# Spring 2018 Final Progress Report

Group 61: iCreate - Generative Design in Virtual Reality

CS 463 | Spring 2018 Term

Hannah Solorzano - Nabeel Shariff - Rhea Mae Edwards

**Abstract**

In regards to the computer science senior capstone project, group 61, Generative Design in Virtual Reality, of the 2017-2018 academic year at Oregon State University, the purpose of this document is to briefly recap the group's project's purpose and goals, describe and state any changes done in regards to the Software Requirements document, Software Specification Design document, and Technology Review and Implementation Plan document of the group's project, present the final Gantt chart of the group's actually project progress and generation, present the final version of the group's Engineering Expo poster, restate each team member's weekly blog posts, describe the installation and instructions in regards to the group's project, describe the group's experiences in going this senior capstone project, present the group's overall rating of the project's client along with reasonings why, and deeper explain the code and software that was used in regards to group 61's senior capstone project.

## PARTICIPANTS

The iCreate - Generative Design in Virtual Reality senior software engineering project consists of the following team members:

**Raffaele de Amicis,** *Mentor*

Rhea Mae Edwards | Nabeel Shariff | Hannah Solorzano

The following persons are associated with Intel Corporation in regards to Hardware, Software and Development Environments in relations to the project:

**Mike Premi,** *Co-Mentor*

The following persons are guidance mentors as a part of the senior software engineering project course for the students:

**Kevin McGrath,** *Instructor* | **Behnam Saeedi,** *Teaching Assistant* | **Kirsten Winters,** *Instructor*

# TABLE OF CONTENTS

# 1 Introduction

## 1.1 Purpose

The purpose of this project is to develop a VR application that allows users to create complex 3D objects by seamlessly creating simpler objects and combining them to form intricate structures.

## 1.2 Goals

The virtual reality (VR) application will utilize a virtual reality headset with input from the user via a controller recognition software. The VR headset will be used to look around in virtual space while the controllers' recognition software will be used by the user to draw curves. The application will also need to utilize the GPU in a computer to both run the VR application and render 3D objects in the virtual space. Additionally, the 3D modeling will be based on generative design techniques, and the assembly of the complex 3D designs will utilize mathematical equations and algorithms to derive the appropriate structure of the design.

The outcome of this Oregon State University computer science senior capstone project is a virtual reality program that allows the user to utilize generative design to develop complex architectural structure.

## 1.3 Request Story

This project was requested by Associate Professor Raffaele De Amicis at Oregon State University planning to work with Mike Premi of Intel and the College of Forestry at Oregon State University. The purpose of this program was to test different methods of generative design in the field of architecture, especially wooden architecture. With generative design, architects would be able to create new and intuitive designs that are not limited to human imagination.

## 1.4 Team Members and Client

The client who sponsored this project is Raffaele de Amicis, an Associate Professor here at Oregon State University who researches advancements in geo-intellectual complex visualization systems and how these technologies can be used in civil protection, eco-terrorism, energy efficiency, and smart cities. His research projects are primarily in virtual reality, augmented reality, geovisualization, visual analytics, and geometric modeling.

This capstone team is comprised of three members: Hannah Solorzano, Rhea Mae Edwards, and Nabeel Shariff. While most of the development work was divided evenly among the group members, there were a few minor roles which revolved around the documentation and communication with the client.

Hannah's role outside of development is that of the document writer. While all three members worked on the documentation, Hannah was in charge of looking over the document to make sure it had the necessary explanations and diagrams. Rhea Mae's role was that of the recorder. She took the meeting notes during both the client and TA meetings and was responsible for the emails that were sent from our team to the client to communicate meeting times and project problems. Additionally, she emailed out the document templates for the progress reports to the team. Lastly, Nabeel's role was where he was the one to submit the progress reports via Canvas or email. This included taking the three video pieces that each member did that described what each have been doing during the term, and stitching them together into one video. Then, Nabeel would take this completed video, and turn it in along side the PowerPoint presentation and written report.

In regards to the role of the client, Raffaele De Amicis had more of a supervisor role, where he gave instruction on which direction he wanted the project to head. Though he sometimes offered advise or other aid when the team was stuck on a particular piece of code.

# 2 Software Requirements Document

## 2.1 Original Document

Continue reading onto the next page.

# Software Requirements 1.1

Group 61: iCreate - Generative Design in Virtual Reality

Fall 2017 Term

Hannah Solorzano - Nabeel Shariff - Rhea Mae Edwards



**Abstract**

The functionality of the virtual reality program iCreate is described in this document. This document is aimed at specifying requirements of the iCreate software to be developed, along with user characteristics, constraints, and terminology of the program. The product functions, assumptions and dependencies are also detailed within the document. Documentation and a detailed development plan for iCreate are also provided.

## PARTICIPANTS

The ICreate - Generative Design in Virtual Reality senior software engineering project consists of the following team members:

**Raffaele de Amicis,** *Mentor*

Rhea Mae Edwards | Nabeel Shariff | Hannah Solorzano

The following persons are associated with Intel Corporation in regards to Hardware, Software and Development Environments in relations to the project:

**Mike Premi,** *Co-Mentor*

The following persons are guidance mentors as a part of the senior software engineering project course for the students:

**Kevin McGrath,** *Instructor* **Behnam Saeedi,** *Teaching Assistant* **Kirsten Winters,** *Instructor*

# TABLE OF CONTENTS

# 1 INTRODUCTION

## 1.1 Purpose

The purpose of this user requirements document is to provide a detailed description of the “iCreate” virtual reality (VR) application. This document will illustrate the functions and features of the applications. Additionally, it will explain the interface and system constraints.

This document is intended to be proposed to Raffaele de Amicis for its approval and a reference for developing the first version of the system for the development team.

## 1.2 Scope

The iCreate software is a virtual reality application that allows users to construct complex architectural designs in VR using simple sketches, gestures, and parameters defined by the user. The application will be available to download for systems that can support VR headsets.

VR users can provide parameters for a base 3D object that will be used to build the user’s complex design. This objective of modifying the base object, can be done either by manually providing the parameters or by altering the object via gestures or virtual sketches. Moreover, both the base objects and the complex designs can be saved for quick access in the future.

Furthermore, the software will need a computer that is capable of running virtual reality applications. The application will also use the proprietary VR software for the respective headset being used.

## 1.3 Glossary

Table 1
Terms and Definitions

| Terms | Definitions |
|---|---|
| VR | Virtual Reality |
| User | Someone who interacts with the iCreate virtual reality application. |
| 3D | 3-dimensional |
| Parameters | The measurements for objects defined by the user. |
| Input | Stimulus provided by user. |
| Output | Feedback from the software based on the user’s input. |
| Virtual Space | A 3D area in virtual reality in which the user can move around and interact with objects. |
| GPU | Graphics processing unit responsible for quick handling and rendering graphics on a computer. |
| Generative Design | A form finding process that mimics nature’s evolutionary approach to design[2]. |
| Render | The process of creating 3D objects and environments. |

## 1.4 References

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

[2] Autodesk, "Generative Design at Airbus | Customer Stories | Autodesk," Autodesk, [Online].
Available: https://www.autodesk.com/customer-stories/airbus. [Accessed 31 October 2017].

# 2 OVERALL DESCRIPTION

## 2.1 Product Perspective

The VR application will utilize a virtual reality headset with input from the user via a controller or gesture recognition software. The VR headset will be used to look around in virtual space while the controllers or gesture recognition software will be used by the user to draw sketches.

The VR application will need to utilize the GPU in a computer to both run the VR application and render 3D objects in the virtual space. Additionally, the 3D modeling will be based on generative design techniques, and the assembly of the complex 3D designs will utilize mathematical equations and algorithms to derive the appropriate structure of the design.

## 2.2 Product Functions

Using the VR application, the user will be able to select and spawn 3D shapes from a provided library and will then be able to adjust the dimensions of the shape. The interface will allow the user to distort or scale the shape to the desired size. Once the shape is finalized, the user sketches a trajectory that represents the curve of the architecture in which they are trying to create. The program will then generate several more of the selected shapes to best fit the curve, forming the structure. If the user wishes, they will be able to store the modified shape and final generated structure in a library for later use.

## 2.3 User Characteristics

There are two main types of users that interact with iCreate: Professional designers and normal users. Each of the types use the system differently, thus, each have their own requirements.

- The professional designers will utilize iCreate for professional designs and projects. They will need access to a wide array of curves and precise transformation and design techniques that will enable them to produce state of the art designs.
- The game designers and students will use iCreate either for entertainment, such as creating simulated environments, or for education, like learning design techniques and concepts.

## 2.4 Constraints

Firstly, the software constraints for the VR application pertain to the functions that generate complex 3D structures from simple 3D objects. For example, when the user creates a 3D shape and trajectory/path, if the object is larger than the path, or if there are too many objects to fit the path, then the program will not be able to properly generate a complex structure.

Secondly, the with regards to hardware constraints, the VR application is created specifically for the HTC Vive using the Unity 3D game engine. A VR headset and a computer capable of running VR software are required to use the VR application. As VR programs are graphics intensive, for iCreate to run successfully, the minimum processor is a Intel Core i5-4590 or AMD FX 8350 (equivalent or better), a NVIDIA GTX 970 or AMD Radeon R9 290

## 2.5 Assumptions and Dependencies

- Users have to have enough computing and graphics power to handle VR software and applications.
- The application must be used with a VR headset.
- The VR application must allow users to spawn and modify 3D objects.

# 3 SPECIFIC REQUIREMENTS

This section contains all the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

## 3.1 External Interface Requirements

### *3.1.1 User Interfaces*

The interface will need to functionally allow the user to spawn a 3D object, then modify it via either a controller or gesture input. Additionally, the user will be able to save to a library for later use or load their creations to create complex structures. Finally, the application will display a way to transform the 3D objects by allowing the user to scale or resize the object.

### *3.1.2 Hardware Interfaces*

The VR application will be able to run on a VR headset (preferably a HTC Vive) and a computer that can run VR software.

The minimum CPU and GPU requirements to successfully run the VR application are:

- Minimum CPU requirements: Intel Core i5-4590 or AMD FX 8350 (equivalent or better)
- Minimum GPU requirements: NVIDIA GTX 970 or AMD Radeon R9 290 (equivalent or better)

### *3.1.3 Software Interfaces*

The VR application will be usable on a Windows operating system capable of running the VR headset's respective proprietary software.

## 3.2 Functional Requirements

### *3.2.1 iCreate VR Environment*

The iCreate software will allow the user to:

- Move around in VR.
- Instantiate a simple 3d object.
- Generate multiple instances of the 3d object.

### *3.2.2 iCreate Object Library*

The iCreate software will allow the user to:

- Obtain 3D objects from the application's library.

### *3.2.3 iCreate Curves*

The iCreate software will allow the user to:

- To draw a 3D curve.
- Draw a trajectory in the form of a curve.
- Create a B-Spline curve.
- Create a Bezier curve.
- Create a ellipse curve.
- Create a circle curve.

- Create a hyperbola curve.
- Create a parabola curve.
- Use the curve to indicate how the user would like to displace multiple instances of the initial object.

#### *3.2.4 iCreate Transformation and Translation*

The iCreate software will allow the user to:

- Extrude the object.
- Resize the object.
- Indicate how they would like to rotate objects across a curve.
- Indicate how they would like to translate objects across a curve.
- Indicate how they would like to scale objects across a curve.
- The 3D objects that make up a complex structure must be connected with each other.

#### *3.2.5 iCreate Save and Load*

The iCreate software will allow the user to:

- Import geometry from a .fbx extension file format.
- Import a 3D object.
- Save a 3D object.
- Save structures.
- Save the entirety of the project as a whole.

