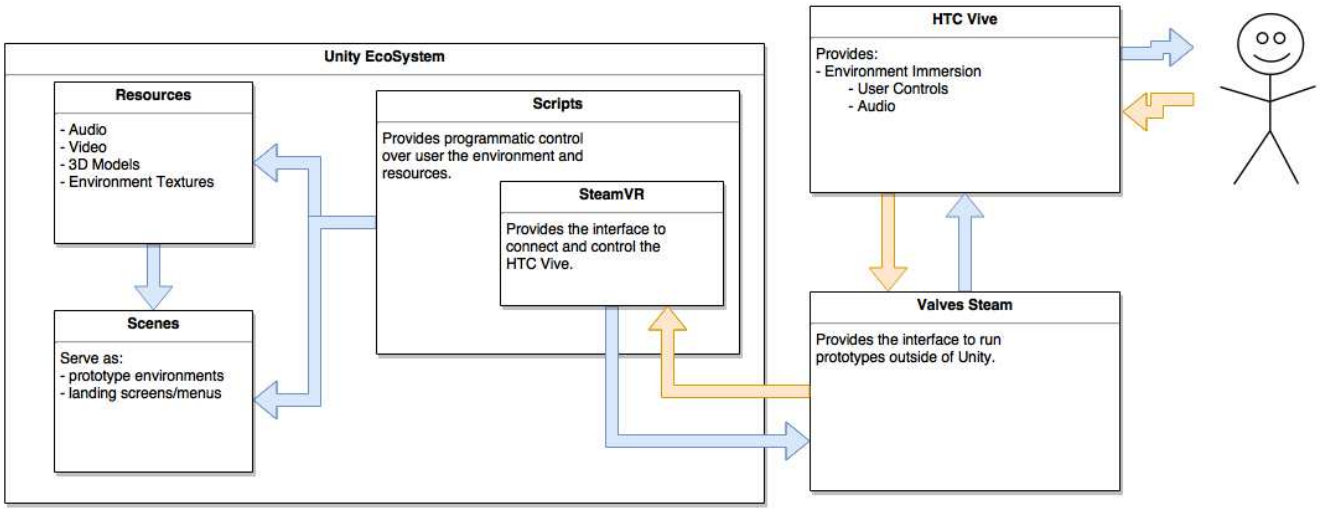


Figure 4: Unity's Internal Ecosystem and Interaction with Users

of this prototypes will be easily transferable for demoing with stakeholders and assets can be easily added or removed as necessary.

4.11 Design Component: .mp4 Video File Format

Introduction

This section, authored by Makiah Merritt, will examine the design aspects of implementing the .mp4 video format within the Meditation and Virtual Reality app.

Scope

Within the Meditation and Virtual Reality app there is a potential that 360 video will be implemented as a user experience. As such, video will only pertain to the aspect of improving a users immersive experience within an environment.

Description

MP4 is a video format created based largely upon the MOV file format. MP4's are widely accepted because of its universal nature and low size. Should the project implement a 360 video aspect, the video file will be added to Unity's resource structure and loaded into a scene through a movie texture.

Purpose within the Project

Video may be implemented through Unity movie texture as a means to provide users with an immersive serene environment.

Stakeholders and Stakeholder Concerns

Marissa Powers and Lindsay Benjamin are the primary stakeholders with regards to prototype content. As such if it is decided that a 360 Video experience would be valuable to observe its effects on an individuals meditative experience, Marissa and Lindsay would provide the desired content for the OSU team to implement. After providing the content the chief concern would be the guarantee that the 360 experience is visually and auditorily seamless for users.

Design Viewpoints

Applicable viewpoints for this design component include context, interface, and interaction. Of these viewpoints interface is the primary concern.

Applicable Design Views

In terms of interface within the application, this file format will need to play nicely with Unity only during the importing process. To import videos (besides .ogg) developers must have Apple Quicktime installed and then they must run Unity as an administrator in order to grant elevated permissions used by the Quicktime Tools exe for the conversion process. Once imported Unity converts the file to Ogg and implements a standard interface for common commands, as defined by the MovieTexture[36]. Any files of this format must be capable of loading within a reasonable amount of time. Additionally, once the video is loaded into a scene it must be controllable via Unity's development tools as well as scripts we develop.

Desired interfaces include manipulating the videos audio level, starting, stopping, pausing, and looping the video.

Listing 1: Unity Scripting API: MovieTexture

```
1  // [01] - http://answers.unity3d.com/answers/28106/view.html
2  // [02] - https://docs.unity3d.com/ScriptReference/MovieTexture.html
3
4  MovieTexture movie = //* Some movie texture in the scene */;
5
6  // Obtaining the Videos Audio Clip Object
7  AudioClip movie_audio = movie.audioClip;
8
9  // Adjusting the audio clips volume
10 float volume = //* Some new volume */;
11 AudioSource.PlayClipAtPoint(movie_audio, transform.position, volume);
12
13 // Checking if the Video is Playing
14 switch(movie.isPlaying) { //* actions to perform */ }
15
16 // Playing the Video
17 movie.Play();
18
19 // Pausing the Video
20 movie.Pause();
21
```

```
22 // Stopping the Video
23 movie.Stop();
24
25 // Looping the Video
26 movie.loop = true;
27
```

Design Rationale

The .mp4 video format provides a lightweight video file that retains most of the original videos quality. Inherently, the smaller .mp4 filesize allows for smaller project builds once a video is imported. Secondly, as Unity supports anything that Apple's Quicktime supports, .mp4 is a natural companion in this aspect as well (being developed by Apple).

4.12 Design Component: Unity Compressed Audio

Introduction

In this section, we will be discussing how we will be using Unity's compressed audio in the application to play almost all necessary audio files.

Scope

In the section we will discuss Unity's audio compression and storage, how it will be used in our application, the persons affected by audio, any design concerns of the stakeholders relating to audio, a listing of the design viewpoints, the design views of those viewpoints, and the design rationale of using compressed audio.

Description

The audio data is compressed before storage, which means a small file size, but it requires the CPU to decompress before playing. This can happen in 3 ways: Decompress On Load, Compressed In Memory, or Streaming. [46] Each method increasingly trades a lower memory footprint for a slower sound response. Since this app will not run on mobile platforms, the encoding will be Ogg Vorbis. (This description was originally written for our Tech Review document [57].)

Purpose within the Project

Compressed audio will allow us to store audio files in a compact amount of disk size. Audio samples will include ambient environment noise, meditation instructions, sound effects, and interface feedback. Unity's built in compression settings will be used, which accepts and imports almost all audio formats [45].

Stakeholders and Stakeholder Concerns

The pertinent stakeholders to this Design Subject are the Oregon State University Capstone team developers, the University of Washington Capstone team in charge of user testing, our Intel Stakeholder group who will need to receive project builds with correct audio, and the end user, who will need clear audio.

Design Rationale

Compressed audio will load relatively quickly (streaming), so that the user can hear the relevant clip without any noticeable delay. A higher quality compression ratio will be chosen as necessary to maintain the high fidelity source material of guided meditations.

4.13 Design Component: Unity Scripting

Introduction

This section will describe how C# will be used for the development of our application, and the design motivations for doing so.

Scope

In this section we will discuss how our application will use Unity scripting, and what role C# plays. We will address these from the standpoint of the persons involved in the creation, modification, and use of C# scripts, as well as look at the design concerns, rationales, views, and viewpoints of scripting. Viewpoints considered will include stakeholders and developers.

Description

Scripting allows developers access to the core functionality of the Unity Game Engine [58], and will drive the actions of our application. Our chosen scripting language, C#, is strongly typed [59], and readily supports a object-oriented model with encapsulation and inheritance, as well as interfaces, generics, and delegates [60].

Purpose within the Project

C# can control every aspect of the Unity engine when creating the application prototype. Its object-oriented model can map directly to controlling Unity GameObjects, which have publicly accessible traits and methods.

Stakeholders and Stakeholder Concerns

The stakeholders involved with this Design Subject are primarily developers. For now, this is the Oregon State University Capstone team developing the prototype. We will need C# to handle all game interactions in a deterministic way, and without run-time performance overhead. Our primary technical stakeholder at Intel, Marissa Powers, will need these goals met as well, so that the prototype can be adopted by other groups at Intel, or by future development teams.

Design Viewpoints

The Design View used within this Design Subject is a Context View of a developer writing C#.

Design Rationale

C# will handle real-time object manipulation, audio playback, and scene transitions. The development team will use it to control when audio starts playing, changes in visual effects, and what actions triggers scene transitions.

4.14 Discussion of Design Changes

The only section of the design of the project that was affected by a change was the storage of project files. Over the course of the year, as new requirements came in from our capstone advisors, we had to shift our initial storage of our project documentation to a Github repository and then later add copies of our work to both a Microsoft One Drive file as well as the intended Google Drive folder. For both the Github repository and the One Drive file, they were class requirements that allowed our advisors easier access to view our work. Though it was a mild inconvenience, we adapted accordingly.

5 Software Requirments Specification

Change History

#	Requirement	What happened to it	Comments
1	Menu System	Removed from requirements based on stakeholder request.	After reviewing our preliminary build that housed multiple builds, the stakeholders decided that stand alone builds were more efficient for ease of use.

5.1 Purpose

The purpose of the Software Requirements Specification (SRS) is to outline the requirements for the Initial prototype that will be used to verify the efficacy of selling this potential product. This prototype will be built using C# in the Unity development platform, and will be designed to be used on the HTC Vive.

5.2 Scope

What will be required as deliverables for this project are a Pre-Alpha, Alpha, Beta, and Release Candidates of the application, consumable in VR to the end user, and a control panel for viewing data and analytics engine to determine effectiveness of meditation. What this software will not be is a finalized application used for sale.

5.3 Overview

This document section contains all of the software requirement specifics for the VRAM Team project. The following will describe what our group plans to deliver over the course of this project. That being said, our industry sponsor and the members of our team both acknowledge that the exploitative nature of our project could lead to changes in expectations as Meditation VR App evolves and takes shape. We will first detail what each group will require to fulfill their part of the project. Next we will break down what we plan to deliver.

5.4 Team Requirements Description

5.4.1 The Intel Stakeholder

The Intel Stakeholder Team will be in charge of project management, communication between the various teams, and providing project related expertise. Throughout this project the stakeholders will require consistent communication with

each team so they can coordinate tasks as necessary. Additionally, when requesting a new task the stakeholders will provide reasonable lead time for teams to develop and return an accurate and quality deliverable. Upon submission of a deliverable the stakeholders will review and share thoughts for improvement to better direct their needs.

5.4.2 The UW Team

The UW Team will be in charge of conducting research on the efficacy of the potential product. The VRAM and UW team will be working together in the development process of deliverables. As such the teams will coordinate a mutual lead time for prototype updates, such that both parties will have adequate time to prepare materials and provide deliverables to each other without issue. UW will provide the VRAM Team with any documents, outlines, diagrams, etc., that they deem necessary for a requested prototype update. The number of prototype updates that the UW team will request will be two, with the chance for more depending on the ability to attend to stretch goals. With this, both teams will have ample time to produce accurate and quality items to be shared.

5.4.3 The OSU Team (V.R.a.M.)

The OSU Team will be in charge of the construction of the prototypes that will be constructed in Unity and will be delivered to the UW Team for user research. In order to complete this project, the OSU team has five requirements and a single request from the stakeholders. The first requirement is that all information that the stakeholders wish that our team uses with discretion is properly associated with a NDA signed by each member of our team. The second requirement is that our team is supplied with a system and HMD capable of being used to develop a VR application. The third requirements is that if there is a request to implement custom or specific assets, models, or technology into the VR application, all fees associated with those custom pieces are covered by a person or entity other than an individual in our team. The fourth requirement is that while the OSU team is eager to learn and help partake in the UX research, development, and validation, we are not held responsible for completing the required research to gain that knowledge. The last requirement that the our team has is that we are allowed to seek consultation from VR development experts so long as what is discussed does not break the previously mentioned NDA. Lastly, the single request that we have is that if there is any need to travel, that we may be reimbursed for out of town (Corvallis, Oregon) transportation costs.

5.5 Product Description

5.5.1 Product Perspective

The developed prototype will ideally be a demonstration that Virtual Reality and Meditation is a plausible venture. The end goal will be to use the prototype as a modular foundation upon which more meditative experiences can be added and/or revamp present experiences with industry professionals to provide an even better meditative experience. The VRAM Team will build this foundation and document section anything necessary for said meditative experiences to be implemented successfully by future developers. If however the UW Team does prove with their conducted research that the efficacy of developing the prototype into a sellable product, the VRAM Team will still produce the prototype as a proof of concept in hopes that the Intel Stakeholder Team can use it for product research at a later date.

5.5.2 Product Perspective Constraints

5.5.2.1 User Interface: Users will operate the prototype through a series of menu selections and scenes. There will be a loading scene as well as individual scenes for meditative experiences. There will be at least one menu released with the prototype. This menu will act as an experience selector and means by which users can exit the application. Future menus may be added for expanding functionality - i.e. option menus for selecting guides or making in game adjustments.

5.5.2.2 Hardware Interfaces: In order to operate within the virtual realm users will require the HTC Vive, which is an HMD device that the program is being tailored for, and some system of input devices such as mouse and keyboard or controllers. The HTC Vive comes stock with controllers provided.

5.5.2.3 Software Interfaces: Currently, the only avenue for this experience is through a windows computer. As such users will need to have the Steam Application[5] in addition to whatever application is required by their HMD device. With Steam installed users will need to install Steam's SteamVR[61] which provides the functional interface for the application and controllers. Lastly, although not necessary for using the prototype, development of additional meditative experiences will require Unity 5.x.x [2] as their development platform.

5.5.2.4 Site Adaption Requirements: This application will require some modifications to accommodate the meditative experiences. Accommodations will vary depending on the intended hardware interface for the application. First, should the application be used without an HMD (i.e. users simply loading the application for an auditory experience) site adaption will be at a minimum. For this instance users will determine their own space requirements and find a suitable quiet location. Secondly, should the application be used with an HMD (such as the HTC Vive) users will need to perform the necessary setup. Noting that HMD setup will be device dependent. HMD sites should be clear of obstacles as users will potentially be moving with a confined region.

5.5.3 Product Functions

Our intended functionality of this project will be as follows:

- User will be able to choose a meditation experience from a list on the main screen.
- Experience will react to user input by altering visual details.
- The final prototype will include a subset of the following meditative experiences:
  - Body Scan
  - Concentration
  - Relaxation
  - Objective Observer
  - Expanded Awareness

5.5.4 User Characteristics

Although we believe this application to have a low learning curve - not requiring formal education or working technical knowledge of the equipment or software development - users of this application will need to posses functioning knowledge of menu systems and must be able to interact with the virtual reality equipment. Because of the nature of this application it is also expected that users be open to, if not familiar with, the practice of meditation.

5.5.5 Constraints

Because this application is a prototype there will be some constraints to consider.

5.5.6 Hardware Limitations

Although the fundamental program architecture is supported on any computer system that can run Unity and Steam, it is recommended that those wishing to experience the application as intended should use these system specifications

Processor	Intel Core i7 6700k (with stock clock rates)
RAM	Crucial 16GB 2133MHz DDR3
GPU	Nvidia 1080 8GB GDDR5 (Note: in most cases a 1070 will also be capable of delivering an equivalent experience[62].)
HMD	HTC Vive
OS	Microsoft Windows 10 64-bit

The above specifications are those present in the development system and therefore should be practical for running the application as a virtual environment as intended. Requiring such a powerful system may seem excessive, however, unlike traditional games where users operate within 30-60+ fps without issue virtual reality applications require frame rates of 90+ to provide a fluid environment for users [62].

5.5.7 Interfaces to Other Applications

This section will be added if a particular portion of the project is made public.

5.5.8 Reliability Requirements

This application has low reliability requirements. We will guarantee that the application exits gracefully should an error occur.

5.5.9 Safety and Security Considerations

Because users will be interacting with our application through an HMD they may not always be present of their surroundings. As such we would like to minimize the ability of the user to become entangled in the HMDs cabling. Furthermore, we must also take considerations of a participant's ability to remain in our virtual reality environment without becoming nauseous.

5.5.10 Assumptions and Dependencies

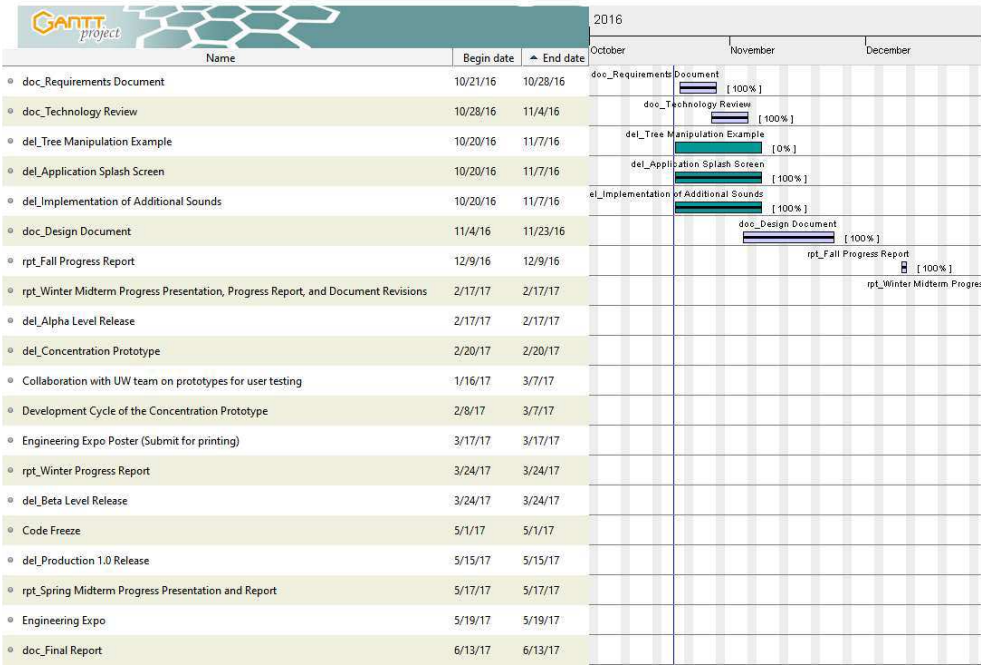
We make no promises that the developed application will function in its original manner on computer systems that fall below the Vive technology minimum requirements.

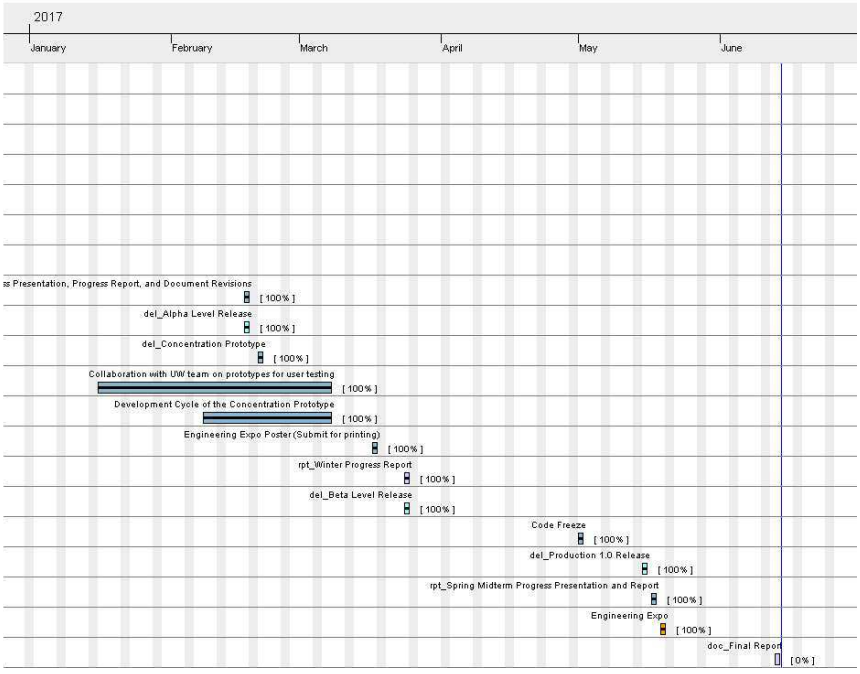
5.6 Addendums

Removal of User Interface (1/6/15)	Initially, the goal was to achieve a collective program with all of the prospective builds located within the application. As we have moved forward, our stakeholders and V.R.a.M. have decided to keep each build as standalone programs. This choice was based on the lack of ease of access to the core content which our stakeholders wish to showcase to their own myriad of stakeholders. In lue of this decision, we have agreed to develop a menu system that our seperate builds could be applied to our stakeholders so choose.

6 GANTT CHARTS

Figure 5





7 Expo Poster

Figure 6

8 Looking Back

8.1 What Did We Learn From All This?

8.1.1 Makiah Merritt

With some background on development teams working for a client, this aspect wasn't foreign; new project = new team + new client. Additionally, all of my jobs have been team based in some fashion. As such most of the knowledge I gained from this project came from the realm of Unity development. Specifically, before this project I've never touched Unity or Virtual Reality and had now concept of how the two would interact. From a programming aspect, the only knowledge I brought with me was a little bit of C#, of which I had played with Window's form development in my spare time. Hitting the ground running, finding a community rich in tutorials, clear documentation, and capable asset packages it wasn't very difficult to get up to a capable level of development. From a project management point of view I found our stakeholders' iterative style very refreshing. Here we developed small builds at first and improved portions that users made note of; keeping development goals very clear. Additionally, this incremental type of development was a breath of fresh air from familiar development practices where new feature requests are frequent and priorities shift on a weekly basis.

If we were to redo the project from scratch I would address our group standards with organization of Unity projects. Within Unity developers have the ability to create hierarchies with project files and assets within a scene. Keeping to a set structure makes it much easier to remember where things are located and thus reduce the complexity of the iterative development we used throughout the project. I think our difficulties came from being unfamiliar with Unity and therefore not necessarily caring about organization as long as we could get something working. Later on, once we started revisiting elements, it became apparent that organization at an earlier stage would've been beneficial; i.e. creating object containers and then attaching scripts targeting inner objects to the parent container. At the end of our project we began implementing hierarchical systems so it's not as big of a deal.

Another aspect I might address are the player controllers we implemented. At the end of our project the SteamVR package, which we used for handling VR controls, was updated to implement player interactions, teleportation, and 2D fallback into a prefab. If we were able to integrate this prefab earlier we would've been able to allow users to move around our scenes and perhaps enabled more interactions with the environment, as we do with our Concentration prototype.

Lastly, we have legacy mockups that are still part of primary builds; namely the introduction screen. These elements were generally created to meet a need, accepted, and never revisited. If we knew that this would be the life cycle of those elements I think we could've put more effort into the visual appeal of the mockups so they would be consistent with the feel of the other portions of the application.

8.1.2 Erik Watterson

The aspects of the project I felt I learned the most about was working within a corporate culture to get a broad task done. Since this project was an efficacy project, proposed in order to prototype an interesting solution, our team had an

interesting look at how a corporation approaches development. To me, this wasn't trivial. All of my other interactions with corporate work was tailored for a specific need. That need had already been fleshed out, and I was the technician to deliver someone elses idea. Here, it was our idea, our drive to get it done, and our hands that gave an output. When I refer to the team, I'm really talking about everyone involved. Even with the looming idea that the UW team or, the Intel team were our clients, it was still apparent that it didn't matter. We needed them, they needed us, and we all wanted this project to work. To me, learning how this relationship formed, why it stayed consistent throughout the year, and why it's so beneficial was the key takeaway from this entire experience.

For hardware technology, I learned quite a bit about hardware specifications for VR equipment, differing kinds of VR equipment, and how VR equipment can influence, or how it can influence, program implementations. For example, in comparing the HTC Vive to the Oculus Rift, we know that room VR will be implemented differently based on the tracking systems in place for both products. For the HTC Vive, you are given two locations that triangulate between themselves and the HMD, allowing for a rectangle to be formed around the play area. For the Oculus Rift, it's tracking system is dependent on a single sensor that spreads out over a large area. In implementation, that single sensor can be problematic since if you turn away from the sensor you might lose tracking when the headset is lost behind the users head. When developing, these hardware related issues were very important to remember and contemplate.

We were particularly lucky to be working very closely with the UW team to determine how they were analyzing their Meditation in VR Design Requirements. As my passion is in the Usability Engineering space, I learned quite a bit about how they approached their user research, developing their questions of interest, and developed their Design Requirements. I was also fortunate to be able to read over their capstone documentation which they allowed our team to view.

Some software technologies were straightforward to learn. Though I learned how to develop C#, became more proficient with Github, and used a plethora of online tools to get work done, the learning of the technologies wasnt as ostentatious as learning about VR hardware, or delving into user experience information. Perhaps I'm bias or jaded, but I'm just not that enthused about knowing yet another language. That said, LaTeX was a good couple headaches to learn, and Unity was interesting to figure out. So, it wasn't all stale from the software side. I was particularly taken with Unity's ability to interact in depth 3D models. For instance, when developing our last build for our stakeholders we had implemented a wind animation to our tree objects that would sway the branches around. Though I implicitly knew that a 3D development platform would allow this kind of interaction, developing on one allowed me a clearer understanding and curiosity about how 3D modeling works.

As the team lead of the OSU team, I learned a lot about how to talk for people who aren't present but are pertinent, schedule events or deadlines based on my teams limitations or skills, and defend or talk about implementations, design decisions, or actions of other individuals. Since I was the one designated to attend the weekly stakeholder meeting, I quickly realized that I needed to speak for my teammates and give them a voice when they weren't present or I couldn't immediately talk with them. I felt that over the course of the school year I could adequately learn the traits of my teammates and represent them effectively when conversing with outside parties.

If I was to start this project over again, I feel I would spend more time developing instead of managing. Because of the deligation of tasks that our team set up for ourselves early Fall term, it was the general idea that Makiah and

Padraig would work primarily on implementation. As things became more hectic later in the academic year, if I would have spent more time helping Makiah and Padraig implement our builds I feel I would have been more proficient and subsequently more useful to them.

8.1.3 Padraig Gillen

I learned a ton from working on this project, often in ways I didn't anticipate when I first signed on back in August. In terms of technical skills, I got a chance to dive into Unity and VR development through SteamVR, and learned that Valve and HTC have done a really good job making it accessible. I also learned some C# tricks, but mostly found that with the help of Visual Studio, everything was pretty intuitive to mesh together and get our prototype running. Though I didn't personally dive deep into the hardware and processing constraints of VR, I was exposed to a lot of the common design choices and got an appreciation for why the industry is where it's at right now, and where it seems to be going next. One motivation I had for joining this project was to work more with graphical development; composing and editing Unity scenes gave me a lot of practice in this and expanded my interests beyond the console.

Moving away from the strictly technical skills, I found it really satisfying to work with the UW team on user centered design, and coordinating what we wanted our shared prototype to look like. Watching their breadth of knowledge and attention to detail, I saw how much goes into making something accessible to a wide range of users, and how important it was to look past my own hunches and think about what could serve the most people. The most satisfying part of this process involved collaborating together on feature choice, scene design, and making realistic deadlines for our deliverables to them. For the class, I had to complete a lot of documentation that I didn't personally think was applicable for this project. However, learning to bring our project into a standard framework was a good learning experience that expanded my writing abilities and I think might benefit me in the long run.

Ultimately, I think the team work and collaboration skills needed for this project have made the most impact on me. Coordinating between our four Intel stakeholders and four UW collaborators took time and effort (e.g. 9pm weekly meetings), and paid off with a great end result. Going forward, I will talk about this experience when looking for work, and think it gave me an appreciation for what a determined team can do in a way that often doesn't happen in a class assignment context. If I had a chance to start over on this project, I think I would want to spend even more time on organizing our projects and refactoring components for reusability. I would also want to do some more bug testing with bird flight towards the end, though this was mainly a time constrained task, not a design issue. Overall, I'm satisfied with how this project went, and feel very fortunate to have been part of a team that took every step of the development process seriously.

8.2 Where Did We Learn About Our Tech

As we were developing in Unity, for the most part, we derived our understand of C# and programming in Unity from the Unity Documentation. Though to note, there were some issues that we had to outsource for more information from random forums here and there. For instance, information about how 3D Models worked. When it came to the specific assets we used within, in large part we couldnt trust the authors documentation. It was either outdated, not specific, or nonexistent, so we mainly stumbled around until we were able to get adequate results. We also consulted individuals

who worked in the related questions field. For instance, any meditation related questions we would ask our stakeholder Lindsay Benjamin, or any 3D model related questions we approached Dr. Mike Bailey. Mike Bailey is a professor at Oregon State who specializes in computer graphics. In large part, when we needed help were generally able to consult someone or find the related information within the technologies related documentation.

9 Fall Blog Posts

Week 03

Erik Watterson

Plans: Work on Shader lab information we got from Intel.
Progress: We got a lab manual for learning 3D shaders, we've all be taking a look at it.
Problems: LaTeX is a killer.

Padraig Gillen

Plans: Continue shader research in Unity, begin thinking about requirements early.
Progress: Got Unity installed on the Chromebook for very portable testing.
Problems: Learning how to compile BibTeX and LaTeX while moving betweek systems.

Makiah Merritt

Plans: TBA
Progress: Participated in necessary assignments. Continued going through tutorials on shaders.
Problems: N/A

Week 04

Erik Watterson

Plans: Look into our Unity tasks given to us by Marissa. Re-evaluate problem statement. Plan to deliver requirements document.
Progress: Set up work station in lab. Introduced the headset to team members. Received feedback on progress report. Received tasks from Marissa. Set up Trello for team. Received UW preliminary time frame.
Problems: The feedback we got for our problem statement is very little. The tasks we received from Marissa might take longer than anticipated. Was approached by another Intel VR team to see the Vive; this team might push to try and use our system.

Padraig Gillen

Plans: Explore Unity audio configurations, and work hard on the two documents we have due.
Progress: Helped setup computer and Vive in Dearborn, and formatted this wiki to specification.
Problems: Finding meeting times has been challenging, but should be better going forward with a team calendar.

Makiah Merritt

Plans: Continue work on the Rapid Prototype implementing leave changes. Help workout the revisions to our problem statement and complete Project Requirements Document.
Progress: Began work on implementing leave color changes.
Problems: None.

Anna Murphy

Plans: Develop Menu/Title screen for prototype. Revise problem statement and begin work on Requirements Document
Progress: Met with stakeholder and decided on course of action for next week. Got feedback on problem statement.
Problems: None

Week 05

Erik Watterson

Plans: Work with Padraig and Anna to go over their respective deliverables before Monday. Begin editing final version of the requirements document to submit by next weeks deadline.
Progress: Makiah has added a usable script to change the trees within the prototypes color. I was able to look at this script and tweek it just slightly so that we can present it to Marissa on Monday.
Problems: We keep cutting the project deadlines a little short. I feel as though if this continues we'll eventually get into a position where we'll be forced to miss a deadline.

Padraig Gillen

Plans: Over the weekend, I will be working on modifying the sound scripts in Unity, and fixing the problems that arose when we updated Unity versions.
Progress: I was able to get familiar with the VR setup / usage of the Vive.
Problems: I have been slammed by other classes, and was not able to contribute nearly enough in this time to the requirements document.

Makiah Merritt

Plans: Help get everyone's changes integrated into the demonstration for Monday. Further cooperation on the documentation.
Progress: Created a Tree Manipulation script and participated in the creation of the Requirements Document.
Problems: Progress on the Tree Manipulation script was slow at first while I determined how best to access objects. Two notable issues still stand. First, although I found a progression that is smooth it is fairly slow. Secondly, I am wondering whether or not it is more efficient to capture all objects during game load vs repeatedly during the scenes update stage.

Week 06

Erik Watterson

Plans: Receive from Marissa, and play with Desktop Ninja's version of a meditation in VR app. Look over the next document due for capstone class. Work with Marissa on the changes needed for the latest prototype.
Progress: Requirements document is signed by all required parties, turned in, and ready to be graded. We have submitted our first deliverable to our client, they have been reviewed, and were met with positive review.
Problems: Not all of our deliverables were fully completed. There is a clear time constraint on each member of this group and I fear that it might cause us to miss a deadline. As it stands, we've completed each assignment on the exact day it was due, and frankly that scares me.

Padraig Gillen

Plans: Refine/quiet sound in prototype to meet stakeholder goals of the focus being on the voice. Get a whole bunch of cool VR apps to do a tech review.
Progress: I was able to get the sound working again in our prototype and learn more about how the SteamVR and Unity interact. I helped a little with the LaTeX formatting.
Problems: Not too many, mainly just time crunches.

Makiah Merritt

Plans: None at this point in time.
Progress: Assisted with this Monday's deliverables and the requirements document.
Problems: LaTeX formatting. . .

Week 07

Erik Watterson

Plans: Plan out design document and quarterly review. Work with Marissa to evaluate next rendition of the prototype.
Progress: Tech review is on it's way to being done, though it will likely run into the next week since everyone is

extremely busy.
Problems: Anna had made the tough decision to take some time away from OSU to take care of her mother. Also, Not enough hours in the day to get things done.

Padraig Gillen

Plans: Finish tech review and continue last week's goal of investigating sound functionality in prototype.
Progress: Worked on tech review with sound and GPU choice.
Problems: I felt incredibly uncertain about what to include in the tech review, given that the hardware has been chosen for our project, and the stakeholder seems pretty confident in the choice of Unity as a development platform. Breaking Unity modules into technologies seems like the way forward as we work to complete the tech review.

Makiah Merritt

Plans: Work on tech review with whatever time is left over from CS 444.
Progress: Generated a section for the tech review.
Problems: Time to work on tech review was limited. As Padraig notes the content for our tech review was also a source of confusion. But we talked with our TA and he said expand upon what we could.