#### *3.2.6 iCreate Stretch Goals*

After the completion of the iCreate program, the developers plan to use a robot to take the design created in the VR space, and recreate it in real life.

- The robot will be provided by the Oregon State University Robotics Club.
- An application programming interface (API) library will be used to allow the robot to read and apply the design.

### 3.3 Performance Requirements

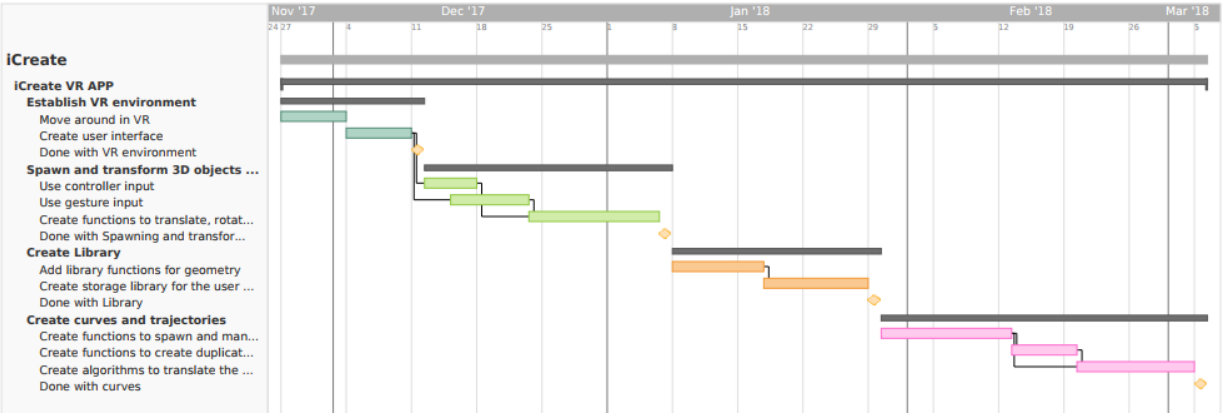
The iCreate program should be able to run on a computer with at least an Intel Core i5-4590 or AMD FX 8350 (equivalent or better), a NVIDIA GTX 970 or AMD Radeon R9 290 (equivalent or better) graphics card, at least 4GB of RAM, at at least 30 frames per second.

The requirements to run iCreate will also depend on the user and the scale or intricacy of the architectural structure that is designed as a larger, more complex structure will require more processing power and a stronger graphics card.

### 3.4 Design Constraints

The VR program will have to run on a HTC Vive VR headset, and be able to support both controller and gesture input through Leap Motion and Lighthouse Tracking Technology. The user interface must be able to allow the user to spawn and transform 3D objects, and save and load previous creations.

## 3.5 Gantt Chart



### 2.2 Document Changes

No changes have been made to the iCreate Software Requirements document, which is still current valid at the document's current version of 1.1. No new requirements were added, no existing requirements were changed, and no requirements were deleted.

### 2.3 Final Gantt Chart

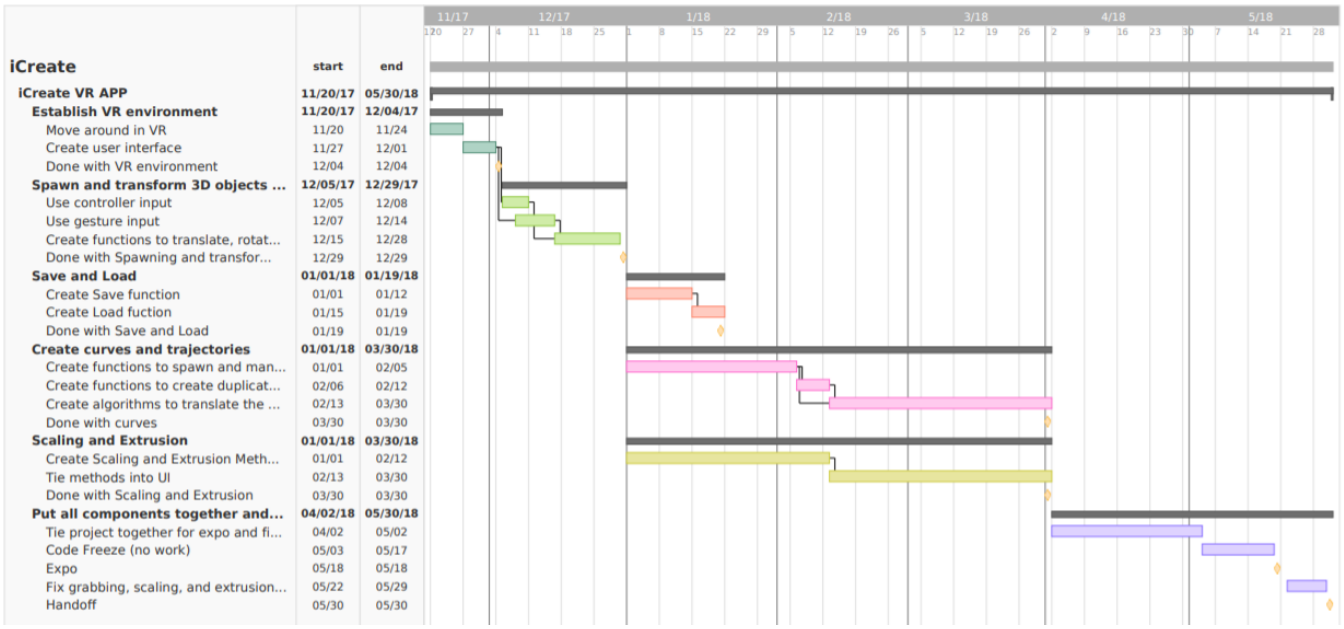



Figure 1. Final Gnatt Chart, illustrating when every piece was actually done when.

## 3 Design Document

### 3.1 Original Document

Continue reading onto the next page.

# Software Specification Design Document

Group 61: iCreate - Generative Design in Virtual Reality

April 23, 2018 - Winter Term

Hannah Solorzano - Nabeel Shariff - Rhea Mae Edwards

**Abstract**

iCreate Generative Design in Virtual Reality is a software program that allows the user to create complex architectural structures using a series of spawned objects. As precision in movements and gestures are important, the UI has been designed in such a way that the user is unhindered and experiences a limited learning curve when using the software for the first time.

## PARTICIPANTS

The ICreate - Generative Design in Virtual Reality senior software engineering project consists of the following team members:

**Raffaele de Amicis,** *Mentor*

**Hannah Solorzano,** *Software Developer* **Nabeel Shariff,** *Software Developer* **Rhea Mae Edwards,** *Software Developer*

The following persons are associated with Intel Corporation in regards to Hardware, Software and Development Environments in relations to the project:

**Mike Premi,** *Co-Mentor*

The following persons are guidance mentors as a part of the senior software engineering project course for the students:

**Kevin McGrath,** *Instructor* **Kirsten Winters,** *Instructor* **Behnam Saeedi,** *Teaching Assistant*

# TABLE OF CONTENTS

# 1 INTRODUCTION

## 1.1 Scope

The software application described in this design document is to be utilized as a tool for the generative design of architectural structures. The goal of this software is not only to be used as a resource for the building of said structures, but also to be a tool for learning more about the limits of a standing structure.

## 1.2 Purpose

The purpose of this Software Specification Document (SSD) is to provide a detailed description of the interface for the iCreate software. Included is the layout and functionality of the user interface.

## 1.3 Intended Audience

This document is intended for the stakeholders and architectural designers who intend to use this software. In addition, this SSD will be used as a reference for the stakeholders and Capstone professors in the case of a differing opinion in regards to the requirements of the design and performance.

## 1.4 Definitions and Acronyms

- **VR** - An abbreviation of Virtual Reality which is described as, "the computer-generated simulation of a three-dimensional image or environment that can be interacted with in a seemingly real or physical way by a person using special electronic equipment, such as a helmet with a screen inside or gloves fitted with sensors." [1]
- **Virtual Space** - A 3D area in VR in which the user can maneuver around in and interact with objects.
- **Generative Design** - A form finding process that can mimic nature's evolutionary approach to design. [2]
- **GUI** - The graphical user interface allows the user to interact with the program via buttons or other types of graphical icons.
- **Primitive** - General 3D shape.
- **Diegetic** - Interface that is included in the game world – i.e., it can be seen and heard by the game characters.[3]

# 2 SYSTEM ARCHITECTURE OVERVIEW

The iCreate software program is a developmental tool that can create architectural structure designs using the generative design process. As the user will be working with delicate structures, the GUI of the iCreate software must be laid out in a way that is intuitive, efficient, and neat.

## 2.1 Design Viewpoints

### *2.1.1 Concerns of Design Stakeholders*

In regards to the developers, this software program should be simple and easily maintainable. The goal is to implement libraries and APIs that provide the functionality in this program to reduce the complexity of the overall codebase.

Users will be concerned about the overall look and feel of the software. The program's interface must be polished, neat, and intuitive, allowing for a seamless and comfortable experience. The goal is for the user to be able to comfortably and successfully design projects with minimal interruptions.

#### 2.1.2 Context

While developing the components of the iCreate software, the main consideration the developers had was the ease of use. As this program is meant for people with different types of technological backgrounds, the GUI must be easy to navigate.

#### 2.1.3 Interface

Since iCreate's user interface is meant for an intuitive experience, the developers have chosen to implement a design for the menu that is neat and robust at the same time, as well as comfortable for the user in terms of hand and eye strain.

#### 2.1.4 Structure

Each of the components are separate objects which means that each primitive is able to have its own physics and settings. This makes manipulation of several primitives easier, and the learning curve smaller.

## 3 COMPONENT DESIGN

### 3.1 VR Environment

The default scene for this software is an empty grey area. It was decided that an empty area would be best as it would reduce obstruction and distractions. Other scenes can have simple skyboxes with different floors to simulate a different scene.

Locomotion will allow users to move around within the virtual environment, so the team will implement the keypad of one of the controllers to provide the user with a method of locomotion.

3D objects can be instantiated in the virtual environment by the users themselves. The user can spawn 3D objects by selecting from a menu of available primitives, and can also create multiple instances of that 3D object.

### 3.2 User Interface

The interface will need to functionally allow the user to spawn a 3D object, then modify it via either a controller or gesture input. Additionally, the user will be able to save to a library for later use or load their creations to create complex structures. Finally, the application will display a way to transform the 3D objects by allowing the user to scale or resize the object.

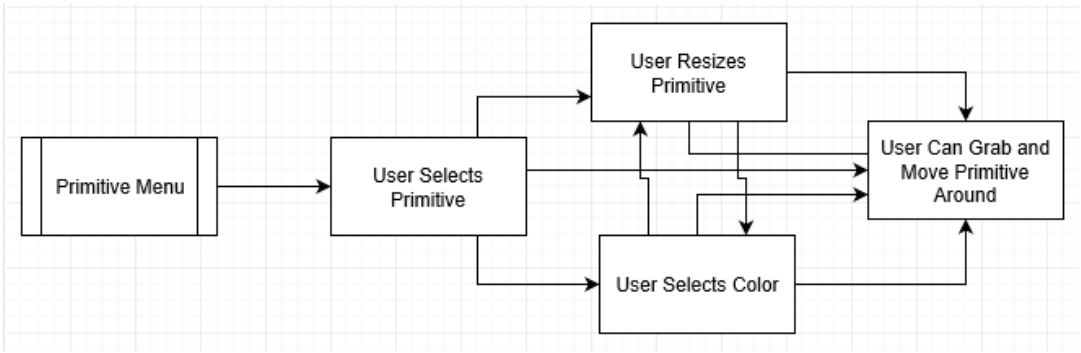



Figure 1. The flow of using the UI menu to spawn a shape.

The interface for iCreate will mostly be based on Diegetic and spatial UI. Diegetic UI will be used for accessing tools and options from a menu attached to the recessive hand's controller, while for all other menus and uses, spatial UI will be used to relay relevant navigation information to the user. The dominant hand will either be used to select options from the menus presented to them, or draw within the virtual Environment. The general flow of spawning a shape is shown in Figure 1.


Figure 2. The UI menu that features the primitive and color options as well as the size slider. This menu will hover over the controller as a diegetic UI component

There are three sections to the menu which can be seen in Figure 2. The first section has four primitives available to use: a square, circle, pyramid, and hexagon. The second part is a slider which can be used to adjust the size of the primitive spawned. Lastly, there are four color options for the primitives. As each primitive are separate objects, it is possible to have different combinations of sizes, colors, and shapes.

### 3.3 Load and Save

The save and load feature for iCreate can be realized through data serialization. To save the data of the scene, we will be using the FileStream and BinaryFormatter classes in Unity.

The FileStream class will be used to first create a save data file. Next, BinaryFormatter will be used to serialize the scene's data into the save file. Once the data is serialized, FileStream can be used to close the file we were writing into..

Finally, to load the save file, BinaryFormatter can be used once again to deserialize the data from the save file and load the scene. The save and load features can be implemented via two buttons on the in game menu, one for saving and one for loading.

### 3.4 Object Library

The Object library will also utilize Unity's serialization to save and load objects to and from a library of 3D objects created by the user. To save the object, BinaryFormatter can be used to save the attributes of the object, like position and rotation (x, y, z), material, etc, into a file or another GameObject. This file or GameObject will be the library.

Finally, to load the saved object from the library into the game world, we can deserialize from the file or GameObject. The user can access and utilize the Object Library from within their in game menu.

### 3.5 Curves

#### *3.5.1 Drawing a Curve*

The generation of a 3D within the program's virtual environment, will be created with the user drawing a 3D curve in their space given. The program offer the user a variety of curves to choose from in order to start the generation of their curve, being a list containing the selections of a bezier curve, b-spline curve, ellipse curve, circle curve, hyperbola curve, and parabola curve. Each of these selections are characterized in the code as an equation with the possibility of an additional process in order to such curves depending on its type.

After the user's selection of a type of curve, the system will offer a base curve once the user selects a starting position within their given environment. Once the user selects the start of the curve, the program will adjust the size and trajectory of the curve itself based off the user's physical movement of moving a controller around the environment from the starting position, generating the curve and sizing accordingly. The user's movements of the controller will provide a variety of inputs for the system given the equation selected by the user earlier. After a button indication from the user, the generated curve will be set in place by the system, giving the finality of that curve within the environment.

#### *3.5.2 Circle Curves, Ellipse Curves, Hyperbola Curves, Parabola Curves*

Circle curves, ellipse curves, hyperbola curves, and parabola curves have set alegbraic equations that can be written in code that represent these four types of code. The program will apply these equations along with the user's curve drawing process explained above.

#### *3.5.3 Bezier Curves*

For generating bezier curves within the program, the Handles.DrawBezier will be able to take in inputs such as the start position, end position, start tangent, and the end tangent of a bezier curve, which we are characteristics and inputs we are interested in creating in regards to the generation of such a curve. These inputs will be given by the user, from their selection of this curve type and then draw given these specifications within the environment. [4]

### *3.5.4 B-Spline Curves*

B-Spline Curves will inherit the same process when it comes to generating bezier curve within the program, but in addition with its slight difference of consisting multiple bezier curves generating a single b-spline curve. Due to a b-spline curve's notable characteristic of smoothness with multiple peaks and lows, such a curve can be generated with multiple bezier curves in order to be physically represented.

## 3.6 Transformation and Translation

The transformation feature will be primarily used while the shape is being instantiated. During the creation, the user can choose to alter the size of the shape via sliders on the diegetic menu attached to the user's controller.

The translation feature allows the shapes to be maneuvered about the environment. The shapes are not influenced by gravity, so they will float in place. The user is then able to grab the shapes and move them around the environment at will. It was decided to not have the shapes under the effects of gravity to aide in the design process to prevent the shapes from rolling around.

# 4 Class Design

This section will describe the classes and methods available in iCreate and how they are related to each other. There are six main classes: Main Menu, Load Project, New Project, Project, Brick, and Algorithm as shown in Figure 3 below.

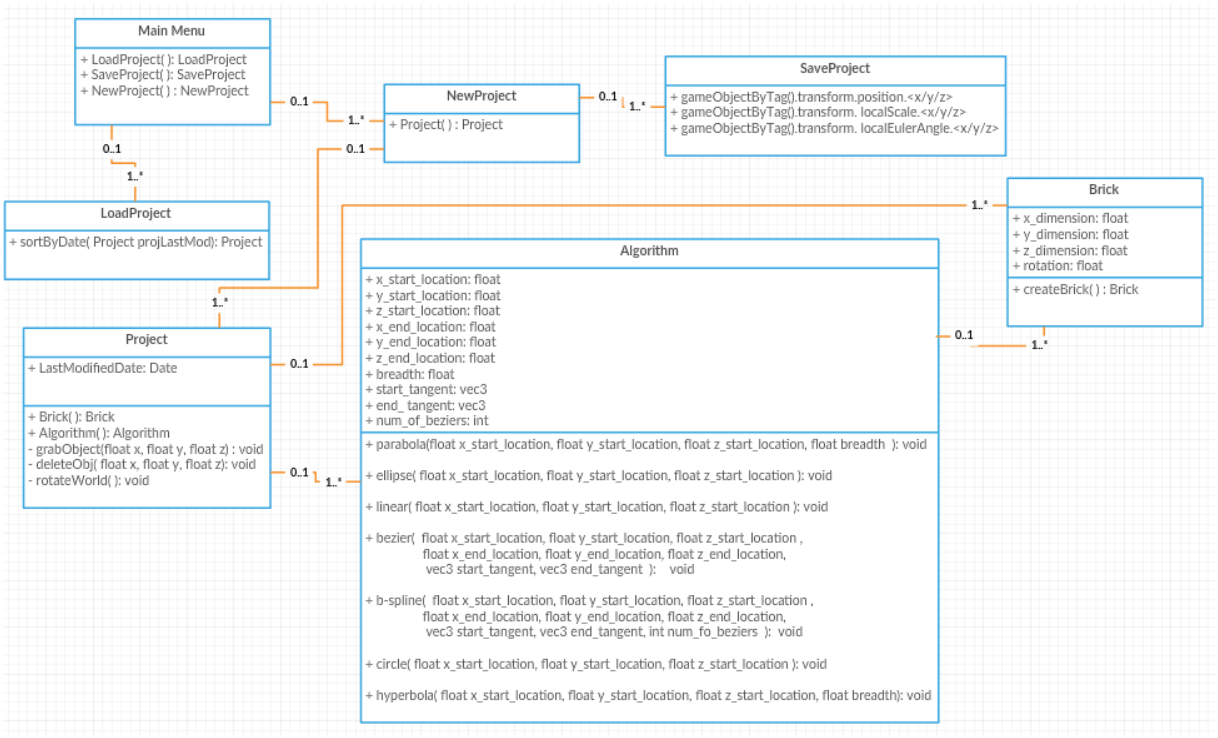



Figure 3. Class diagram that displays the methods available in iCreate, as well as the relationships between the different classes.

## 4.1 Main Menu

The main menu is the first screen that the user encounters when opening the iCreate program and has three functionalities: Load Project, New Project, and Exit. While the *Load( )* and *NewProject( )* selections are passed to the LoadProject and NewProject classes respectively which lead the user into the VR environment. The *Exit( )* selection exits the program entirely.

### *4.1.1 Available Functions:*

- *LoadProject( ) -*
  Calls the LoadProject class.
- *NewProject( ) -*
  Calls the NewProject class.
- *Exit( ) -*
  A public function that exits the program.

## 4.2 Load Project

When Load Project is selected from the main menu, a list of projects is presented which are sorted by the projects' last date of modification using the *sortByDate( )* function. This function cycles through each project and reads the Project's date variable *lastModified*. After a project is selected, that project is then loaded into the VR environment to resume development.

This load feature works by reading from the saved file created by the SaveProject( ) function. It reads each of the location, rotation, and scale values and creates a block with those specifications using Unity's *Instantiate( )* function.

### *4.2.1 Available Functions:*

- *sortByDate( Project projectLastMod ) -*
  A public function that is used by MainMenu's Load( ) to list the available projects. The parameter is a list of Projects, which will provide the last modification dates from calling *Project.lastModifiedDate*. Returns a Project.
- *Project( ) -*
  Calls the Project class.

## 4.3 New Project

The New Project class enters into a new VR environment with no additional parameters or function calls.

### *4.3.1 Available Functions:*

- *Project( ) -*
  Calls the Project class.

## 4.4 Save Project

The Save Project class allows the current project to be saved by reading in the positions, rotation, and scale of each block and writing them to a file. As each block is being created, they are added to a block list. Present in this save function is a loop that cycles through every block in the list of blocks and reading the values of each block. These values will then be transformed into a string which will be written to a file in a data structure type format.

#### 4.4.1 Available Functions:

- *gameObjectByTag().transform.position.<x/y/z>* -
  This function returns the x, y, and z coordinate location of a block.
- *gameObjectByTag().transform. localScale.<x/y/z>* -
  This function returns the x, y, and z coordinates for the scale of a block.
- *gameObjectByTag().transform. localEulerAngle.<x/y/z>* -
  This function returns the x, y, and z coordinates for the angle of the block.

### 4.5 Project

The Project class is the VR environment where the user is able to design architectural structures. This class is related to the Brick and Algorithm class which enable the ability to spawn blocks in a specified pattern. The user is able to grab and delete blocks using the functions *grabObj( )* and *deleteObj( )*.The function *rotateWorld( )* is used by the user to rotate the VR environment around for better viewing.

#### 4.5.1 Available Variables:

- *lastModifiedDate* -
  A public variable that returns the date of the last modification to that Project. This variable is a Date.

#### 4.5.2 Available Functions:

- *grabObj(float x, float y, float z )* -
  A private function that takes an x, y, z coordinate of the location of the object that is to be manipulated through the aid of a bounding box surrounding the head of the controller. Returns nothing.
- *deleteObj( )* -
  A private function that takes an x, y, z coordinate of the location of the object that is to be deleted. Returns nothing.
- *rotateWorld( )* -
  A private function that has no parameters. This function rotates the environment around the user. Returns nothing.

### 4.6 Brick

The brick class is used to instantiate the brick object, or primitive, and requires an x, y, and z dimension. The function *createBrick( )*, takes the x, y, and z dimension parameters and spawns a brick with the specified size at the controllers' current location.

#### 4.6.1 Available Variables:

- *x_dimension* -
  A public variable that specifies the x dimension of the brick. This variable is a float.
- *y_dimension* -
  A public variable that specifies the y dimension of the brick. This variable is a float.
- *z_dimension* -
  A public variable that specifies the z dimension of the brick. This variable is a float.

#### *4.6.2 Available Functions:*

- *createBrick( float x, float y, float z ) -*

  A public function that takes three parameters which specify the size of the brick. Returns a Brick object.