Anna Murphy

(Attending to family matters indefinitely)

Week 08

Erik Watterson

Plans: Continue development on Design document. Communicate with Marissa on Monday about the next set of deliverable. Then start implementing the next set of deliverable.
Progress: The Tech Document is done, for now. Marissa has had time to demo the last prototype we made to some of her Intel stakeholders and they love the idea and where the project is headed. When we met Last Thursday (11/17/16) she had outlined the next set of changes that she'd like to see.
Problems: The tech review was a mess due to time constraints for all of us.

Padraig Gillen

Plans: Keep brainstorming and understanding requirements of Design Document. Investigate time needed to add more features to prototype.
Progress: Tech review sections were completed somehow.
Problems: Determining what sections to work on in the tech review was a major struggle for me. Furthermore, trying to get a sense for a wide variety of things at once while meeting formatting and length requirements was a challenge.

Makiah Merritt

Plans: After our weekly meeting with Marissa some new tasks have popped up that she'd like done, currently Padraig has volunteered for them but I'll be available should he need assistance. Besides that we're aiming to get a good start on our Design Document and progress reports.
Progress: Assisted with the Tech Review with two sections of my own and helping get the tex document put together.
Problems: There was a lot of stress this week between CS 444 and CS 461 but hopefully that was a one time deal.

Week 09

Erik Watterson

Plans: Trying to wrap as much stuff of before end of term.
Progress: Working on Design Document.
Problems: Drowning in documentation.

Padraig Gillen

Plans: I need to plan and write my three sections for the *Design Document*.
Progress: Meetings with stakeholders have been going well, and we have begun planning and outlining the Design Doc.
Problems: We are going to to have a major time crunch getting a signed design document in by Friday at noon. We are getting pretty good at working together on these documents, but it will be a close deadline for sure. Additionally, as I begin to implement a few small features, some technical snags have come up with protocols. A couple more hours of careful debugging should hopefully fix that though.

Makiah Merritt

Plans: If there's time, design document and progress report.
Progress: Outlining of the design document.
Problems: Not enough time.

10 Winter Blog Posts

Week 01

Erik Watterson

Plans: Talk with Ricki Mudd about expected deliverables to UW team for this term.
Progress: Worked heavily on delivering workable demo builds of current prototype to Marissa. We needed to split a single build with multiple pieces of functionality into 4 seperate builds. Also, the last Friday of Winter break we met with the Intel stakeholder team. We had a blast visiting the Jones Farm campus in Hillsboro. We ended up going on

another unrelated teams lab tour then did a bit of code debugging with Marissa to fix a build transfer issue, then played with VR all day until the final meeting. In Marissa's words, "Today was really productive".
Problems: We forgot to invite McGrath to the meeting at Intel! He lives right down the street, might as well see if he's interested in seeing the cool stuff!

Padraig Gillen

Plans: Receive feedback from Marissa and work on new changes needed.
Progress: Finished my part of splitting our former mega prototype into more manageable chunks.
Problems: Weather predictions caused a hasty retreat back to Corvallis after the Intel visit. Forgot to update this blog on time :)

Makiah Merritt

Plans: Make it to class on time.
Progress: Attempted to get a menu implementation working to send off to Marissa. **Update 20170113.2353** got a menu system implemented into a single build. Will work on replicating it to the other builds later.
Problems: Progress with implementing a menu using Rays is posing a larger problem than anticipated.

Week 02

Erik Watterson

Plans: Prep for meeting with UW team. Re-implement changes for Marissa. Other classes homework.
Progress: Relatively slow week. Nothing to report.
Problems: The last builds I delivered to Marissa were faulty. Next week I will be re-doing them.

Padraig Gillen

Plans: Verify documentation is complete and accurate.
Progress: Worked on fixing builds that were uploaded to Google Drive, and started documentation to what each build contains.
Problems: Not many to report, need to allocate more time to this project in the coming weeks.

Makiah Merritt

Plans: After I've finished with stats I'll hopefully have some time to wrap up my lingering controller issue and migrate these updates to the other builds.
Progress: We had a meeting with the UW team on Wednesday evening to go over some of their ideas for what they'd like shipped. Additionally, I've made alot of progress with the menu system. From what I could tell my lingering issue was keeping inactive controllers from activating when the menu is pulled up. Otherwise menus open and close and

controllers disappear and reappear with the menu and the menu can handle many elements including scene changes with fading, audio and volume changes (only 2D for now but the port to VR shouldn't be difficult at all), and controlling the current track (play/stop/pause/restart).
Problems: Stats homework has kept me pretty bogged down and unable to do some of the work I was wanting to get done over the weekend. Namely I wanted to update the split builds as well as one of my personal builds with the progress I've made with the menu system. Then ship these to Marissa (all) and the UW team (my personal build).

Week 03

Erik Watterson

Plans: A bit more research for the UW team so that we can better address procedrial generation.
Progress: After talking with the UW team, we realized that the project might diverge a bit more than anticipated. We'll be talking with them next Tuesday to discuss moving forward.
Problems: Marissa has a family emergency and needed to step aside for a moment. She should be back some time this week or next week.

Padraig Gillen

Plans: Meet with Marissa to get new instructions for builds. Meet with UW team to brainstorm new directions for prototypes.
Progress: Had 2 successful meetings with UW, and addressed getting them familiar with our prototypes, as well as some common issues they might have.
Problems: Other classes piled on this week, so next will require a lot of heavy up front research.

Makiah Merritt

Plans: Continue refining the menu build if I have time. Attempt to be on time for class. Whatever else needs done.
Progress: Assisted Erik with his builds and attended a new weekly meeting with the UW team, for the meeting I had prepared a rough draft of 360 video in VR.
Problems: Stats kicked my butt last week and there's a midterm the first week of February.

Week 04

Erik Watterson

Plans: This week, I'll be constructing the outline of the written documentation that is due on the 21st.
Progress: As of right now, we have a solid idea of what we would like to do to deliver to the UW team. We think the low poly procedural is a good idea, and also feel that we can create a small prototype of a tai chi slow object movement demo. We just need to buckle down and do them now.
Problems: Because of the lack of dedicated classes, I feel like our team is a bit more disconnected than it was last term.

Because of this, I'm a little concerned that we haven't implemented as much of our program as we should have by this point.

Padraig Gillen

Plans: Begin work on our first prototype for UW, which should involve object interaction in the environment.
Progress: Researched 360 video implementations and issues.
Problems: Too many assignments + activities! This is a self-inflicted problem, but I'm looking forward to having a quiet, semi-productive weekend.

Makiah Merritt

Plans: Assist with whatever development the UW team would like us to do.
Progress: Drafted rough prototypes for procedural generation and serialization of data to a file (so we can record user stats. Updated the 360 video draft with an actual 360 video, increasing its usefulness as a proof of concept.
Problems: It's been a pretty good week.

Week 05

Erik Watterson

Plans: Work on Documentation while Padraig and Makiah work on the Prototype. When Padraig is ready to do animations, I'll try to help assist.
Progress: We have a Nice beach prototype constructed by Padraig that we'll use for the base of the UW deliverable. Padraig did an awesome job with this. Great communication with the UW team, allowing us to ask questions about what they would like on the fly.
Problems: Working on Documentation isn't going as planned. Other classes are temporarily swamping me with work.

Padraig Gillen

Plans: Work to bring the team into finishing our prototype by Saturday, and figure out animations in Unity. Work on getting the documentation we need done, done.
Progress: Got a solid beach prototype started, with room enough to do what we need. The assets should be mostly usable in the final edition, though I'll need to find public domain or free to use audio for the final version.
Problems: Had trouble with my home Unity setup, and decided to stick to developing on the campus machines for now. 5400 rpm hard drives plague my existence.

Makiah Merritt

Plans: After midterms are over we will begin working on the progress report. Additionally, we have a deliverable for the UW team to deliver by the 18th.

Progress: *Edit:* I forgot that I mocked up a system (2D) for logging a players answers to survey questions (how are you, stress level, comments). Data provided is input into pre and post meditation forms and then can be viewed on a final screen that lists out 10 logs and can provide details for a specific log if the user "Opens" it. This can be adapted to import other data that won't be mentioned here.
Problems: Unfortunately, studying for stats has taken up most of my time.

Week 06

Erik Watterson

Plans: Help Makiah and Padraig with the build for UW for Saturday.
Progress: The build for UW is going great but it needed to be temporarily put on hold for a moment due to documentation. We have the UW team an estimated time frame of Saturday. So now we are trying to get the progress report done by Friday.
Problems: We were under the assumption that we could get done with the build for UW, the progress report, and the updated documentation/OneNote file all done by Friday, but last Friday night, we realized that wasn't possible. After asking Kirsten and McGrath for an extension, we have till next Friday to get the documentation done. So, now we only have to worry about 2/3 things this week. I've also been putting off a request from Marissa to get 360 video extracted from the RFP builds since I don't have the time to invest in doing that at the moment. Hopefully next week will be better.

Padraig Gillen

Plans: Merge my work with Makiah's, and finish the UW prototype by Saturday night.
Progress: This week I was able to learn about Unity animations, and how to implement them in the newest prototype for UW. I wanted to see if I could record motion using the VR controller we have, and found a Github project that has scripts to do this. I forked it here, and have started working on making minor corrections as needed, so that it can be used to record our animations. Also, a lot of documentation work, particularly fixing issues as we transition to the new template, and dealing with non-page breaking long tables.
Problems: The above script has some issues, notably that it incorrectly caps at a certain frame count. Balancing my time between finding and fixing these problems, and working on documentation was a crunch.

Makiah Merritt

Plans: Get the Concentration Build to UW.
Progress: Worked on collision detection with Padraig.
Problems: ST314 midterm == senioritis (even though I've got a year to go). We should've started on the progress report sooner.

Week 07

Erik Watterson

Plans: Look more into 360 video for Marissa
Progress: I've been dinking around with the 360 video solutions in the asset store, but none seem to get me anywhere. I'll keep poking around and see what comes up. I've also had a hand in updating the documentation. Specifically organizing the one note with all of the latest renditions of the documents.
Problems: No solutions for the 360 video deliverable.

Padraig Gillen

Plans: Look into getting 360 video recording working with Erik, or find out what limitations are present. Use any new found free time to catch up on other classes projects.
Progress: Worked to get the Concentration prototype animations recorded and running on orbs. Attended a very satisfying meeting. Updated my work in our fall term documentation.
Problems: Revising all the docs. . . .

Makiah Merritt

Plans: Playing around, enjoying life. *Edit:* nevermind, schools still going strong #NoSenioritis
Progress: Transitioned documents to the capstone template. Participated in delivering the Concentration prototype to UW. Thursday I was also able to go through the Technology Review and Design Document and make some updates.
Problems: Some rush but not bad.

Week 08

Erik Watterson

Plans: Finish up the next itteration on whatever is asked of us. Visit UW for the UX expo. Woo!
Progress: Helped facilitate 1.3.1 for UW. I wasn't able to contribute much to building it, but I kept a close eye on it's progress since we had a very short window to deliver it.
Problems: 360 video recording isn't going so well. All of the free recording packs on the asset store are old and broken. It may be that we need to take one and rebuild it. I'll shoot an email to Brian Mckenzie, an Intel Unity contact, to see if he has a solution.

Padraig Gillen

Plans: Record a newer, slower concentration prototype.
Progress: Not much on my end, mainly wrapping things up with UW.
Problems: The Discord web app doesn't seem to let me use push to talk.

Makiah Merritt

Plans: Progress with whatever Marissa or the UW team asks for as well as any research I delve into.
Progress: Mocked up a Concentration 1.3.1 for the UW team. This build can handle one or two controller's and it has minor adjustments to the speed and height. If I have time maybe I will look into building something that will automatically adjust the orbs to the players height. Additionally, I would like to determine which of the two SteamVR controllers is a better prefab to work off of, there is an Interactables/Player.prefab as well as the basic CameraRig.prefab. Apparent differences so far is that the player prefab provides for the capture of object interactions (hover, attachments, etc.) as well as a built in 2D fallback mode. Where as the CameraRigprefab provides a better interface to the controllers themselves. Perhaps there is a way to combine the two and get the best of both worlds.
Problems: Getting the OrbController.cs scripts to recognize controller configurations (1 vs 2 controllers). There is still an issue when a user changes controller configurations without restarting the scene.

Week 09

Erik Watterson

Plans: Need to take a look at the poster and talk more with Marissa about what she would like to accomplish post UW testing.
Progress: Done with UW side of things. Visiting UW! Wooo!
Problems: UW wanted one final change to the build which was a little taxing on us since it was very last second. Also, now that we're done… need to figure out how to keep in touch with these wonderful people…

Padraig Gillen

(Next) Week's Plans: Panic a little as I realize how much is left for the class, as I have thought very little about our poster.
(This) Week's Progress: Recorded a slower, more centrally focused concentration prototype. Visited UW Friday to see their final presentation.
(This) Week's Problems: Apparently we got lucky on the first animation recording in Unity, as it seems impossible to get something steady the second go around.

Makiah Merritt

Plans: Like Padraig next week is going to need a "panic" button. But we'll begin work on the poster, presentation/pptx, and I will do my document.
Progress: Attended the Monday and Tuesday weekly meetings to make sure everything is on track. Will visit UW Friday for their expo.
Problems: N/A

Week 10

Erik Watterson

Plans: Do progress report then take a break from the project.
Progress: Talked with Marissa about what we plan to do going forward with the project. She outlined that the next step is to get the project presentable by the end of April. This entails new assets, a redesign of the scene, and new features. Since the discussion, I informed Marissa that we would like to take two weeks off to collect ourselves. She requested that we find applicable midterm assets to add to the build and implement them before Friday (3/17/17). I was able to implement the tree assets within the time frame. Also, I finally updated my previous not filled in weekly status reports.
Problems: The leading reason why we wanted to do a re-haul on the tree assets was because one of the more prominent type of tree in the RFP scene was constantly being affected by what's called Billboarding. I've come to find out that this free tree model can't have their Billboarding feature changes, but a paid model can. Which is incredibly important now, since one of the two tree models purchased has a large Billboarding issue. First thing Spring term is to look into how exactly to stop the Billboarding process, promptly followed by how to change the Billboarding process to effect performance.

Padraig Gillen

Plans: Work on final progress report, and fix any bugs that come up with deliverable.
Progress: Found a bird bundle for Marissa to purchase, and got it implemented into the RFP. Contributed to poster.
Problems: Said bundle thinks that documentation in the form of examples is good enough.

Makiah Merritt

Plans: Collaborate and finish the video for capstone. Apply any lingering assets to the build for Marissa.
Progress: Did some work on the expo poster, my progress report, and research on sky box/weather management assets. For the latter, I messed around with free assets and found that they were fairly distracting. However, Marissa provided us with access to paid asset packs on Thursday, hoping for a Friday turn around (I'm headed back to the lab now).
Problems: Time was somewhat constrained for working on deliverable's (post obtaining paid assets) thanks to a busy work week.

Week 11

Erik Watterson

Plans: TBA
Progress: TBA
Problems: TBA

Padraig Gillen

Plans: Spring break.
Progress: Got the birds to sufficiently flit around the scene. Wrote a progress report, and did my part in recording our best mutual update yet.
Problems: Birds are still dive bombing the dirt, and I need to teach them to avoid flying at the user.

Makiah Merritt

Plans: If unable to attend to it this week, between work next week I want to look into why the skybox asset Marissa purchased isn't rendering with our current CameraRig setup.
Progress: This week I (will) have turned in my progress report, signed our midterm progress report, and particpipated in the recording of our final winter term presentation. Also we have signed up for our spring term weekly meetings with Vee.
Problems: Because of work last Thursday and Friday and finals this week I haven't been able to apply much time to updating my portion of our current RFP-v6 build with the skybox, beyond the 2 hours or so last Friday evening I spent after work to merge it into the scene.

11 Spring Blog Posts

Week 01

Padraig Gillen

Plans: Refine stakeholder documentation, and continue minor bird bug fixes. Get poster going!
Progress: Attend class, and get back into the relevant thinking.
Problems: I focused on other classes because this didn't seem pressing, will have to quickly jump back into it.

Makiah Merritt

Plans: Continue researching and documentation.
Progress: Attended this weeks meeting. Also, because I wasn't able to do so over break I began looking into the interfaces of the Sky Master ULTIMATE asset pack. Reiterated during this meeting, we're looking to start with something simple such as changing the time of day, adding or subtracting from the cloud bed, etc. The route I think I want to tackle is transitioning from a morning to afternoon feel. The second request was the creation of documentation in preparation to pass the project off. We'll be documenting our scripts and their interfaces as well as each individual build and how to get them running as well as any tips and tricks necessary for modifying them. I had begun a document outline for this and put it on the Google Drive.
Problems: Time constraints only allowed me to be in the lab one day. I can't really do the necessary research at home because I need access to the Vive and my desktop currently can't handle it.

Erik Watterson

Plans: Start on build manuals and keep trecking on last deliverable build for Marissa.
Progress: Getting caught up on the stakeholder meeting I missed. Began planning out last steps of the project.
Problems: This week has been slow for me. There have been a few side projects that have taken me away from my capstone project and need to be completed this week. Next week will be better.

Week 02

Padraig Gillen

Plans: Continue docs, poster work.
Progress: OSC rebuilds.
Problems: OS 2.

Makiah Merritt

Plans: Continue work on OSC and the SkyMaste pack.
Progress: Made some pretty decent progress interfacing with the Sky Master asset pack.
Problems: We have yet to do any major revisions to our poster and prototype documentation has alot to do.

Erik Watterson

Plans: Work on documentation, poster, and help Makiah with final build.
Progress: This week has been mainly associated around Marissa's documentation request and finding out the capstone classes final due dates.
Problems: I'm feeling like I'm falling behind in my work ethic coming into this last term of the school year.

Week 03

Padraig Gillen

Plans: Finish OSC docs and work to make it a modular inclusion for our latest build.
Progress: Start revisiting old code to properly create documentation.
Problems: TBH not very much progress for this class.

Makiah Merritt

Plans: TBA
Progress: Made alot of progress with our documentation. I've added documentation for most of the scripts and builds I was the primary on, minus the most recent RFP-v7.0 build in progress.
Problems: It took us a while to get feedback on our poster and we missed the turn in due date.

Erik Watterson

Plans: Finish up Marissas manual and RFP build. Submit the final poster.
Progress: We have a nice format for the manual, have started adding content to it and we have a good idea how finish up the build.
Problems: We had to pull double time on our deliverable to Marissa. We started them, but didn't realize how quickly the month was passing and we were nearing the due date. I also completely missed a poster draft due date.

Week 04

Padraig Gillen

Plans: Figure out WIRED assignment.
Progress: Write OSC docs, worked with Makiah to integrate OSC into build. Finalize poster as team and submit it to instructors.
Problems: Had a meeting with Vee where we realized our menu requirement had changed and was not being reflected in our documentation.

Makiah Merritt

Plans: Work on the WIRED assignment.
Progress: Figured out the last bit of manipulation for the Sky Master Ultimate asset package I wanted (clouds). With this, I worked with Padraig to get OSC brought into the build, after which we delivered it to Marissa. Also, we reviewed and sent off our capstone poster to Kevin.
Problems: As Padraig had noted we had an update for our requirements document to reflect the separation of builds => a single build with the menu system, where other's don't have it.

Erik Watterson

Plans: Edit the requirements doc. Put all of the code we have done on github. Add a readme on how to interface with the code via Unity. Get final poster approved then submitted to print. Get done with our wired article. Get dinner with the team.
Progress: We got the manual and build done! Makiah really pulled through for the team this week. We were also able to verify our final poster edit and figured out a time to celebrate with dinner next week.
Problems: We ran into a small issue with the requirements document that articulates a menu build that our stakeholders decided against. We'll need to get that settled soon.

Week 05

Padraig Gillen

Plans: I want to go back now after some time has passed, and make sure the documentation I wrote is complete enough and makes sense. We will also have to write / shoot the progress report together.
Progress: As Makiah mentioned, we have been in a small lull after poster submission, which I've needed for other classes.
Problems: Not many. In a perfect world we would have build feedback already, but I understand that there's a lot to go through and play with, and our stakeholders are busy people.

Makiah Merritt

Plans: Gear up to work on our progress report and presentation deliverables as well as mentally prep for expo.
Progress: Last week we had gotten Marissa the RPF-v7.0 build which included new visual improvements and script interfaces. Since then we haven't heard back from her concerning any new actions to take. As such the only update is that Erik has turned in the poster and all of us have submitted our model release forms.
Problems: None.

Erik Watterson

Plans: Begin working on progres report and getting ready for expo.
Progress: Just worked on Wired Article. Nothing exciting this week.
Problems: Since we officially handed in our last thing to Marissa, no new issues have arose.

Week 06

Padraig Gillen

Plans: Finish and record the progress reports with the other two on Monday. We also still need to have some visuals ready for our monitors at expo, which I want to help get ready. Given our previous issues with 360 video, this might require some creativity.
Progress: Since I didn't mention it, I also was able to submit my WIRED article last week. Not much, most of my effort went into preparing for other classes midterms, that ended up being bumped on Thursday.
Problems: Much of the time I planned to spend on the progress report was interrupted as I went back to Portland early Friday to help a friend with a family emergency.

Makiah Merritt

Plans: EXPO! Also, we need to record our Midterm presentation.
Progress: Created and worked on the base files for our midterm deliverables (LaTeX document and pptx).
Problems: Still in limbo as far as Expo goes; what devices are available? where will our booth will be? etc.

Erik Watterson

Plans: Expo is next week, so we're focusing pretty heavily in preparing for it.
Progress: Mainly just the midterm progress report.
Problems: No feedback on if our expo requests have been granted.

Week 07 - Post Expo Reflection

Padraig Gillen

Expo has come and gone, meaning our project is effectively wrapped up. I will reflect a little in this blog post, which I would like to point out was assigned on the due date!

This has been a whirlwind project, beginning forever ago last summer. I have really enjoyed working with my team, even as it grew and shrank throughout the year. I learned a lot from my teammates, most of which involved collaborating together and appreciating the novel way they solved problems.
Inspiration highlights:

- Marissa's ability to validate our work done, cut through to what mattered in the project, and point us in the right direction as we worked
- The UW team's collective eye for aesthetically pleasing, intuitive presentation, and their collaborative-by-default spirit
- Makiah's endless knowledge of Unity's functionality, combined with perpetual good humor
- Erik steering our project ship towards success, and continual focus on keeping us all in sync
- TA meetings with Vee that handled project business and made for some fun Monday / Tuesday mornings

Skills / Project Review:
I enjoyed working with the Unity / Steam / Vive combo, and appreciated all that Valve & HTC have done to make the VR part of development a breeze. Intel was also a great help in funding of this project, and our project looks good because of the generous hardware and asset packs they bought for us. The major frustrations I had were dealing with portability problems as Unity version were bumped.

I think the biggest skill I acquired was working simultaneously with the amount of people we had, and making sure everybody was on the same page with things I worked on. I also gained a ton of intuition for game development, and debugging graphical problems, something I would have shunned at the beginning of the project. Both of these will be useful going forward.

If I were to start over on this project, I think I would advise myself to dive deeper when I solve a problem and make sure I understand what root cause I addressed. This would also mean cleaner code. I would be satisfied with our work as the project client, and think that I would focus on making one cohesive app with a few more novel experiences if we had another year.

Makiah Merritt

When Erik first approached us (Padraig and I) concerning the project I was a little worried about accepting the invention. To join a project sponsored by Intel with the added issue of having to quickly learn a new development kit (Unity) as well as its languages (Shaders and C#), it was quite daunting. However, looking back I tell myself to jump on the opportunity as it would be a learning opportunity from multiple perspectives. First, there is working with a team to develop something that would be used for user testing; I have developed for clients before but have never been part of a research study, perk 1. Next, in being sponsored by Intel there was a lot of knowledge and resources that they brought to the table, all of which was provided to support us throughout the project; networking with and learning from individuals in industry, perk 2. Lastly, in being part of a team project everyone is there to help each other. Knowing each team member excelled in particular areas it made work distribution much easier; learning to rely on team members, perk 3.

Skills learned? Obviously, there is picking up the Unity environment and languages. Beyond this however, is the continual application of the skills behind being able to pickup new development processes; such as performing the research necessary to implement a task. This skill will be invaluable in our future work.

I can't say there was anything I didn't like in this project. Working with Marissa and the UW team was a blast! Marissa was a great project lead. Her organization and sense of direction was impeccable. After talking with her there was little doubt about what we were supposed to be doing. Additionally, my team members were great as well. Erik was a champ working as a mediator between our stakeholders and UW team. Attending some of the meetings, it was pretty difficult to filter out information pertinent to us but Erik was able to throughout the term. Padraig was/is a Git wizard. When we were working through all of our documentation in Fall term we could rely on him to keep things moving smoothly. He was also very apt at transferring this skill to other areas of our project, such as finding and implementing the OSC plugin.

I believe we could have done more work with our project but perhaps the biggest hurdle of doing more was the work behind NDA, where all of the interesting bits were happening. With this I would say as a client our team did a great job in delivering the foundation for future work. Speaking of future work, many individuals at expo were interested in the quantification of our study on Meditation In VR. This is likely where one would pickup our project. Although there are still some visual aspects to be improved upon, our groundwork yields itself to quantitative studies.

Erik Watterson

This project was a bit of a dream come true for me. The following are some loose reasons off the top of my head. We worked directly with a corporate sponsor. I was allowed to hand pick my team. I lead my team and we were effective at getting our tasks done. We had very few road blocks that required project pivots. The issues that did come up were quickly addressed and taken care of. Between the three teams, there was very little miscommunication, which was likely the biggest miracle of them all. Finally, I got to experience participating in a full on CS project.

Something that I'm a little unsure about is the lack of coding I personally accomplished for this project. I realize that my primary role was to make sure to make my team was efficient, but I would have liked to spend a bit more time

looking at the programming problems myself. I think it was around the time of the second poster draft submission, I was working on it and I realized I hadn't even really seen half the code that we implemented. So there I was, listing out our accomplishments, and part of me felt a little undeserving.

In terms of teammates, I chose to reach out to Padraig and Makiah early on when the project was being formulated, and since that decision they have done above and beyond what I've expected of them. No team dynamic will be perfect, and ours particularly needed to work on our procrastination, but we tried our hardest to always hit due dates, and if we didn't make them, we would always have a legitimate reason. For instance, when we had 4 weeks of back to back deliverables that the UW team needed during Winter term. With a lesser group, I could honestly see us not making their schedule.

In terms of skills, nothing really screams out at me that I'm now a expert, or semi-expert in. I've become more efficient at managing people and scheduling. I've learned how to operate within Unity, C# programming language, and the Vive's Steam VR asset pack. I've grown a bit in being able to sell an idea or network with others to expand an idea. But none of those I feel I'm an expert at. . . yet.

Overall, the quality of work that we put out was fine. I think that if we would of had a more fleshed out schedule during Winter term we would have produced better quality work. That being said, I've asked Marissa several times how she feels about the work that we have submitted and she's told me she thinks it's great every time I've asked. I'll have to defer to her judgement on this.

The last interaction we will be having with our stakeholder, for this project, is to meet with them one last time at their Intel Office. This meeting will be a final handoff of any lingering work that we've done, and a debriefing on what to expect moving forward. One thing I plan to address at the meeting are references from our stakeholders, for our team, and to ask them if they might have any work for Padraig to do since he's graduating.

12 CODE LISTINGS

Listing 2: Game Controller

```
1   /*
2  * This script was based off of a tutorial.
3  * Ref[1]: https://unity3d.com/learn/tutorials/topics/scripting/persistence-saving-and-loading-data
4  *
5  * This script must be attached to a root GameObject.
6  * Ref[2]: http://answers.unity3d.com/answers/456306/view.html
7  */
8 using System;
9 using System.Collections;
10 using System.Collections.Generic;
11 using System.IO;
12 using System.Runtime.Serialization.Formatters.Binary;
13 using UnityEngine;
14 using UnityEngine.SceneManagement;
15
16 public class GameController : MonoBehaviour
17 {
18     /* VARIABLE DECLARATIONS ================================================*/
19     // Public Variables
20     public bool G_Debug = true;
21     public static GameController G_GameController;
22     private GameData G_GameData = new GameData();
23
24     // Private Variables
25     private int G_CurrentScene = 0;
26     private string G_DataPath = ""; // For us it was: C:/Users/ME!/AppData/LocalLow/DefaultCompany/
       project_template/gameData.dat
27     /*====  /VARIABLE DECLARATIONS ====*/
28
29
30     /* UNITY SPECIFIC FUNCTIONS ============================================*/
31     private void Awake()
32     {
33         G_DataPath = Application.persistentDataPath + "/gameData_" + System.DateTime.Now.ToString("
       yyyyMMdd") + ".dat";
34         G_CurrentScene = SceneManager.GetActiveScene().buildIndex;
35
36         if (G_GameController == null)
37         {   // If there is no game controller create one.
38             DontDestroyOnLoad(gameObject);
39             G_GameController = this;
40         }
41         else if (G_GameController != this)
42         {   // If this isn't the the current game controller destroy it, we will
43             // use 'this' one.
44             Destroy(gameObject);
45             G_GameController = this;
```

```
46          }
47          LoadGameData();
48
49          if (G_Debug) { print(G_DataPath); }
50      } /*-- /Awake() declaration --*/
51
52      private void Update()
53      {
54      } /*-- /Update() declaration --*/
55
56      private void OnGUI ()
57      {
58      //    GUI.Label(new Rect(Screen.width / 2 - 50, Screen.height / 2 - 80, 200
59      //            , 30), "Current Scene: " + SceneManager.GetActiveScene().name);
60      } /*-- /OnGUI() declaration --*/
61      /*====  /UNITY SPECIFIC FUNCTIONS ====*/
62
63
64      /* MY FUNCTIONS ============================================================*/
65      /*-------------------------------------------------------------------------
66       * Description       : Saves desired data to a binary file for later use.
67       */
68      public void SaveGameData()
69      {
70          // Grab/create objects to handle data transfer
71          BinaryFormatter bf = new BinaryFormatter();
72          FileStream file = File.Open(G_DataPath, FileMode.OpenOrCreate);
73          GameData data = null;
74
75          // Gather game data
76          data = G_GameData;
77          data.log_timestamp = System.DateTime.Now.ToString("g");
78
79          // Write data to our file
80          bf.Serialize(file, data);
81          file.Close();
82
83          print("File Saved: " + System.DateTime.Now.ToString("yyyyMMdd.HHmm"));
84      } /*-- /SaveGameData() declaration --*/
85
86
87      /*-------------------------------------------------------------------------
88       * Description       : Loads game data from a previously binary file.
89       */
90      public void LoadGameData()
91      {
92          if(!File.Exists(G_DataPath)) { return; }
93
94          // Grab/create objects to handle data transfer
95          BinaryFormatter bf = new BinaryFormatter();
```

```
96          FileStream file = File.Open(G_DataPath, FileMode.Open);
97          GameData data = new GameData();
98
99          // Obtain data from the file
100         data = (GameData)bf.Deserialize(file);
101
102         G_GameData = data;
103
104         // Store data for later use
105         file.Close();
106
107         print("File Loaded: " + System.DateTime.Now.ToString("yyyyMMdd.HHmm"));
108     } /*-- /LoadGameData() declaration --*/
109
110
111     /*-----------------------------------------------------------------------
112      * Description       : Sets a parameter of the GameData class.
113      * Parameters        : @member   - the member to set.
114      *                   : @value    - the value to set the member to.
115      */
116     public void SetGameData(string member, int value)
117     {
118         switch(member)
119         {
120             case "pre_survey_question1":
121                 G_GameData.pre_survey_question1 = (float)value;
122                 break;
123             case "pre_survey_question2":
124                 G_GameData.pre_survey_question2 = (float)value;
125                 break;
126             case "post_survey_question1":
127                 G_GameData.post_survey_question1 = (float)value;
128                 break;
129             case "post_survey_question2":
130                 G_GameData.post_survey_question2 = (float)value;
131                 break;
132             default:
133                 break;
134         }
135     } /*-- /SetGameData(string, float) declaration --*/
136
137     public void SetGameData(string member, float value)
138     {
139         switch(member)
140         {
141             case "pre_survey_question1":
142                 G_GameData.pre_survey_question1 = value;
143                 break;
144             case "pre_survey_question2":
145                 G_GameData.pre_survey_question2 = value;
```

```
146                 break;
147             case "post_survey_question1":
148                 G_GameData.post_survey_question1 = value;
149                 break;
150             case "post_survey_question2":
151                 G_GameData.post_survey_question2 = value;
152                 break;
153             default:
154                 break;
155         }
156     } /*-- /SetGameData(string, float) declaration --*/
157
158     public void SetGameData(string member, string value)
159     {
160         switch(member)
161         {
162             case "pre_survey_timestamp":
163                 G_GameData.pre_survey_timestamp = value;
164                 break;
165             case "pre_survey_question3":
166                 G_GameData.pre_survey_question3 = value;
167                 break;
168             case "post_survey_timestamp":
169                 G_GameData.post_survey_timestamp = value;
170                 break;
171             case "post_survey_question3":
172                 G_GameData.post_survey_question3 = value;
173                 break;
174             default:
175                 break;
176         }
177     } /*-- /SetGameData(string, string) declaration --*/
178
179
180     /*-------------------------------------------------------------------------
181      * Description      : Sets a parameter of the GameData class.
182      * Parameters       : @member     - the member to get.
183      *                  : @return_val - a reference to the container storing
184      *                  :               the value of the member.
185      */
186     public void GetGameData(string member, ref int return_val)
187     {
188         switch(member)
189         {
190             case "pre_survey_question1":
191                 return_val = (int)G_GameData.pre_survey_question1;
192                 break;
193             case "pre_survey_question2":
194                 return_val = (int)G_GameData.pre_survey_question2;
195                 break;
```

```
196             case "post_survey_question1":
197                 return_val = (int)G_GameData.post_survey_question1;
198                 break;
199             case "post_survey_question2":
200                 return_val = (int)G_GameData.post_survey_question2;
201                 break;
202             default:
203                 return_val = (int)-1f;
204                 break;
205         }
206     } /*-- /SetGameData(string, int) declaration --*/
207
208     public void GetGameData(string member, ref float return_val)
209     {
210         switch(member)
211         {
212             case "pre_survey_question1":
213                 return_val = G_GameData.pre_survey_question1;
214                 break;
215             case "pre_survey_question2":
216                 return_val = G_GameData.pre_survey_question2;
217                 break;
218             case "post_survey_question1":
219                 return_val = G_GameData.post_survey_question1;
220                 break;
221             case "post_survey_question2":
222                 return_val = G_GameData.post_survey_question2;
223                 break;
224             default:
225                 return_val = -1f;
226                 break;
227         }
228     } /*-- /SetGameData(string, float*) declaration --*/
229
230     public void GetGameData(string member, ref string return_val)
231     {
232         switch(member)
233         {
234             case "pre_survey_timestamp":
235                 return_val = G_GameData.pre_survey_timestamp;
236                 break;
237             case "pre_survey_question3":
238                 return_val = G_GameData.pre_survey_question3;
239                 break;
240             case "post_survey_timestamp":
241                 return_val = G_GameData.post_survey_timestamp;
242                 break;
243             case "post_survey_question3":
244                 return_val = G_GameData.post_survey_question3;
245                 break;
```

```
246             default:
247                 return_val = "";
248                 break;
249         }
250     } /*-- /SetGameData(string, string) declaration --*/
251
252
253     /*-------------------------------------------------------------------------
254      * Description       : Exits the application.
255      */
256     public void ExitApplication()
257     {
258 #if UNITY_EDITOR
259         UnityEditor.EditorApplication.isPlaying = false;
260 #else
261         Application.Quit();
262 #endif
263     } /*-- /ExitApplication() declaration --*/
264     /*====  /MY FUNCTIONS ====*/
265 } /*====  /public class GameController : MonoBehaviour ====*/
266
267
268 /*
269  * This class is used to write the desired data to a file.
270  */
271 [Serializable] class GameData
272 {
273     public string log_timestamp;
274
275     public string pre_survey_timestamp;
276     public float pre_survey_question1;
277     public float pre_survey_question2;
278     public string pre_survey_question3;
279
280     public string post_survey_timestamp;
281     public float post_survey_question1;
282     public float post_survey_question2;
283     public string post_survey_question3;
284 } /*====  /[Serializable] class GameData ====*/
```