### 4.7 Algorithms

The Algorithms class provides the user with the ability to spawn multiple Bricks along a calculated trajectory. This class offers four path types: Linear, Bezier, Ellipse, and Parabola. Each path function requires an x, y, and z coordinate to specify the starting location of the trajectory.

#### *4.7.1 Available Variables:*

- ***x_start_loc*** *-*

  A public variable that specifies the x position of the starting point. This variable is a float.
- ***y_start_loc*** *-*

  A public variable that specifies the y position of the starting point. This variable is a float.
- ***z_start_loc*** *-*

  A public variable that specifies the z position of the starting point. This variable is a float.
- ***x_end_loc*** *-*

  A public variable that specifies the x position of the ending point of the given curve. This variable is a float.
- ***y_end_loc*** *-*

  A public variable that specifies the y position of the ending point of the given curve.. This variable is a float.
- ***z_end_loc*** *-*

  A public variable that specifies the z position of the ending point of the given curve. This variable is a float.
- ***breadth*** *-*

  A public variable that is calculated for the width of a trajectory based off the current angle of the controller and its given location.
- ***start_Tangent*** *-*

  A public variable for the beginning tangent for a specified curve.
- ***end_Tangent*** *-*

  A public variable for the beginning tangent for a specified curve.
- ***num_of_beziers*** *-*

  A public variable that specifies the number of bezier curves to use when creating a B-spline trajectory.

#### *4.7.2 Available Functions:*

- ***Parabola***( float **x_start_loc** , float **y_start_loc**, float **z_start_loc**, vec3 **breadth** ) -

  A public function that takes four parameters: three for the initial starting location of the trajectory, and one for the breadth of the curve. This function generates a parabolic curve using 30 control points that spread out a certain distance based off of the rotation of the controller - the higher degree of angle that the controller is in, the narrower the breadth of the curve is, and vise versa as seen in Figure 4.

- ***Ellipse***( float **x_start_loc** , float **y_start_loc**, float **z_start_loc** ) -

  A public function that takes three parameters which specify the beginning location of the ellipse trajectory. This function uses the click of the trigger button to begin a *LineRenderer* component which initializes a circular line with a radius of 10 units. This circle's width can then be adjusted by dragging the controller while the trigger button is pressed. This method is illustrated in Figure 6.

- ***Linear***( float **x_start_loc** , float **y_start_loc**, float **z_start_loc** ) -

  A public function that takes three parameters - x, y, and z coordinate points for the beginning location. Once the beginning point is specified, the function generates a mesh cylinder of length 10 using the *Mesh* component that represents the line. Calculations are done to change the center of rotation of the cylinder line from the center, to the origin point. This allows the line to be rotated around in a circle as seen in Figure 5.

- ***Bezier***( float **x_start_loc**, float **y_start_loc**, float **z_start_loc**,
  float **x_end_loc**, float **y_end_loc**, float **z_end_loc**,
  vec3 **start_tangent**, vec3 **end_tangent** ) -

  A public function that takes 8 parameters which specify the beginning and ending location of the Bezier trajectory along with their respective tangent vectors. This function uses the *DrawBezier* method of the *Handles* class that is built into Unity. The *DrawBezier* method requires the three coordinates of both the starting and ending locations, the two tangent vectors associated with each location, the color of the bezier, the texture, and the width. While the starting and ending locations will be passed into the function along with the tangent vectors, the color of the bezier curve will be set to red, the texture set to null, and the width set to the handle size * 0.1. This method is illustrated in Figure 7.

- ***B-Spline***( float **x_start_loc**, float **y_start_loc**, float **z_start_loc**,
  float **x_end_loc**, float **y_end_loc**, float **z_end_loc**,
  vec3 **start_tangent**, vec3 **end_tangent**, int **num_of_beziers** ) -

  This function utilizes the Bezier function to create a Bezier Spline (B-spline) trajectory. The function takes 9 parameters: 2 for the beginning of the bezier, 2 for the ending location of the bezier, 2 tangent vectors which act as the middle two control points, and the last parameter is the number of bezier curves used to create the B-spline curve. Using the same functional methods as the Bezier function, this function cycles through creating and connecting the specified number of bezier curves to form the B-spline trajectory. This method is illustrated in Figure 7.

- ***Circle***( float **x_start_loc** , float **y_start_loc**, float **z_start_loc** ) -

  A public function that takes three parameters which specify the beginning location of the ellipse trajectory. This function is similar to the Ellipse function which uses the click of the trigger button to begin a *LineRenderer* component that initializes a circular line with a radius of 10 units. This method is illustrated in Figure 6.

- ***Hyperbola***( float **x_start_loc** , float **y_start_loc**, float **z_start_loc**, vec3 **breadth** ) -

  A public function that takes three parameters which specify the beginning location of the ellipse trajectory. This function utilizes the Parabola function to create an initial parabolic curve. Once this curve has been instantiated, the function uses the current angle of the controller to tilt the hyperbolic curve. This method is illustrated in Figure 4.

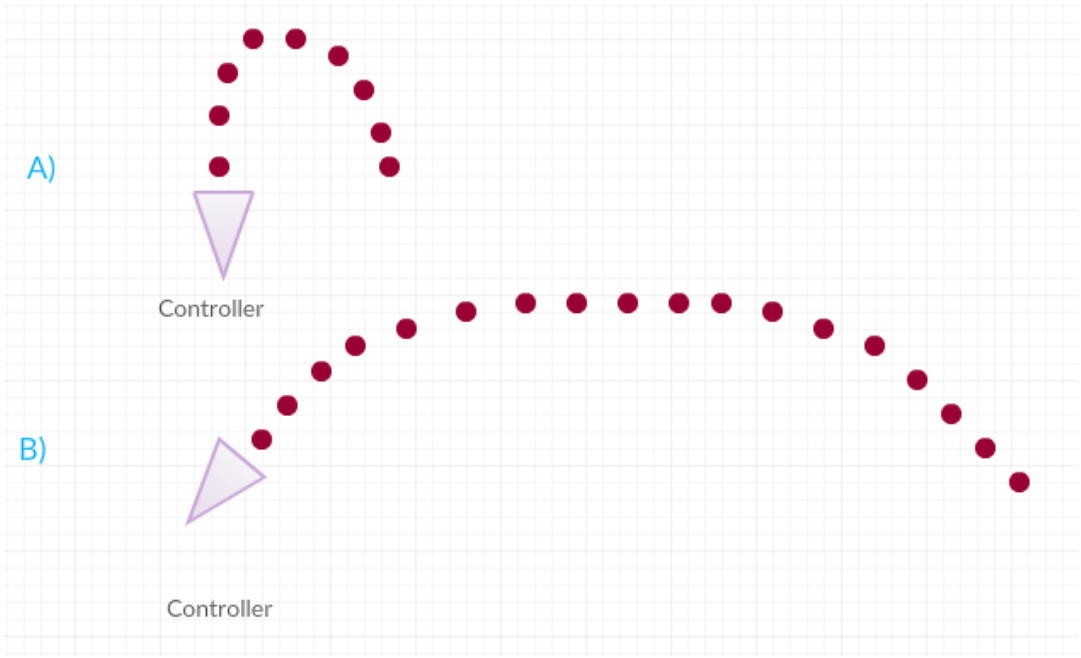



Figure 4. The width of the parabola created using the Parabola( ) function is determined by the angle of the controller as seen by the difference between A and B. The dots represent the units that make up the trajectory. In the program, there will be 30 units per parabola. This method is also used for the Hyperbola( ) function, though the difference is that the tilting the controller on it's side will cause the trajectory to be drawn on it's side as well.

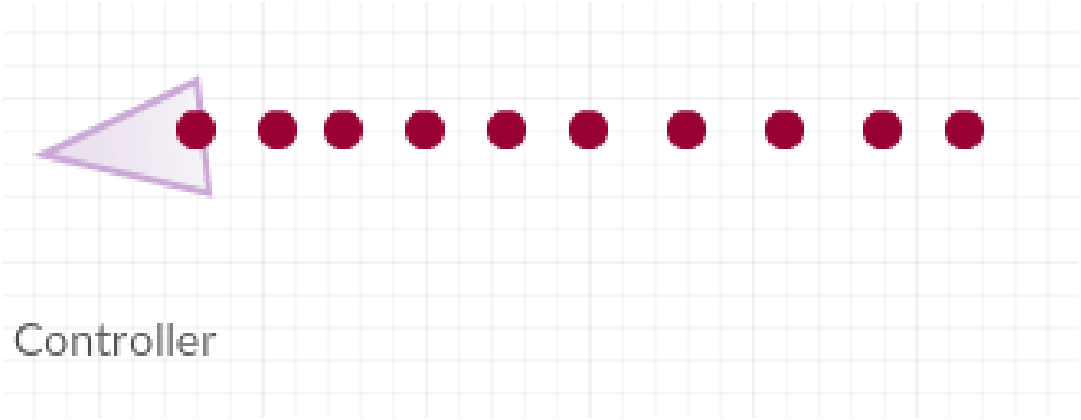



Figure 5. The Linear( ) function uses the head of the controller as the starting location for the linear trajectory. The trajectory is 10 units long.

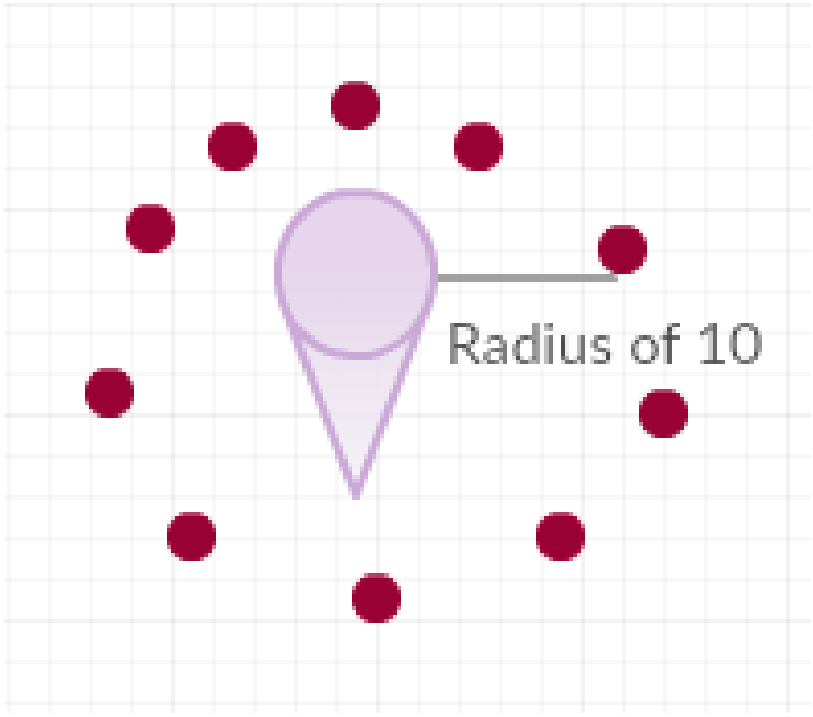



Figure 6. For both the Ellipse( ) and Circle( ) functions, the controller is set to the middle of the circle/ellipse while the trajectory is drawn around it. The circle/ellipse has a radius of 10, and depending on if it is a circle or ellipse being generated, the sides of the trajectory can be brought in to create a narrower path.


Figure 7. The Bezier( ) function uses the head of the controller as the first point, and creates the end point 10 units away. In between, the two control points are created. The B-Spline( ) function uses the Bezier( ) function multiple times to create the B-spline path. For each call to the Bezier( ) function that is made, the ending point becomes the new starting point, with a new ending point being created 10 units away.