Listing 3: JS OSC Receiver

```
// Source: https://github.com/heaversm/unity-osc-receiver
// Modified Dec 4 2016 to use in VRaM project

public var RemoteIP : String = "127.0.0.1"; //127.0.0.1 signifies a local host (if testing locally
public var ListenerPort : int = 9090; //the port you will be listening on
//public var OSCAddress : String = "/filter"; // what OSC messages should be addressed to
public var Target : GameObject;
public var showMessages = true;

private var SendToPort : int = 99999; //the port you will be sending from (not currently used)
private var handler : Osc;
private var TargetScript : Component;
public var TargetScriptName : String = "SkyMasterInterface";


public function Start ()
{
        //Initializes on start up to listen for messages
        //make sure this game object has both UDPPackIO and OSC script attached

        var udp : UDPPacketIO = GetComponent("UDPPacketIO");
        udp.init(RemoteIP, SendToPort, ListenerPort);
        handler = GetComponent("Osc");
        handler.init(udp);
        handler.SetAllMessageHandler(AllMessageHandler);

        TargetScript = Target.GetComponent(TargetScriptName);

        Debug.Log("OSC receiver listening to " + RemoteIP + ", on port " + ListenerPort);
}

function Update () {

}

//These functions are called when messages are received
//Access values via: oscMessage.Values[0], oscMessage.Values[1], etc

public function AllMessageHandler(oscMessage: OscMessage){
    var msgString = Osc.OscMessageToString(oscMessage); //the message and value combined
        var msgAddress = oscMessage.Address; //the message parameters
        var msgValue = oscMessage.Values[0].ToString(); //the message value
        //Debug.Log(msgString); //log the message and values coming from OSC

    //FUNCTIONS YOU WANT CALLED WHEN A SPECIFIC MESSAGE IS RECEIVED
        switch (msgAddress){
            default: // don't filter by address right now
                if (showMessages) Debug.Log("OSC value read: " + msgValue);
                // call external function to update tree colors
```

```
50                  TargetScript.G_CurrentOSCValue = msgValue;
51                          break;
52          }
53
54  }
```

Listing 4: Meditation History Controls

```
 using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using UnityEngine;
using UnityEngine.UI;

public class MeditationHistoryControls : MonoBehaviour
{
    public bool G_Debug = true;
    public GameObject G_GameController = null;
    public GameObject G_MoodSurveyHistory = null;
    public GameObject G_SurveyDataTemplatePanel = null;
    public GameObject G_LogDetailsPanel = null;
    public int G_LogCount = 10;      // Number of logs to load
    private string G_DataPath = ""; // For us it was: C:/Users/ME!/AppData/LocalLow/DefaultCompany/
    project_template/gameData.dat

    // Use this for initialization
    void Start()
    {
        // Do we need to run?
        if (!G_MoodSurveyHistory.activeInHierarchy) { return; }

        // Capture the game controller if necessary
        if (G_GameController == null)
        {
            G_GameController = GameObject.Find("Component_Holder");
        }

        // Prepare to gather files
        G_DataPath = Application.persistentDataPath;
        GetLogs();
    }

    public void GetLogs()
    {
        DirectoryInfo dirInfo = new DirectoryInfo(G_DataPath);
        FileInfo[] files = null;

        // Get Files
        if(dirInfo.Exists)
        {
            files = dirInfo.GetFiles("gameData*.dat");
        }

        /* Capture the n most recent meditation experiences to display
         * The for loop is reversed because it appears that the files are
         * ordered by name. Thus our naming scheme gameData_YYYYMMDD.dat
```

```
49          * will obtain them with later dates at the end.
50          */
51         int loadedLogCount = 0;
52         Vector3 panelPosition = new Vector3(150, 0, 0);
53         for(int i = files.Length; i > 0;)
54         {
55             i--;
56             LoadLog(files[i], ref panelPosition);
57             loadedLogCount++;
58             if(loadedLogCount > G_LogCount) { break; }
59         }
60     }
61
62     private void LoadLog(FileInfo log, ref Vector3 panelPosition)
63     {
64         GameObject logPanel = null;
65         string logTimestamp = "";
66         string preTimestamp = "";
67         float preOverall = 0f;
68         float preStress = 0f;
69         string preNotes = "";
70         string postTimestamp = "";
71         float postOverall = 0f;
72         float postStress = 0f;
73         string postNotes = "";
74
75         // Open File
76         BinaryFormatter bf = new BinaryFormatter();
77         FileStream fs = log.OpenRead();
78         GameData data = new GameData();
79
80         // Obtain Data from file
81         data = (GameData)bf.Deserialize(fs);
82         logTimestamp = data.log_timestamp;
83         preTimestamp = data.pre_survey_timestamp;
84         preOverall = data.pre_survey_question1;
85         preStress = data.pre_survey_question2;
86         preNotes = data.pre_survey_question3;
87         postTimestamp = data.post_survey_timestamp;
88         postOverall = data.post_survey_question1;
89         postStress = data.post_survey_question2;
90         postNotes = data.post_survey_question3;
91         fs.Close();
92
93         // Copy the template survey data panel to display the data of this log
94         logPanel = Object.Instantiate(G_SurveyDataTemplatePanel);
95         logPanel.transform.SetParent(G_SurveyDataTemplatePanel.transform.parent);
96         logPanel.transform.localPosition = panelPosition;
97
98         // Load the data into the new panel
```

```
99          logPanel.transform.Find("Timestamp").GetComponent<Text>().text = logTimestamp;
100
101         logPanel.transform.Find("Pre-Timestamp").GetComponent<Text>().text = preTimestamp;
102         logPanel.transform.Find("Pre-Overall").GetComponent<Slider>().value = preOverall;
103         logPanel.transform.Find("Pre-Stress").GetComponent<Slider>().value = preStress;
104         logPanel.transform.Find("Pre-Notes").GetComponent<Text>().text = preNotes;
105
106         logPanel.transform.Find("Post-Timestamp").GetComponent<Text>().text = postTimestamp;
107         logPanel.transform.Find("Post-Overall").GetComponent<Slider>().value = postOverall;
108         logPanel.transform.Find("Post-Stress").GetComponent<Slider>().value = postStress;
109         logPanel.transform.Find("Post-Notes").GetComponent<Text>().text = postNotes;
110
111         // Update our scroll view width and panel positions
112         RectTransform rt = G_SurveyDataTemplatePanel.transform.parent.GetComponent<RectTransform>();
113         if(panelPosition.x > rt.rect.width)
114         {
115             rt.sizeDelta = new Vector2(rt.rect.width + 325, rt.rect.height);
116         }
117         panelPosition.x += 300; // panel widths of 300 + 25 gap between
118         logPanel.SetActive(true);
119     }
120
121     public void OpenLog(GameObject logPanel)
122     {
123         string logTimestamp = "";
124         string preTimestamp = "";
125         float preOverall = 0f;
126         float preStress = 0f;
127         string preNotes = "";
128         string postTimestamp = "";
129         float postOverall = 0f;
130         float postStress = 0f;
131         string postNotes = "";
132
133         // Capture data
134         logTimestamp = logPanel.transform.Find("Timestamp").GetComponent<Text>().text;
135
136         preTimestamp = logPanel.transform.Find("Pre-Timestamp").GetComponent<Text>().text;
137         preOverall = logPanel.transform.Find("Pre-Overall").GetComponent<Slider>().value;
138         preStress = logPanel.transform.Find("Pre-Stress").GetComponent<Slider>().value;
139         preNotes = logPanel.transform.Find("Pre-Notes").GetComponent<Text>().text;
140
141         postTimestamp = logPanel.transform.Find("Post-Timestamp").GetComponent<Text>().text;
142         postOverall = logPanel.transform.Find("Post-Overall").GetComponent<Slider>().value;
143         postStress = logPanel.transform.Find("Post-Stress").GetComponent<Slider>().value;
144         postNotes = logPanel.transform.Find("Post-Notes").GetComponent<Text>().text;
145
146         // Load data
147         G_LogDetailsPanel.transform.Find("Title").GetComponent<Text>().text = "Log Details for " +
    logTimestamp;
```

```
148         if (preTimestamp != "")
149         {
150             G_LogDetailsPanel.transform.Find("Pre Survey Timestamp").GetComponent<InputField>().text =
    preTimestamp;
151 
152             switch ((int)preOverall)
153             {
154                 case 1:
155                     G_LogDetailsPanel.transform.Find("response1").GetComponent<InputField>().text = "
    Poor";
156                     break;
157                 case 2:
158                     G_LogDetailsPanel.transform.Find("response1").GetComponent<InputField>().text = "
    Could be better";
159                     break;
160                 case 3:
161                     G_LogDetailsPanel.transform.Find("response1").GetComponent<InputField>().text = "OK"
    ;
162                     break;
163                 case 4:
164                     G_LogDetailsPanel.transform.Find("response1").GetComponent<InputField>().text = "
    Pretty Good";
165                     break;
166                 case 5:
167                     G_LogDetailsPanel.transform.Find("response1").GetComponent<InputField>().text = "
    Great!";
168                     break;
169                 default:
170                     break;
171             }
172 
173             switch ((int)preStress)
174             {
175                 case 0:
176                     G_LogDetailsPanel.transform.Find("response2").GetComponent<InputField>().text = "Not
    At All";
177                     break;
178                 case 1:
179                     G_LogDetailsPanel.transform.Find("response2").GetComponent<InputField>().text = "
    Normal Stuff";
180                     break;
181                 case 2:
182                     G_LogDetailsPanel.transform.Find("response2").GetComponent<InputField>().text = "
    Kinda Stressed";
183                     break;
184                 case 3:
185                     G_LogDetailsPanel.transform.Find("response2").GetComponent<InputField>().text = "
    Feeling Pressured";
186                     break;
187                 case 4:
```

```
188                 G_LogDetailsPanel.transform.Find("response2").GetComponent<InputField>().text = "
    Crumbling";
189                 break;
190             case 5:
191                 G_LogDetailsPanel.transform.Find("response2").GetComponent<InputField>().text = "
    Extremely Stressed";
192                 break;
193             default:
194                 break;
195         }
196 
197         G_LogDetailsPanel.transform.Find("response3").GetComponent<InputField>().text = (preNotes ==
     "" ? "" : preNotes);
198     }
199     else
200     {
201         G_LogDetailsPanel.transform.Find("Pre Survey Timestamp").GetComponent<InputField>().text = "
    ";
202         G_LogDetailsPanel.transform.Find("response1").GetComponent<InputField>().text = "";
203         G_LogDetailsPanel.transform.Find("response2").GetComponent<InputField>().text = "";
204         G_LogDetailsPanel.transform.Find("response3").GetComponent<InputField>().text = "";
205     }
206 
207     if (postTimestamp != "")
208     {
209         G_LogDetailsPanel.transform.Find("Post Survey Timestamp").GetComponent<InputField>().text =
    postTimestamp;
210 
211         switch ((int)postOverall)
212         {
213             case 1:
214                 G_LogDetailsPanel.transform.Find("response4").GetComponent<InputField>().text = "
    Much Worse!";
215                 break;
216             case 2:
217                 G_LogDetailsPanel.transform.Find("response4").GetComponent<InputField>().text = "A
    Little Worse";
218                 break;
219             case 3:
220                 G_LogDetailsPanel.transform.Find("response4").GetComponent<InputField>().text = "
    Pretty Much the Same";
221                 break;
222             case 4:
223                 G_LogDetailsPanel.transform.Find("response4").GetComponent<InputField>().text = "A
    Little Better";
224                 break;
225             case 5:
226                 G_LogDetailsPanel.transform.Find("response4").GetComponent<InputField>().text = "
    Much Better!";
227                 break;
```

```
228                 default:
229                     break;
230             }
231 
232             switch ((int)postStress)
233             {
234                 case 0:
235                     G_LogDetailsPanel.transform.Find("response5").GetComponent<InputField>().text = "Not
    At All";
236                     break;
237                 case 1:
238                     G_LogDetailsPanel.transform.Find("response5").GetComponent<InputField>().text = "
    Normal Stuff";
239                     break;
240                 case 2:
241                     G_LogDetailsPanel.transform.Find("response5").GetComponent<InputField>().text = "
    Kinda Stressed";
242                     break;
243                 case 3:
244                     G_LogDetailsPanel.transform.Find("response5").GetComponent<InputField>().text = "
    Feeling Pressured";
245                     break;
246                 case 4:
247                     G_LogDetailsPanel.transform.Find("response5").GetComponent<InputField>().text = "
    Crumbling";
248                     break;
249                 case 5:
250                     G_LogDetailsPanel.transform.Find("response5").GetComponent<InputField>().text = "
    Extremely Stressed";
251                     break;
252                 default:
253                     break;
254             }
255 
256             G_LogDetailsPanel.transform.Find("response6").GetComponent<InputField>().text = (postNotes
    == "" ? "" : postNotes);
257         }
258         else
259         {
260             G_LogDetailsPanel.transform.Find("Post Survey Timestamp").GetComponent<InputField>().text =
    "";
261             G_LogDetailsPanel.transform.Find("response4").GetComponent<InputField>().text = "";
262             G_LogDetailsPanel.transform.Find("response5").GetComponent<InputField>().text = "";
263             G_LogDetailsPanel.transform.Find("response6").GetComponent<InputField>().text = "";
264         }
265 
266         G_LogDetailsPanel.SetActive(true);
267     }
268 
269 
```

```
270     /*-------------------------------------------------------------------------------------
271      * Description       : Exits the application.
272      */
273     public void ExitApplication()
274     {
275 #if UNITY_EDITOR
276         UnityEditor.EditorApplication.isPlaying = false;
277 #else
278         Application.Quit();
279 #endif
280     } /*-- /ExitApplication() declaration --*/
281 }
```

Listing 5: Menu System 2D

```
using UnityEngine;
using UnityEngine.Audio;
using UnityEngine.EventSystems;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.VR;
using UnityStandardAssets.Characters.FirstPerson;
using System.Collections;

public class MenuSystem2D : MonoBehaviour
{
    /* VARIABLE DECLARATIONS ================================================*/
    // Public Variables
    public bool G_Debug = false;            // Print Debug Statements Y/N
    public GameObject G_Player;             // Player Object
    public GameObject G_FPSController;      // The FPS Controller a the "FirstPersonController" script component
    public GameObject G_MainMenu;           // Reference to the MainMenu object under VRMenu
    public AudioMixer G_MasterAudioMixer;   // Applications Audio Mixer
    public EventSystem G_EventSystem;       // Scenes EventSystem(?) Or Canvas' EventSystem(?)
    public AudioSource G_AudioTrack;        // Targeted Audio Track, if not assigned it will grab the object tagged "AudioTrack" on Start
    public Sprite G_PlayImage;              // Instead of hunting through the projects assets
    public Sprite G_PauseImage;             // Instead of hunting through the projects assets

    // Private Variables
    private bool G_MenuOpen = false;        // Current state of the menu
    private string G_CurrentMenu = "Main";  // Current menu screen
    private GameObject G_MenuPanels;        // Collection of menu panels
    private GameObject G_MenuTitleBar;      // The menus title bar
    private GameObject G_SceneFader;        // An image object that is used to "dim" the screen when the menu is open
    private float G_TimeScale = 1f;         // Saves the games time scale on menu open and restores it on close
    private float G_MasterVolume = 1f;      // Saves the games volume on menu open and restores it on close
    private string G_AudioTrackState;       // State of the tagged audio track
    /*====  /VARIABLE DECLARATIONS ====*/


    /* UNITY SPECIFIC FUNCTIONS ============================================*/
    private void Start()
    {
        G_MenuPanels = G_MainMenu.transform.Find("MenuPanels").gameObject;
        G_MenuTitleBar = G_MainMenu.transform.Find("TitleBar").gameObject;
        G_SceneFader = G_MainMenu.transform.Find("SceneFader").gameObject;

        //float fadeTime = GameObject.FindWithTag("ComponentHolder").GetComponentInChildren<SceneFader>().BeginFade(-1);
```

```
44          //new WaitForSeconds(fadeTime);
45      }
46
47
48      private void FixedUpdate()
49      {   // https://community.unity.com/t5/Scripting/Enable-Disable-GameObjects/m-p/2031123/highlight/
        true
50          if((Input.GetKeyDown(KeyCode.P)) || (Input.GetKeyDown(KeyCode.M)))
51          {
52              ToggleMainMenu();
53          }
54      }
55      /*====  /UNITY SPECIFIC FUNCTIONS ====*/
56
57
58      /* MY FUNCTIONS ============================================================*/
59      /*-------------------------------------------------------------------------
60       * Description       : Handles the opening and closing of the main menu.
61       * Pre-Conditions    : a) G_MenuOpen corresponds to the menus current state
62       */
63      public void ToggleMainMenu()
64      {
65          if (G_Debug) { print("Function Start: ToggleMainMenu(G_MenuOpen = " + G_MenuOpen + ",
        G_AudioTrackState =  " + G_AudioTrackState + ")"); }
66          if (!G_MenuOpen)
67          {
68              OpenMenu();
69          }
70          else
71          {
72              CloseMenu();
73          }
74          if (G_Debug) { print("Function Complete: ToggleMainMenu()"); }
75      } /*-- /ToggleMainMenu() declaration --*/
76
77
78      /*-------------------------------------------------------------------------
79       * Description       : Performs actions necessary for opening the main menu.
80       *                   : 1) Pauses audio, scene time, and player controls
81       *                   : 2) Activates the MainMenu panel and the Title bar
82       * Pre-Conditions    : a) Menu elements have been reset to their default visibilities
83       *                   : b) All necessary objects are accessible
84       */
85      private void OpenMenu()
86      {
87          if (G_Debug) { print("Function Start: OpenMenu()"); }
88
89          // Pause the audio track if its playing
90          if(G_AudioTrack.isPlaying)
91          {
```

```
92              G_AudioTrackState = "UnPause";
93              G_AudioTrack.Pause();
94          }
95          else if (G_AudioTrackState == "" || G_AudioTrackState == null)
96          {
97              G_AudioTrackState = "Stopped";
98          }
99
100         // Capture the audio listeners volume - mutes gameplay
101         G_MasterVolume = AudioListener.volume;
102         AudioListener.volume = 0f;
103
104         // Stop the scenes time - stops object movement
105         G_TimeScale = Time.timeScale;
106         Time.timeScale = 0f;
107
108         // Pause Player Controls
109         G_FPSController.GetComponent<FirstPersonController>().enabled = false;
110         Cursor.visible = true;
111         Cursor.lockState = CursorLockMode.None;
112
113         // Display menu
114         G_MenuPanels.SetActive(true);
115         G_MenuTitleBar.SetActive(true);
116         G_SceneFader.SetActive(true);
117         G_MenuOpen = true;
118         if (G_Debug) { print("Function Complete: OpenMenu()"); }
119     } /*-- /OpenMenu() declaration --*/
120
121
122     /*-------------------------------------------------------------------------
123      * Description      : Handles reseting menu visibilities and restoring the
124      *                  : scene state (audio, time, and player controls).
125      */
126     private void CloseMenu()
127     {
128         if (G_Debug) { print("Function Start: CloseMenu()"); }
129         // Restore default menu visibilities
130         ResetMenuPanels();
131         G_MenuPanels.SetActive(false);
132         G_MenuTitleBar.SetActive(false);
133         G_SceneFader.SetActive(false);
134
135         // Restore game state
136         Time.timeScale = G_TimeScale;                // Restore TimeScale
137         AudioListener.volume = G_MasterVolume;       // Restore AudioListener
138         ResumeAudioStateAfterMenuClose();            // Resume Track
139
140         // Resume Player Controls
141         G_FPSController.GetComponent<FirstPersonController>().enabled = true;
```

```
142         Cursor.visible = false;
143         Cursor.lockState = CursorLockMode.Locked;
144 
145         // Finished
146         G_MenuOpen = false;
147         if (G_Debug) { print("Function Complete: CloseMenu()"); }
148     } /*-- /CloseMenu() declaration --*/
149 
150 
151     /*-------------------------------------------------------------------------
152      * Description      : Resets the menu panel visibilities to their default
153      *                  : state.
154      */
155     private void ResetMenuPanels()
156     {
157         if(G_Debug) { print("Function Start: ResetMenuPanels()"); }
158         GameObject temp;
159 
160         // Hide Sub Menus and Show Main when complete
161         temp = G_MenuPanels.transform.Find("Main").gameObject;
162         if(temp != null) temp.SetActive(true);
163         G_CurrentMenu = "Main";
164 
165         temp = G_MenuPanels.transform.Find("SceneSelection").gameObject;
166         if(temp != null) temp.SetActive(false);
167 
168         temp = G_MenuPanels.transform.Find("TrackSelection").gameObject;
169         if(temp != null) temp.SetActive(false);
170 
171         temp = G_MenuPanels.transform.Find("SceneOptions").gameObject;
172         if(temp != null) temp.SetActive(false);
173 
174         temp = G_MenuPanels.transform.Find("ApplicationSettings").gameObject;
175         if(temp != null) temp.SetActive(false);
176 
177         temp = G_MenuPanels.transform.Find("AudioSettings").gameObject;
178         if(temp != null) temp.SetActive(false);
179 
180         if(G_Debug) { print("Function Complete: ResetMenuPanels()"); }
181     } /*-- /ResetMenuPanels() declaration --*/
182 
183 
184     /*-------------------------------------------------------------------------
185      * Description      : Handles navigating the menu system by showing and
186      *                  : hiding panels.
187      */
188     public void ChangeMenu(string selectedMenu)
189     {
190         if(G_Debug) { print("Function Start: ChangeMenu(G_CurrentMenu = " + G_CurrentMenu + ", 
    selectedMenu = " + selectedMenu + ")"); }
```

```
191
192         if(selectedMenu != null && selectedMenu != "")
193         {
194             GameObject menuA, menuB;
195             menuA = G_MenuPanels.transform.FindChild(G_CurrentMenu).gameObject;
196             menuB = G_MenuPanels.transform.FindChild(selectedMenu).gameObject;
197
198             if(menuA != null && menuB != null)
199             {
200                 menuA.SetActive(false);      // Hide Previous Menu
201                 menuB.SetActive(true);       // Display Desired Menu
202                 G_CurrentMenu = selectedMenu; // Update menu reference
203             }
204         }
205
206         if(G_Debug) { print("Function Complete: ChangeMenu()"); }
207     } /*— /ChangeMenu() declaration —*/
208
209
210     /*
211      * Description      : If implemented this would allow the user to change between
212      *                  : various meditations including Body Scan, Guide, Self, etc.
213      */
214     public void ChangeMeditationType()
215     {
216         // As one can see I don't have code for this.
217     } /*— /ChangeMeditationType() declaration —*/
218
219
220     /*————————————————————————————————————————————————————————————————
221      * Description      : Handles the tagged AudioTrack's state after the menu closes
222      * Pre-Conditions   : a) The desired AudioTrack is captured in G_AudioTrack
223      *                  : b) The desired AudioTrack state is saved in G_AudioTrackState
224      * Post-Conditions  : The G_AudioTrack will be set to whatever state the user desires.
225      */
226     private void ResumeAudioStateAfterMenuClose()
227     {
228         if(G_Debug) { print("Function Start: ResumeAudioStateAfterMenuClose(G_AudioTrackState = " +
    G_AudioTrackState + ")"); }
229
230         float transitionPeriod = 1f;
231         switch(G_AudioTrackState)
232         {
233             case "Restarted":
234                 G_AudioTrack.time = 0f;
235                 G_AudioTrackState = "Playing";
236                 G_MasterAudioMixer.FindSnapshot("Meditation").TransitionTo(transitionPeriod);
237                 G_AudioTrack.Play();
238                 break;
239             case "Stopped":
```

```
240                 G_MasterAudioMixer.FindSnapshot("Default").TransitionTo(transitionPeriod);
241                 G_AudioTrack.Stop();
242                 break;
243             case "Playing":
244                 G_MasterAudioMixer.FindSnapshot("Meditation").TransitionTo(transitionPeriod);
245                 G_AudioTrack.Play();
246                 break;
247             case "UnPause":
248                 G_AudioTrackState = "Playing";
249                 G_MasterAudioMixer.FindSnapshot("Meditation").TransitionTo(transitionPeriod);
250                 G_AudioTrack.Play();
251                 break;
252             case "Paused":
253                 G_MasterAudioMixer.FindSnapshot("Default").TransitionTo(transitionPeriod);
254                 G_AudioTrack.Pause();
255                 break;
256         }
257 
258         if(G_Debug) { print("Function Complete: ResumeAudioStateAfterMenuClose()"); }
259     } /*-- /ResumeAudioStateAfterMenuClose() declaration --*/
260 
261 
262     /*-------------------------------------------------------------------------
263      * Description      : Toggles (play/pause) the tagged audio source.
264      * Pre-Conditions   : a) The target source is tagged "AudioTrack"
265      */
266     public void AudioTrackControls(string action)
267     {
268         if(G_Debug) { print("Function Start: AudioTrackControls(action = " + action + ")"); }
269 
270         GameObject button = null;
271         button = G_MenuPanels;
272         button = button.transform.Find("Main").gameObject;
273         button = button.transform.Find("TrackQuickAccess").gameObject;
274         button = button.transform.Find("btn_ToggleMeditationTrack").gameObject;
275 
276         switch(action)
277         {
278             case "Play/Pause":
279                 if(G_AudioTrackState == "Playing"
280                 || G_AudioTrackState == "UnPause")
281                 {
282                     G_AudioTrackState = "Paused";
283                     button.GetComponent<Image>().sprite = G_PlayImage;
284                 }
285                 else
286                 {
287                     G_AudioTrackState = "Playing";
288                     button.GetComponent<Image>().sprite = G_PauseImage;
289                 }
```

```
290                 break;
291             case "Stop":
292                 G_AudioTrackState = "Stopped";
293                 button.GetComponent<Image>().sprite = G_PlayImage;
294                 break;
295             case "Restart":
296                 G_AudioTrackState = "Restarted";
297                 button.GetComponent<Image>().sprite = G_PauseImage;
298                 break;
299             default:
300                 // leave audio track in its current state.
301                 break;
302         }
303         if(G_Debug) { print("New state: " + G_AudioTrackState); }
304 
305         if(G_Debug) { print("Function Complete: AudioTrackControls()"); }
306     } /*-- /AudioTrackControls() declaration --*/
307 
308 
309     /*-------------------------------------------------------------------------------
310      * Description      : Changes the current audio track of the Meditation Track audio source.
311      * Pre-Conditions   : Audio tracks must be inside of [Assets Folder]/Resources/Audio/
    MeditationTracks
312      *                  : as the Resources.Load() function uses pathing relative to the Resources folder
    .
313      */
314     public void ChangeTracks(string trackName)
315     {
316         AudioClip nextTrack = null;
317         GameObject button = null;
318         button = G_MenuPanels;
319         button = button.transform.Find("Main").gameObject;
320         button = button.transform.Find("TrackQuickAccess").gameObject;
321         button = button.transform.Find("btn_ToggleMeditationTrack").gameObject;
322 
323         if (trackName == "none")
324         {
325             G_AudioTrack.GetComponent<AudioSource>().Stop();
326             button.GetComponent<Image>().sprite = G_PlayImage;
327         }
328         else if(trackName == "Cycle")
329         {
330             // Play all tracks, I don't have code for this obviously
331         }
332         else
333         {
334             nextTrack = Resources.Load<AudioClip>("Audio/Meditation Tracks/" + trackName);
335             G_AudioTrack.GetComponent<AudioSource>().clip = nextTrack;
336             button.GetComponent<Image>().sprite = G_PauseImage;
337             G_AudioTrackState = "Playing";
```

```
338         }
339     } /*— /ChangeTracks() declaration —*/
340
341
342     /*————————————————————————————————————————————————————————————————————
343      * Description       : Updates values of the Master Audio Mixer based upon the sliders value.
344      * Pre-Conditions    : Volumes of the Audio mixer are exposed as parameters
345      *                   : See: https://docs.unity3d.com/Manual/AudioMixerOverview.html
346      */
347     public void UpdateVolume(string mixerGroup)
348     {
349         if (G_MenuPanels != null)
350         {
351             GameObject audioSettingsPanel = G_MenuPanels.transform.Find("AudioSettings").gameObject;
352             if (audioSettingsPanel != null)
353             {
354                 Slider obj = audioSettingsPanel.transform.Find(mixerGroup).GetComponent<Slider>();
355
356                 // Update Scene Volume
357                 if (obj != null)
358                 {
359                     G_MasterAudioMixer.SetFloat(mixerGroup, obj.value);
360                 }
361             }
362         }
363     } /*— /UpdateVolume() declaration —*/
364
365
366     /*————————————————————————————————————————————————————————————————————
367      * Description       : Handles the changing of scenes.
368      * Parameters        : (int) sceneIndex - Index of scene to load
369      * Dependencies      : (Script) Scene Fader
370      * Pre-Conditions    : a) Desired Scene is part of the current set of scenes built
371      *                   : b) The scene fader script is present and attached as a child of the Scene
    Components object
372      * Notes             : Scenes are obtained via zero based indexes as set during the projects build.
373      *                   : See File -> Build Settings... for indexes
374      */
375     public void ChangeScene(int sceneIndex)
376     {
377         if (G_Debug) { print("Function Start: ChangeScene(sceneIndex = " + sceneIndex + ")"); }
378
379         float fadeTime = GameObject.FindWithTag("ComponentHolder").GetComponentInChildren<SceneFader>().
    BeginFade(1);
380         if (G_Debug) { print("WaitForSeconds(" + Time.time * fadeTime + ")"); }
381         new WaitForSeconds(fadeTime);
382
383         // Restore scene defaults so we have them on the next scene
384         CloseMenu();
385
```

```
386             // Perform scene change
387             SceneManager.LoadScene(sceneIndex);
388 
389             if (G_Debug) { print("Function Complete: ChangeScene()"); }
390         } /*-- /ChangeScene() declaration --*/
391 
392 
393         /*-----------------------------------------------------------------------------
394          * Description       : Exits the application.
395          */
396         public void ExitApplication()
397         {
398 #if UNITY_EDITOR
399             UnityEditor.EditorApplication.isPlaying = false;
400 #else
401             Application.Quit();
402 #endif
403         } /*-- /ExitApplication() declaration --*/
404         /*====  /MY FUNCTIONS ====*/
405 } /*====  /public class MenuSystem2D : MonoBehaviour ====*/
```

Listing 6: Menu System VR

```
using UnityEngine;
using UnityEngine.Rendering;
using UnityEngine.Audio;
using UnityEngine.EventSystems;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using UnityEngine.VR;
using System.Collections;
using Valve.VR;

public class MenuSystemVR : MonoBehaviour
{
    /* VARIABLE DECLARATIONS ================================================*/
    // Public Variables
    public bool G_Debug = false;              // Print Debug Statements Y/N
    public GameObject G_Player;               // Player Object
    public GameObject G_SteamVRCameraRig;     // SteamVR Camera Object
    public GameObject G_MainMenu;             // Reference to the MainMenu object under VRMenu
    public AudioMixer G_MasterAudioMixer;     // Applications Audio Mixer
    public EventSystem G_EventSystem;         // Scenes EventSystem(?) Or Canvas' EventSystem(?)
    public AudioSource G_AudioTrack;          // Targeted Audio Track, if not assigned it will grab the object tagged "AudioTrack" on Start
    public Sprite G_PlayImage;                // Instead of hunting through the projects assets
    public Sprite G_PauseImage;               // Instead of hunting through the projects assets

    // Private Variables
    private bool G_MenuOpen = false;          // Current state of the menu
    private string G_CurrentMenu = "Main";    // Current menu screen
    private float G_TimeScale = 1f;           // Saves the games time scale on menu open and restores it on close
    private float G_MasterVolume = 1f;        // Saves the games volume on menu open and restores it on close
    private SteamVR_Controller.Device G_LeftControllerDev;
    private SteamVR_Controller.Device G_RightControllerDev;
    private string G_AudioTrackState;         // State of the tagged audio track
    /*====  /VARIABLE DECLARATIONS ====*/


    /* UNITY SPECIFIC FUNCTIONS ============================================*/
    private void Awake()
    {
    }

    private void Start()
    {
        if (SteamVR.active)
        {
            GameObject button = null;

```

```
47              // Capture the current menu state
48              G_MenuOpen = G_MainMenu.activeInHierarchy;
49
50              // Capture the Audio Track and save its state
51              button = G_MainMenu.transform.FindChild("AudioTrackControls").FindChild("ToggleAudioTrack").
        gameObject;
52              G_AudioTrack = GameObject.FindGameObjectWithTag("AudioTrack").GetComponent<AudioSource>();
53              if (G_AudioTrack.isPlaying)
54              {
55                  G_AudioTrackState = "Playing";
56                  button.GetComponent<Image>().sprite = G_PauseImage;
57              }
58              else
59              {
60                  G_AudioTrackState = "Stopped";
61                  button.GetComponent<Image>().sprite = G_PlayImage;
62              }
63
64              // Capture the controllers so we can reference them in Update()
65              UpdateControllerDeviceReferences();
66
67              // Capture the players location and set the menus offset
68              //G_MenuOffset = transform.position - G_SteamVRCameraRig.transform.FindChild("Camera (eye)")
        .transform.position;
69          }
70
71          float fadeTime = GameObject.FindWithTag("ComponentHolder").GetComponentInChildren<SceneFader>().
        BeginFade(-1);
72          if (G_Debug) { print("Function Start: Wait at " + Time.time); }
73          StartCoroutine(wait(fadeTime));
74          if (G_Debug) { print("Function Start: Resume at " + Time.time); }
75      }
76
77      IEnumerator wait(float duration)
78      {
79          yield return new WaitForSeconds(duration);
80      }
81
82      private void Update()
83      {
84          UpdateControllerDeviceReferences(); // Uncomment this if we can't rely on the references
        gathered from Start()
85
86          if ((G_LeftControllerDev != null && G_LeftControllerDev.GetPressUp(SteamVR_Controller.ButtonMask
        .ApplicationMenu))
87          || (G_RightControllerDev != null && G_RightControllerDev.GetPressUp(SteamVR_Controller.
        ButtonMask.ApplicationMenu))
88          || (Input.GetKeyDown(KeyCode.P)) || (Input.GetKeyDown(KeyCode.M)))
89          {
90              ToggleMainMenu();
```

```
91          }
92      }
93
94      private void FixedUpdate()
95      {
96      }
97
98      private void LateUpdate()
99      {
100         if (!G_MenuOpen)
101         {
102             G_SteamVRCameraRig.transform.Find("Controller (left)").gameObject.SetActive(false);
103             G_SteamVRCameraRig.transform.Find("Controller (right)").gameObject.SetActive(false);
104         }
105         //transform.position = G_SteamVRCameraRig.transform.FindChild("Camera (eye)").transform.position
     + G_MenuOffset;
106     }
107     /*====  /UNITY SPECIFIC FUNCTIONS ====*/
108
109
110
111     /* MY FUNCTIONS ============================================================*/
112     /*-------------------------------------------------------------------------
113      * Description      : Obtains the SteamVR_Controller.Device for the left and right
114      *                  : controllers.
115      */
116     private void UpdateControllerDeviceReferences()
117     {
118         int controllerIndex = -1;
119
120         controllerIndex = SteamVR_Controller.GetDeviceIndex(SteamVR_Controller.DeviceRelation.Leftmost,
    ETrackedDeviceClass.Controller);
121         G_LeftControllerDev = SteamVR_Controller.Input(controllerIndex);
122         if (G_Debug)
123         {
124             print("Left Controller Device Found? " + (G_LeftControllerDev == null ? "No" : "Yes"));
125             print("Left Controller Index? " + G_LeftControllerDev.index);
126         }
127
128
129         controllerIndex = SteamVR_Controller.GetDeviceIndex(SteamVR_Controller.DeviceRelation.Rightmost,
     ETrackedDeviceClass.Controller);
130         G_RightControllerDev = SteamVR_Controller.Input(controllerIndex);
131         if (G_Debug)
132         {
133             print("Right Controller Device Found? " + (G_RightControllerDev == null ? "No" : "Yes"));
134             print("Right Controller Index? " + G_RightControllerDev.index);
135         }
136
137
```

```
138         if (G_LeftControllerDev.index == G_RightControllerDev.index)
139         {
140             if (G_Debug) { print("Player is only using one controller."); }
141 
142             // We'll disable the left controller, because the right is the first assigned (see
    SteamVR_ControllerManager::Refresh()
143             G_SteamVRCameraRig.transform.Find("Controller (left)").gameObject.SetActive(false);
144             G_LeftControllerDev = null;
145 
146             if (G_Debug) { print("Left controller disabled."); }
147         }
148     }/*-- /UpdateControllerDeviceReferences() declaration --*/
149 
150 
151     /*-------------------------------------------------------------------------
152      * Description      : Handles the opening and closing of the main menu.
153      * Pre-Conditions   : a) G_MenuOpen corresponds to the menus current state
154      */
155     public void ToggleMainMenu()
156     {
157         if (G_Debug) { print("Function Start: ToggleMainMenu(G_MenuOpen = " + G_MenuOpen + ",
    G_AudioTrackState = " + G_AudioTrackState + ")"); }
158         if (!G_MenuOpen)
159         {
160             OpenMenu();
161         }
162         else
163         {
164             CloseMenu();
165         }
166         if (G_Debug) { print("Function Complete: ToggleMainMenu()"); }
167     } /*-- /ToggleMainMenu() declaration --*/
168 
169 
170     /*-------------------------------------------------------------------------
171      * Description      : Opens default menu objects, captures the scenes time
172      *                  : scale and volume then sets them to 0f, pauses user's
173      *                  : ability to move.
174      * Pre-Conditions   : a) Menu elements have been reset to their default visibilities
175      *                  : b) All necessary objects are accessible
176      */
177     private void OpenMenu()
178     {
179         if (G_Debug) { print("Function Start: OpenMenu()"); }
180         GameObject controller = null;
181 
182 
183         // Pause the audio track if its playing
184         if (G_AudioTrack.isPlaying)
185         {
```

```
186             G_AudioTrackState = "UnPause";
187             G_AudioTrack.Pause();
188         }
189 
190         // Capture the audio listeners volume - mutes gameplay
191         G_MasterVolume = AudioListener.volume;
192         AudioListener.volume = 0f;
193 
194         // Stop the scenes time - stops object movement
195         G_TimeScale = Time.timeScale;
196         Time.timeScale = 0f;
197 
198         // Turn on the controllers model and laser pointer
199         if (G_LeftControllerDev != null)
200         {
201             controller = G_SteamVRCameraRig.transform.Find("Controller (left)").gameObject;
202             controller.SetActive(true);
203             controller.transform.Find("Model").GetComponent<SteamVR_RenderModel>().SetModel(controller.
    transform.Find("Model").GetComponent<SteamVR_RenderModel>().renderModelName);
204         }
205         if (G_RightControllerDev != null)
206         {
207             controller = G_SteamVRCameraRig.transform.Find("Controller (right)").gameObject;
208             controller.SetActive(true);
209             controller.transform.Find("Model").GetComponent<SteamVR_RenderModel>().SetModel(controller.
    transform.Find("Model").GetComponent<SteamVR_RenderModel>().renderModelName);
210         }
211 
212 
213         // Open the menu
214         G_MainMenu.SetActive(true);
215         G_MenuOpen = true;
216         if (G_Debug) { print("Function Complete: OpenMenu()"); }
217     } /*-- /OpenMenu() declaration --*/
218 
219 
220     /*-------------------------------------------------------------------------
221      * Description      : Resets menu objects to their default visibilities,
222      *                  : restores the scenes time scale and volume, and
223      *                  : restores user controls.
224      */
225     private void CloseMenu()
226     {
227         if (G_Debug) { print("Function Start: CloseMenu()"); }
228         // Turn off the controllers model and laser pointer
229         if (G_LeftControllerDev != null) G_SteamVRCameraRig.transform.Find("Controller (left)").
    gameObject.SetActive(false);
230         if (G_RightControllerDev != null) G_SteamVRCameraRig.transform.Find("Controller (right)").
    gameObject.SetActive(false);
231 
```

```
232         // Restore default menu visibilities
233         G_MainMenu.SetActive(false);
234 
235         // Restore game state
236         Time.timeScale = G_TimeScale;                   // Restore TimeScale
237         AudioListener.volume = G_MasterVolume;          // Restore AudioListener
238         G_MenuOpen = false;
239         ResumeAudioStateAfterMenuClose();               // Resume Track
240         if (G_Debug) { print("Function Complete: CloseMenu()"); }
241     } /*-- /CloseMenu() declaration --*/
242 
243 
244     /*-----------------------------------------------------------------------------
245      * Description      : Handles the tagged AudioTrack's state after the menu closes
246      * Pre-Conditions   : a) The desired AudioTrack is captured in G_AudioTrack
247      *                  : b) The desired AudioTrack state is saved in G_AudioTrackState
248      * Post-Conditions  : The G_AudioTrack will be set to whatever state the user desires.
249      */
250     private void ResumeAudioStateAfterMenuClose()
251     {
252         if (G_Debug) { print("Function Start: ResumeAudioStateAfterMenuClose(G_AudioTrackState = " +
    G_AudioTrackState + ")"); }
253 
254         float transitionPeriod = 1f;
255         switch (G_AudioTrackState)
256         {
257             case "Restarted":
258                 G_AudioTrack.time = 0f;
259                 G_AudioTrackState = "Playing";
260                 G_MasterAudioMixer.FindSnapshot("Meditation").TransitionTo(transitionPeriod);
261                 G_AudioTrack.Play();
262                 break;
263             case "Stopped":
264                 G_MasterAudioMixer.FindSnapshot("Default").TransitionTo(transitionPeriod);
265                 G_AudioTrack.Stop();
266                 break;
267             case "Playing":
268                 G_MasterAudioMixer.FindSnapshot("Meditation").TransitionTo(transitionPeriod);
269                 G_AudioTrack.Play();
270                 break;
271             case "UnPause":
272                 G_AudioTrackState = "Playing";
273                 G_MasterAudioMixer.FindSnapshot("Meditation").TransitionTo(transitionPeriod);
274                 G_AudioTrack.Play();
275                 break;
276             case "Paused":
277                 G_MasterAudioMixer.FindSnapshot("Default").TransitionTo(transitionPeriod);
278                 G_AudioTrack.Pause();
279                 break;
280         }
```

```
281
282         if (G_Debug) { print("Function Complete: ResumeAudioStateAfterMenuClose()"); }
283     } /*-- /ResumeAudioStateAfterMenuClose() declaration --*/
284
285
286     /*-------------------------------------------------------------------------
287      * Description      : Toggles (play/pause) the tagged audio source.
288      * Pre-Conditions   : a) The target source is tagged "AudioTrack"
289      */
290     public void AudioTrackControls(string action)
291     {
292         if (G_Debug) { print("Function Start: AudioTrackControls(action = " + action + ")"); }
293
294         GameObject button = GameObject.Find("ToggleAudioTrack");
295         switch (action)
296         {
297             case "Play/Pause":
298                 if (G_AudioTrackState == "Playing"
299                 || G_AudioTrackState == "UnPause")
300                 {
301                     G_AudioTrackState = "Paused";
302                     button.GetComponent<Image>().sprite = G_PlayImage;
303                 }
304                 else
305                 {
306                     G_AudioTrackState = "Playing";
307                     button.GetComponent<Image>().sprite = G_PauseImage;
308                 }
309                 break;
310             case "Stop":
311                 G_AudioTrackState = "Stopped";
312                 button.GetComponent<Image>().sprite = G_PlayImage;
313                 break;
314             case "Restart":
315                 G_AudioTrackState = "Restarted";
316                 button.GetComponent<Image>().sprite = G_PauseImage;
317                 break;
318             default:
319                 // leave audio track in its current state.
320                 break;
321         }
322         if (G_Debug) { print("New state: " + G_AudioTrackState); }
323
324         // ResumeAudioStateAfterMenuClose();            // Resume Track, comment out if the main menu is
        implemented
325
326         if (G_Debug) { print("Function Complete: AudioTrackControls()"); }
327     } /*-- /AudioTrackControls() declaration --*/
328
329
```

```
330     /*-------------------------------------------------------------------------------
331      * Description       : Handles the changing of scenes.
332      * Parameters        : (int) sceneIndex - Index of scene to load
333      * Dependencies      : (Script) Scene Fader
334      * Pre-Conditions    : a) Desired Scene is part of the current set of scenes built
335      *                   : b) The scene fader script is present and attached as a child of the Scene
     Components object
336      * Notes             : Scenes are obtained via zero based indexes as set during the projects build.
337      *                   : See File -> Build Settings... for indexes
338      */
339     public void ChangeScene(int sceneIndex)
340     {
341         if (G_Debug) { print("Function Start: ChangeScene(sceneIndex = " + sceneIndex + ")"); }
342 
343         float fadeTime = GameObject.FindWithTag("ComponentHolder").GetComponentInChildren<SceneFader>().
     BeginFade(1);
344         if (G_Debug) { print("WaitForSeconds(" + Time.time * fadeTime + ")"); }
345         new WaitForSeconds(fadeTime);
346 
347         // Restore scene defaults so we have them on the next scene
348         CloseMenu();
349 
350         // Perform scene change
351         SceneManager.LoadScene(sceneIndex);
352 
353         if (G_Debug) { print("Function Complete: ChangeScene()"); }
354     } /*-- /ChangeScene() declaration --*/
355 
356 
357     /*-------------------------------------------------------------------------------
358      * Description       : Exits the application.
359      */
360     public void ExitApplication()
361     {
362 #if UNITY_EDITOR
363         UnityEditor.EditorApplication.isPlaying = false;
364 #else
365         Application.Quit();
366 #endif
367     } /*-- /ExitApplication() declaration --*/
368     /*====  /MY FUNCTIONS ====*/
369 } /*====  /public class MenuSystemVR : MonoBehaviour ====*/
```

Listing 7: Menu Trigger Pull

```
1  using UnityEngine;
2 using System.Collections;
3
4 public class MenuTriggerPull : MonoBehaviour {
5
6     private SteamVR_Controller.Device device;
7     private SteamVR_TrackedObject controller;
8
9     void Awake()
10    {
11        controller = GetComponent<SteamVR_TrackedObject>();
12    }
13
14    void FixedUpdate()
15    {
16        //sets device to the tracking index
17        device = SteamVR_Controller.Input((int)controller.index);
18        var playerCamera = GetComponentInParent<Camera>();
19
20
21        if (device.GetPress(SteamVR_Controller.ButtonMask.Trigger))
22        {
23            Debug.Log("trigger pressed, in PlayMedAudio.cs: FixedUpdate()");
24            // Transition scene
25            Application.LoadLevel("Boxed In");
26
27
28        }
29    }
30 }
```

Listing 8: Orb Controller

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR;
using Valve.VR.InteractionSystem;

public class OrbController : MonoBehaviour
{
    /* VARIABLE DECLARATIONS ================================================*/
    // Public Variables
    public bool G_Debug = false;
    public bool G_PrintOnce = true;
    public bool G_SlowOnHoverEnd = false;
    public GameObject G_TargetController = null;
    public OrbController G_OtherOrbController = null;
    public Animator G_OrbAnimation = null;
    public GameObject G_EasterEgg = null;

    [HideInInspector]
    public bool G_HoldingCollission = false;
    [HideInInspector]
    public bool G_HoldingOpposite = false;
    [HideInInspector]
    public int G_ConnectedControllerCount = 0;

    // Private Variables
    private float G_FullAnimationSpeed = .75f;
    private float G_SlowAnimationSpeed = 0.35f;
    private SteamVR_Controller.Device G_TargetControllerDev = null;
    private SteamVR_Controller.Device G_OtherControllerDev = null;


    /* UNITY FUNCTIONS ======================================================*/
    // Use this for initialization
    void Start()
    {
        G_FullAnimationSpeed = G_OrbAnimation.speed;
        G_OrbAnimation.speed = 0;
    }

    // Update is called once per frame
    void Update()
    {
        /*
         * Get Controller Devices
         */
        SteamVR_TrackedObject obj = null;
        if (G_TargetControllerDev == null || !G_TargetControllerDev.connected)
        {
```

```
50              obj = G_TargetController.GetComponent<SteamVR_TrackedObject>();
51              if (obj.index > 0)
52                  G_TargetControllerDev = SteamVR_Controller.Input((int)obj.index);
53          }
54          if (G_OtherControllerDev == null || !G_OtherControllerDev.connected)
55          {
56              obj = G_OtherOrbController.G_TargetController.GetComponent<SteamVR_TrackedObject>();
57              if (obj.index > 0)
58                  G_OtherControllerDev = SteamVR_Controller.Input((int)obj.index);
59          }
60
61
62          /*
63           * Get the number of connected controllers
64           */
65          if (G_TargetControllerDev != null
66          && G_TargetControllerDev.connected)
67          {
68              if (G_Debug)
69              {
70                  print(G_TargetController.name + " detected with index = " + G_TargetControllerDev.index)
    ;
71              }
72
73              if (G_OtherControllerDev != null
74              && G_OtherControllerDev.connected)
75              {   // Both Controllers Detected
76                  if (G_Debug) { print("Both controllers detected."); }
77                  G_ConnectedControllerCount = 2;
78              }
79              else
80              {   // Just This Controller Detected
81                  if (G_Debug) { print("Target controller detected."); }
82                  G_ConnectedControllerCount = 1;
83              }
84          }
85          else if (G_OtherControllerDev != null
86          && G_OtherControllerDev.connected)
87          {   // Just The Other Controller Detected
88              if (G_Debug) { print("Opposite controller detected."); }
89              G_ConnectedControllerCount = 1;
90          }
91          else
92          {   // No Controllers Detected
93              if (G_Debug) { print("No controllers detected."); }
94              G_ConnectedControllerCount = 0;
95          }
96          if (G_Debug || (G_PrintOnce))
97          {
98              print("G_ConnectedControllerCount = " + G_ConnectedControllerCount);
```

```
99          }
100
101
102         /*
103          * Adjust for and begin animation based upon the number of controllers
104          * present.
105          *
106          * If there is only one controller, disable orb2
107          * Else enable both controllers
108          */
109         switch(G_ConnectedControllerCount)
110         {
111             case 0:             // No controllers detected
112                 // Do nothing?
113                 break;
114
115
116             case 1:             // One controller detected
117                 /*
118                  * Disable orb2
119                  */
120                 if (gameObject.name != "Orb 2")
121                 {
122                     G_OtherOrbController.gameObject.SetActive(false);
123                 }
124                 else
125                 {
126                     gameObject.SetActive(false);
127                     return;
128                 }
129
130
131                 /*
132                  * ---- Display easter egg ----
133                  *
134                  * If player crosses and holds controllers and
135                  * presses opposite directions on the touch pad.
136                  */
137                 if (G_HoldingOpposite
138                 && G_TargetControllerDev != null
139                 && G_TargetControllerDev.GetPressDown(SteamVR_Controller.ButtonMask.Touchpad))
140                 {
141                     //G_TargetControllerDev.TriggerHapticPulse();
142                     //G_OtherControllerDev.TriggerHapticPulse();
143                     G_EasterEgg.SetActive(true);
144                 }
145
146
147                 /*
148                  * ---- Begin/Resume Animation ----
```

```
149                  *
150                  * If G_SlowOnHoverEnd is enabled
151                  *      if controller is contacting its orb -> full speed
152                  *      else -> slowed speed
153                  * else if both controller contacts its orb -> full speed
154                  */
155                 if (G_SlowOnHoverEnd)
156                 {
157                     if (G_HoldingCollission
158                     && G_OtherOrbController.G_HoldingCollission
159                     && G_OrbAnimation.speed == G_SlowAnimationSpeed)
160                     {   // Begin or resume animation
161                         G_OrbAnimation.speed = G_FullAnimationSpeed;
162                     }
163                     else if ((!G_HoldingCollission
164                     || !G_OtherOrbController.G_HoldingCollission)
165                     && (G_OrbAnimation.speed != G_SlowAnimationSpeed
166                     && G_OrbAnimation.speed != 0f))
167                     {   // Slow animation if it's not already
168                         G_OrbAnimation.speed = G_SlowAnimationSpeed;
169                     }
170                 }
171                 else if (G_HoldingCollission)
172                 {   // Begin animation at full speed
173                     G_OrbAnimation.speed = G_FullAnimationSpeed;
174                 }
175                 break;
176 
177 
178             case 2: default:    // Both controllers detected
179                 /*
180                  * Enable orb2
181                  */
182                 if (gameObject.name != "Orb 2")
183                 {
184                     G_OtherOrbController.gameObject.SetActive(true);
185                 }
186                 else
187                 {
188                     gameObject.SetActive(true);
189                 }
190 
191 
192                 /*
193                  * ---- Display easter egg ----
194                  *
195                  * If player crosses and holds controllers and
196                  * presses opposite directions on the touch pad.
197                  */
198                 if (G_HoldingOpposite
```

```
199             && G_TargetControllerDev != null
200             && G_OtherControllerDev != null
201             && G_TargetControllerDev.GetPressDown(SteamVR_Controller.ButtonMask.Touchpad)
202             && G_OtherControllerDev.GetPressDown(SteamVR_Controller.ButtonMask.Touchpad))
203             {
204                 //G_TargetControllerDev.TriggerHapticPulse();
205                 //G_OtherControllerDev.TriggerHapticPulse();
206                 G_EasterEgg.SetActive(true);
207             }
208
209
210             /*
211              * ---- Begin/Resume Animation ----
212              *
213              * If G_SlowOnHoverEnd is enabled
214              *      if both controllers are contacting their orbs -> full speed
215              *      else -> slowed speed
216              * else if both controllers contact their orbs -> full speed
217              */
218             if (G_SlowOnHoverEnd)
219             {
220                 if (G_HoldingCollission
221                 && G_OtherOrbController.G_HoldingCollission
222                 && G_OrbAnimation.speed == G_SlowAnimationSpeed)
223                 {   // Begin or resume animation
224                     G_OrbAnimation.speed = G_FullAnimationSpeed;
225                 }
226                 else if ((!G_HoldingCollission
227                 || !G_OtherOrbController.G_HoldingCollission)
228                 && (G_OrbAnimation.speed != G_SlowAnimationSpeed
229                 && G_OrbAnimation.speed != 0f))
230                 {   // Slow animation if it's not already
231                     G_OrbAnimation.speed = G_SlowAnimationSpeed;
232                 }
233             }
234             else if (G_HoldingCollission
235             && G_OtherOrbController.G_HoldingCollission)
236             {   // Begin animation at full speed
237                 G_OrbAnimation.speed = G_FullAnimationSpeed;
238             }
239             break;
240         }
241
242
243         /*
244          * Manual Overrides
245          *
246          * Alpha1 - starts the animation
247          * Alpha2 - toggles G_SlowOnHoverEnd
248          * //Alpha3 - stops the animation
```

```
249             * //Alpha4 - restarts the animation
250             */
251             if (Input.GetKeyDown(KeyCode.Alpha1))
252             {   // Start animation
253                 G_OrbAnimation.speed = G_FullAnimationSpeed;
254             }
255             else if (Input.GetKeyDown(KeyCode.Alpha2))
256             {   // Enable/Disable pause on hover end
257                 print("Pause on hand hover end toggled, active = " + !G_SlowOnHoverEnd);
258                 G_SlowOnHoverEnd = !G_SlowOnHoverEnd;
259 
260                 if (G_OrbAnimation.speed == G_FullAnimationSpeed)
261                 {   // Set to slow speed
262                     G_OrbAnimation.speed = G_SlowAnimationSpeed;
263                 }
264                 else if (G_OrbAnimation.speed == G_SlowAnimationSpeed)
265                 {   // Set to full speed
266                     G_OrbAnimation.speed = G_FullAnimationSpeed;
267                 }
268             }
269             //else if (Input.GetKeyDown(KeyCode.Alpha3))
270             //{   // Stop animation
271             //}
272             //else if (Input.GetKeyDown(KeyCode.Alpha4))
273             //{   // Restart animation
274             //}
275 
276 
277             G_PrintOnce = false;
278         }
279 
280 
281         /* STEAM VR FUNCTIONS ================================================*/
282         // Sent when the hand first starts hovering over the object
283         public void OnHandHoverBegin(Hand hand)
284         {
285             if (G_Debug) print("OnHandHoverBegin: " + hand + " = " + G_TargetController + "?");
286             if (hand.name == G_TargetController.name
287             || G_ConnectedControllerCount == 1)
288             {
289                 G_HoldingCollission = true;
290             }
291             else if (hand.name == G_OtherOrbController.G_TargetController.name)
292             {
293                 G_HoldingOpposite = true;
294             }
295         }
296 
297         // Sent when the hand stops hovering over the object
298         public void OnHandHoverEnd(Hand hand)
```

```
299     {
300         if (G_Debug) print("OnHandHoverEnd: " + hand + " = " + G_TargetController + "?");
301         if (hand.name == G_TargetController.name
302         || G_ConnectedControllerCount == 1)
303         {
304             G_HoldingCollission = false;
305         }
306         else if (hand.name == G_OtherOrbController.G_TargetController.name)
307         {
308             G_HoldingOpposite = false;
309         }
310     }
311
312
313     /* CUSTOM FUNCTIONS ========================================================*/
314     private void __WaitForSeconds(float duration)
315     {
316         if (G_Debug) print("private void __WaitForSeconds(duration = " + duration + ") start time = " +
    Time.time + " Function Start");
317
318         float resumeTime = Time.time + duration;
319         while (resumeTime > Time.time)
320         {
321             print("private void __WaitForSeconds(): busy waiting: " + resumeTime + " > " + Time.time);
322         }
323
324         if (G_Debug) print("private void __WaitForSeconds() end time = " + Time.time + " Function
    Complete");
325     }
326 }/*-- /public class OrbController : MonoBehaviour --*/
```

Listing 9: Osc Test Sender

```
#!/usr/bin/env python3
# this is quite longer than it needs to be, but I wanted it to be easy to reconfigure
# modifed from https://pypi.python.org/pypi/python-osc
# on a new computer, first run from a command prompt:
#
#      pip3 install python-osc
#

### Edit the following variables as needed to send OSC messages

# where to send the messages, default is 127.0.0.1 (localhost)
# just make sure the IP and port match the values set in Unity, OSCaddress shouldn't matter for now
ip          = "127.0.0.1"
port        = 9090
OSCaddress = "/filter"

# Do you want to increase linearly, or send random numbers?
# Linear: set to 'False', Random: set to 'True'
randomValues = True #False

# highest value to send
highBound = 10

# time between messages (seconds)
updateInterval = 5

### Begin actual code
import random
import time
from pythonosc import osc_message_builder, udp_client

print("Opening connection to {}:{}".format(ip, port))
client = udp_client.SimpleUDPClient(ip, port)

newValue = highBound # actually starts at 0 since we add 1 later

while (True):
    if (randomValues):
        newValue = random.randint(0, highBound)
    else:
        newValue = (newValue + 1) % (highBound + 1)

    print ("Sending", newValue)
    client.send_message(OSCaddress, newValue)
    time.sleep(updateInterval)
```

Listing 10: Post Meditation Survey Controls

```
 using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PostMeditationSurveyControls : MonoBehaviour {
    public GameObject G_GameController = null;
    public GameObject G_PostMeditationSurvey = null;

    private int G_Question1Value = 0;
    private GameObject G_Question1ValueDisplay = null;
    private Slider G_Question1Slider = null;
    private Slider G_PreMeditationResponse1Slider = null;
    private GameObject G_PreMeditationResponse1ValueDisplay = null;
    private int G_PreMeditationResponse1Value = 0;

    private int G_Question2Value = 0;
    private GameObject G_Question2ValueDisplay = null;
    private Slider G_Question2Slider = null;
    private Slider G_PreMeditationResponse2Slider = null;
    private GameObject G_PreMeditationResponse2ValueDisplay = null;
    private int G_PreMeditationResponse2Value = 0;

    private InputField G_Question3InputField = null;
    private InputField G_PreMeditationResponse3InputField = null;
    private string G_PreMeditationResponse3Value = "";


    // Use this for initialization
    void Start ()
    {
        // Do we need to run?
        if (!G_PostMeditationSurvey.activeInHierarchy) { return; }

        // Capture the game controller if necessary
        if (G_GameController == null)
        {
            G_GameController = GameObject.Find("Component Holder");
        }

        // Get Objects for Question & Response 1
        G_Question1ValueDisplay = G_PostMeditationSurvey.transform.Find("Question 1").Find("Current Value").gameObject;
        G_Question1Slider = G_PostMeditationSurvey.transform.Find("Question 1").Find("Slider").gameObject.GetComponent<Slider>();
        G_PreMeditationResponse1Slider = G_PostMeditationSurvey.transform.Find("Pre-Meditation Response 1").Find("Slider").gameObject.GetComponent<Slider>();
        G_PreMeditationResponse1ValueDisplay = G_PostMeditationSurvey.transform.Find("Pre-Meditation
```

```
Response 1").Find("Current Value").gameObject;
47          G_GameController.GetComponent<GameController>().GetGameData("pre_survey_question1", ref
        G_PreMeditationResponse1Value);
48
49          // Get Objects for Question & Response 2
50          G_Question2ValueDisplay = G_PostMeditationSurvey.transform.Find("Question 2").Find("Current
        Value").gameObject;
51          G_Question2Slider = G_PostMeditationSurvey.transform.Find("Question 2").Find("Slider").
        gameObject.GetComponent<Slider>();
52          G_PreMeditationResponse2Slider = G_PostMeditationSurvey.transform.Find("Pre-Meditation Response
        2").Find("Slider").gameObject.GetComponent<Slider>();
53          G_PreMeditationResponse2ValueDisplay = G_PostMeditationSurvey.transform.Find("Pre-Meditation
        Response 2").Find("Current Value").gameObject;
54          G_GameController.GetComponent<GameController>().GetGameData("pre_survey_question2", ref
        G_PreMeditationResponse2Value);
55
56          // Get Objects for Question & Response 3
57          G_Question3InputField = G_PostMeditationSurvey.transform.Find("Question 3").Find("InputField").
        gameObject.GetComponent<InputField>();
58          G_PreMeditationResponse3InputField = G_PostMeditationSurvey.transform.Find("Pre-Meditation
        Response 3").Find("InputField").gameObject.GetComponent<InputField>();
59          G_GameController.GetComponent<GameController>().GetGameData("pre_survey_question3", ref
        G_PreMeditationResponse3Value);
60      }
61
62
63      void LoadSurveyResponses()
64      {
65          G_GameController.GetComponent<GameController>().GetGameData("pre_survey_question1", ref
        G_PreMeditationResponse1Value);
66          switch(G_PreMeditationResponse1Value)
67          {
68              case 1: // Poor
69                  G_PreMeditationResponse1ValueDisplay.GetComponent<Text>().text = "Poor";
70                  G_PreMeditationResponse1Slider.value = G_PreMeditationResponse1Value;
71                  break;
72              case 2: // Could be better
73                  G_PreMeditationResponse1ValueDisplay.GetComponent<Text>().text = "Could be better";
74                  G_PreMeditationResponse1Slider.value = G_PreMeditationResponse1Value;
75                  break;
76              case 3: // OK
77                  G_PreMeditationResponse1ValueDisplay.GetComponent<Text>().text = "OK";
78                  G_PreMeditationResponse1Slider.value = G_PreMeditationResponse1Value;
79                  break;
80              case 4: // Pretty Good
81                  G_PreMeditationResponse1ValueDisplay.GetComponent<Text>().text = "Pretty Good";
82                  G_PreMeditationResponse1Slider.value = G_PreMeditationResponse1Value;
83                  break;
84              case 5: // Great!
85                  G_PreMeditationResponse1ValueDisplay.GetComponent<Text>().text = "Great!";
```

```
86                  G_PreMeditationResponse1Slider.value = G_PreMeditationResponse1Value;
87                  break;
88              default:
89                  break;
90          }
91
92          G_GameController.GetComponent<GameController>().GetGameData("pre_survey_question2", ref
        G_PreMeditationResponse2Value);
93          switch(G_PreMeditationResponse2Value)
94          {
95              case 1: // Much Worse!
96                  G_PreMeditationResponse2ValueDisplay.GetComponent<Text>().text = "Much Worse!";
97                  G_PreMeditationResponse2Slider.value = G_PreMeditationResponse2Value;
98                  break;
99              case 2: // A Little Worse
100                 G_PreMeditationResponse2ValueDisplay.GetComponent<Text>().text = "A Little Worse";
101                 G_PreMeditationResponse2Slider.value = G_PreMeditationResponse2Value;
102                 break;
103             case 3: // Pretty Much the Same
104                 G_PreMeditationResponse2ValueDisplay.GetComponent<Text>().text = "Pretty Much the Same";
105                 G_PreMeditationResponse2Slider.value = G_PreMeditationResponse2Value;
106                 break;
107             case 4: // A Little Better
108                 G_PreMeditationResponse2ValueDisplay.GetComponent<Text>().text = "A Little Better";
109                 G_PreMeditationResponse2Slider.value = G_PreMeditationResponse2Value;
110                 break;
111             case 5: // Much Better!
112                 G_PreMeditationResponse2ValueDisplay.GetComponent<Text>().text = "Much Better!";
113                 G_PreMeditationResponse2Slider.value = G_PreMeditationResponse2Value;
114                 break;
115             default:
116                 break;
117         }
118
119         G_GameController.GetComponent<GameController>().GetGameData("pre_survey_question3", ref
        G_PreMeditationResponse3Value);
120         G_PreMeditationResponse3InputField.text = G_PreMeditationResponse3Value;
121     }
122
123     public void UpdateHowAreYouFeelingNow(Slider slider)
124     {
125         int value = (int)slider.value;
126         switch(value)
127         {
128             case 1: // Poor
129                 G_Question1ValueDisplay.GetComponent<Text>().text = "Much Worse!";
130                 G_Question1Value = value;
131                 break;
132             case 2: // Could be better
133                 G_Question1ValueDisplay.GetComponent<Text>().text = "A Little Worse";
```