# 5 State Design

This section describes the flow from state to state, as well as each choice presented at each step. Figure 8 shows the relationships between the various states and the destinations with the solid blue circle being the starting point, the squares representing the states, the orange circles representing the choices, and the blue dot with a ring being the end.

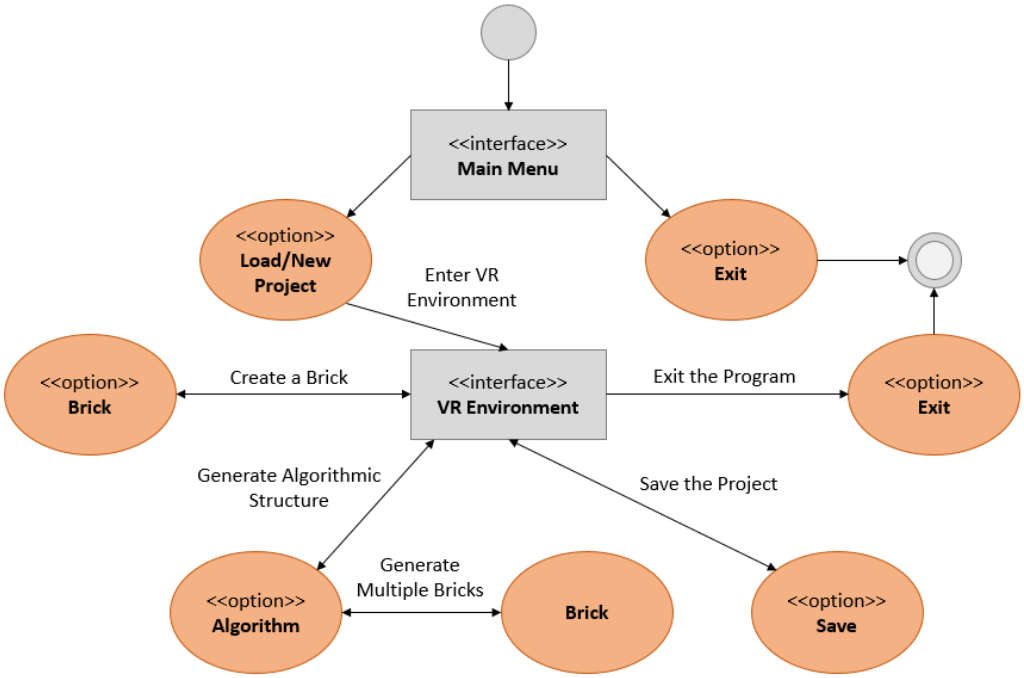



Figure 8. State diagram that discloses the various states and choices the user is presented with when using the iCreate program.

When beginning the program, the user is shown the Menu which has several options that have two different destinations. If the user selects the Exit, then they will leave the program entirely, whereas if the individual selects either Load or New Project, they will enter into the main VR Environment. From this state, the user has four options: Create a brick, Save the current project, use an Algorithm, and Exit. The Brick, Save, and Algorithm choices will redirect the user back to the VR Environment state after finishing the choices' requested task. The Algorithm option will call on the Brick method multiple times to spawn the required number of bricks.The Exit choice will result in leaving the program. It is possible to leave the program without saving the current build.

# List of Figures



## REFERENCES

[1] M. Webster (2018). *Definition of Virtual Reality* . [Online]. Available: https://www.merriam-webster.com/dictionary/virtual%20reality

[2] AutoDesk (2018). *What is Generative Design* . [Online]. Available: https://www.autodesk.com/solutions/generative-design

[3] Gamasutra (2018). *Game UI Discoveries: What Gamers Want* . [Online]. Available: https://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_.php?print=1

[4] Unity Technologies (2018). *Handles.DrawBezier* . [Online]. Available: https://docs.unity3d.com/ScriptReference/Handles.DrawBezier.html

### 3.2 Document Changes

The original version of the Design Document contained brief, but informative, information on the different features of what iCreate has to offer. The team later came back to add more detailed information to section 3, the Component Design, to include more specificities on the main features such as the load/save functionality, the object library, and the different selection of curves.

After review with our client, we added further detail to section 4, where we added explanations as to how the curves are mathematically generated, and diagrams showing how the final curves would appear in the program. In addition, a detailed class diagram was included to show the functions of each class, and how they all interact with each other.

## 4 Technology Review and Implementation Plan

### 4.1 Original Document

Continue reading onto the next page.

# Technology Review and Implementation Plan

Group 61: iCreate - Generative Design in Virtual Reality

Fall 2017 Term

Rhea Mae Edwards
Software Developer

Nabeel Shariff
Software Developer

Hannah Solorzano
Software Developer

**Abstract**

The purpose of this document is to represent further research on different aspects related to the creation of the senior capstone project generative design in virtual reality. Topics that are discussed within this paper are types of headsets, a variety of programming languages used, a handful of software that can be used along with a virtual reality program, type of developing environments, user interfaces, functionality of different types of APIs, GPUs, frames, and user controls in relation to the project. Each of these topics described consists of three different options and how they are common and also differ from one another. Out of the three options, one of the options has been chosen to be the initial choice of being implemented for the generative design project. Overall, this document identifies components of the problem the project consists of, identifies technologies for building solutions, researches alternatives, analyzes trade-offs using criteria, and works toward persuading its reader of the best choice based on its criteria.

# TABLE OF CONTENTS

# 1 INTRODUCTION

There are many tools that our team will have to use in order to fulfill all of the requirements for our senior capstone generative design virtual reality program project. A few of the tools that we will have to implement are a headset, a programming language to write in, and the possibility of including some kind of other software that can be used with our finalize virtual reality program in order to intensify such a virtual user experience. Other tools for our project includes a developing environment, the structure of the program's user interface, functionality of an application programming interfaces (APIs), graphics processing units (GPUs), the development of the program's user controls, and frames.

The generative design virtual reality program our team will be creating involves the construction of simple 3D structures initiated by the user and further development by the software program. Such a program involves the user placing simple 3D object in some environment, and then setting constraints for the program to then build some structure by using the simple 3D object. These tools of a headset, programming language, and connective software, will have to satisfy such a overall functionality and purpose.

The options that we have chosen to research for headsets are the following:

1) HTC Vive
2) Oculus Rift
3) PlayStation VR

The options that we have chosen to research for programming languages are the following:

1) C#
2) C/C++
3) Java

The options that we have chosen to research for connective software are the following:

1) Dynamo
2) Grasshopper - Iris VR
3) Leap Motion

The options that we have chosen to research for user interfaces are the following:

1) Non-Diegetic UI
2) Spatial UI
3) Diegetic UI

The options that we have chosen to research for the Operating Systems for development are the following:

1) Microsoft Windows
2) Apple Macintosh
3) Linux

The options that we have chosen to research for distribution methods are the following:

1) Steam Store
2) Oculus Home
3) Executable File

The options that we have chosen to research for GPUs are the following:

1) GeForce GTX 970
2) NVIDIA TITAN Xp
3) GeForce GTX 1060

The options that we have chosen to research for developing environments are the following:

1) Unity
2) Unreal Engine
3) OpenGL

The options that we have chosen to research for user controls are the following:

1) Oculus Rift Touch Controllers
2) Hands
3) Sony Controllers

This document will further discuss the options stated above, and which ones out of each of the topics we have chosen to initial use in our project implementation. Each section will describe each of the options individually, and then compare each of them in regards to our project requirements and preferred implementation choices.

# 2 HEADSETS

## 2.1 Overview

In order for our virtual reality program to be useful and actually be implemented for its full purpose, there will need to be a headset for us to download our program onto. A headset is the main piece of hardware that our team will need in order for us to properly test and use our virtual program with. Choosing a headset that will fulfill all of our needs and requirement is important, and highly sufficient enough for us for our project.

## 2.2 Criteria

Headsets are wore by individual users, allowing them to view computation altered reality with or through a screen, such as virtual and augmented reality. The price of a headset can greatly vary from just tens of dollars to a couple thousands of dollar depending on the complexity and usability of the device. Every headset has its own set of a variety of uses, where some headsets can be more relevant for some purposes and software program than others.

## 2.3 Potential Choices

### *2.3.1 HTC Vive*

The HTC Vive provides a decent space for its users to move within a room or in other words ”play area”. This headset allows a maximum of 15 feet for its user to move from its base-stations while in use. Such a play area does not have to be a perfect square to work effectively either. These boundaries built for the HTC Vive have been made to be fairly flexible. The HTC Vive also can detect objects and can possible even replace any obstacles within its play area, preventing any overlap or object confusion within its visuals. This specification also helps with the safety and real-life physical movements. [1]

A downside with the HTC Vive though, involves this idea of its use of effectively tracking its user. According to the Gadget review of 2015, ”Here’s EXACTLY How The HTC Vive Works,” there is a certain way the headset should be worn, begin ”mounted above head height, [...] faced down at a 30-45 degree angle.” [1] Such accuracy can be very particular in its use for a user to wear.

### *2.3.2 Oculus Rift*

One thing about the Oculus Rift, this headset has Windows 10 cross-device compatibility, where streaming through the device is a possibility when connected to a Xbox One. The Oculus Rift has similar head motion controls with its user when moving through a play area, being about a 3-foot by 3-foot square in any given space. This headset can also connect to a computer for further use allowing future installs and upgrades for its software. [2]

Also, the game engines Unity and Unreal Engine are supported by the Oculus Rift.

### *2.3.3 PlayStation VR*

The PlayStation VR is a fairly new headset which has not been out on the market for barely even a year. The PlayStation VR is highly compatible with game consoles, begin a true console-based user experience. This headset also provides ”HDR (high dynamic range) - compatible TVs” meaning its support with HDR - compatible games. Also for a consumer’s perspective on the headset, the price one pays for the device is the best value provide for how much it costs, which is high consumer’s benefit. [3]

## 2.4 Discussion

| | Virtual Reality Capability | Abundant Available Software Resources for New Users | ”Large” Play Area |
|---|---|---|---|
| HTC Vive | ✓ | ✓ | ✓ |
| Oculus Rift | ✓ | | ✓ |
| PlayStation VR | ✓ | | |

Comparison of Headset Technology Options

## 2.5 Conclusion

We chose the HTC Vive headset setup because its play area specifics and available resources we have through the internet and personal user experiences in using the device. With our virtual reality program in relation with generative design, using the HTC Vive provides a great amount of physical space for a user to move and design their elements within the program, and hardware specifics that allow a user to get a maximum virtual reality experience.

| | Virtual Reality Capability | Abundant Available Software Resources for New Users | ”Large” Play Area |
|---|---|---|---|
| HTC Vive | ✓ | ✓ | ✓ |

HTC Vive Headset Choice

# 3 Programming Languages

## 3.1 Overview

Before writing any code, we need to figure out what programming language we will want to write in. Basis of the basics.

## 3.2 Criteria

Individually, programming languages were made with there in own purpose with it comes to coding. There are programming languages that are more appropriate to write in when it comes to coding for a virtual reality program than others. Each language can also provide its own set of unique functionality. Also, when it comes to what a developer is programming, there are more sources out there that can help with such programming problems depending on the language one uses.