```
134                 G_Question1Value = value;
135                 break;
136             case 3: // OK
137                 G_Question1ValueDisplay.GetComponent<Text>().text = "Pretty Much the Same";
138                 G_Question1Value = value;
139                 break;
140             case 4: // Pretty Good
141                 G_Question1ValueDisplay.GetComponent<Text>().text = "A Little Better";
142                 G_Question1Value = value;
143                 break;
144             case 5: // Great!
145                 G_Question1ValueDisplay.GetComponent<Text>().text = "Much Better!";
146                 G_Question1Value = value;
147                 break;
148             default:
149                 break;
150         }
151     }
152 
153     public void UpdateHowStressedDoYouFeel(Slider slider)
154     {
155         int value = (int)slider.value;
156         switch(value)
157         {
158             case 0: // Not At All
159                 G_Question2ValueDisplay.GetComponent<Text>().text = "Not At All";
160                 G_Question2Value = value;
161                 break;
162             case 1: // Normal Stuff
163                 G_Question2ValueDisplay.GetComponent<Text>().text = "Normal Stuff";
164                 G_Question2Value = value;
165                 break;
166             case 2: // Kinda Stressed
167                 G_Question2ValueDisplay.GetComponent<Text>().text = "Kinda Stressed";
168                 G_Question2Value = value;
169                 break;
170             case 3: // Feeling Pressured
171                 G_Question2ValueDisplay.GetComponent<Text>().text = "Feeling Pressured";
172                 G_Question2Value = value;
173                 break;
174             case 4: // Crumbling
175                 G_Question2ValueDisplay.GetComponent<Text>().text = "Crumbling";
176                 G_Question2Value = value;
177                 break;
178             case 5: // Extremely Stressed
179                 G_Question2ValueDisplay.GetComponent<Text>().text = "Extremely Stressed";
180                 G_Question2Value = value;
181                 break;
182             default:
183                 break;
```

```
184             }
185         }
186
187         public void Skip() { }
188
189         public void Reset()
190         {
191             G_Question1Slider.value = 1;
192             G_Question1ValueDisplay.GetComponent<Text>().text = "Poor";
193
194             G_Question2Slider.value = 0;
195             G_Question2ValueDisplay.GetComponent<Text>().text = "Not At All";
196
197             G_Question3InputField.text = "";
198         }
199
200         public void Cancel() { }
201
202         public void Submit()
203         {
204             G_GameController.GetComponent<GameController>().SetGameData("post_survey_timestamp", System.
        DateTime.Now.ToString("g"));
205             G_GameController.GetComponent<GameController>().SetGameData("post_survey_question1",
        G_Question1Value);
206             G_GameController.GetComponent<GameController>().SetGameData("post_survey_question2",
        G_Question2Value);
207             G_GameController.GetComponent<GameController>().SetGameData("post_survey_question3",
        G_Question3InputField.text);
208             G_GameController.GetComponent<GameController>().SaveGameData();
209
210             //G_PostMeditationSurvey.SetActive(false);
211             SceneManager.LoadSceneAsync(2); // Load Meditation History
212         }
213 }
```

Listing 11: Preference Loader

```
1   using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PreferenceLoader : MonoBehaviour
6  {
7      /* VARIABLE DECLARATIONS ============================================*/
8      // Public Variables
9      // Private Variables
10     /*====  /VARIABLE DECLARATIONS ====*/
11
12
13     /* UNITY SPECIFIC FUNCTIONS ========================================*/
14     // Use this for initialization
15     void Start()
16     {
17
18     }
19
20     // Update is called once per frame
21     void Update()
22     {
23
24     }
25     /*====  /UNITY SPECIFIC FUNCTIONS ====*/
26
27
28     /* MY FUNCTIONS ====================================================*/
29     /*-----------------------------------------------------------------------
30      * Description       : asdf
31      */
32     public void ExampleFunction()
33     {
34     } /*-- /ExampleFunction() declaration --*/
35
36
37     /*-----------------------------------------------------------------------
38      * Description       : Exits the application.
39      */
40     public void ExitApplication()
41     {
42 #if UNITY_EDITOR
43         UnityEditor.EditorApplication.isPlaying = false;
44 #else
45         Application.Quit();
46 #endif
47     } /*-- /ExitApplication() declaration --*/
48     /*====  /MY FUNCTIONS ====*/
49 } /*====  /public class PreferenceLoader : MonoBehaviour ====*/
```

Listing 12: Pre Meditation Survey Controls

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PreMeditationSurveyControls : MonoBehaviour {
    public GameObject G_GameController = null;
    public GameObject G_PreMeditationSurvey = null;

    private int G_Question1Value = 0;
    private GameObject G_Question1ValueDisplay = null;
    private Slider G_Question1Slider = null;

    private int G_Question2Value = 0;
    private GameObject G_Question2ValueDisplay = null;
    private Slider G_Question2Slider = null;

    private InputField G_Question3InputField = null;


    // Use this for initialization
    void Start ()
    {
        // Do we need to run?
        if(!G_PreMeditationSurvey.activeInHierarchy) { return; }

        // Capture the game controller if necessary
        if (G_GameController == null)
        {
            G_GameController = GameObject.Find("Component Holder");
        }

        G_Question1ValueDisplay = G_PreMeditationSurvey.transform.Find("Question 1").Find("Current Value
    ").gameObject;
        G_Question1Slider = G_PreMeditationSurvey.transform.Find("Question 1").Find("Slider").gameObject
    .GetComponent<Slider>();

        G_Question2ValueDisplay = G_PreMeditationSurvey.transform.Find("Question 2").Find("Current Value
    ").gameObject;
        G_Question2Slider = G_PreMeditationSurvey.transform.Find("Question 2").Find("Slider").gameObject
    .GetComponent<Slider>();

        G_Question3InputField = G_PreMeditationSurvey.transform.Find("Question 3").Find("InputField").
    gameObject.GetComponent<InputField>();
    }

    public void UpdateHowAreYouFeelingToday(Slider slider)
    {
```

```
45          int value = (int)slider.value;
46          switch(value)
47          {
48              case 1: // Poor
49                  G_Question1ValueDisplay.GetComponent<Text>().text = "Poor";
50                  G_Question1Value = value;
51                  break;
52              case 2: // Could be better
53                  G_Question1ValueDisplay.GetComponent<Text>().text = "Could be better";
54                  G_Question1Value = value;
55                  break;
56              case 3: // OK
57                  G_Question1ValueDisplay.GetComponent<Text>().text = "OK";
58                  G_Question1Value = value;
59                  break;
60              case 4: // Pretty Good
61                  G_Question1ValueDisplay.GetComponent<Text>().text = "Pretty Good";
62                  G_Question1Value = value;
63                  break;
64              case 5: // Great!
65                  G_Question1ValueDisplay.GetComponent<Text>().text = "Great!";
66                  G_Question1Value = value;
67                  break;
68              default:
69                  break;
70          }
71      }
72
73      public void UpdateHowStressedDoYouFeel(Slider slider)
74      {
75          int value = (int)slider.value;
76          switch(value)
77          {
78              case 0: // Not At All
79                  G_Question2ValueDisplay.GetComponent<Text>().text = "Not At All";
80                  G_Question2Value = value;
81                  break;
82              case 1: // Normal Stuff
83                  G_Question2ValueDisplay.GetComponent<Text>().text = "Normal Stuff";
84                  G_Question2Value = value;
85                  break;
86              case 2: // Kinda Stressed
87                  G_Question2ValueDisplay.GetComponent<Text>().text = "Kinda Stressed";
88                  G_Question2Value = value;
89                  break;
90              case 3: // Feeling Pressured
91                  G_Question2ValueDisplay.GetComponent<Text>().text = "Feeling Pressured";
92                  G_Question2Value = value;
93                  break;
94              case 4: // Crumbling
```

```
95                  G_Question2ValueDisplay.GetComponent<Text>().text = "Crumbling";
96                  G_Question2Value = value;
97                  break;
98              case 5: // Extremely Stressed
99                  G_Question2ValueDisplay.GetComponent<Text>().text = "Extremely Stressed";
100                 G_Question2Value = value;
101                 break;
102             default:
103                 break;
104         }
105     }
106
107     public void Skip() { }
108
109     public void Reset()
110     {
111         G_Question1Slider.value = 1;
112         G_Question1ValueDisplay.GetComponent<Text>().text = "Poor";
113
114         G_Question2Slider.value = 0;
115         G_Question2ValueDisplay.GetComponent<Text>().text = "Not At All";
116
117         G_Question3InputField.text = "";
118     }
119
120     public void Cancel() { }
121
122     public void Submit()
123     {
124         G_GameController.GetComponent<GameController>().SetGameData("pre_survey_timestamp", System.
        DateTime.Now.ToString("g"));
125         G_GameController.GetComponent<GameController>().SetGameData("pre_survey_question1",
        G_Question1Value);
126         G_GameController.GetComponent<GameController>().SetGameData("pre_survey_question2",
        G_Question2Value);
127         G_GameController.GetComponent<GameController>().SetGameData("pre_survey_question3",
        G_Question3InputField.text);
128         G_GameController.GetComponent<GameController>().SaveGameData();
129
130         //G_PreMeditationSurvey.SetActive(false);
131         SceneManager.LoadSceneAsync(1); // Load Post Meditation Survey
132     }
133 }
```

Listing 13: Scene Fader

```
/*
 * Description  : Handles scene transition.
 * Author       : Asbj rn Thirslund
 * https://unity3d.com/learn/tutorials/topics/graphics/fading-between-scenes
 *
 * Fading out a level
 * float fadeTime = GameObject.FindWithTag("SceneComponents").GetComponentInChildren<SceneFader>().
    BeginFade(1);
 * yield return new WaitForSeconds(fadeTime);
 * Application.LoadLevel(<nextLevel>);
 */
using UnityEngine;
using UnityEngine.VR;
using UnityEngine.UI;
using System.Collections;

public class SceneFader : MonoBehaviour {
    // Public variables
    public bool         G_Debug = false;        // Print debug statements, this WILL reduce FPS because
    statements are printed during the OnGUI event (every frame).
    public Texture2D    G_FadeOutTexture;       // The image texture that will overlay the screen
    //public float        G_FadeSpeed = 0.8f;     // The transition speed, Commented out for now to
    differentiate the VR and FPS
    public float        G_FPSFadeSpeed = 0.08f; // The transition speed for the FPSController
    public float        G_VRFadeSpeed = 100f;   // The transition speed for the FPSController
    public Material     G_BlackBox;

    // Private Variables
    private int         G_DrawDepth = -1000;    // The texture's order in the draw hierarchy: lower
    numbers render on top
    private float       G_Alpha = 1.0f;         // The texture's alpha value between 0 and 1
    private float       G_FadeDir = -1;         // The direction to fade: in = -1, out = 1
    private bool        G_FadeComplete = false; // Has our transition worked?

    private void Start()
    {
        G_BlackBox.color = Color.black;
        G_FadeComplete = false;
    }

    private void Update()
    {
        if (!G_FadeComplete) { PerformFade(); }
    }

    private void OnLevelWasLoaded()
    {
        G_Alpha = 1;        // Just to make sure its defaulted.
        BeginFade(-1);
```

```
46      }
47
48      private void OnApplicationQuit()
49      {
50          G_BlackBox.color = Color.black;
51          G_FadeComplete = false;
52      }
53
54
55      /*-------------------------------------------------------------------------
56       * Description  : Sets G_FadeDir to the parameter passed and returns
57       *              : G_FadeSpeed for use in later calculations.
58       * Return       : G_FadeSpeed
59       */
60      public float BeginFade(int direction)
61      {
62          G_FadeComplete = false;
63          G_FadeDir = direction;
64          if (SteamVR.active)
65          {
66              return (G_VRFadeSpeed);
67          }
68          else
69          {
70              return (G_FPSFadeSpeed);
71          }
72      } /*-- /OnGUI() declaration ------------------------------------------------*/
73
74
75      /*-------------------------------------------------------------------------
76       * Description  : Perfoms a screen fade in/out. (Handles fading for the HMD)
77       *              : OnGUI Handles the fading for FPSController.
78       */
79      private void PerformFade()
80      {
81          // Perform Fade
82          //if (SteamVR.active)
83          {
84              // Calculate the fade effect
85              G_Alpha += G_FadeDir * G_VRFadeSpeed * Time.deltaTime;
86              G_Alpha = Mathf.Clamp01(G_Alpha);
87
88              Color endColor = new Color(G_BlackBox.color.r, G_BlackBox.color.g, G_BlackBox.color.b,
    G_Alpha);
89              G_BlackBox.color = Color.Lerp(G_BlackBox.color, endColor, (Time.time * G_VRFadeSpeed));
90              if(G_Debug) { print("New G_BlackBox.color: " + G_BlackBox.color); }
91          }
92      } /*-- /PerformFade() declaration ------------------------------------------*/
93
94
```

```
95      /*————————————————————————————————————————————————————————————————————
96       * Description  : Fades out/in the alpha value using a direction, speed, and
97       *              : Time.deltatime (to convert the operation to sections)
98       */
99      private void OnGUI()
100     {
101         // Perform Fade
102         if (!SteamVR.active)
103         {
104             // Calculate the fade effect
105             G_Alpha += G_FadeDir * G_FPSFadeSpeed * Time.deltaTime;
106             G_Alpha = Mathf.Clamp01(G_Alpha);
107
108             // Set GUI's alpha value
109             GUI.color = new Color(GUI.color.r, GUI.color.g, GUI.color.b, G_Alpha);
110
111             // Make sure that the G_FadeTexture draws on top
112             GUI.depth = G_DrawDepth;
113
114             // Draw texture across the entire screen
115             GUI.DrawTexture(new Rect(0, 0, Screen.width, Screen.height), G_FadeOutTexture);
116         }
117     } /*—— /OnGUI() declaration ——————————————————————————————————————*/
118 }/*==== /public class SceneFader : MonoBehaviour ====*/
```

Listing 14: (Osc) Server

```
#!/usr/bin/env python3

"""Small example OSC server

This program listens to several addresses, and prints some information about
received packets.
"""
import math

from pythonosc import dispatcher, osc_server

ip = "127.0.0.1"
port = 9090

def print_volume_handler(unused_addr, args, volume):
  print("[{0}] ~ {1}".format(args[0], volume))

def print_compute_handler(unused_addr, args, volume):
  try:
    print("[{0}] ~ {1}".format(args[0], args[1](volume)))
  except ValueError: pass

dispatcher = dispatcher.Dispatcher()
dispatcher.map("/filter", print)
dispatcher.map("/volume", print_volume_handler, "Volume")
dispatcher.map("/logvolume", print_compute_handler, "Log volume", math.log)
dispatcher.map("/1/push7", print)

server = osc_server.ThreadingOSCUDPServer(
    (ip, port), dispatcher)
print("Serving on {}".format(server.server_address))
server.serve_forever()
```

Listing 15: Sky Master Interface

```
/* ============================================================================
 * Author        : MakiahMerritt:merrittm@oregonstate.edu
 * Created       : 20170411.1037
 * Last Updated  : 20170427.1103 - Working on cloud manipulation
 * Purpose       : Manipulate the SkyMaster asset package based upon user input.
 *               :
 * --------------:-------------------------------------------------------------
 * Public Vars   : G_Debug       - determines whether or not to print debugging
 *               :                 statements in console.
 *               : G_SMContainer - A reference to the game object acting as the
 *               :                 sky master container.
 *               : G_CurrentTime - Current time of day, will interface with
 *               :                 SkyMasterManager.Current_Time
 *               :
 * --------------:-------------------------------------------------------------
 * Notes         : 01) Members of interest from SkyMasterManager
 *               :   - SkyMasterManager.Volume_Weather_types
 *               :   - SkyMasterManager.currentWeather
 *               :   - SkyMasterManager.currentWeatherName
 *               :   - SkyMasterManager.Current_Time
 *               :
 * --------------:-------------------------------------------------------------
 */
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace Artngame.SKYMASTER
{
    public class SkyMasterInterface : MonoBehaviour
    {
        /* VARIABLE DECLARATIONS ============================================*/
        /*------ Public Variables ------*/
        [Header("Generic Variables")]
        public bool G_Debug = false;                        // Print Debug Statements Y/N

        [Space(10)]
        [Header("Object References")]
        public GameObject G_SMContainer = null;             // Reference to the sky master container

        [Space(10)]
        [Header("OSC Controls")]
        public int G_MinOSCValue = -10;                     // Minimum value recieved across user input
        public int G_MaxOSCValue = 10;                      // Maximum value recieved across user input
        [HideInInspector]
        public string G_CurrentOSCValue = null;             // Current value recieved from user input

        [Space(10)]
        [Header("Time of Day Manipulation")]
```

```
50          public bool G_ManipulateTOD = true;                 // Toggle Time of Day manipulation
51          public float G_TODMultiplier = 0.001f;              // We'll multiply G_CurrentTime by this
52          [Range(0, 24)]
53          public float G_TODMin = 9;                          // Minimum time of day value, morningish
        look
54          [Range(0, 24)]
55          public float G_TODMax = 24;                         // Maximum time of day value, afternoon look
56
57          [Space(10)]
58          [Header("Sun Shaft Manipulation")]
59          public bool G_ManipulateSS = false;
60          public float G_SSMin = 0.03f;
61          public float G_SSMax = 0.25f;
62          public float G_SSMultiplier = 0.001f;
63
64          [Space(10)]
65          [Header("Cloud Manipulation")]                      // Items on the Clouds Tab of SkyMaster
66          public bool G_ManipulateClouds = true;              // Toggle cloud movement over OSC
67
68          /* Variables for "Volumetric & Shader based Clouds" > "Shader based Cloud dome"
69          public float G_CloudShiftMultiplier = 0.25f;        // Speed of cloud shifts
70          public float G_CloudShiftDensityControl = 0.43f;
71          public float G_CloudLowerLayerSize = 2.45f;
72          public float G_CloudCoverageOffset = 0f;
73          public float G_CloudSpeedOffset = 1f;
74          public float G_CloudAmbience = 1.5f;
75          */
76
77          /* Variables for "Volumetric Clouds" > "Shader based Volumetric Clouds"
78          // General Interface
79          public float G_OffsetSunIntensity = -0.05f;
80          public float G_OffsetShadowDifference = -0.02f;
81          public float G_OffsetFogDensity = 0f;
82          public int G_CloudHorizon = 0;
83          public float G_CloudDensity = 0.0001f;
84          public float G_CloudCoverageClearDay = -0.22f;
85          public float G_CloudCoverageStorm = -0.06f;
86
87          // Base Controls for "Multi quad clouds"
88          public int G_MultiQuadCloudsHeight = 1005;
89          public int G_MultiQuadCloudsHorizonOffset = 863;
90          public int G_MultiQuadCloudsScale = 3;
91          public int G_MultiQuadCloudsRotation = 6;
92          */
93
94          /* Variables from the SkyMasterManager.cs */
95          // Controls for "Shader based Cloud dome"
96          public bool G_ManipulateL1CloudCoverOffset = false;
97          public float G_L1CloudCoverOffsetMultiplier = 0f;   // Don't touch this!
98          public bool G_ManipulateL1CloudDensOffset = false;
```

```
99          public float G_L1CloudDensOffsetMultiplier = -00.001f;
100         public bool G_ManipulateL1CloudSize = false;
101         public float G_L1CloudSizeMultiplier = 0.01f;
102         public float G_L1CloudSize_Min = 0f;
103         public float G_L1CloudSize_Max = 4f;
104         public bool G_ManipulateL1Ambience = false;
105         public float G_L1AmbienceMultiplier = 0f;           // Don't touch this either!
106
107
108         [Space(10)]
109         [Header("Terrain Manipulation")]                    // Items on the Terrain Tab of SkyMaster
110         public bool G_ManipulateTerrain = true;             // Toggle terrain manipulation
111         public Color32 G_TerrainTint_Wet = new Color32(255, 255, 255, 255);
112         public Color32 G_TerrainSpecular_Wet = new Color32(142, 162, 195, 255);
113         public float G_TerrainShininess_Wet = 0.25f;
114         public Color32 G_TerrainTint_Dry = new Color32(255, 255, 255, 255);
115         public Color32 G_TerrainSpecular_Dry = new Color32(0, 0, 0, 255);
116         public float G_TerrainShininess_Dry = 0f;
117         public float G_TerrainRGBMultiplier_R = 0.01f;      // We'll multiply G_NumericOSCValue by this
    to get a new RGB Color
118         public float G_TerrainRGBMultiplier_B = 0.01f;      // We'll multiply G_NumericOSCValue by this
    to get a new RGB Color
119         public float G_TerrainRGBMultiplier_G = 0.01f;      // We'll multiply G_NumericOSCValue by this
    to get a new RGB Color
120         public float G_TerrainShininessMultiplier = 0.01f;
121
122
123         /*------ Private Variables ------*/
124         // References
125         private SkyMaster G_SMScript = null;                // Reference to the sky master script
126         private SkyMasterManager G_SMManagerScript = null; // Reference to the sky master manager
    script
127         private SeasonalTerrainSKYMASTER G_SMTerrainScript = null;    // Reference to the sky master
    terrain manager
128         private Material G_TerrainMaterial = null;          // Reference to the SeasonalTerrainSKYMASTER
    .TerrainMat
129         private AtmosphericScatteringSkyMaster G_AtmoScatteringScript = null;     // Reference to the
    AtmosphericScatteringSkyMaster script
130         private Material G_CloudDome01Material = null;      // Reference the the DOME CLOUDS L1(Clone)
    in the Sky Master Objects hierarchy collection
131
132         // OSC
133         private bool G_OSCOverride = false;                 // Overrides the guard against
    G_CurrentOSCValue == G_OldOSCValue
134         private string G_OldOSCValue = null;                // Last OSC value recieved
135         private int G_NumericOSCValue = 0;                  // Conversion of G_CurrentOSCValue to a
    usable integer
136
137         // Time of Day
138         private float G_CurrentTOD = 0f;                    // Current time of day, will interface with
```

```
    SkyMasterManager.Current_Time
139         private float G_PreviousTOD = 0f;
140 
141         // Sun Shafts
142         private float G_OldSSIntensity = 0f;
143         private float G_CurrentSSIntensity = 0f;
144 
145         // Clouds
146         private int G_CloudShiftDirection = 1;
147         private float G_OldCloudDome01Density = 0f;
148         private float G_CurrentCloudDome01Density = 0f;
149         private float G_L1CloudCoverOffset = 00.00f;
150         private float G_L1CloudDensOffset = 00.47f;
151         private float G_L1CloudSize = 02.54f;
152         private float G_L1Ambience = 01.50f;
153 
154         // Terrain
155         private Color32 G_OldTerrainTintRGB;
156         private Color32 G_OldTerrainSpecularRGB;
157         private float G_OldTerrainShininess;
158         /*====  /VARIABLE DECLARATIONS ====*/
159 
160 
161         /* UNITY SPECIFIC FUNCTIONS ========================================*/
162         // Use this for initialization
163         void Start()
164         {
165             /*------ Gather references ------*/
166             G_SMScript = G_SMContainer.GetComponent<SkyMaster>();
167             G_SMManagerScript = G_SMContainer.GetComponent<SkyMasterManager>();
168             G_SMTerrainScript = G_SMScript.TerrainManager;
169             G_TerrainMaterial = G_SMTerrainScript.TerrainMat;
170             G_AtmoScatteringScript = GameObject.Find("AtmosphericScattering(Clone)").GetComponent<
    AtmosphericScatteringSkyMaster>();
171             G_CloudDome01Material = G_SMManagerScript.CloudDomeL1Mat;
172 
173             /*------ Initialize variables ------*/
174             G_CurrentOSCValue = "0";
175             G_NumericOSCValue = 0;
176             G_OldOSCValue = "-1";
177             G_CurrentTOD = G_SMManagerScript.Current_Time;
178             G_CurrentSSIntensity = G_AtmoScatteringScript.heightRayleighIntensity;
179             G_CurrentCloudDome01Density = G_CloudDome01Material.GetFloat("_CloudDensity");
180             G_L1CloudCoverOffset = G_SMManagerScript.L1CloudCoverOffset;
181             G_L1CloudDensOffset = G_SMManagerScript.L1CloudDensOffset;
182             G_L1CloudSize = G_SMManagerScript.L1CloudSize;
183             G_L1Ambience = G_SMManagerScript.L1Ambience;
184 
185             /*------ Set Defaults ------*/
186             //G_TerrainMaterial.SetColor("_SpecColor", G_TerrainSpecular_Dry);
```

```
187             //G_TerrainMaterial.SetFloat("_Shininess", 0f);
188
189             if(G_Debug)
190             {
191                 print("G_CurrentOSCValue = " + G_CurrentOSCValue
192                     + " | G_NumericOSCValue = " + G_NumericOSCValue
193                     + " | G_OldOSCValue = " + G_OldOSCValue
194                     + " | G_OSCOverride = " + G_OSCOverride
195                     + " | G_CurrentTOD = " + G_CurrentTOD
196                     + " | G_CurrentSSIntensity = " + G_CurrentSSIntensity
197                     + " | G_CurrentCloudDome01Density = " + G_CurrentCloudDome01Density
198                     + " | G_L1CloudCoverOffset = " + G_L1CloudCoverOffset
199                     + " | G_L1CloudDensOffset = " + G_L1CloudDensOffset
200                     + " | G_L1CloudSize = " + G_L1CloudSize
201                     + " | G_L1Ambience = " + G_L1Ambience);
202             }
203         } /*-- /Start() declaration --*/
204
205
206         // Update is called once per frame
207         void Update()
208         {
209             GameObject tempGamObj = null;
210
211
212             /*---- Simulate OSC input ----*/
213             if (Input.GetKey(KeyCode.Alpha1))
214             {   // Increment G_CurrentOSCValue
215                 int temp = (int)(Random.Range(G_MinOSCValue, G_MaxOSCValue));
216                 temp = ((temp > 0) ? temp : temp * -1);
217                 G_CurrentOSCValue = temp.ToString();
218                 G_OSCOverride = true;
219             }
220             else if(Input.GetKey(KeyCode.Alpha2))
221             {   // Decrement G_CurrentOSCValue
222                 int temp = (int)(Random.Range(G_MinOSCValue, G_MaxOSCValue));
223                 temp = ((temp < 0) ? temp : temp * -1);
224                 G_CurrentOSCValue = temp.ToString();
225                 G_OSCOverride = true;
226             }
227
228             /*---- Verify Script References ----*/
229             if(G_SMManagerScript == null)
230             {
231                 G_SMManagerScript = G_SMContainer.GetComponent<SkyMasterManager>();
232             }
233             if(G_AtmoScatteringScript == null)
234             {
235                 G_AtmoScatteringScript = GameObject.Find("AtmosphericScattering(Clone)").GetComponent<
    AtmosphericScatteringSkyMaster>();
```

```
236             }
237
238             /*—— Handle Interfering Game Objects ——*/
239             tempGamObj = GameObject.Find("SETA1_DAY_CLOUDS(Clone)");
240             if (tempGamObj != null) { tempGamObj.SetActive(false); }
241
242
243             /*—— Handle OSC input ——*/
244             if (G_Debug)
245             {
246                 print("G_CurrentOSCValue = " + G_CurrentOSCValue
247                     + " | G_OldOSCValue = " + G_OldOSCValue
248                     + " | G_OSCOverride = " + G_OSCOverride);
249             }
250             if(G_CurrentOSCValue != G_OldOSCValue
251             || G_OSCOverride)
252             {
253                 /*—— Handle OSC Value ——*/
254                 int oldOSCValue = int.Parse(G_OldOSCValue);
255                 G_NumericOSCValue = int.Parse(G_CurrentOSCValue);
256                 G_NumericOSCValue = Mathf.Clamp(G_NumericOSCValue, G_MinOSCValue, G_MaxOSCValue);
257                 if (G_NumericOSCValue < oldOSCValue) { G_NumericOSCValue = G_NumericOSCValue * -1; }
258                 G_OldOSCValue = G_CurrentOSCValue;
259                 G_OSCOverride = false;
260
261                 /*—— Manipulate the TOD ——*/
262                 if(G_ManipulateTOD && (G_SMManagerScript != null))
263                 {
264                     float newTOD = 0f;
265                     newTOD = G_CurrentTOD + (G_NumericOSCValue * G_TODMultiplier);
266                     newTOD = Mathf.Clamp(newTOD, G_TODMin, G_TODMax);
267
268                     G_PreviousTOD = G_CurrentTOD;
269                     G_SMManagerScript.Current_Time = Mathf.Lerp(G_SMManagerScript.Current_Time, newTOD,
    1);
270                     G_CurrentTOD = newTOD;
271
272                     if(G_Debug)
273                     {
274                         print("Time of Day Manipulation[previous = "
275                             + G_PreviousTOD + "|new = "
276                             + G_CurrentTOD + "]");
277                     }
278                 }/*— /if (G_ManipulateTOD) —*/
279
280                 /*—— Manipulate sunshafts ——*/
281                 if(G_ManipulateSS && (G_AtmoScatteringScript != null))
282                 {
283                     float newIntensity = 0f;
284                     if(G_ManipulateTOD)
```

```
285                 {   // Manipulate sun shafts, taking into account current TOD
286                     newIntensity = G_CurrentSSIntensity + (G_NumericOSCValue * G_SSMultiplier * (
    G_CurrentTOD > 18 ? (G_NumericOSCValue > 0 ? -1 : 1) : (G_NumericOSCValue < 0 ? -1 : 1)));
287                 }
288                 else
289                 {   // Simply manipulate the sun shafts
290                     newIntensity = G_CurrentSSIntensity + (G_NumericOSCValue * G_SSMultiplier);
291                 }
292                 newIntensity = Mathf.Clamp(newIntensity, G_SSMin, G_SSMax);
293 
294                 G_OldSSIntensity = G_AtmoScatteringScript.heightRayleighIntensity;
295                 G_AtmoScatteringScript.heightRayleighIntensity = Mathf.Lerp(G_AtmoScatteringScript.
    heightRayleighIntensity, newIntensity, 1);
296                 G_CurrentSSIntensity = newIntensity;
297 
298                 if(G_Debug)
299                 {
300                     print("Sun Shaft Manipulation[previous = "
301                         + G_OldSSIntensity + "|new = "
302                         + G_CurrentSSIntensity + "]");
303                 }
304             }/*-- /if (G_ManipulateSS) --*/
305 
306             /*---- Manipulate Clouds ----*/
307             /* Commented out because we're not trying to target materials anymore
308             if(G_ManipulateClouds && (G_CloudDome01Material != null))
309                 float newDensity = 0f;
310 
311                 // Set the clouds density / acts as a source of moement
312                 G_OldCloudDome01Density = G_CloudDome01Material.GetFloat("_CloudDensity");
313                 newDensity = G_CurrentCloudDome01Density + (G_NumericOSCValue *
    G_CloudShiftMultiplier * G_CloudShiftDirection);
314                 newDensity = Mathf.Clamp(newDensity, 0f, 10f);
315                 if(newDensity == 0) { G_CloudShiftDirection = 1; }
316                 else if(newDensity == 10) { G_CloudShiftDirection = -1; }
317                 G_CloudDome01Material.SetFloat("_CloudDensity", Mathf.Lerp(
    G_CurrentCloudDome01Density, newDensity, 1));
318                 G_CurrentCloudDome01Density = newDensity;
319 
320                 if(G_Debug)
321                 {
322                     print("Cloud Manipulation[previous = "
323                         + G_OldCloudDome01Density + "|new = "
324                         + G_CurrentCloudDome01Density + "]");
325                 }
326             }/*-- /if (G_ManipulateClouds) --*/
327 
328             if(G_ManipulateClouds)
329             {
330                 // Below are the min|max limits enforced by the Sky Master Manager script
```

```
331                 /* Variables for "Volumetric & Shader based Clouds" > "Shader based Cloud dome"
332                 G_CloudShiftDensityControl      : min =  00.00 | max = 10.00
333                 G_CloudLowerLayerSize           : min =  00.10 | max = 10.00
334                 G_CloudCoverageOffset           : min =  00.00 | max = 00.20
335                 G_CloudSpeedOffset              : min = -05.00 | max = 05.00
336                 G_CloudAmbience                 : min =  00.10 | max = 01.50
337                 */
338
339                 /* Variables for "Volumetric Clouds" > "Shader based Volumetric Clouds"
340                 // General Interface
341                 G_OffsetSunIntensity            : min = -02.00 | max = 02.00
342                 G_OffsetShadowDifference        : min = -02.00 | max = 02.00
343                 G_OffsetFogDensity              : min = -12.00 | max = 12.00
344                 G_CloudHorizon                  : min =  00.00 | max = 1.5
345                 G_CloudDensity                  : min =  4e-05 | max = 0.0002
346                 G_CloudCoverageClearDay         : min = -15.00 | max = 5
347                 G_CloudCoverageStorm            : min = -15.00 | max = 5
348
349                 // Base Controls for "Multi quad clouds"
350                 G_MultiQuadCloudsHeight         : min = -1500.00 | max = 1500.00
351                 G_MultiQuadCloudsHorizonOffset  : min = -1500.00 | max = 1500.00
352                 G_MultiQuadCloudsScale          : min = -15.00 | max = 15
353                 G_MultiQuadCloudsRotation       : min =  00.00 | max = 15
354
355                 /* Variables from the SkyMasterManager.cs
356                 // Controls for "Shader based Cloud dome"
357                 G_L1CloudCoverOffset            : min =  00.00 | max = 00.20
358                 G_L1CloudDensOffset             : min =  n/a   | max = n/a
359                 G_L1CloudSize                   : min =  n/a   | max = n/a
360                 G_L1Ambience                    : min =  n/a   | max = n/a
361                 */
362                 float newL1CloudCoverOffset = 0f;
363                 float newL1CloudDensOffset = 0f;
364                 float newL1CloudSize = 0f;
365                 float newL1Ambience = 0f;
366
367                 if(G_ManipulateL1CloudCoverOffset)
368                 {
369                     newL1CloudCoverOffset = G_L1CloudCoverOffset + (G_L1CloudCoverOffsetMultiplier *
    G_NumericOSCValue);
370                     G_SMManagerScript.L1CloudCoverOffset = Mathf.Lerp(G_L1CloudCoverOffset,
    newL1CloudCoverOffset, 1);
371                     G_L1CloudCoverOffset = newL1CloudCoverOffset;
372                 }
373                 if(G_ManipulateL1CloudDensOffset)
374                 {
375                     newL1CloudDensOffset = G_L1CloudDensOffset + (G_L1CloudDensOffsetMultiplier *
    G_NumericOSCValue);
376                     G_SMManagerScript.L1CloudDensOffset = Mathf.Lerp(G_L1CloudDensOffset,
    newL1CloudDensOffset, 1);
```

```
377                         G_L1CloudDensOffset = newL1CloudDensOffset;
378                     }
379                     if(G_ManipulateL1CloudSize)
380                     {
381                         newL1CloudSize = G_L1CloudSize + (G_L1CloudSizeMultiplier * G_NumericOSCValue);
382                         newL1CloudSize = Mathf.Clamp(newL1CloudSize, G_L1CloudSize_Min,
    G_L1CloudSize_Max);
383                         G_SMManagerScript.L1CloudSize = Mathf.Lerp(G_L1CloudSize, newL1CloudSize, 1);
384                         G_L1CloudSize = newL1CloudSize;
385                     }
386                     if(G_ManipulateL1Ambience)
387                     {
388                         newL1Ambience = G_L1Ambience + (G_L1AmbienceMultiplier * G_NumericOSCValue);
389                         G_SMManagerScript.L1Ambience = Mathf.Lerp(G_L1Ambience, newL1Ambience, 1);
390                         G_L1Ambience = newL1Ambience;
391                     }
392
393                     if(G_Debug)
394                     {
395                         print("Cloud_Manipulation"
396                         + (G_ManipulateL1CloudCoverOffset == true ? "G_SMManagerScript.L1CloudDensOffset
    [previous=" + G_L1CloudDensOffset + "|new=" + newL1CloudDensOffset + "]" : "")
397                         + (G_ManipulateL1CloudDensOffset == true ? "G_SMManagerScript.L1CloudDensOffset[
    previous=" + G_L1CloudDensOffset + "|new=" + newL1CloudDensOffset + "]" : "")
398                         + (G_ManipulateL1CloudSize == true ? "G_SMManagerScript.L1CloudSize[previous=" +
     G_L1CloudSize + "|new=" + newL1CloudSize + "]" : "")
399                         + (G_ManipulateL1Ambience == true ? "G_SMManagerScript.L1Ambience[previous=" +
    G_L1Ambience + "|new=" + newL1Ambience + "]" : "")
400                         );
401                     }
402                 }/*-- /if (G_ManipulateClouds) --*/
403
404                 /*------ Manipulate the terrain ------*/
405                 /* Commented out because manging the terrains shininess is too irksome
406                 if(G_ManipulateTerrain && (G_SMManagerScript != null))
407                 {
408                     Color new_terrainTintRGB = new Color();
409                     Color new_terrainSpecularRGB = new Color();
410                     float new_terrainShininess = 0f;
411
412                     /*-- Terrain Tint Color --*
413                     // Generate new terrain tint colors
414                     // Set terrain tint colors
415
416                     /*-- Terrain Specular Color --*
417                     // Generate new terrain specular colors
418                     G_OldTerrainSpecularRGB = G_TerrainMaterial.GetColor("_SpecColor");
419
420                     new_terrainSpecularRGB.r = G_OldTerrainTintRGB.r + (G_NumericOSCValue *
    G_TerrainRGBMultiplier_R);
```

```
421                 new_terrainSpecularRGB.g = G_OldTerrainTintRGB.g + (G_NumericOSCValue *
    G_TerrainRGBMultiplier_G);
422                 new_terrainSpecularRGB.b = G_OldTerrainTintRGB.b + (G_NumericOSCValue *
    G_TerrainRGBMultiplier_B);
423
424                 new_terrainSpecularRGB.r = Mathf.Clamp(new_terrainSpecularRGB.r,
    G_TerrainSpecular_Dry.r, G_TerrainSpecular_Wet.r);
425                 new_terrainSpecularRGB.g = Mathf.Clamp(new_terrainSpecularRGB.g,
    G_TerrainSpecular_Dry.g, G_TerrainSpecular_Wet.g);
426                 new_terrainSpecularRGB.b = Mathf.Clamp(new_terrainSpecularRGB.b,
    G_TerrainSpecular_Dry.b, G_TerrainSpecular_Wet.b);
427
428                 // Set terrain specular colors
429                 G_TerrainMaterial.SetColor("_SpecColor", new_terrainSpecularRGB);
430
431                 /*-- Terrain shininess --*
432                 // Generate new shininess
433                 G_OldTerrainShininess = G_TerrainMaterial.GetFloat("_Shininess");
434                 new_terrainShininess = G_OldTerrainShininess + (G_NumericOSCValue *
    G_TerrainShininessMultiplier);
435                 new_terrainShininess = Mathf.Clamp(new_terrainShininess, 0f, 0.25f);
436
437                 // Set shininess
438                 G_TerrainMaterial.SetFloat("_Shininess", Mathf.Lerp(G_OldTerrainShininess,
    new_terrainShininess, 1));
439
440
441                 if(G_Debug)
442                 {
443                     print("Manipulating the terrain: \n"
444                         + "Tint Color[previous = "
445                         + G_OldTerrainTintRGB
446                         + " |new = " + new_terrainTintRGB
447                         + "]\n"
448                         + "Specular Color[previous = "
449                         + G_OldTerrainSpecularRGB
450                         + "|new = " + new_terrainSpecularRGB
451                         + "]\n"
452                         + "Shininess[previous = "
453                         + G_OldTerrainShininess
454                         + "|new = " + new_terrainShininess
455                         + "]");
456                 }
457             }/*-- /if (G_ManipulateTerrain) --*/
458         } /*-- /if (handle new osc value) --*/
459     } /*-- /Update() declaration --*/
460     /*====  /UNITY SPECIFIC FUNCTIONS ====*/
461
462
463     /* ADDED FUNCTIONS ================================================*/
```

```
464         /*-------------------------------------------------------------------------
465          * Description       : TBA
466          * Pre-Conditions    : TBA
467          */
468         public void template()
469         {
470         } /*-- /template() declaration --*/
471         /*====  /ADDED FUNCTIONS ====*/
472     } /*-- /public class SkyMasterInterface : MonoBehaviour --*/
473 } /*-- /namespace Artngame.SKYMASTER --*/
474
475
476
477 /*
478 SkyMaster Notes:
479 - Setup isn't difficulty but is also by no means pnp
480 - SkyMaster adjusts the terrain in many different ways. Using the terrain tab
481 -- Set "Unity Terrains"
482 --- "Size": 1
483 --- "Element 0": Field (Terrain), or the scenes terrain
484 This will add the SeasonalTerrainSKYMASTER.cs script to the selected terrain.
485 The material of which becomes "SM_TerrainSNOW_Lite". Notable because if we wanted to adjust elements
        such as the terrian's "Unity terrain tint color," "Terrain specular color," or "Unity terrain
        shininess" we would adjust it on this material.
486 -- Click "Setup Unity terrain(s)", blows out a new section
487 --- Adjust as desired
488 If the ground becomes super dark after loading the scene, expand the Sky Master Manager (Script) and
        adjust the "Terrain_Spring_Col" member to something lighter.
489 - Using the Camera FX tab
490 -- Click "Add Volumetric lighting", leave default values
491 -**** Based upon the demo scene SkyMasterManager.cs seems to provide the interface for scene changes
492
493
494
495
496 Adjustments I might try:
497 - Time of day
498 -- Start AM, 9
499 -- End PM, 21
500 - A little bit of sunlight in the morning to daytime rays
501 -- Take "Volumetric Sun shafts intensity" (under Camera FX) from 0.03 to the default 0.25
502 - Drying wet ground (recal note about material:SM_TerrainSNOW_Lite)
503 -- Wet ground settings
504 --- "Unity terrain tint color"  : rgba(255, 255, 255, 255)
505 --- "Terrain specular color"    : rgba(142, 162, 195, 255)
506 --- "Unity terrain shininess"   : 0.25
507 -- Dry ground settings
508 --- "Unity terrain tint color"  : rgba(255, 255, 255, 255)
509 --- "Terrain specular color"    : rgba(0, 0, 0, 255)
510 --- "Unity terrain shininess"   : 0.00
```

```
511 */
```

Listing 16: Tree Object Manipulation

```
/*****************************************************************************
 * Developer     : MakiahMerritt:merrittm@oregonstate.edu
 * Created       : 20161027.1416
 * Purpose       : Changes the colors of a trees leaves over time.
 * Public Vars   : print_debug      - determines whether or not print
 *               :                    statements will show in the console.
 *               : animation_active - determines whether or not the animation
 *               :                    will be active.
 *               : change_interval  - the number of seconds between color
 *               :                    updates
 * Notes         : 1) Make sure all affected tree objects are tagged "Tree"
 *               : 2) This will only update materials that have "Leaves" or
 *               :    "Fronds" in their names.
 *               :
 * TODO          : [HIGH] - Figure out color smooth transitions
 *               : [LOW]  - Determine best method of gathering tree objects
 *               :              a) bulk collect at start
 *               :              b) collect during update process
*****************************************************************************/
using UnityEngine;
using System.Collections;

public class Tree_Object_Manipulation : MonoBehaviour
{
    /*==== DECLARATIONS ====*/
    /*---- Public Varaibles ----*/
    public bool print_debug = true;
    public bool animation_active = true;
    public int change_interval = 1;
    public float change_duration = 0.00125F;

    /*---- Private Varaibles ----*/
    private GameObject[] tree_objects;
    private float prev_alarm_timestamp = 0F;


    /*==== INITIALIZATIONS ====*/
    void Start()
    {
        tree_objects = GameObject.FindGameObjectsWithTag("Tree");

        // Would getting all of our trees at the start of the scene help with
        // fps during updates?
        /*---- SET TREES ARRAY ----
        for(var k = 0; k < tree_objects.Length; k++)
        {
            trees = new Tree[tree_objects.Length][];
```

```
48              trees = tree_objects[k].FindObjectsOfType<Tree>();
49              if(print_debug) print("Tree Count: " + tree_objects.Length);
50
51              /*---- SET MATERIALS ARRAY ----*
52              tree_materials = new Material[trees.Length][];
53              material_shaders = new Shader[trees.Length][];
54              for(var i = 0; i < tree_materials.Length; i++)
55              {
56                  temp = trees[i].GetComponent<Renderer>().materials.Length;
57                  tree_materials[i] = new Material[temp];
58                  tree_materials[i] = trees[i].GetComponent<Renderer>().materials;
59
60                  /*---- SET SHADERS ARRAY ----*
61                  // Every material has one shader
62                  temp = tree_materials[i].Length;
63                  material_shaders[i] = new Shader[temp];
64                  for(var j = 0; j < temp; j++)
65                  {
66                      material_shaders[i][j] = tree_materials[i][j].shader;
67                  }
68              }
69          }*/
70      }/*--//END// void Start --*/
71
72
73      /*==== PROCESS ====*/
74      // Update is called once per frame
75      void Update()
76      {
77          /*---- DECLARATIONS ----*/
78          Tree[] trees;
79          Material[] tree_materials;
80          Color new_tree_color;
81          float new_tree_color_r;
82          float new_tree_color_g;
83          float new_tree_color_b;
84          float cur_time = Time.realtimeSinceStartup;
85
86
87          /*---- PROCESS ----*/
88          if ((animation_active)
89          && ((cur_time - prev_alarm_timestamp) > change_interval))
90          {
91              // Store new timestamp
92              prev_alarm_timestamp = cur_time;
93              if (print_debug) print("Update time: " + cur_time);
94
95              // Walk through and update trees
96              for (int i = 0; i < tree_objects.Length; i++)
97              {
```

```
98              // if (print_debug) print("Tree Count: " + tree_objects.Length);
99              trees = tree_objects[i].GetComponentsInChildren<Tree>();
100             for (int j = 0; j < trees.Length; j++)
101             {
102                 tree_materials = trees[j].GetComponent<Renderer>().materials;
103                 for (int k = 0; k < tree_materials.Length; k++)
104                 {
105                     //material_shader = tree_materials[k].shader;
106                     if ((tree_materials[k].name.ToLower().IndexOf("leaves") != -1)
107                     && (tree_materials[k].color.r < 255))
108                     {
109                         if(print_debug) print("[Tree " + (i + 1) + "] Old Leaves Color: " +
    tree_materials[k].color);
110                         // Generate new colors
111                         //Static color change. Increases red pigment prodominantly.
112                         new_tree_color_r = tree_materials[k].color.r + 5.0F;
113                             if (new_tree_color_r > 2.5F) { new_tree_color_r = 2.5F; }
114                         new_tree_color_g = tree_materials[k].color.g + 0.1F;
115                             if (new_tree_color_g > 1.75F) { new_tree_color_g = 1.75F; }
116                         new_tree_color_b = tree_materials[k].color.b;
117
118
119                         // Set new color
120                         new_tree_color = new Color(new_tree_color_r, new_tree_color_g,
    new_tree_color_b);
121                         tree_materials[k].color = Color.Lerp(tree_materials[k].color, new_tree_color
    , change_duration);
122                         if(print_debug) print("[Tree " + (i + 1) + "] New Leaves Color: " +
    tree_materials[k].color);
123                     }
124                     else if ((tree_materials[k].name.ToLower().IndexOf("fronds") != -1)
125                         && (tree_materials[k].color.g < 200))
126                     {
127                         if (print_debug) print("[Tree " + (i + 1) + "] Material: " + tree_materials[
    k].name);
128                         if (print_debug) print("[Tree " + (i + 1) + "] Old Frond Color: " +
    tree_materials[k].color);
129                         // Generate new colors
130                         //Static color change. Increases red pigment prodominantly.
131                         new_tree_color_r = tree_materials[k].color.r + 5.0F;
132                             if (new_tree_color_r > 2F) { new_tree_color_r = 2F; }
133                         new_tree_color_g = tree_materials[k].color.g + 0.1F;
134                             if (new_tree_color_r > 1.25F) { new_tree_color_r = 1.25F; }
135                         new_tree_color_b = tree_materials[k].color.b;
136
137
138                         // Set new color
139                         new_tree_color = new Color(new_tree_color_r, new_tree_color_g,
    new_tree_color_b);
140                         tree_materials[k].color = Color.Lerp(tree_materials[k].color, new_tree_color
```

```
    , change_duration);
141                             if(print_debug) print("[Tree " + (i + 1) + "] New Frond Color: " +
    tree_materials[k].color);
142                         }
143                     }/*--//END// for each tree material --*/
144                 }/*--//END// for each tree object--*/
145             }/*--//END// for each tree game object --*/
146         }/*--//END// if animate --*/
147     }/*--//END// void Update --*/
148 }/*--//END// class Tree_Object_Manipulation --*/
```

Prototype Guide

Virtual Reality and Meditation

By | OSU Capstone Team:
Padraig Gillen, Makiah Merritt, and Erik Watterson (Team Lead)
For | Capstone Stakeholders:
Marissa Powers (Project Lead), Mike Premi (Stakeholder)
Published | N/A

Contents

Acronyms & Definitions 3
Project Asset List 3
Current Builds 4
RFP-v6.1 4
RFP-NFdbk-Demo-40s-v1 / 80s-v1 6
RFP-NFdbk-OSC-v1 8
Concentration v1.4.1 10
RFP Menu Build v1.4 13
Testing Player Logs v1.3 15
Testing Cloud Animations v1.1 ..... **Error! Bookmark not defined.**
Scripts 17
Game Controller (GameController.cs) 17
Meditation History Controls (MeditationHistoryControls.cs) 22
Menu System 2D (MenuSystem2D.cs) 26
Menu System VR (MenuSystemVR.cs) 34
Menu Trigger Pull (MenuTriggerPull.cs) 43
Orb Controller (OrbController.cs) 44
Sky Master Interface (SkyMasterInterface.cs) 48
Post Meditation Survey Controls (PostMeditationSurveyControls.cs) 53
Preference Loader (PreferenceLoader.cs) 59
Pre Meditation Survey Controls (PreMeditationSurveyControls.cs) 60
Scene Fader (SceneFader.cs) 65
Tree Object Manipulation (TreeObjectManipulation.cs) 69
Tree Object Manipulation OSC (TreeObjectManipulationOSC.cs) 71
OSC (Osc.cs) 73
UDP Packet IO (UDPPacketIO.cs) 74
JavaScript OSC Receiver (JS_OSCReceiver.js) 75
References 76

Figures

Figure 1: Screenshot from the RFP Prototype 4
Figure 2: Screenshot from the Concentration v1.4.1 Prototype 10

Figure 3: Screenshot from the RFP Menu Build v1.4 Prototype 13
Figure 4: Screenshot from the Testing Player Logs v1.3 Prototype 15

Tables

No table of figures entries found.

Code Listings

No table of figures entries found.

Acronyms & Definitions

Term	Definition
Billboarding	When a graphics model stays at the same angle no matter where the camera is facing.
OSC	Open Sound Control
RFP	Rapid Forest Prototype

Project Asset List

Asset	Acquired Via	Developer	Cost	Included In
Standard Assets Package	Unity Asset Store - https://www.assetstore.unity3d.com/en/#!/content/32351	Unity Technologies	Free	All
Bird Flock Bundle Asset Package	Unity Asset Store - https://www.assetstore.unity3d.com/en/#!/content/25576	Unluck Software	$29.95	RFP-v6.1
Sky Master ULTIMATE Asset Package	Unity Asset Store - https://www.assetstore.unity3d.com/en/#!/content/25357	ARTnGame (Nasos T.)	$90.00	RFP-v6.1
SteamVR Asset Package	Unity Asset Store - https://www.assetstore.unity3d.com/en/#!/content/32647	Valve Corporation	Free	All
Global Illumination Proxy Asset Package	Unity Asset Store - https://www.assetstore.unity3d.com/en/#!/content/21197	ARTnGame (Nasos T.)	Included with Sky Master ULTIMATE	RFP-v6.1
Unity-VRInputModule Asset Package	GitHub - https://github.com/wacki/Unity-VRInputModule	Wacki	Free	
Unity OSC Receiver	GitHub - https://github.com/heaversm/unity-osc-receiver	Mike Heavers	Free	RFP-OSC
Unity Animation Reorder	https://github.com/newyellow/Unity-Runtime-Animation-Recorder	GitHub user: Newyellow	Free	Conentration

Current Builds

RFP-v6.1

Figure 1: Screenshot from the RFP Prototype

Description

This version of the RFP is used for demonstrational purposes. By reworking the original RFP, we have revamped the models within the scene by applying bought assets, we have added the OSC script to this build, and we have added a weather changing effect based on an input given to the OSC script.

Highlights

- Birds Fly around scene and contain 360 audio bird chirps.
- Scene is surrounded by hills making it feel as though there is a natural barrier for the user.
- (Weather Information)
- (OSC information)

Known Bugs

- There is a billboarding issue with the American_Elm_Hero tree leaves. To rectify the issue, the leaf object needed to be edited outside of Unity, so instead, we turned the leaves that were billboarding invisible. If you look closely at the trees, you'll see ends of branches where leaves should be.
- Occasionally, birds will fly into, and through, the ground. The included bird asset is supposed to dodge obstacles, but appears to not properly detect the ground at the time of this release. This behavior can vary from a gentle gliding through the earth, all the way to diving directly at the ground as if the bird was stunned.

Assets Used

Asset Name	Usage
Standard Assets	General assets.
SteamVR Plugin	Interfacing with the HTC Vive.

Custom Scripts Used

Script Name	Usage

Modifying the Scene

To Modify…

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris ut magna accumsan neque placerat lobortis. Pellentesque ultricies, dui in efficitur venenatis, enim magna luctus ligula, et vulputate nibh augue a eros.

RFP-NFdbk-Demo-40s-v1 / 80s-v1

Description

RFP-NFdbk-Demo-40s-v1 and RFP-NFdbk-Demo-80s-v1, stands for Rapid Forrest Prototype Neural Feedback 40 second / 80 second demo. In these demos, users are presented with a grey welcome screen acting; at which they must pull a trigger to continue into the primary scene. The primary scene of the prototype is the Rapid Forrest Prototypes "Boxed In" scene; consisting of a valley like setting with tree objects surrounding the user and a bird audio track as ambient sound.

Highlights

- The color of the tree leaves and fronds change over the course of 40 or 80 seconds.
- A custom meditation track instructs users to focus on this change.
- Instruction text is applied to each controller.

Known Bugs

- At this time the prototypes appear to operate as expected.

Assets Used

Asset Name	Usage

Standard Assets	Provided Assets:
	• Environment Textures
	• Trees Prefab
	  ○ Conifer
	  ○ Broadleaf
	• FPSController Prefab

SteamVR Plugin	Interfacing with the HTC Vive.
Leaves changing take 1 – processed 49s	Recorded Meditation track spanning 49 seconds from Lindsay and Marissa.
Nightingale-song	Ambient audio of Nightingale birds.

Custom Scripts Used

Script Name	Usage
Tree_Object_Manipulation.cs	Used to manipulate tree assets, tagged with "Tree," within the scene. Attached to the "Trees" GameObject.
Tree_Object_Manipulation_80s.cs	Used to manipulate tree assets, tagged with "Tree," within the scene. Attached to the "Trees" GameObject.
Tree_Object_Manipulation_OSC.cs	Used to manipulate tree assets, tagged with "Tree," within the scene. Attached to the "Trees" GameObject.

Script Name	Usage
PlayMedAudio.cs	Used to start the meditation track applied to the "Awake" game object. Attached to the "Controller (left)" and "Controller (right)" GameObjects.
StartTreeChange.cs	Handles starting the Standard (untimed) or the timed (80 second) experience. Attached to the "Controller (left)" and "Controller (right)" GameObjects.

Modifying the Scene

To modify the meditation track played for the user…

1. In the scenes' Hierarchy panel expand the "Sounds/Dialog Tracks" GameObjects till the "Awake" GameObject can be seen.
2. Selecting the "Awake" GameObject, the current AudioSource can be seen in the Inspector panel.
    a. Changing the AudioSource's AudioClip will change the played track.

To modify the meditation track played for the user…

1. In the scenes' Hierarchy panel expand the "Sounds/Ambient Sounds" GameObjects till the "Nightingale" GameObject can be seen.
2. Selecting the "Nightingale" GameObject, the current AudioSource can be seen in the Inspector panel.
    a. Changing the AudioSource's AudioClip will change the played track.

To modify the scripted tree manipulations…

1. In the scenes' Hierarchy panel select the "Trees" GameObject.
2. In the inspector view the attached Tree_Object_Manipulation*.cs scripts can be seen.

To modify when the tree manipulations start..

1. In the scenes' Heirarchy panel, Tree manipulations are started with the StartTreeChange.cs script attached to the left and right CameraRig controllers.

To modify the instruction text labeling each controller…

1. In the scenes' Hierarchy panel, expand the "[CameraRig]" GameObject to view the "Controller (left)" and "Controller (right)" GameObjects.
2. Expand the desired controller GameObject to see an "Instructions" GameObject.
3. Attached to the "Instructions" GameObject is a TextMesh component, containing the displayed text.

RFP-NFdbk-OSC-v1

Description

This modification of the original RFP build serves to demonstrate a scene changing based on OSC input. The scene is similar to the RFP 40s / 80s builds, but lacks meditation audio, and has no automatic leaf changes.

Highlights

- Based on original RFP design.
- Used for OSC integration for Bio-feedback
- The leaves of all trees change color based on the OSC input value.
- The green leaf color is associated with a 0 input, and a vibrant orange with the maximum OSC value (default is 10). Every other input value between the two scales the leaf color appropriately.

Known Bugs + Caveats

- No known bugs at this time.
- Note: The leaf changes will happen very suddenly if two consecutive OSC values are far apart. This is intended behavior, as smoothing is expected to be done by the sender.
- Since this was intended as a development build, no bounds checking is done on the numbers. If a number is sent beyond the specified maximum, the leaf colors will approach yellow.

Assets Used

Asset Name	Usage
Standard Assets	General assets.
SteamVR Plugin	Interfacing with the HTC Vive.
Unity OSC Receiver	Receiving and processing OSC messages.

Custom Scripts Used

Script Name	Usage
UDPPacketIO.cs	Implement base UDP sending /receiving.
Osc.cs	Implement OSC message sending / receiving / processing.
JS_OSC_Receiver.js	Read OSC values and update the Trees object.
Tree_Object_Manipulation_OSC.cs	Scale the color of all leaves based on the received OSC value.

Modifying the Scene

To modify the OSC maximum value…

1. Find the Trees object in the hierarchy, and then look for the Tree_Object_Manipulation_OSC script.
2. Change the “Max_OSC_val” to whatever you wish the maximum to be. This can be updated while the scene is actively running.

To modify the remote IP or port number…

1. Find the OSC object in the hierarchy, and navigate to its member script “JS_OSC_Receiver”.
2. Update the “Remote IP” or “Port” fields to match the sender.

a. If you are on the same computer as the sender, the IP address will typically be 127.0.0.1.

Concentration v1.4.1

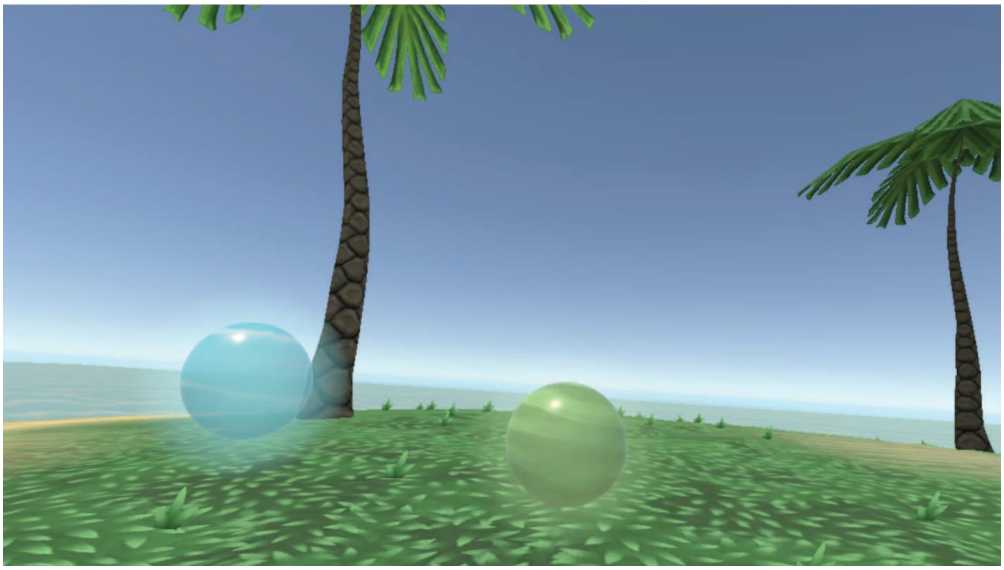


Figure 2: Screenshot from the Concentration v1.4.1 Prototype

Description

The Concentration prototype was prepared for our partner UW team. The main premise is tasking users with concentrating on following orbs in front of them. The scene's setting is that of an island with a few trees, surrounded by water.

Highlights

- This prototype presents users with a concentration task to facilitate a state of flow.
- Custom orb pattern recording using the Unity Runtime Recorder GitHub project.
- Unlike the other prototypes which use the Camera Rig prefab of SteamVR; this prototype is created using the "Player" prefab from the SteamVR asset packages Interaction System.

Known Bugs

- None at this time, the prototype appears to be working as intended.

Assets Used

Asset Name	Usage
Standard Assets	General assets.
SteamVR Plugin	Interfacing with the HTC Vive.
Cartoon PalmTree and Umbrella	Provides the assets for the environment.
Unity Runtime Recorder	Use to record the animation patterns applied to the concentration orbs.

Custom Scripts Used

Script Name	Usage
OrbController.cs	Handles application of orb + controller collisions and orb animations. Applied to Orb GameObjects.
Interactable.cs	From the SteamVR InteractionSystem. Applying this to a GameObject allows collisions to be handled properly. Applied to Orb GameObjects.

Modifying the Scene

To manipulate the orb positions (namely height)…

1. Within the Hierarchy panel find and expand the “Orb Positioner” GameObject.
    a. The two orbs, “Orb 1” and “Orb 2,” are contained within this GameObject. As such, manipulating the transform of this GameObject will change the orbs contained within.

To manipulate the colors of the orbs…

1. Within the Project asset panel find the “Materials” folder.
    a. In this folder there is an orb material and a hover highlight material. The orb material is applied to an orb within the scene. While the hover highlight material is applied to the controller model.
2. Choose the material pairs to manipulate and adjust as desired.
    a. After modifying the material colors the orbs aura must be modified as well.
3. In the Hierarchy panel expand the “Orb Positioner” GameObject and expand the modified orb.
4. Select the “Area Light” GameObject contained within and modify the attached light component color property to match the material color.

To modify the orb animation speed…

1. In the Project asset panel navigate to “Unity Runtime Recorder/Recording”
2. Open the orb animation controller. This should open the Animator panel.
3. In the Animator panel select the “hand*” state.
4. In the Inspector panel modify the Animation states speed property as desired.
5. Repeat steps 2 – 4 for the second orb animation controller.

To record a new orb animation…

1. Drag the script ‘Unity Runtime Recorder/UnityAnimSaver/UnityAnimationRecorder.cs” to the game object you wish to track, which in our case is the “Hand 1” object under “SteamVR objects”.
2. Verify the “Start Record Key” and “Stop Record Key” fields are set to your preference.
3. Set the “Save Path” and “File name” fields to appropriate values for where you want the animation to end up.
4. Start the scene, and press the “Start Record Key” when you are ready to record.
5. Press the “Stop Record Key” when you are done. The recording should be located in your desired folder.

To apply a new orb animation…

1. Drag the animation onto whatever object you wish to animate.
2. To loop the animation, navigate to the animation itself under “Assets”, open it, and check the “Loop time” box that is listed in the Inspector.

RFP Menu Build v1.4

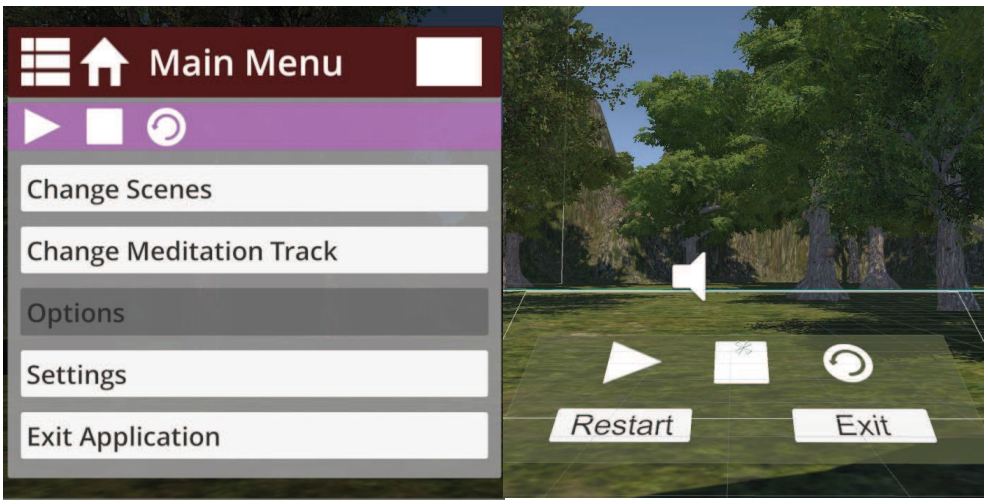


Figure 3: Screenshot from the RFP Menu Build v1.4 Prototype

Description

Extends the Rapid Forest Prototype with the implementation of a fade in (when loading the Boxed In scene) and a Menu System for 2D and VR. The main aspect of this build is the menu system, created using Unities UI Component System.

Highlights

- Implements the VRInputModule GitHub project, providing laser pointers from the HTC Vive controller models.
    - VRInputModule prefab added to the scene within the [CameraRig] GameObject.
- Features two menu systems, one for VR and one for 2D.
    - 2D Menu
        - Open and close the menu with an on-screen button
        - Change Scenes
        - Change Meditation Tracks
        - Adjust scene audio levels
        - Play/Pause, Stop, and Restart a meditation track
        - Restart the scene
        - Exit the app
    - VR Menu
        - Menu can be toggled using the HTC Vive controller application menu button
        - Play/Pause, Stop, and Restart a meditation track
        - Restart the scene
        - Exit the app
- Features a scene fading effect.

Known Bugs

- Solved, but simply attaching the ViveUILaserPointer script to the CameraRig's controllers didn't handle toggling the pointer visibilities with the controller. Once displayed they would show even if the controller was disabled.

Custom Scripts Used

Asset Name	Usage
Standard Assets	General assets.
SteamVR Plugin	Interfacing with the HTC Vive.
VRInputModule	Provides laser pointers for the HTC Vive controllers.
Simple UI Elements	Used for menu icons.

Scripts Used

Script Name	Usage
PreferenceLoader.cs	Applied to the Component Holder GameObject.
SceneFader.cs	Applied to the Component Holder GameObject.
ViveUILaserPointer.cs	Applied to the left and right [CameraRig] controller GameObjects.
PlayMedAudioLeft.cs	Applied to the left [CameraRig] controller GameObject.
MenuSystem2D.cs	Handles interactions with the 2D menu.
MenuSystemVR.cs	Handles interactions with the VR menu

Modifying the Scene

To add panels...