## 3.3 Potential Choices

### *3.3.1 C#*

A note from C# developers is their love with the language of C# ”for begin pleasant to use and well-designed”. C# was originally designed by Microsoft for developing apps. The language is also highly recommended to be used for creating games through the Unity game engine being easy to start with, which is relevant to our team’s over project, in relation to writing a virtual reality program. C# is a high level language, where there is a focus more on programming than the little details of the language itself, which can be slightly annoy for using other programming languages out there. [4]

On the scalability side, C# is easy to maintain and fast for being a statically typed language. Also, there are many resources and examples written in C# for developers to reference and use. [4]

### *3.3.2 C/C++*

C++ is a language that allows its programmer to have a lot of power and control over its computers resources through the act of coding. With sufficient knowledge in using the language, its programs can be created cheaply with high speeds and ability to run compare to other programming languages. C++ also allows its programmers ”to develop game engines, games, and desktop apps. Many AAA title video games are built with C++.” [5]

A downside of coding in C++, is its beginner friendliness. C++ is a lower level language that can be very complex, especially in regards to brand-new programmers coding a virtual reality program such as every member in our team. Such a complex with the language, also makes it a little difficult to maintain in general.

### *3.3.3 Java*

According to Codementor, ”this general-purpose language [of Java] was designed to be easier to use than C++.” Companies have used Java to develop desktop apps and website backend systems. Being more related with virtual reality programs, Java is highly used in developing Android apps. Java is beginner friendly and can be fairly easy to use to optimize performance within a written program, but can be a frustrating the use from the start, due to its great lengths in code when writing. [6]

### 3.4 Discussion

| | Able to Use for Virtual Reality Programs | Simple to Learn for Beginners | Many Available Resources |
|---|---|---|---|
| C# | ✓ | ✓ | ✓ |
| C/C++ | ✓ | | ✓ |
| Java | ✓ | | ✓ |

Comparison of Programming Language Technology Options

### 3.5 Conclusion

We chose to use C# as our main programming language because mainly with our initial chosen game engine, Unity, for our group to code in, C# is the more widely use language to code in. Many of the tutorials and examples when coding in Unity are in C# , along with C# begin the most often recommended programming language when creating games through the Unity game engine. [4]

| | Able to Use for Virtual Reality Programs | Simple to Learn for Beginners | Many Available Resources |
|---|---|---|---|
| C# | ✓ | ✓ | ✓ |

C# Programming Language Choice

# 4 CONNECTIVE SOFTWARE

## 4.1 Overview

Suggested by our client, was a handful of separate software and development tools that we as a team, can think of using with our program once we get our basic implementation working for our main project. The idea behind these additions, is to further our basic idea of a program and provide a more diverse user experience with our generative design virtual program. There are hopes also that adding a single or multiple additional software and tools will provide a more realistic experience for the program's users overall.

## 4.2 Criteria

There are only certain types of software that will be able to connect with a virtual headset, and especially depending on the one our team decides on using. Along with the additional software's ability to integrate, the software's purpose is to enhance and benefit the main program in some way. Such enhancement can be done by the addition of useful features or accessibility and functionality of the program overall.

## 4.3 Potential Choices

### *4.3.1 Dynamo*

Dynamo is a software that is useful to implement in regards to "easing the modeling, visualization, and analysis tools" for most connected devices. In relation to our generative designing virtual reality project, Dynamo can help users "generate sophisticated designs from simple data, logic, and analysis" due to its structural work-flow in the software along with is such implementation. [7]

### *4.3.2 Grasshopper - Iris VR*

Grasshopper - Iris VR helps provide exported "surfaces, meshes, and breps" that can be found very useful in our generative design virtual reality program. This software can also help apply material and color to a structure or object in a given program. In regards in providing a greater designing ability with generative design virtual reality program for our users, this addition can be helpful in providing such capabilities. [8]

### *4.3.3 Leap Motion*

Through the cameras of a headset and infrared light, Leap Motion can help integrate a user's real hand motions within a virtual reality program. Leap Motions is rated to be fairly simple to use and implement by a developer, which is a great benefit for beginner programmers wanting to just simply create a highly realistic program through virtual reality in a short amount of given time. [9]

## 4.4 Discussion

| | Capable with the HTC Vive | Movement Relevant | Visually Relevant |
|---|---|---|---|
| Dynamo | ✓ | | ✓ |
| Grasshopper - Iris VR | ✓ | | ✓ |
| Leap Motion | ✓ | ✓ | |

Comparison of Connective Software Technology Options

### 4.5 Conclusion

We chose Leap Motion as our initial main software to connect our main virtual program to because it allows its users to use their bare hands when maneuvering through a virtual reality program. Allowing this type of availability for our user with generative designing will be a very convenient addition, and allow our users to experience a more realistic building of a virtual physical structural design with their own hands in virtual reality.

| | Capable with the HTC Vive | Movement Relevant | Visually Relevant |
|---|---|---|---|
| Leap Motion | ✓ | ✓ | |

Leap Motion Connective Software Choice

# 5 User Interface

## 5.1 Overview

A user interface (UI) is the way the user interacts with the software system. Virtual Reality (VR) has allowed designers and developers to break the limits of UI and literally think outside the box. Since the team will be using the Unity game engine [10], the interface will depend on the UI options available through Unity.

## 5.2 Criteria

The criteria for choosing the appropriate UI is based on maximizing the quality and comfort of the user's experience. The user should be able to have easy access to essential tools through an intuitive UI. The UI should also work seamlessly for both controller and gesture output, and allow the user to maximize their efficiency when designing 3D models.

## 5.3 Potential Choices

### *5.3.1 Non-Diegetic UI*

A non-diegetic UI is an UI that is overlaid on top of a screen, usually in the form of a heads-up display (HUD) [11]. It doesn't exist within an environment, but makes sense for the user in the context of viewing the application. Examples of non-diegetic UI include health bars and scores.

### *5.3.2 Spatial UI*

Spatial UI is meant to place the UI within the environment itself, being attached to the virtual camera of the headset, allowing the user's eyes to focus on the UI [11]. An example of spatial UI is a virtual notification appearing in front of the user in VR.

### *5.3.3 Diegetic UI*

Diegetic UI provides a way for elements within the environment to display information to the user. This type of UI can be attached to elements within the virtual environment. A virtual holographic display on the user's controller is an example of diegetic UI.

## 5.4 Discussion

Non-diegetic UI is a very common UI and is probably the best option for non-VR applications. However, iCreate is a VR application, so the non-diegetic UI will be too close for the user to focus on with their eyes, causing great discomfort to the user. In contrast, spatial and diegetic UI offer an alternative that allows the user to comfortably interact with their VR environment without feeling discomfort, i.e. nausea.

Spatial UI will allow the program to relay information directly in front of the user. This type of UI is useful for showing the user information, but isn't as useful for receiving input from the user. In contrast, diegetic UI allows not only a way to display information to the user, but also receive input from the user. Compared to spatial UI, diegetic UI is attached to the environment. This means the user can directly interact with almost any element within the virtual environment if it has an UI attached.

| | VR Applicable | Comfortable Usage for User | Allows Wide Range of Input from User |
|---|---|---|---|
| Non-Diegetic UI | | | ✓ |
| Spatial UI | ✓ | ✓ | |
| Diegetic UI | ✓ | ✓ | ✓ |

Comparison of User Interface Technology Options

### 5.5 Conclusion

The team has chosen diegetic UI as the main UI for the iCreate VR application. Diegetic UI can be used to attach small menus with sliders and buttons on the user's controllers, allowing the user to comfortably have their tools within arm's reach. Additionally, to relay notifications directly to the user, the team may use spatial UI, but the main UI will still be diegetic.

| | VR Applicable | Comfortable Usage for User | Allows Wide Range of Input from User |
|---|---|---|---|
| Diegetic UI | ✓ | ✓ | ✓ |

Diegetic User Interface Choice

# 6 Operating System

## 6.1 Overview

Apart from the VR headset's propriety operating system (OS), the VR application also needs to run on the OS that is running on the computer itself. The OS manages the resources needed by the headset and VR software to interact with each other.

## 6.2 Criteria

The appropriate OS chosen for iCreate's development should give the team access to the chosen software tools and should be highly compatible with those tools. Additionally, the OS shouldn't hinder the performance of the VR application, and should also be compatible with the headset and its proprietary software.

## 6.3 Potential Choices

### *6.3.1 Microsoft Windows*

Windows [12] is the most commonly used OS, and offers great compatibility with Unity, the HTC Vive headset, and VR applications. Additionally, there is a plethora of documentation related to the chosen software tools, facilitating the team's development.

### *6.3.2 Apple Macintosh*

Macintosh [13] is also widely used around the world, but is not as open as some of the other operating systems. It doesn't have great compatibility with VR applications, and only recently supported the HTC Vive with the latest Macintosh OS update [14]. It does however support Unity very well, and is also compatible with several design applications that can help the team with 3D development. Moreover, documentation for Unity and other software is present for Macintosh as well.

### *6.3.3 Linux*

Linux [15] offers the team the most freedom when it comes to programming. Moreover, Linux is also widely used and is open source. However, although Linux supports the HTC Vive, it does not offer stability and compatibility with Unity. Additionally, Unity is only available experimentally for Linux [16].

## 6.4 Discussion

Macintosh offers a great design experience for the development team, and Linux offers a more direct approach for development. However, both Macintosh OS and Linux are lacking when it comes to compatibility and stability of the software tools chosen by the team . In contrast, Windows is highly compatible with Unity and HTC Vive, and both are stable on Windows. Additionally, most of the team's experience with developing in Windows is much greater compared to the experience with developing on the other operating systems. Moreover, there is plenty of documentation for Windows about Unity, Leap Motion, and the HTC Vive headset, making Windows the most effective and efficient option.

| | Quality of Design Experience | Compatible and Stable | Comfortability and Available Resources in Usage |
|---|---|---|---|
| Microsoft Windows | | ✓ | ✓ |
| Apple Macintosh | ✓ | | |
| Linux | ✓ | | |

Comparison of Operating System Technology Options

### 6.5 Conclusion

The team has decided to develop the iCreate VR application on the Microsoft Windows operating system because it offers the most compatibly and support for the software tools chosen by the team, and also allows for the most efficient and effective development with a great deal of documentation available for Windows. Additionally, Windows is also the most commonly used OS for VR application use, and thus will give the team access to the widest share of users [17].

| | Quality of Design Experience | Compatible and Stable | Comfortability and Available Resources in Usage |
|---|---|---|---|
| Microsoft Windows | | ✓ | ✓ |

Microsoft Windows Operating System Choice

# 7 Distribution Method

## 7.1 Overview

Once the VR application is developed, it has to find its way to the user via some channel of distribution. A distribution method is the way the user can access and install the VR application on their system.

## 7.2 Criteria

The criteria for choosing the appropriate distribution method has to take into account the simplest and most convenient way for the user to access and install the iCreate VR application. It should also be compatible for the chosen operating system, hardware, and software chosen for development.

## 7.3 Potential Choices

### *7.3.1 Steam Store*

The Steam Store [18] is most compatible with the HTC Vive. It also has a large number of users. Additionally, it supports the use of Oculus Rift for VR applications. Furthermore, Steam makes it slightly easier for users to access the VR application since the users on the Steam Store know exactly what their looking for. However, it also requires the use of a proprietary development kit, and involves a $100 fee to join the program [19].

### *7.3.2 Oculus Home*

Oculus Home is similar to the Steam Store, but is exclusive to Oculus products, and does not support the HTC Vive [19]. This also limits the user base and development options. Although there is no fee, a developer account is required to publish on Oculus Home [10].

### *7.3.3 Executable File*

Distributing the VR application as a standalone executable program provides the most freedom and direct access to the user. However, since this distribution method does not involve a 3rd party store, it is the least secure method of distribution. Additionally, the team will have to find a way to make the iCreate application available for download for users.