1. The simplest method is to copy one of the existing menu panels and modify the text and actions of the controls it contains.
    a. This requires understanding of Unity UI components and interaction with the applied Menu System script.
    b. Updating the Menu System scripts may also be necessary.

Testing Player Logs v1.3

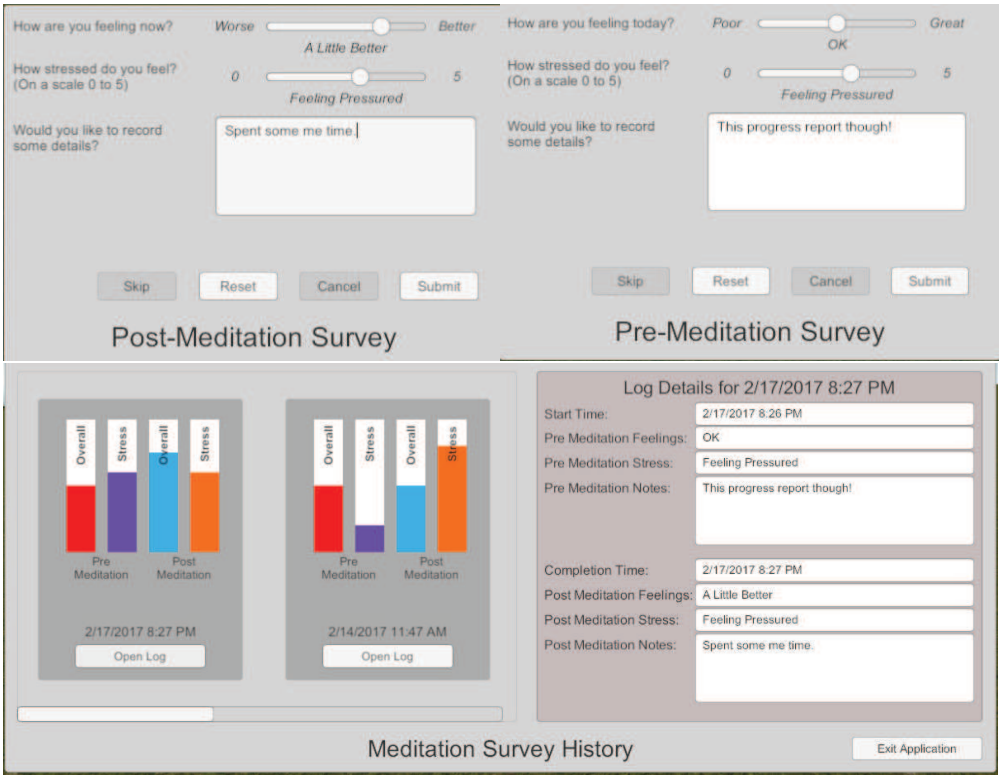


Figure 4: Screenshot from the Testing Player Logs v1.3 Prototype

Description

An experiment based upon a concept drawn up by the UW team, to record and chart player meditative experiences over time. To achieve the goal of storing data over time a script interface converts a class to a serialized object and then saves it to a file. The reason there are three identical scenes containing the same GameObject's is to test data persistence between scenes. The data being persisted is the GameData class, found in the GameController.cs script attached to the "Component Holder" GameObject.

Highlights

- Data serialization
- Saving and reading from a file
- Object persistence across scenes
- Two input forms and a history form

Known Bugs

- When reading files and creating new log history panels in the "Meditation Survey History" form, the panels aren't positioning properly within the Viewport.
    - Changing of screen size may rearrange log history panels as well.

Assets Used

Asset Name	Usage
Standard Assets	General assets.
SteamVR Plugin	Interfacing with the HTC Vive.
VRInputModule	Provides laser pointers for the HTC Vive controllers.

Custom Scripts Used

Script Name	Usage
PreferenceLoader.cs	Attached to the "Component Holder" GameObject.
GameController.cs	Attached to the "Component Holder" GameObject.
LaserPointerInputModule.cs	Unused, as the prototype is 2D
ViveUILaserPointer.cs	Unused, as the prototype is 2D
PreMeditationSurveyControls.cs	Attached to the "2D User Inventory Survey" GameObject.
PostMeditationSurveyControls.cs	Attached to the "2D User Inventory Survey" GameObject.
MeditationHistoryControls.cs	Attached to the "2D User Inventory Survey" GameObject.

Modifying the Scene

To modify form elements…

1. Modify the desired form(s)
    a. Knowledge of the Unity UI Component system is required.
2. If new elements are added or data types of elements are modified one must update the three survey control scripts to handle the interactions.
    a. The GameData class matches the elements currently shown on the Pre- and Post-Meditation Survey forms. Changing this class requires old files to be removed, else they can't be read during the serialization process.

Scripts

Game Controller (GameController.cs)

Description

This script provides the base functionality for the user survey reports. It contains the GameData class and provides interfaces to it via the GameController class. Logs created by GameController.SaveGameData() are created in the path given by G_DataPath with a filename of *gameData_yyyyMMdd* (where *yyyyMMdd*, is the date of creation).

Dependencies

- This script has no dependencies.

Usage Instructions

1. Attach this script to a GameObject within the desired scene.
2. Provide references to this script from the other three survey scripts (Pre MeditationSurveyControls.cs, PostMeditationSurveyControls.cs, and MeditationHistoryControls.cs)
3. Modification of the GameData class (which becomes serialized binary file upon form submission) requires updating any reference to the GameData class to handle the desired changes.
    a. I.e. Changing pre_survey_question1 to a string would at least require updates to each of the Set- and Get- GameData functions within the GameController class.
    b. Currently, the structure of GameData class and the interfaces provided by the GameController class has dependency by the Pre- and Post- Meditation survey scripts.
    c. Modification of the GameData class, beyond adding elements requires one to erase old "gameData*.dat" files as they can no longer be read directly into a GameData object.

Interface

Script Members

GameController Member	Details
`public bool G_Debug`	Controls whether or not to print debugging statements to the console as the script runs.
`public static GameController G_GameController`	A reference to this script within the scene. If one is not found an instance is instantiated. Once instantiated the script will persist for the duration of the apps runtime, allowing data to be kept across scenes.
`private GameData G_GameData`	A reference to the current instantiation of the GameData class, as retained by the GameController persistence.
`private int G_CurrentScene`	A reference to the currently loaded scene.
`private string G_DataPath`	Location of the generated binary files on the local machine. For us it was: C:/Users/ME!/AppData/LocalLow/DefaultCompany/project_template/*gameData_yyyyMMdd.dat*

Script Members

GameData Member	Details
`public string log_timestamp`	A time stamp formatted as System.DateTime.Now.ToString("g");
`public string pre_survey_timestamp`	A timestamp of when the user submitted the Pre-Meditation survey.
`public float pre_survey_question1`	Answer to the surveys first question.
`public float pre_survey_question2`	Answer to the surveys second question.
`public string pre_survey_question3`	Answer to the surveys third question.
`public string post_survey_timestamp`	A timestamp of when the user submitted the Post-Meditation survey.
`public float post_survey_question1`	Answer to the surveys first question.
`public float post_survey_question2`	Answer to the surveys second question.
`public string post_survey_question3`	Answer to the surveys third question.

Script Methods

GameController Method	Details
`private void Awake`	Description: Performs necessary initializations and loads any game data if present. Parameters: • None Pre-Conditions: • None Post-Conditions: • An instance of GameController is set to persist while the app is running. • An instance of GameData is initialized and a call to LoadGameData is performed to grab any logs from the current day.

Script Methods

GameController Method	Details
`public void SaveGameData`	Description: Saves the current instantiation of GameData to a serialized file. Parameters: • None Pre-Conditions: • G_GameData is instantiated. Post-Conditions: • If a file for the current day isn't found, one is created. • The file is saved to G_DataPath.
`public void LoadGameData`	Description: Loads the serialized file pointed to by G_DataPath if it exists. Parameters: • None Pre-Conditions: • None Post-Conditions: • File contents are parsed during the serialization process and read into G_GameData.

Script Methods

GameController Method	Details

`public void SetGameData`	Description:
	Sets a parameter of the GameData class.
	Function overloads are provided for @values of type int, float, and string.
	
	Parameters:
	• @member
	  ○ The member of GameData to set
	• @value
	  ○ The value of which G_GameData.@member will be set to.
	
	Pre-Conditions:
	• G_GameData is initialized.
	• @member and @value are valid.
	
	Post-Conditions:
	• G_GameData.@member is set.

`public void GetGameData`	Description:
	Sets a parameter of the GameData class.
	Function overloads are provided for @values of type int, float, and string.
	
	Parameters:
	• @member
	  ○ The member of GameData to set
	• @return_val
	  ○ A reference in which to store the value of G_GameData.@member.
	
	Pre-Conditions:
	• G_GameData is initialized.
	• @member and @value are valid.
	
	Post-Conditions:
	• G_GameData.@member is retrieved into @return_val.

Script Methods	
GameController Method	Details
`public void ExitApplication`	Description: Exits the Unity project. Parameters: • None Pre-Conditions: • None Post-Conditions: • The application instance is closed.

Meditation History Controls (MeditationHistoryControls.cs)

Description

This script provides the interface for retrieval of data files created by the Pre- and Post- Meditation Survey Control scripts.

Dependencies

- The GameController.cs script is a required reference for this script to function properly, as it provides the necessary interface to the GameData structure.

Usage Instructions

1. Attach the script to the GameObject containing the UI for the survey controls.
2. In the Inspector apply the appropriate object references.

Interface

Script Members

Member	Details
`public bool G_Debug`	Controls whether or not to print debugging statements to the console as the script runs.
`public GameObject G_GameController`	A reference to the GameController.cs script within the scene. In the Testing Player Logs 1.3 build this script is attached to the "Component Holder."
`public GameObject G_MoodSurveyHistory`	A reference to the GameObject containing the form elements: a scroll view of past logs, a panel display for log details, and an exit button.
`public GameObject G_SurveyDataTemplatePanel`	A hidden template panel containing the default display for log panels. Based upon the template created in the Testing Player Logs 1.3 build, this panel contains four graphs a datetime stamp and an open log button.
`public GameObject G_LogDetailsPanel`	The panel in which to display a logs full information.
`public int G_LogCount`	Controls the maximum number of logs to gather from G_DataPath.
`private string G_DataPath`	Location of the generated binary files on the local machine. For us it was: *C:/Users/ME!/AppData/LocalLow/DefaultCompany/project_template/gameData_yyyyMMdd.dat*

Script Methods

Method	Details
`void Start`	Description: Performs variable initialization. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_GameController is set if it wasn't already. • G_DataPath is set to Application.persistentDataPath.
`public void GetLogs`	Description: Performs a query of G_DataPath, retrieving any files matching "gameData*.dat". With this collection it then runs through G_LogCount files, making calls to LoadLog() for each. Parameters: • None Pre-Conditions: • G_DataPath is valid. Post-Conditions: • Log panels are created for each log found up to G_LogCount.

Script Methods

Method	Details

`private void LoadLog`	Description:
	An helper function for GetLogs(). This performs the task of loading data from the passed log reference.
	
	Parameters:
	• @log
	  • A reference to the file log for which a panel will be created.
	• @panelPosition
	  • Panels position within the scroll view. Currently, this is used to align based upon X coordinates for a horizontal view.
	
	Pre-Conditions:
	• @log is a valid file reference.
	• @log contains serialized data matching that of the GameData class.
	
	Post-Conditions:
	• A new log panel based upon G_SurveyDataTemplatePanel is created for @log.
	• The log position as provided via @panelPosition is updated to position the next log 300 units to the right of the panel created for @log.

Script Methods

Method	Details
`public void OpenLog`	Description: Serves as the action for the “Open Log” button in the log history scroll view of the Testing Player Logs 1.3 build. This function loads the selected log into the details pane. Parameters: • @logPanel • The log panel to open. Hidden fields of this panel provide the log details skipping the need to reach out to the disk to reopen the file. Pre-Conditions: • @logPanel exists and has information loaded from the appropriate log file. Post-Conditions: • G_LogDetailsPanel is loaded with the logs details.

Menu System 2D (MenuSystem2D.cs)

Description

Handles user interaction with a 2D menu system. The menu system is based upon the RFP Menu Builds and expects a GameObject containing each menu element within it. Panels make up the different menus within the system and each panel consists of buttons or sliders as controls with a back button to return to its parent menu. UI controls are tied to the appropriate script interfaces.

Dependencies

- Although this script has no dependencies. It expects the same scene setup as present in our RFP Menu Builds. Within these builds are all the expected panels and game objects.

Usage Instructions

1. To add controls to this script begin by adding UI elements to the scene.
2. After the UI elements are added, update/modify/create the necessary interfaces within the script.
3. After the script is ready to handle input from the scene set the new/updated UI elements to reference to appropriate script functions and pass required parameters.
    a. Note: UI element actions can only have one parameter.

Interface

Script Members

Member	Details
`public bool G_Debug`	Print Debug Statements Y/N
`public GameObject G_Player`	A reference to the *Player* GameObject. In many of our prototypes the *Player* GameObject encapsulates both the FPSController (part of Unity's Standard Asset package) and the CameraRig (part of the SteamVR asset package).
`public GameObject G_FPSController`	A reference to the FPSController GameObject.
`public GameObject G_MainMenu`	A reference to the GameObject containing the menu system.
`public AudioMixer G_MasterAudioMixer`	A reference to the Master AudioMixer.
`public EventSystem G_EventSystem`	A reference to the scenes EventSystem. EventSystems are automatically created with the first Canvas added to the scene.
`public AudioSource G_AudioTrack`	A reference to the GameObject holding the audio track to play. If unassigned when the scene is started, the script will search for an object with the "AudioTrack" tag.
`public Sprite G_PlayImage`	A reference to the "Play" sprite image. Saves the hassle of hunting through the projects assets programmatically.
`public Sprite G_PauseImage`	A reference to the "Pause" sprite image. Saves the hassle of hunting through the projects assets programmatically.
`private bool G_MenuOpen`	Current state of the menu.

Script Members

Member	Details
`private string G_CurrentMenu`	Current menu screen visible to the player.
`private GameObject G_MenuPanels`	A reference to the GameObject containing the menu's panels.
`private GameObject G_MenuTitleBar`	A reference to the menus title bar panel.
`private GameObject G_SceneFader`	A reference to the image object used to "dim" the scene when the menu is open.
`private float G_TimeScale`	Stores the scenes time scale so it can be set to 0 when the menu is opened, and restored when the menu is closed.
`private float G_MasterVolume`	Stores the scenes current master volume. Allowing for volume adjustments in transition with opening the menu (dim or silence the scenes volume) and closing the menu (restore pre-menu volume).
`private string G_AudioTrackState`	Stores the state of the audio track on menu open; allowing the script to restore it on menu close.

Script Methods

Method	Details
`private void Start`	Description: Performs variable initialization; capturing references to objects within the scene. Parameters: • None Pre-Conditions: • G_MainMenu is set to reference the scenes menu system. • G_MenuPanels will search for a GameObject called "MenuPanels." • G_MenuTitleBar will search for a GameObject called "TitleBar." • G_SceneFader will search for a GameObject called "SceneFader." Post-Conditions: • G_MenuPanels, G_MenuTitleBar, and G_SceneFader references are set using G_MainMenu.

Script Methods

Method	Details
`private void FixedUpdate`	Description: Listens for user keyboard input (KeyCodes *P* or *M*). If input is detected ToggleMainMenu() is called. Parameters: • None Pre-Conditions: • None Post-Conditions: • None
`public void ToggleMainMenu`	Description: Calls OpenMenu() or CloseMenu() based upon the state of G_MenuOpen. Parameters: • None Pre-Conditions: • G_MenuOpen corresponds with the menus current visible state. Post-Conditions: • None

Script Methods

Method	Details
`private void OpenMenu`	Description: Performs actions necessary for opening the main menu. Step 1) pauses the scenes audio, time, and player controls (releasing the cursor). Step 2) activates the appropriate menu objects. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_AudioTrackState is set to the “UnPause” or “Stopped” depending on what the G_AudioTrack will resume to when the menu is closed. • G_MasterVolume is set to the AudioListeners current volume. The AudioListener is then set to 0. • G_TimeScale is set to the scenes current time scale. The time scale is then set to 0. • G_FPSController is deactivated. • The mouse cursor is enabled. • G_MenuPanels, G_MenuTitleBar, and G_SceneFader are activated. • G_MenuOpen is set to true.

Script Methods

Method	Details
`private void CloseMenu`	Description: Handles resetting menu visibilities and restoring the scene state (audio, time, and player controls). Parameters: • None Pre-Conditions: • None Post-Conditions: • The main meu panel is set to open on the next menu open. • G_MenuPanels, G_MenuTitleBar, and G_SceneFader are deactivated. • The scenes time scale and AudioListender are restored to their previous values. • G_FPSController is reactivated. • The mouse cursor is hidden. • G_MenuOpen is set to false.
`private void ResetMenuPanels`	Description: Resets the menu panel visibilities to their default state. Parameters: • None Pre-Conditions: • None Post-Conditions: • All menu panels, except for “Main,” are deactivated.

Script Methods

Method	Details
`public void ChangeMenu`	Description: Handles navigating the menu system by showing and hiding panels. Parameters: • @selectedMenu ○ A string representing the menu to load. Pre-Conditions: • None Post-Conditions: • G_CurrentMenu is deactivated. • @selectedMenu is set to active and G_CurrentMenu now points to @selectedMenu.
`public void ChangeMeditationType`	Description: Unused function. This was originally intended to handle transitioning between various meditative experiences (body scan, relaxation, concentration, etc.) Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Script Methods	
Method	Details
`private void ResumeAudioStateAfterMenuClose`	Description: Handles restoring the state of G_AudioTrack based upon G_AudioTrackState. Parameters: • None Pre-Conditions: • G_AudioTrackState represents the state to resume. Post-Conditions: • G_AudioTrack state is restored. • G_AudioTrackState is set to “Playing” if the user is unpausing or restarting the audio track.
`public void AudioTrackControls`	Description: Handles the transition of the UI’s Play/Pause track button. Parameters: • @action ○ The state of the audio track. Pre-Conditions: • None Post-Conditions: • The image sprite of the button tagged “btn_ToggleMeditationTrack” is set to G_PlayImage or G_PauseImage.

Script Methods	
Method	Details
`public void ChangeTracks`	Description: Changes the current audio track of the Meditation Track audio source. Parameters: • @trackName ○ The name of an audio track to play. Pre-Conditions: • Audio tracks must be inside of [Assets Folder]/Resources/Audio/MeditationTracks as the Resources.Load() function uses pathing relative to the Resources folder. Post-Conditions: • The AudioTrack is set to play @trackName when the menu closes.
`public void UpdateVolume`	Description: Updates the values of a mixer group within the G_MasterAudioMixer. Parameters: • @mixerGroup ○ The mixer group who's volume will be updated. Pre-Conditions: • @mixerGroup matches the name of UI Slider control and matches the name of a group within G_MasterAudioMixer. Post-Conditions: • The mixer group is updated to the value of its corresponding Slider.

Script Methods	
Method	Details
`public void ChangeScene`	Description: Handles changing scenes. Parameters: • @sceneIndex ○ Build index of the scene to load. Pre-Conditions: • The desired scene is part of the current set of scenes build when the app is started. • The scene fader script is present and attached as a child of a GameObject called "ComponentHolder." Post-Conditions: • The desired scene is loaded.
`public void ExitApplication`	Description: Exits the application. Parameters: • None Pre-Conditions: • None Post-Conditions: • The application is closed.

Menu System VR (MenuSystemVR.cs)

Description

Handles user interactions with a simple menu system prepared for VR. Based upon the RFP Menu Builds, the expected menu system consists of one panel with play/pause, stop, and restart audio track buttons, a restart scene button, and an exit app button. Each of these UI elements are tied to a function within the script.

Dependencies

- Although this script has no dependencies. It expects the same scene setup as present in our RFP Menu Builds. Within these builds are all the expected panels and game objects.

Usage Instructions

1. To add controls to this script begin by adding UI elements to the scene.
2. After the UI elements are added, update/modify/create the necessary interfaces within the script.

3. After the script is ready to handle input from the scene set the new/updated UI elements to reference to appropriate script functions and pass required parameters.
    a. Note: UI element actions can only have one parameter.

Interface

Script Members

Member	Details
`public bool G_Debug`	Print Debug Statements Y/N
`public GameObject G_Player`	A reference to the *Player* GameObject. In many of our prototypes the *Player* GameObject encapsulates both the FPSController (part of Unity's Standard Asset package) and the CameraRig (part of the SteamVR asset package).
`public GameObject G_SteamVRCameraRig`	A reference to the CameraRig GameObject.
`public GameObject G_MainMenu`	A reference to the GameObject containing the menu system.
`public AudioMixer G_MasterAudioMixer`	A reference to the Master AudioMixer.
`public EventSystem G_EventSystem`	A reference to the scenes EventSystem. EventSystems are automatically created with the first Canvas added to the scene.
`public AudioSource G_AudioTrack`	A reference to the GameObject holding the audio track to play. If unassigned when the scene is started, the script will search for an object with the "AudioTrack" tag.
`public Sprite G_PlayImage`	A reference to the "Play" sprite image. Saves the hassle of hunting through the projects assets programmatically.
`public Sprite G_PauseImage`	A reference to the "Pause" sprite image. Saves the hassle of hunting through the projects assets programmatically.
`private bool G_MenuOpen`	Current state of the menu.
`private string G_CurrentMenu`	Current menu screen visible to the player.
`private float G_TimeScale`	Stores the scenes time scale so it can be set to 0 when the menu is opened, and restored when the menu is closed.
`private float G_MasterVolume`	Stores the scenes current master volume. Allowing for volume adjustments in transition with opening the menu (dim or silence the scenes volume) and closing the menu (restore pre-menu volume).
`private SteamVR_Controller.Device G_LeftControllerDev`	A reference to the device registered to be the left controller by SteamVR.
`private SteamVR_Controller.Device G_RightControllerDev`	A reference to the device registered to be the right controller by SteamVR.
`private string G_AudioTrackState`	Stores the state of the audio track on menu open; allowing the script to restore it on menu close.

Script Methods

Method	Details
private void Awake	Description: Does nothing. Parameters: • None Pre-Conditions: • None Post-Conditions: • None
private void Start	Description: Variable initialization. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_MenuOpen, G_AudioTrack, and G_AudioTrackState references are set. • UpdateControllerDeviceReferences() is called. • The scene begins fading in.

Script Methods	
Method	Details
private void Update	Description: Checks controller references, using UpdateControllerDeviceReferences(), and listens for the user to click the controllers "ApplicationMenu" button. If button press is detected, ToggleMainMenu() is called. Parameters: • None Pre-Conditions: • None Post-Conditions: • None
private void FixedUpdate	Description: Does nothing. Parameters: • None Pre-Conditions: • None Post-Conditions: • None
private void LateUpdate	Description: Deactivates the left and right controllers if the G_MenuOpen is false. This is a fix to hide the laser pointers displayed when the menu is open. Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Script Methods

Method	Details
private void UpdateControllerDeviceReferences	Description: Obtains and sets controller references, from SteamVR_Controller, to the left and right controllers. Note: the right controller is set first on scene load. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_LeftControllerDev and G_RightControllerDev references are set. • If only one controller is used G_LeftControllerDev is set to null.
public void ToggleMainMenu	Description: Calls OpenMenu() or CloseMenu() based upon the state of G_MenuOpen. Parameters: • None Pre-Conditions: • G_MenuOpen corresponds with the menus current visible state. Post-Conditions: • None

Script Methods

Method	Details
private void OpenMenu	Description: Opens default menu objects, captures the scenes time scale and volume then sets them to 0f, pauses user's ability to move. Parameters: • None Pre-Conditions: • Menu elements are set to their default visibilities. • Necessary references and objects are accessible. Post-Conditions: • G_AudioTrackState, G_MasterVolume, and G_TimeScale are set. • The audio track is paused, if playing. • The AudioListener's volume is set to 0. • The scenes time scale is set to 0. • Laser pointers are activated for active controllers. • G_MainMenu is activated. • G_MenuOpen is set to true.

Script Methods	
Method	Details
private void CloseMenu	Description: Resets the menu objects to their default visibilities and restores the scenes state. Parameters: • None Pre-Conditions: • None Post-Conditions: • The controllers are deactivated, thus hiding the laser pointers. • G_MainMenu is deactivated. • The scenes AudioListener volume and time scale are restored. • G_MenuOpen is set to false. • ResumeAudioStateAfterMenuClose() is called.
private void ResumeAudioStateAfterMenuClose	Description: Handles restoring the state of G_AudioTrack based upon G_AudioTrackState. Parameters: • None Pre-Conditions: • G_AudioTrackState represents the state to resume. Post-Conditions: • G_AudioTrack state is restored. • G_AudioTrackState is set to “Playing” if the user is unpausing or restarting the audio track.

Script Methods	
Method	Details
public void AudioTrackControls	Description: Handles the transition of the UI's Play/Pause track button. Parameters: • @action ○ The state of the audio track. Pre-Conditions: • None Post-Conditions: • The image sprite of the button tagged "btn_ToggleMeditationTrack" is set to G_PlayImage or G_PauseImage.
public void ChangeScene	Description: Handles changing scenes. Parameters: • @sceneIndex ○ Build index of the scene to load. Pre-Conditions: • The desired scene is part of the current set of scenes build when the app is started. • The scene fader script is present and attached as a child of a GameObject called "ComponentHolder." Post-Conditions: • The desired scene is loaded.

Script Methods	
Method	Details
public void ExitApplication	Description: Exits the application. Parameters: • None Pre-Conditions: • None Post-Conditions: • The application is closed.

Menu Trigger Pull (MenuTriggerPull.cs)

Description

Dependencies

- This script has no dependencies.

Usage Instructions

1. asdf

Interface

Script Members	
Member	Details

Script Methods	
Method	Details
	Description: TBA Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Orb Controller (OrbController.cs)

Description

This script handles the activation of orb Animations for the Concentration Prototype. Additional options provide the interface for adjusting orb animation speeds based upon orb and controller collissions.

Dependencies

- This script has no inter script dependencies. However, because this script was created for the Concentration Prototype a similar scene setup is expected because of GameObject and Animation references.

Usage Instructions

1. Apply the script to an orb within the scene.
2. Set the required references for public members.
    a. G_EasterEgg doesn't need to be set.

Interface

Script Members

Member	Details
`public bool G_Debug`	Determines whether or not to print debugging statements to the console.
`public bool G_PrintOnce`	If set, certain debugging statements will print once.
`public bool G_SlowOnHoverEnd`	If set, orb animations will slow when the user disconnects one or more controllers from designated orb's.
`public GameObject G_TargetController`	A reference to the controller GameObject, within the CameraRig, that this orb should respond to.
`public OrbController G_OtherOrbController`	A reference to the OrbController.cs script attached to the other orb GameObject.
`public Animator G_OrbAnimation`	A reference to the Animation applied to the orb.
`public GameObject G_EasterEgg`	A reference to an Easter Egg GameObject within the scene.
`public bool G_HoldingCollission`	A flag set when this orb is in contact with G_TargetController.
`public bool G_HoldingOpposite`	A flag set when this orb is in contact with G_OtherOrb
`public int G_ConnectedControllerCount`	The number of controllers detected by SteamVR.
`private float G_FullAnimationSpeed`	The speed of the animation when full. Used to restore the default speed to a slowed orb Animation.
`private float G_SlowAnimationSpeed`	The speed of the animation when slowed. Used to slow an orb Animation from full speed.
`private SteamVR_Controller.Device G_TargetControllerDev`	A reference to the controller device, from SteamVR, that this orb will be targeting.
`private SteamVR_Controller.Device G_OtherControllerDev`	A reference to the controller device, from SteamVR, that the other orb will be targeting.

Script Methods

Method	Details
`void Start`	Description: Variable initialization. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_FullAnimationSpeed and G_OrbAnimation speed are initialized.
`void Update`	Description: Checks for updated controller device references, adjusts G_ConnectedControllerCount – deactivating the second orb controller if only one controller device is detected, and then checks for orb collisions. The collision checks performed will start the orb animation, display the Easter Egg, or adjust the speed of orb animations if G_SlowOnHoverEnd is set. Manual overrides are prepared for use in 2D setups. KeyCode.Alpha1 begins the animation and KeyCode.Alpha2 toggles G_SlowOnHoverEnd. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_TargetControllerDev, G_OtherControllerDev, and G_ConnectedControllerCount references are updated. • The GameObject to which G_OtherOrbController is attached is deactivated if the orb is “Orb 2” and G_ConnectedControllerCount is 1. • If G_SlowOnHoverEnd, orb Animation speed is set to G_FullAnimationSpeed if

Script Methods	
Method	Details
	designated collisions are detected and G_SlowAnimationSpeed if not.
`public void OnHandHoverBegin`	Description: Detects when the orb is begins colliding with a Vive controller. Parameters: • @hand ○ The hand colliding with the orb. Pre-Conditions: • None Post-Conditions: • If the target controller is colliding with the orb, G_HoldingCollission is set to true. • If the opposite controller is colliding with the orb, G_HoldingOpposite is set to true.
`public void OnHandHoverEnd`	Description: Sent when the orb is no longer colliding with a Vive controller. Parameters: • @hand ○ The hand colliding with the orb. Pre-Conditions: • None Post-Conditions: • If the target controller was colliding with the orb, G_HoldingCollission is set to false. • If the opposite controller was colliding with the orb, G_HoldingOpposite is set to false.
`private void __WaitForSeconds`	Description: A busy wait function intended to inhibit processing for a given amount of time. Unfortunately, it doesn't work as expected and therefore isn't used.

Script Methods	
Method	Details
	Parameters: • @duration ○ Duration to wait. Pre-Conditions: • None Post-Conditions: • The function returns after @duration has passed.

Sky Master Interface (SkyMasterInterface.cs)

Description

This script acts as an interface to the Sky Master ULTIMATE asset package. Specifically, users are able to manipulate a scenes Time of Day, Atmospheric Scattering (Sun Shafts), Clouds, and Terrain; all of which are handled by the Sky Master pack through its internal scripts.

Dependencies

- This script requires the Sky Master ULTIMATE Unity asset package, as it is an interface to various environment properties. Specifically, it looks for references to:
  - SkyMaster.cs
  - SkyMasterManager.cs
  - AtmosphericScatteringSkyMaster.cs
  - Various materials
- Also this script encapsulates the SkyMasterInterface class within the Artngame.SKYMASTER namespace to access sibling components more easily.

Usage Instructions

1. Apply this script to a component holder after the Sky Master asset package has been added and setup within the desired scene.
2. Select this object
3. In the Inspector panel
   a. Apply required references
   b. Toggle the desired manipulations
   c. Manipulate exposed values as desired
      i. Note setting a “multiplier” value to 0 effectively cancels the manipulations effect
4. After playing the scene pressing Alpha1 will act as an increase in OSC value and Alpha2 a decrease.

Interface

Script Members

Member	Details
`public bool G_Debug`	A toggle to display debug statements in the console.
`public GameObject G_SMContainer`	A reference to the GameObject holding the Sky Master components.
`public int G_MinOSCValue`	The minimum value the script will handle via OSC.
`public int G_MaxOSCValue`	The maximum value the script will handle via OSC.
`public string G_CurrentOSCValue`	The received OSC value. Exposed for interfacing with the OSC plugin scripts but hidden in the Inspector panel.

Script Members

Member	Details
`public bool G_ManipulateTOD`	Toggle whether or not to manipulate the scenes time of day with OSC. Enabling this will increase or decrease the scenes TOD. If G_ManipulateSS is enabled as well this will decrease sun shaft intensity as the TOD increases past noon, and increase as the TOD decreases at evening.
`public float G_TODMultiplier`	A multiplier to apply when manipulating the scenes TOD.
`public float G_TODMin`	The minimum TOD allowed. This is defaulted to an early afternoon.
`public float G_TODMax`	The maximum TOD allowed. This is defaulted to an early evening.
`public bool G_ManipulateSS`	Toggle whether or not to manipulate the scenes sun shafts with OSC. Enabling this will increase or decrease the intensity of sun shafts sun shaft.
`public float G_SSMin`	The minimum value to reduce sun shaft intensity to.
`public float G_SSMax`	The maximum value to increase sun shaft intensity to. It is recommended to keep this value fairly low as the shaft intensities get very strong.
`public float G_SSMultiplier`	A multiplier to apply when manipulating the scenes sun shafts.
`public bool G_ManipulateClouds`	Toggle whether or not to manipulate the scenes clouds with OSC.
`public bool G_ManipulateL1CloudCoverOffset`	Toggle whether or not to manipulate the scenes cloud cover offset with OSC. Not recommended.
`public float G_L1CloudCoverOffsetMultiplier`	A multiplier to apply when manipulating the scenes cloud cover offset.
`public bool G_ManipulateL1CloudDensOffset`	Toggle whether or not to manipulate the scenes cloud density offset with OSC.
`public float G_L1CloudDensOffsetMultiplier`	A multiplier to apply when manipulating the scenes cloud density offset.
`public bool G_ManipulateL1CloudSize`	Toggle whether or not to manipulate the scenes cloud size with OSC.
`public float G_L1CloudSizeMultiplier`	A multiplier to apply when manipulating the scenes cloud size.
`public float G_L1CloudSize_Min`	The minimum cloud size allowed. Although there is no limit applied in the Sky Master scripts, the effect cycles when increasing and decreasing size. This prohibits that cycling.

Script Members

Member	Details
`public float G_L1CloudSize_Max`	The maximum cloud size allowed. Although there is no limit applied in the Sky Master scripts, the effect cycles when increasing and decreasing size. This prohibits that cycling.
`public bool G_ManipulateL1Ambience`	Toggle whether or not to manipulate the scenes cloud ambience with OSC. Not recommended.
`public float G_L1AmbienceMultiplier`	A multiplier to apply when manipulating the scenes cloud ambience.
`public bool G_ManipulateTerrain`	Toggle whether or not to manipulate the scenes terrain with OSC. Not recommended. The goal of which is to transition the terrain between a wet and dry look.
`public Color32 G_TerrainTint_Wet`	The terrain color tint to transition to for a “wet” look.
`public Color32 G_TerrainSpecular_Wet`	The terrain color specular value (color) to transition to for a “wet” look.
`public float G_TerrainShininess_Wet`	The terrain shininess to transition to for a “wet” look.
`public Color32 G_TerrainTint_Dry`	The terrain color tint to transition to for a “dry” look.
`public Color32 G_TerrainSpecular_Dry`	The terrain color specular value (color) to transition to for a “dry” look.
`public float G_TerrainShininess_Dry`	The terrain shininess to transition to for a “dry” look.
`public float G_TerrainRGBMultiplier_R`	A multiplier to apply when manipulating a terrain color RGB.r value.
`public float G_TerrainRGBMultiplier_B`	A multiplier to apply when manipulating a terrain color RGB.b value.
`public float G_TerrainRGBMultiplier_G`	A multiplier to apply when manipulating a terrain color RGB.g value.
`public float G_TerrainShininessMultiplier`	A multiplier to apply when manipulating the scenes terrain towards a wetter look.
`private SkyMaster G_SMScript`	A reference to the SkyMaster.cs script as found via G_SMContainer. This acts as a primary interface to the Sky Master components.
`private SkyMasterManager G_SMManagerScript`	A reference to the SkyMasterManager.cs script as found via G_SMContainer. This acts as a primary interface to the Sky Master components.
`private SeasonalTerrainSKYMASTER G_SMTerrainScript`	A reference to the SeasonalTerrainSKYMASTER.cs script as found via G_SMScript.
`private Material G_TerrainMaterial`	A reference to a material applied to the scenes terrain by the SkyMaster.cs script.

Script Members

Member	Details
`private AtmosphericScatteringSkyMaster G_AtmoScatteringScript`	A reference to the AtmosphericScatteringSkyMaster.cs script found via the "AtmosphericScattering(Clone)" GameObject. This provides the interface for manipulating sun shafts.
`private Material G_CloudDome01Material`	A reference to the material used to display clouds. Not used anymore.
`private bool G_OSCOverride`	Overrides a guard in Update() against G_CurrentOSCValue == G_OldOSCValue.
`private string G_OldOSCValue`	The last OSC value processed.
`private int G_NumericOSCValue`	The numeric representation of G_CurrentOSCValue.
`private float G_CurrentTOD`	The current time of day.
`private float G_PreviousTOD`	The previous time of day.
`private float G_OldSSIntensity`	The previous sun shaft intensity.
`private float G_CurrentSSIntensity`	The current sun shaft intensity.
`private int G_CloudShiftDirection`	The direction to shift clouds. Not used anymore.
`private float G_OldCloudDome01Density`	The old density of G_ CloudDome01Material. Not used anymore.
`private float G_CurrentCloudDome01Density`	The current density of G_ CloudDome01Material. Not used anymore.
`private float G_L1CloudCoverOffset`	The current cloud cover offset.
`private float G_L1CloudDensOffset`	The current cloud density offset.
`private float G_L1CloudSize`	The current cloud size.
`private float G_L1Ambience`	The current cloud ambience.
`private Color32 G_OldTerrainTintRGB`	The previous terrain tint.
`private Color32 G_OldTerrainSpecularRGB`	The previous terrain specular (color).
`private float G_OldTerrainShininess`	The previous terrain shininess.

Script Methods

Method	Details
`void Start()`	Description: Handles reference capturing and variable initialization. Parameters: • None Pre-Conditions: • G_SMContainer is set. • The Sky Master components are setup within the scene. Post-Conditions: • G_SMScript, G_SMManagerScript, G_SMTerrainScript, G_TerrainMaterial, G_AtmoScatteringScript, G_CloudDome01Material, G_CurrentTOD, G_CurrentSSIntensity, G_CurrentCloudDome01Density, G_L1CloudCoverOffset, G_L1CloudDensOffset, G_L1CloudSize, and G_L1Ambience references are captured. • G_CurrentOSCValue, G_NumericOSCValue, and G_OldOSCValue values are initialized.
`void Update()`	Description: Handles manual or OSC input. Verifies script references. Checks for an “SETA1_DAY CLOUDS(Clone)” GameObject and deactivates it if present. Handles toggled manipulations. Parameters: • None Pre-Conditions: • Script references are valid. Post-Conditions: • Toggled manipulations have been performed.

Post Meditation Survey Controls (PostMeditationSurveyControls.cs)

Description

This script was designed with three questions in mind. As such it is created to work seamlessly with the GameController.cs, PreMeditationSurveyControls.cs, and MeditationHistoryControls.cs scripts.

Dependencies

- This script requires GameController.cs as the interface to the GameData class.

Usage Instructions

1. Because this script is based upon the Testing Player Logs build it is necessary for similar GameObject's and UI components to be present.
2. After the necessary GameObject's and UI components are imported, add or modify UI components as desired.
3. After adding or modifying UI components, create or update the necessary functions within this script.
    a. Also, updating GameController.cs as necessary.
4. After the script has been prepared to handle the new or updated elements, update the UI components to use the scripts interface, passing parameters as necessary.

Interface

Script Members

Member	Details
`public GameObject G_GameController`	A reference to the GameObject to which the GameController.cs script is attached.
`public GameObject G_PostMeditationSurvey`	A reference to the UI container for the Post Meditation Survey form.
`private int G_Question1Value`	The value of the Question 1 Slider for the users Post Meditation response.
`private GameObject G_Question1ValueDisplay`	A reference to the Question 1 value label for the users Post Meditation response.
`private Slider G_Question1Slider`	A reference to the Question 1 Slider for the users Post Meditation response.
`private Slider G_PreMeditationResponse1Slider`	A reference to the Question 1 Slider for the users Pre-Meditation response.
`private GameObject G_PreMeditationResponse1ValueDisplay`	A reference to the Question 1 value label for the users Pre-Meditation response.
`private int G_PreMeditationResponse1Value`	The value of the Question 1 Slider for the users Pre-Meditation response.
`private int G_Question2Value`	The value of the Question 2 Slider for the users Post Meditation response.
`private GameObject G_Question2ValueDisplay`	A reference to the Question 2 value label for the users Post Meditation response.
`private Slider G_Question2Slider`	A reference to the Question 2 Slider for the users Post Meditation response.
`private Slider G_PreMeditationResponse2Slider`	A reference to the Question 2 Slider for the users Pre-Meditation response.

Script Members

Member	Details
`private GameObject G_PreMeditationResponse2ValueDisplay`	A reference to the Question 2 value label for the users Pre-Meditation response.
`private int G_PreMeditationResponse2Value`	The value of the Question 2 Slider for the users Pre-Meditation response.
`private InputField G_Question3InputField`	A reference to the Question 3 input field for the users Post Meditation response.
`private InputField G_PreMeditationResponse3InputField`	A reference to the Question 3 input field for the users Pre-Meditation response.
`private string G_PreMeditationResponse3Value`	The value of the Question 3 input field for the users Pre-Meditation response.

Script Methods

Method	Details
void Start	Description: Initialization of variable references. After references are retrieved data for Pre-Meditation Survey responses is loaded. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_Question1ValueDisplay, G_Question1Slider, G_PreMeditationResponse1Slider, G_PreMeditationResponse1ValueDisplay, G_GameController, G_Question2ValueDisplay, G_Question2Slider, G_PreMeditationResponse2Slider, G_PreMeditationResponse2ValueDisplay, G_Question3InputField, and G_PreMeditationResponse3InputField references are set. • G_PreMeditationResponse1Value, G_PreMeditationResponse2Value, and G_PreMeditationResponse3Value values are set.

Script Methods

Method	Details
void LoadSurveyResponses	Description: Performs a call to GetGameData() from G_GameController to set the values of the Pre-Meditation Response UI controls. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_PreMeditationResponse1Value, G_PreMeditationResponse2Value, and G_PreMeditationResponse3Value are set to values from the active GameData object.
public void UpdateHowAreYouFeelingNow	Description: Called with each update of the question 1 Slider; this function updates the sliders label, translating a numeric value to a text representation. Parameters: • @slider ○ A reference to the Slider object being manipulated. Pre-Conditions: • @slider is a valid Slider object. • G_Question1ValueDisplay is a valid reference. Post-Conditions: • G_Question1ValueDisplay is set to one of five options.

Script Methods

Method	Details
public void UpdateHowStressedDoYouFeel	Description: Called with each update of the question 2 Slider; this function updates the sliders label, translating a numeric value to a text representation. Parameters: • @slider ○ A reference to the Slider object being manipulated. Pre-Conditions: • @slider is a valid Slider object. • G_Question2ValueDisplay is a valid reference. Post-Conditions: • G_Question2ValueDisplay is set to one of six options.
public void Skip	Description: Does nothing. If the Testing Player Logs prototype was further flushed out this would enable players to skip the survey and transition to their next task. Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Script Methods

Method	Details
public void Reset	Description: Resets the survey form to its default values. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_Question1Slider is set to 1 • G_Question1ValueDisplay is set to “Poor” • G_Question2Slider is set to 0 • G_Question2ValueDisplay is set to “Not At All” • G_Question3InputField is set to “”
public void Cancel	Description: Does nothing. If the Testing Player Logs prototype was further flushed out this would enable players to skip the survey and transition to their next task. Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Script Methods

Method	Details
public void Submit	Description: Makes a call to the SetGameData() interface of G_GameController to store the received values into the active GameData object. Then calls SaveGameData() from G_GameController to store the active GameData object to a serialized file. Parameters: • None Pre-Conditions: • None Post-Conditions: • GameData contains the values of the UI elements. • GameData is saved to the local harddrive.

Preference Loader (PreferenceLoader.cs)

Description

Currently, an empty and unused script (if present in any build except the Menu Builds). Its original purpose was to provide app initialization from a local file containing user preferences; the thought of which came up during the creation of the Menu Builds. Its primary purpose would've been handling loading player preferences for volumes (environment, SFX, music, meditation tracks, etc.) as those were the manipulatable options within the Menu2D build. Additional preferences were intended for an orb acting as a meditation/scene guide; present in a Waterfall build, separate of this project.

Within the Menu Builds this script handles detecting the player's setup (2D or VR) and activates the appropriate set of GameObject's to handle menu interactions. It then updates the menu elements with values corresponding to the scenes current state; i.e. volume Sliders in the 2D menu are set to the corresponding values of the Master AudioMixer.

Dependencies

- This script has no dependencies.

Usage Instructions

1. N/A

Interface

Script Members

Member	Details

Script Methods

Method	Details

Pre Meditation Survey Controls (PreMeditationSurveyControls.cs)

Description

Dependencies

- This script requires GameController.cs as the interface to the GameData class.

Usage Instructions

1. Because this script is based upon the Testing Player Logs build it is necessary for similar GameObject's and UI components to be present.
2. After the necessary GameObject's and UI components are imported, add or modify UI components as desired.
3. After adding or modifying UI components, create or update the necessary functions within this script.
    a. Also, updating GameController.cs as necessary.
4. After the script has been prepared to handle the new or updated elements, update the UI components to use the scripts interface, passing parameters as necessary.

Interface

Script Members

Member	Details
`public GameObject G_GameController`	A reference to the GameObject to which the GameController.cs script is attached.
`public GameObject G_PreMeditationSurvey`	A reference to the UI container for the Pre-Meditation Survey form.
`private int G_Question1Value`	The value of the Question 1 Slider.
`private GameObject G_Question1ValueDisplay`	A reference to the label for the Question 1 Slider.
`private Slider G_Question1Slider`	A reference to the Question 1 Slider.
`private int G_Question2Value`	The value of the Question 2 Slider.
`private GameObject G_Question2ValueDisplay`	A reference to the label for the Question 2 Slider.
`private Slider G_Question2Slider`	A reference to the Question 2 Slider.
`private InputField G_Question3InputField`	A reference to the Question 3 InputField.

Script Methods

Method	Details
`void Start`	Description: Reference variable initialization. Parameters: • None Pre-Conditions: • Expected GameObjects and UI components exist within the scene. • G_PreMedtiationSurvey is set to a valid GameObject with expected child elements. Post-Conditions: • G_GameController, G_Question1ValueDisplay, G_Question1Slider, G_Question2ValueDisplay, G_Question2Slider, and G_Question3InputField references are set.
public void UpdateHowAreYouFeelingNow	Description: Called with each update of the question 1 Slider; this function updates the sliders label, translating a numeric value to a text representation. Parameters: • @slider ○ A reference to the Slider object being manipulated. Pre-Conditions: • @slider is a valid Slider object. • G_Question1ValueDisplay is a valid reference. Post-Conditions: • G_Question1ValueDisplay is set to one of five options.

Script Methods

Method	Details
public void UpdateHowStressedDoYouFeel	Description: Called with each update of the question 2 Slider; this function updates the sliders label, translating a numeric value to a text representation. Parameters: • @slider ○ A reference to the Slider object being manipulated. Pre-Conditions: • @slider is a valid Slider object. • G_Question2ValueDisplay is a valid reference. Post-Conditions: • G_Question2ValueDisplay is set to one of six options.
public void Skip	Description: Does nothing. If the Testing Player Logs prototype was further flushed out this would enable players to skip the survey and transition to their next task. Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Script Methods

Method	Details
public void Reset	Description: Resets the survey form to its default values. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_Question1Slider is set to 1 • G_Question1ValueDisplay is set to “Poor” • G_Question2Slider is set to 0 • G_Question2ValueDisplay is set to “Not At All” • G_Question3InputField is set to “”
public void Cancel	Description: Does nothing. If the Testing Player Logs prototype was further flushed out this would enable players to skip the survey and transition to their next task. Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Script Methods

Method	Details
public void Submit	Description: Makes a call to the SetGameData() interface of G_GameController to store the received values into the active GameData object. Then calls SaveGameData() from G_GameController to store the active GameData object to a serialized file. Parameters: • None Pre-Conditions: • None Post-Conditions: • GameData contains the values of the UI elements. • GameData is saved to the local harddrive.

Scene Fader (SceneFader.cs)

Description

Handles fading of a scene on load. This provides an easy transition into the experience for users. When in 2D mode the entire GUI is faded, however this technique doesn't work for VR thus the Camera Rig needs to be surrounded with objects using a manipulatable material. The early implementation of this script used multiple objects to surround the Camera Rig, however, later on it was found that objects can be rendered inside out. Therefore, it should be possible to surround the Camera Rig with one object using the manipulatable material.

Dependencies

- This script has no dependencies.

Usage Instructions

1. Attach this script to a GameObject within the scene.
2. In the inspector set the reference to for G_FadeOutTexture and G_BlackBox parameters.
3. Adjust the fade speeds as desired.

Interface

Script Members

Member	Details
`public bool G_Debug`	Print debug statements, this WILL reduce FPS because statements are printed during the OnGUI event (every frame).
`public Texture2D G_FadeOutTexture`	The image texture that will overlay the screen.
`public float G_FPSFadeSpeed`	The transition speed when in 2D.
`public float G_VRFadeSpeed`	The transition speed when in VR.
`public Material G_BlackBox`	A reference to material applied to the box cubes surrounding the Camera Rig.
`private int G_DrawDepth`	The texture's order in the draw hierarchy: lower numbers render on top.
`private float G_Alpha`	The texture's alpha value between 0 and 1.
`private float G_FadeDir`	The direction to fade: in = -1, out = 1.
`private bool G_FadeComplete`	A flag to detect whether or not the fade has completed.

Script Methods

Method	Details
`private void Start`	Description: Basic initializations. Parameters: • None Pre-Conditions: • None Post-Conditions: • Resets G_BlackBox.color to a solid black in case it wasn't when the scene loads. (Changes to materials persist after exiting the scene.)
`private void Update`	Description: Calls PerformFade() should the fading process be incomplete. Parameters: • None Pre-Conditions: • None Post-Conditions: • None
`private void OnLevelWasLoaded`	Description: Begins the fading process. Parameters: • None Pre-Conditions: • None Post-Conditions: • None

Script Methods

Method	Details
`private void OnApplicationQuit`	Description: Resets the G_BlackBox.color material. Parameters: • None Pre-Conditions: • None Post-Conditions: • Resets G_BlackBox.color to a solid black in case it wasn't when the scene loads. (Changes to materials persist after exiting the scene.)
`public float BeginFade`	Description: Sets G_FadeDir to the parameter passed and returns G_VRFadeSpeed or G_FPSFadeSpeed for use in later calculations. Parameters: • None Pre-Conditions: • None Post-Conditions: • None
`private void PerformFade`	Description: Perfoms a screen fade in/out. (Handles fading for the HMD) OnGUI Handles the fading for FPSController. Parameters: • None Pre-Conditions: • None Post-Conditions: • G_BlackBox.color's alpha value is faded with each consecutive call.

Script Methods

Method	Details
`private void OnGUI`	Description: Fades out/in the alpha value using a direction, speed, and Time.deltatime (to convert the operation to sections). Parameters: • None Pre-Conditions: • None Post-Conditions: • Draws a texture across the GUI using G_FadeOutTexture. • G_FadeOutTexture is faded with each consecutive call.

Tree Object Manipulation (TreeObjectManipulation.cs)

Description

Changes tree colors over time.

Dependencies

- This script has no dependencies.

Usage Instructions

1. Attach this script to a GameObject within the scene.
2. Tag each tree to be manipulated with “Tree.”
3. Adjust change_interval and change_duration as desired.

Interface

Script Members

Member	Details
`public bool print_debug`	Controls whether or not to print debugging statements to the console as the script runs.
`public bool animation_active`	Determines whether or not the animation will be active.
`public int change_interval`	The number of seconds between color updates.
`public float change_duration`	The duration of transition for color updates, used by Color.Lerp().
`private GameObject[] tree_objects`	Objects tagged with “Tree” within the scene.
`private float prev_alarm_timestamp`	The timestamp of the last color update.

Script Methods

Method	Details
`void Start`	Description: Variable initialization. Parameters: • None Pre-Conditions: • Trees to be manipulated are tagged with “Tree” • OSC has been properly setup Post-Conditions: • The tree_objects member contains a reference to all tree objects in the scene.

Script Methods

Method	Details
`void Update`	Description: If the animation_active flag is set and the set interval, change_interval, has passed, Update() handles the color transition of trees collected in the tree_objects array by walking through each tree and adding to their red and green values. Parameters: • None Pre-Conditions: • tree_objects contains references to all trees within the scene. Post-Conditions: • Tree objects in the scene have been transitioned towards a deeper hue of red.

Tree Object Manipulation OSC (TreeObjectManipulationOSC.cs)

Description

This script handles the gradual manipulation of tree colors based upon received OSC values. Tree colors transition between a fall like color from a spring green.

Dependencies

- This script requires the application of Osc.cs, UDPPacketIO.cs, JS_OSCReceiver.js

Usage Instructions

1. Attach this script to a GameObject within the scene.
2. Tag each tree to be manipulated with “Tree.”
3. Adjust Max_OSC_val as desired.

Interface

Script Members

Member	Details
`public bool print_debug`	Controls whether or not to print debugging statements to the console as the script runs.
`public bool animation_active`	Determines whether or not the animation will be active.
`public int max_OSC_val`	The maximum OSC value that can be handled.
`public string current_OSC_val`	The most recent OSC value received.
`private GameObject[] tree_objects`	Objects tagged with “Tree” within the scene.
`private float prev_alarm_timestamp`	UNUSED
`private int color_scale`	UNUSED
`private int current_numeric_OSC_val`	The numeric conversion of current_OSC_val.
`private string old_OSC_val`	The previous OSC value.
`private float color_fraction`	The ratio of tree color adjustment based upon the OSC input.
`private static float max_leaf_red_val`	The maximum amount of red a tree can transition to.
`private static float max_frond_red_val`	The maximum amount of red a frond can transition to.
`private static float max_green_val`	The maximum amount of green to transition to.

Script Methods

Method	Details
`void Start`	Description: Variable initialization. Parameters: • None Pre-Conditions: • Trees to be manipulated are tagged with "Tree" • OSC has been properly setup Post-Conditions: • The tree_objects member contains a reference to all tree objects in the scene.
`void Update`	Description: Handles the manipulation of tree colors based upon OSC if the animation_active flag is set. Parameters: • None Pre-Conditions: • tree_objects contains references to all trees within the scene. Post-Conditions: • Tree objects in the scene have been updated based upon a ratio of OSC values and set maximums.

OSC (Osc.cs)

Description

Used without modification from an open source example project on Github. This Unity "plugin" implements OSC functionality.

Dependencies

- UDPPacketIO.cs

Usage Instructions

1. Place in the "Plugins" folder in the root of the Project Assets.
2. This script requires no user modification, as it serves only as a dependency for the OSC receiver script.

Interface

Relevant Script Members

Member	Details
`OSCMessage`	

Relevant Script Methods

Method	Details
	Description: TBA Parameters: • None Pre-Conditions: • None Post-Conditions: • None

UDP Packet IO (UDPPacketIO.cs)

Description

Used without modification from an open source example project on Github. This Unity "plugin" provides UDP functionality.

Dependencies

- This script has no dependencies.

Usage Instructions

1. Place in the "Plugins" folder in the root of the Project Assets.
2. This script requires no user modification, as it serves only as a dependency for the OSC plugin.

Interface

Script Members

Member	Details

Script Methods

Method	Details
	Description: TBA Parameters: • None Pre-Conditions: • None Post-Conditions: • None

JavaScript OSC Receiver (JS_OSCReceiver.js)

Description

Based on an open source example project on Github, with modifications to suit this project. This script runs an OSC listening server on a specified IP / port. It can update another specified component's object to this value.

The script operates by checking for

Dependencies

- Osc.cs
- UDPPacketIO.cs

Usage Instructions

1. Verify the OSC and UDP Packet scripts are present in the "Plugins" folder.
2. Add this script to an object
3. Drag the object yo want to send the value to

Interface

Script Members

Member	Details

Script Methods

Method	Details
	Description: TBA Parameters: • None Pre-Conditions: • None Post-Conditions: • None

References

Geig, M. (n.d.). *Persistence - Saving and Loading Data*. Retrieved from Unity - Tutorials: https://unity3d.com/learn/tutorials/topics/scripting/persistence-saving-and-loading-data

sotirosn. (n.d.). *DontDestroyOnLoad not working*. Retrieved from Unity Answers: http://answers.unity3d.com/questions/441721/dontdestroyonload-not-working.html#answer-456306

REFERENCES

[1] "Unity manual," 2016. [Online]. Available: https://docs.unity3d.com/Manual/index.html

[2] "Unity home page," 2016. [Online]. Available: https://unity3d.com/

[3] "Github home page," 2016. [Online]. Available: https://github.com/

[4] "Google drive home page," 2016. [Online]. Available: https://www.google.com/drive/

[5] "Drop box home page," 2015. [Online]. Available: https://www.dropbox.com/home

[6] "Lighting and rendering," 2015. [Online]. Available: https://unity3d.com/learn/tutorials/topics/graphics/unity-5-lighting-and-rendering

[7] "Real-time lighting vs. baked lightmaps," 2016. [Online]. Available: http://minddesk.com/learn/article.php?id=54

[8] G. Kumparak, "A brief history of oculus," 2016. [Online]. Available: https://techcrunch.com/2014/03/26/a-brief-history-of-oculus/

[9] "Spec comparison: Does the rifts touch update make it a true vive competitor?" 2016. [Online]. Available: http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/

[10] M. Swider, "Htc vive vs oculus rift: which vr headset is better?" 2016. [Online]. Available: http://www.techradar.com/news/wearables/htc-vive-vs-oculus-rift-1301375

[11] "Oculus rift home page," 2016. [Online]. Available: https://www.oculus.com/

[12] A. Souppouris, "How htc and valve built the vive," 2016. [Online]. Available: https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/

[13] "Htc vive home page," 2016. [Online]. Available: https://www.vive.com/us/

[14] "Google cardboard home page," 2016. [Online]. Available: https://vr.google.com/cardboard/

[15] "Samsung gear vr home page," 2016. [Online]. Available: http://www.samsung.com/us/explore/gear-vr/?cid=ppc-

[16] R. Holy, "Google cardboard versus samsung gear vr," 2016. [Online]. Available: http://www.androidcentral.com/google-cardboard-vs-samsung-gear-vr

[17] "I am cardboard faq," 2016. [Online]. Available: http://www.imcardboard.com/faq

[18] "Compatible android smartphones," 2016. [Online]. Available: http://www.samsung.com/global/galaxy/gear-s2/device-compatibility/

[19] Staff, "Welcome to unity," 2016. [Online]. Available: https://store.unity.com/

[20] ——, "3d formats," 2016. [Online]. Available: https://docs.unity3d.com/Manual/3D-formats.html

[21] ——, "Welcome to the unity scripting reference!" 2016. [Online]. Available: https://docs.unity3d.com/ScriptReference/index.html

[22] ——, "What is unreal engine 4?" 2016. [Online]. Available: https://www.unrealengine.com/what-is-unreal-engine-4

[23] ——, "Programming guide," 2016. [Online]. Available: https://docs.unrealengine.com/latest/INT/Programming/index.html

[24] ——, "Get cryengine today," 2016. [Online]. Available: https://www.cryengine.com/get-cryengine

[25] Community, "Programing language?" 2016. [Online]. Available: https://www.cryengine.com/community/viewtopic.php?f=314&t=113367

[26] "Game programming tutorials," 2017. [Online]. Available: https://www.digitaltutors.com/subject/game-development-programming-tutorials

[27] V. Lau, "Vr game engines," 2016. [Online]. Available: https://infinityleap.com/vr-game-engines/

[28] Community, "Best game engine: Unity vs cryengine vs unreal engine 4," 2016. [Online]. Available: http://www.ign.com/boards/threads/best-game-engine-unity-vs-cryengine-vs-unreal-engine-4.454692197/

[29] Staff, "Difference between mov vs. avi," 2016. [Online]. Available: http://www.differencebetween.net/technology/difference-between-mov-vs-avi/

[30] ——, "Opening avi files," N.D. [Online]. Available: http://file.org/extension/avi

[31] Community, "Comparison of video container formats," 2016. [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_video_container_formats

[32] G. S., "Expert opinion: What is the difference between an mpeg4 & an avi or mov file?" 2016. [Online]. Available: http://mtdigital.com/differencebetweenmpeg4avimovfile/

[33] Staff, "Mp4 vs mov comparison: Which is better?" 2016. [Online]. Available: https://www.macxdvd.com/mac-dvd-video-converter-how-to/mp4-vs-mov.htm

[34] ——, "H.264 profiles and levels," 2008. [Online]. Available: http://blog.mediacoderhq.com/h264-profiles-and-levels/

[35] ——, "The ogg container format," N.D. [Online]. Available: https://xiph.org/ogg/

[36] ——, "Movie texture," 2016. [Online]. Available: https://docs.unity3d.com/Manual/class-MovieTexture.html

[37] ——, "An overview of h.264 advanced video coding," N.D. [Online]. Available: https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/

[38] ——, "H.264/mpeg-4 part 10 advanced video coding," N.D. [Online]. Available: http://www.elecard.com/en/technology/avc.html

[39] ——, "Mov vs h.264," N.D. [Online]. Available: http://www.iorgsoft.com/compare/mov-vs-h.264-comparison.html

[40] ——, "Mov vs mp4," N.D. [Online]. Available: http://www.iorgsoft.com/compare/mov-vs-mp4-comparison.html

[41] ——, "Mp4 vs h.264," N.D. [Online]. Available: http://www.iorgsoft.com/compare/mp4-vs-h.264-comparison.html

[42] ——, "Overview of qtff," 2016. [Online]. Available: https://developer.apple.com/library/content/documentation/QuickTime/QTFF/QTFFChap1/qtff1.html#//apple_ref/doc/uid/TP40000939-CH203-BBCGDDDF

[43] cpapciak, "Most common types of video files and containers," 2012. [Online]. Available: http://www.dvdyourmemories.com/blog/types-of-video-files-containers/

[44] L. Case, "All about video codecs and containers," 2010. [Online]. Available: http://www.techhive.com/article/213612/all_about_video_codecs_and_containers.html?page=2

[45] Staff, "Audio files," 2014. [Online]. Available: https://docs.unity3d.com/462/Documentation/Manual/AudioFiles.html

[46] ——, "Audio clip," 2016. [Online]. Available: https://docs.unity3d.com/Manual/class-AudioClip.html

[47] ——, "Tracker modules," 2014. [Online]. Available: https://docs.unity3d.com/462/Documentation/Manual/TrackerModules.html

[48] Aleksandr, "Documentation, unity scripting languages and you - unity blog," 2016. [Online]. Available: https://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/

[49] "Unityscript versus javascript," 2016. [Online]. Available: http://wiki.unity3d.com/index.php/UnityScript_versus_JavaScript

[50] R. B. de Oliveira, “A scarily powerful language for .net,” 2009. [Online]. Available: https://boo-language.github.io/

[51] “Creating and using scripts,” 2016. [Online]. Available: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

[52] “Nvidia titan x graphics card with pascal — geforce,” 2016. [Online]. Available: http://www.geforce.com/hardware/10series/titan-x-pascal

[53] “Gtx 1080 graphics card — geforce,” 2016. [Online]. Available: http://www.geforce.com/hardware/10series/geforce-gtx-1080

[54] “Radeon rx 480 graphics card,” 2016. [Online]. Available: http://www.amd.com/en-us/products/graphics/radeon-rx-series/radeon-rx-480

[55] “Sony recommends 90fps at minimum for best playstation vr experience on ps4,” 2016. [Online]. Available: http://cukusa.com/cuk-custom-vr-extreme-gaming-pc-the-best-new-vr-ready-tower-desktop-computer-for-gamers-glossy-white-blue-lights.html

[56] “Human experimentation: An introduction to the ethical issues,” 2016. [Online]. Available: http://www.pcrm.org/research/healthcare-professionals/research-compendium/human-experimentation-an-introduction-to-the

[57] M. M. Erik Watterson, Padraig Gillen, “Meditation in vr technlogy review,” 2016.

[58] “Unity - manual: Scripting,” 2016. [Online]. Available: https://docs.unity3d.com/Manual/ScriptingSection.html

[59] “Types (c# programming guide),” 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms173104.aspx

[60] “Object-oriented programming (c#),” 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/mt656686.aspx

[61] “Steamvr home page,” 2016. [Online]. Available: http://store.steampowered.com/steamvr

[62] S. Hollister, “Don’t buy a pc graphics card until you read this,” 2016. [Online]. Available: https://www.cnet.com/news/nvidia-gtx-1080-vs-1070-vs-amd-rx480-gpu-price-availability/

[63] Staff, “System requirements for unity 5.4,” 2016. [Online]. Available: https://unity3d.com/unity/system-requirements

[64] ——, “Hardware & software specifications,” 2016. [Online]. Available: https://docs.unrealengine.com/latest/INT/GettingStarted/RecommendedSpecifications/

[65] ——, “System requirements,” 2016. [Online]. Available: http://docs.cryengine.com/display/CEMANUAL/System+Requirements

[66] ——, “Build once, deploy anywhere,” 2016. [Online]. Available: https://unity3d.com/unity/multiplatform

[67] ——, “Frequently asked questions (faq),” 2016, section: General Information - What platforms are supported? [Online]. Available: https://www.unrealengine.com/faq

[68] ——, “Platforms,” 2016. [Online]. Available: https://www.cryengine.com/features/platforms

[69] ——, “Cryengine platform specific,” 2016. [Online]. Available: http://docs.cryengine.com/display/CEPROG/CRYENGINE+Platform+Specific

[70] G. S., “Expert opinion: What is the difference between an mpeg4 & an avi or mov file?” 2016. [Online]. Available: http://mtdigital.com/differencebetweenmpeg4avimovfile/

[71] Staff, “Art asset file types,” 2014. [Online]. Available: http://docs.cryengine.com/display/SDKDOC2/Art+Asset+File+Types

[72] ——, “Avi vs h.264,” N.D. [Online]. Available: http://www.iorgsoft.com/compare/avi-vs-h.264-comparison.html

[73] C. Bryant, “What are the pros and cons when comparing the mp4 and mov video containers?” 2014. [Online]. Available: https://www.quora.com/What-are-the-pros-and-cons-when-comparing-the-mp4-and-MOV-video-containers

[74] B. DeFrees, ".mov vs .mp4 image quality vs. file size," 2011. [Online]. Available: https://www.quora.com/What-are-the-pros-and-cons-when-comparing-the-mp4-and-MOV-video-containers

[75] K. D. McGrath, "Cs senior capstone," 2016. [Online]. Available: http://eecs.oregonstate.edu/capstone/cs/capstone.cgi?home=1

[76] M. Premi, "Project description: Virtual reality meditation app," 2016. [Online]. Available: http://eecs.oregonstate.edu/capstone/cs/capstone.cgi?project=200

[77] C. Minasians, "Nvidia geforce gtx 1080, 1070 & 1060 vs amd radeon rx 480," 2016. [Online]. Available: http://www.pcadvisor.co.uk/review/graphics-cards/nvidia-geforce-gtx-1080-1070-1060-vs-amd-radeon-rx-480-3641216/

[78] "Cuk sentinel custom vr extreme gaming pc the best new vr ready tower desktop computer for gamers (glossy white/blue lights)," 2016. [Online]. Available: http://cukusa.com/cuk-custom-vr-extreme-gaming-pc-the-best-new-vr-ready-tower-desktop-computer-for-gamers-glossy-white-blue-lights.html

[79] "What is intellectual property?" [Online]. Available: http://www.wipo.int/about-ip/en/

[80] "What is virtual reality?" 2016. [Online]. Available: http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html

[81] "User experience basics," 2016. [Online]. Available: https://www.usability.gov/what-and-why/user-experience.html