## 7.4 Discussion

Since the team is using the HTC Vive as the main hardware to develop for, Oculus Home does not provide any benefit as it is only available for Oculus products. In contrast, both the Steam Store and executable file distribution methods make iCreate accessible for HTC Vive users, as well as for other headsets including the Oculus Rift. However, compared to the Steam Store option, the executable file distribution is free and gives the development team the most freedom and control for distribution. Moreover, the executable file can be made available for download directly from the iCreate website, so the team will not be limited to using a proprietary development kit and will not be required to pay any fees either.

| | Usable with the HTC Vive Headset | Cost Effective | Simplicity in Usage |
|---|---|---|---|
| Steam Store | ✓ | | |
| Oculus Home | | ✓ | |
| Executable File | ✓ | ✓ | ✓ |

Comparison of Distribution Method Technology Options

### 7.5 Conclusion

The team has decided to distribute iCreate via an executable file, available for download directly from the iCreate website. For now, the VR app will be distributed for free, so there is no need to go through a 3rd party channel at the moment. In the future, should the team decide to distribute iCreate more aggressively and for profit, the Steam Store will serve as a good backup option.

| | Usable with the HTC Vive Headset | Cost Effective | Simplicity in Usage |
|---|---|---|---|
| Executable File | ✓ | ✓ | ✓ |

Executable File Distribution Method Choice

# 8 Graphics Card

## 8.1 Overview

The graphics card, or graphics processing unit (GPU), in the computer that is running the program is important as the GPU is what allows the computer to render and clearly display graphics. Without the proper GPU, the user will not be able to use iCreate.

## 8.2 Criteria

The GPU is required to effortlessly render the constantly updating program. As such, the minimum requirements are at least a GeForce GTX 970 [20] or an AMD Radeon R9 290 [21]. These GPU's are not very powerful in terms of their rendering power, but they are still capable of running a VR program. In addition to a strong GPU, it should also be noted that the computer needs at least a i5 core central processing unit (CPU), 8GB of RAM, and a 1.3 HDMI port.

## 8.3 Potential Choices

### *8.3.1 GeForce GTX 970*

This graphics card has the minimum powerlevel required to run VR programs. It is affordable and easy to access.

### *8.3.2 NVIDIA TITAN Xp*

NVIDIA claims that the NVIDIA TITAN Xp [22] is the strongest GPU currently on the market. It has more rendering power and produces highly detailed displays. It is not as affordable as the GeForce GTX 970, as it retails for $1200.

### *8.3.3 GeForce GTX 1060*

The GeForce GTX 1060 is a powerful graphics card that is very capable of performing great visual displays and redering power. This card is highly accessable and very affordable.

## 8.4 Discussion

The team considered the budget for the project when considering which GPU to use. As the GeForce GTX 1060 is both affordable and accessible, the team went with this option. The NVIDIA TITAN Xp is clearly not an option as the budget is not large enough, and the GeForce GTX 970 is affordable, but it's rendering power is not strong enough.

| | Affordability | Accessibility | Rendering Power |
|---|---|---|---|
| GeForce GTX 970 | ✓ | ✓ | |
| NVIDIA TITAN Xp | | | ✓ |
| GeForce GTX 1060 | ✓ | ✓ | ✓ |

Comparison of Graphics Card Technology Options

## 8.5 Conclusion

In the end, the team decided to use the GeForce GTX 1060 as we already possess this GPU and we do not have the budget to acquire new graphics cards.

| | Affordability | Accessibility | Rendering Power |
|---|---|---|---|
| GeForce GTX 1060 | ✓ | ✓ | ✓ |

GeForce GTX 1060 Graphic Card Choice

# 9 Development Environment

## 9.1 Overview

The libraries that the chosen game engine should have must fully be able to complete a given task. This way, the team does not have to spend time reinventing the wheel and creating our own methods while designing the program's functionality.

## 9.2 Criteria

The game engine that will be used in the development of this project is required to have an easy learning curve,as well as the capability to render detailed graphics using a series of libraries.

## 9.3 Potential Choices

### *9.3.1 Unity*

Unity [24], a game engine known for it's easy learning curve offers several tools for programming simulations and video games for platforms, computers, and mobile phones. It utilizes the C# programming language whose classes and object oriented items allow for the easier creation of 3D items and interface. Unity also offers an abundance of libraries and tutorials for virtual reality as there is a large user base who share their 3D objects and libraries so that others may use those items in their own project.

### *9.3.2 Unreal Engine*

Unreal Engine [25] is known for it's highly detailed graphics and rendering capabilities. It is written in C++, another object oriented language, which allows for game objects to have more detailed specifications. This engine is much more difficult to learn, and there is a smaller casual user base as Unreal has only gone free to use for a short time. As such, there are less tutorials and user written libraries available for use.

### *9.3.3 OpenGL*

Unlike the other two choices, OpenGL [26] is not a game engine. Instead, it is a application programming interface (API) that specializes in communicating directly with the graphics processing unit (GPU). This is advantageous because it allows for more customization and control over how an object renders. This API is written in C, an object-oriented language, which uses classes to give objects more customization options such as position, color by pixel, shininess/reflectiveness, and smoothness. There are few libraries that add additional features, and customizing or creating our own library is a daunting task.

## 9.4 Discussion

The team considered the benefits of each game engine and thought about how important the number of libraries present will be once we begin to design the program. It was decided that it is important for the engine to have a large user base as this means that there are more tutorials, forums, books, or people to consult if we run into a problem or are struggling to figure out how to design a certain feature.

| | Large User Base | Relevant User Libraries | Ease of Learning Usage of Environment |
|---|---|---|---|
| Unity | ✓ | ✓ | ✓ |
| Unreal Engine | ✓ | ✓ | |
| OpenGL | ✓ | ✓ | |

Comparison of Development Environment Technology Options

### 9.5 Conclusion

In the end, it was decided that Unity would be the best tool for us to design the iCreate program on. With its large user base, numerous libraries available, and easy learning curve, this tool would be the easiest engine to use for development. Compared to the smaller user bases of Unreal, or the more complicated library structure of OpenGL.

| | Large User Base | Relevant User Libraries | Ease of Learning Usage of Environment |
|---|---|---|---|
| Unity | ✓ | ✓ | ✓ |

Unity Development Environment Choice

# 10 User Controls

## 10.1 Overview

This technology is how the user controls how they manipulate the 3D objects in the virtual space. Since this requires an extreme amount of precision, the technology that we end up going with needs to have pinpoint accuracy.

## 10.2 Criteria

Due to the nature of this program, the user needs to be able to grab and drag the points of an object, as well as sketch precise trajectory paths.

## 10.3 Potential Choices

### *10.3.1 Oculus Rift Touch Controllers*

The Oculus Rift Touch controllers [27] feature a small controller that fits in the palm of your hand, along with a piece that wraps around the entirety of your hand. It is designed to add touch sensitivity to increase the VR experience.

### *10.3.2 Hands*

This form of a controller is very basic as there are no physical controllers required. Certain types of VR programs use the headset to identify where the fingers of the hand are positioned, and allows the user to pinch, drag, and drop items with hand gestures.

### *10.3.3 Sony Controllers*

The Sony controllers [28] are small analog type controllers that feature an analog stick and arrow keys. This controller is widely available, but is not as precise in detecting movement.

## 10.4 Discussion

The team wanted a controller that would maximize the level of preciseness that the user experiences. This is important because the preciseness of movements is what allows the user to create intricate structures. Affordability of the controller was also considered.

| | Controls' Preciseness | Affordability |
|---|---|---|
| Oculus Rift Touch Controllers | ✓ | ✓ |
| Hands | ✓ | ✓ |
| Sony Controllers | | ✓ |

Comparison of User Controls Technology Options

## 10.5 Conclusion

In the end, the team decided that we would implement the iCreate program to incorporate hand gestures. We found this to be the most precise of all the types of controllers, and there is no additional costs to including them. In addition, being able to implement the Oculus Rift Touch Controllers would be a valid option also depending on how the overall program implementation progresses during development.

| | Controls' Preciseness | Affordability |
|---|---|---|
| Hands | ✓ | ✓ |

Hands User Controls Choice (Main)

| | Controls' Preciseness | Affordability |
|---|---|---|
| Oculus Rift Touch Controllers | ✓ | ✓ |

Oculus Rift Touch Controllers User Controls Choice (Alternate)

## 11 CONCLUSION

In conclusion, the iCreate development team will be using components that are most useful and beneficial to both the users and the developers. The team has chosen to develop for the HTC Vive using the Unity 3D game engine, utilizing programs written in C#. Additionally, the team will be using Leap Motion to implement the user's hands as one of the primary methods of input, and the user interface will mostly consist of diegetic UI. Moreover, the team will be using an Nvidia GeForce GTX 970 (or better) video card as the primary dedicated graphics card. Finally, the iCreate application will be developed for the Windows operating system, and to be distributed as an executable file via a download page on the iCreate website.

Table summaries of all of the technology options chosen:

| | Virtual Reality Capability | Abundant Available Software Resources for New Users | "Large" Play Area |
|---|---|---|---|
| HTC Vive | ✓ | ✓ | ✓ |

HTC Vive Headset Choice

| | Able to Use for Virtual Reality Programs | Simple to Learn for Beginners | Many Available Resources |
|---|---|---|---|
| C# | ✓ | ✓ | ✓ |

C# Programming Language Choice

| | Capable with the HTC Vive | Movement Relevant | Visually Relevant |
|---|---|---|---|
| Leap Motion | ✓ | ✓ | |

Leap Motion Connective Software Choice

| | VR Applicable | Comfortable Usage for User | Allows Wide Range of Input from User |
|---|---|---|---|
| Diegetic UI | ✓ | ✓ | ✓ |

Diegetic User Interface Choice

| | Quality of Design Experience | Compatible and Stable | Comfortability and Available Resources in Usage |
|---|---|---|---|
| Microsoft Windows | | ✓ | ✓ |

Microsoft Windows Operating System Choice

| | Usable with the HTC Vive Headset | Cost Effective | Simplicity in Usage |
|---|---|---|---|
| Executable File | ✓ | ✓ | ✓ |

Executable File Distribution Method Choice

| | Affordability | Accessibility | Rendering Power |
|---|---|---|---|
| GeForce GTX 1060 | ✓ | ✓ | ✓ |

GeForce GTX 1060 Graphic Card Choice

| | Large User Base | Relevant User Libraries | Ease of Learning Usage of Environment |
|---|---|---|---|
| Unity | ✓ | ✓ | ✓ |

Unity Development Environment Choice

| | Controls' Preciseness | Affordability |
|---|---|---|
| Hands | ✓ | ✓ |

Hands User Controls Choice (Main)

| | Controls' Preciseness | Affordability |
|---|---|---|
| Oculus Rift Touch Controllers | ✓ | ✓ |

Oculus Rift Touch Controllers User Controls Choice (Alternate)

## REFERENCES


[1] D. Peppiatt. (2015). *Here's EXACTLY How The HTC Vive Works* . [Online]. Available: https://www.gadgetdaily.xyz/raspberry-pi-3-is-on-sale-now/

[2] M. Andronico, S. Smith. (2016). *What is the Oculus Rift?* . [Online]. Available: https://www.tomsguide.com/us/what-is-oculus-rift,news-18026.html

[3] W. Fulton. (2017). *PlayStation VR (2017) review* . [Online]. Available: https://www.digitaltrends.com/vr-headset-reviews/playstation-vr-2017-review/

[4] Codementor. (2016). *Why Learn C# ?* . [Online]. Available: http://www.bestprogramminglanguagefor.me/why-learn-c-sharp

[5] Codementor. (2016). *Why Learn C++?* . [Online]. Available: http://www.bestprogramminglanguagefor.me/why-learn-c-plus-plus

[6] Codementor. (2016). *Why Learn Java?* . [Online]. Available: http://www.bestprogramminglanguagefor.me/why-learn-java

[7] J. Dellel. (2017). *Dynamo: Autodesk's answer to Computational Design* . [Online]. Available: https://enterprisehub.autodesk.com/articles/dynamo-autodesk-s-answer-to-the-computational-design

[8] IrisVR. *Grasshopper* . [Online]. Available: https://help.irisvr.com/hc/en-us/articles/220686068-Grasshopper

[9] A. Colgan. (2014). *How Does the Leap Motion Controller Work?* . [Online]. Available: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/

[10] Unity Technologies. (2017). *Unity-Game Engine* . [Online]. Available: https://unity3d.com/

[11] Unity Technologies. (2017). *Unity-User Interfaces for VR* . [Online]. Available: https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr

[12] Microsoft. (2017). *"Windows — Official Site* . [Online]. Available: https://www.microsoft.com/en-us/windows/

[13] Apple. (2017). *macOS High Siera - Apple* . [Online]. Available: https://www.apple.com/macos/high-sierra/

[14] L. Painter. (2017). *How to use VR on a Mac* . [Online]. Available: https://www.macworld.co.uk/how-to/mac/how-use-vr-on-mac-3640213/

[15] Linux. (2017). *Linux.org* . [Online]. Available: https://www.linux.org/

[16] N. Bard. (2015). *The State of Unity on Linux* . [Online]. Available: https://blogs.unity3d.com/2015/07/01/the-state-of-unity-on-linux/

[17] Steam. (2017) *Steam Hardware & Software Survey: October 2017* . [Online]. Available: Available:http://store.steampowered.com/hwsurvey/

[18] Steam. (2017). *Welcome to Steam* . [Online]. Available: http://store.steampowered.com/

[19] T. Andrade. (2016). *VR Games & Apps-Where to Sell?* . [Online]. Available: https://virtualrealitypop.com/vr-games-apps-where-to-sell-2127697bb70c

[20] Nvidia. (2017). *GeForceGTX970* . [Online]. Available: https://www.geforce.com/hardware/desktopgpus/geforce-gtx-970

[21] AMD. (2017). *AMD RadeonTM R9 Series Gaming Graphics Cards with High-Bandwidth Memory.* . [Online]. Available: http://www.amd.com/en-us/products/graphics/desktop/r9

[22] Nvidia. (2016). *NVIDIA TITAN Xp Graphics Card with Pascal Architecture.* . [Online]. Available: https://www.nvidia.com/en-us/design-visualization/products/titan-xp/

[23] AKiTiO. (2017). *AKiTiO Node — Thunderbolt$^{TM}$ 3 eGXF expansion chassis for eGPUs.* . [Online]. Available: www.akitio.com/expansion/node.

[24] Unity. (2017). *Game Engine* . [Online]. Available: unity3d.com/

[25] Unreal Engine. (2017). *Game Engine* . [Online]. Available: unrealengine.com/en-US/what-is-unreal-engine-4.com/

[26] Group, K. (2017). *The Industry's Foundation for High Performance Graphics* . [Online]. Available: https://www.opengl.org/

[27] Pino,N. (2017). *Oculus Touch review* . [Online]. Available: http://www.techradar.com/reviews/oculustouch-controller

[28] PlayStationMove. (2017). *PlayStation Move Accessories* . [Online]. Available: https://www.playstation.com/en-us/explore/accessories/vraccessories/playstation-move/

### 4.2 Document Changes

No changes have been made to the iCreate Technology Review and Implementation Plan document, which is still current valid at the document's current and original version.

### 4.3 Implementation Changes

There have been a few discrepancies between the Technology Review and the actual implementation. While most of the components have remained consistent, such as the utilization of the HTC Vive, C#, Microsoft Windows, GeForce GTX 1060, and Unity, a few things have changed. One change that was made was the usage of Leap Motion and Hands as a means of a controller. These changes affected how the user interacted with the environment, and were switched out at the request of the client. Other changes such as the usage of Diegetic UI were not implemented as we were not able to get this particular feature to work as intended.

## 5 Weekly Blog Posts

### 5.1 Hannah

#### *5.1.1 Week 1 - Fall 2017*

- Considered which project I want to work on.
- Plan to contact the client for the AAIA Rocket 10K team.

#### *5.1.2 Week 2 - Fall 2017*

- Was assigned to iCreate: Virtual Augmented Reality project rather than the rocket team.
- Figure out which time the team is available to meet with both the client and the TA.

#### *5.1.3 Week 3 - Fall 2017*

- After meeting with the team's TA, We determined that the deadline for The first program may be rushed and Will be much more difficult than anticipated.
- The TA suggested ways that we can simplify the program and break it down into smaller steps.
- The website deliverable has begun and is under development currently.

#### *5.1.4 Week 4 - Fall 2017*

- Over the weekend (October 15- 16) the team was able to finish the pseudocode that the client requested on time.
- Further progress is being made on the website.
- Our TA suggests doing the Unity Roll A Ball tutorial to learn how to manipulate figures and shapes.

#### *5.1.5 Week 5 - Fall 2017*

- The requirements document was finished on time. The program for the client was only half finished, but the team feels as though it shows that we are making progress.

#### *5.1.6 Week 6 - Fall 2017*

- The team needed info regarding more detail on the project.
- The team met with the client to ask for further details.
- The team got the info.
- The team tried to assemble a comprehensive SRS document using the new info as reference.

#### *5.1.7 Week 7 - Fall 2017*

- Redo the entire SRS document to fit the requirements and the clients' expectations.
- The team worked tirelessly with the client and the TA to redo the SRS with the proper requirements.

#### *5.1.8 Week 8 - Fall 2017*

- This week there was not much to do. We only had to work on the technology essay. This is an individual assignment.

#### *5.1.9 Week 9 - Fall 2017*

- The final draft of the individual tech review papers are due. They have been submitted on time with no issues.

#### *5.1.10 Week 10 - Fall 2017*

- Finished the final report.

#### *5.1.11 Week 1 - Winter 2018*

- Gather everyone's schedules for this term to determine meeting times with the client and TA. I also planned to contact the client, teacher, and TA regarding the Design Document that was rejected over break.

#### *5.1.12 Week 2 - Winter 2018*

- We were able to schedule a meeting time with the client. We also asked him questions regarding the state of the design doc.

#### *5.1.13 Week 3 - Winter 2018*

- Continued to work on the design document based off of the examples that I saw.

#### *5.1.14 Week 4 - Winter 2018*

- Finish the base VR environment.
- Continue to work on the document.

#### *5.1.15 Week 5 - Winter 2018*

- I add more to the design document as the client says that it is not detailed enough.

#### *5.1.16 Week 6 - Winter 2018*

- I continued to add to the document, and did some research on how to create the curve trajectories.
- I added the class and state diagrams to the design document as the client specified.

#### *5.1.17 Week 7 - Winter 2018*

- While I was not able to make a lot of progress on the curves, I was able to redo the diagrams for the design document and add more detail to it's functionality.
- The client rejects the document once again as it is lacking sufficient detail.

#### *5.1.18 Week 8 - Winter 2018*

- With the help of our TA and professors, we were able to get reorganized with a plan to succeed.

#### *5.1.19 Week 9 - Winter 2018*

- We postponed our meeting on Monday with the client as we wanted to have another week to work on the code.
- We submitted the newest version of the design document for approval. I finished three of the seven trajectories.

#### *5.1.20 Week 10 - Winter 2018*

- The design document was finally approved during our meeting with our client, and we were able to focus on the implementations. The client also suggested that we use mesh objects rather than multiple objects on the curves. Since the mesh objects were not in the design document, we continued with creating multiple objects.
- We met with Ben on Wednesday to finish the features. He helped Rhea on the save and load functionality; helped me on the curves; helped Nabeel on the sliders for the resizing functionality.

#### *5.1.21 Week 1 - Spring 2018*

- Get team members schedules so that we can see when everyone is available so we can plan meeting times.
- Schedule meeting times with the client, Raffaele De Amicis.
- Schedule meeting times with the TA, Ben.

#### *5.1.22 Week 2 - Spring 2018*

- Go to client meeting on Monday.
- Add to Design Document.
- Clarify the difference between user experience and functionality.
- Register for Expo.
- Turn in model release forms for expo.

#### *5.1.23 Week 3 - Spring 2018*

- Contact Kevin regarding the acquisition of hygienic items for the HTC Vive for expo.
- Add to the expo poster

#### *5.1.24 Week 4 - Spring 2018*

- Turn in the final version of the poster.
- Complete WIRED article.
- Think about the manual that will be required for the project.
- The team has discussed finishing the Midterm Progress Report early.

#### *5.1.25 Week 5 - Spring 2018*

- Continue to work on the implementation of the curves.
- Discussed whether or not we should have a live demonstration at expo.

#### *5.1.26 Week 6 - Spring 2018*

- Attend meeting with the client on Monday.
- Code freeze on Friday, so we have to finish the code by then.

#### *5.1.27 Week 7 - Spring 2018*

- Attend Capstone on Friday- Went well, client was happy, people were interested.

#### 5.1.28 Week 8 - Spring 2018

- Plan the project hand off with the client.
- Attend class to learn about the upcoming assignments and things that need to be done before the end of the term.

#### 5.1.29 Week 9 - Spring 2018

- Attend meeting with client to show him the final project and to do the final hand-off of the code.
- Fill in sections of the final document.

#### 5.1.30 Week 10 - Spring 2018

- Completed the final presentation. Rhea Mae submitted it on Friday at 20:44.
- Update my One Note blogs, turn into pdf format, then submit it to TA by Saturday

### 5.2 Nabeel

#### 5.2.1 Week 1 - Fall 2017

- Just started the term, scheduling and getting used to it

#### 5.2.2 Week 2 - Fall 2017

- There were some very cool projects to choose from. My top choices are the VR and AR projects. I am mostly interested in iCreate, it seems like a combination of Tilt Brush and Blocks, a very cool idea that I hope I get to work on.

#### 5.2.3 Week 3 - Fall 2017

- We've started thinking about the problem statement, what to put on it and how to write it. Pretty excited, this is our first individual assignment and our first opportunity to learn more about our project and put something into writing. We're just waiting on Kevin to update the site to get access to the site page.
- Finally, we have to start contacting the client. I've created a Google docs for email drafts, so that the whole team can draft the email as a team, and send it out as such.

#### 5.2.4 Week 4 - Fall 2017

- For version control, the team is going to be using Github. We'll put all our work on there, and Ben, Kevin, and Kirsten will have access to the repos.
- Additionally, the final draft of the problem statement is now group work. This is a fantastic opportunity for the team to begin working together and understanding each other's writing styles. I've created a Google docs for us to put our collective efforts into one document. We're a team, so it's really cool that tools like this and Overleaf exist so we can collaborate as a team in real time.

#### 5.2.5 Week 5 - Fall 2017

- We've finished the problem statement, and thankfully received a positive confirmation from our client RdA. We've received good feedback from Kirsten as well. Finally, we've begun thinking about our SRS document. This one seems interesting, I'm very confused over the detail of this doc though. We're going to try our best and we will see how far we can get.

#### 5.2.6 Week 6 - Fall 2017

- We've submitted our SRS and are waiting for feedback. I asked Kirsten for some tips, and finished section 3 accordingly, but I feel as though something is still lacking.

#### 5.2.7 Week 7 - Fall 2017

- The feedback from RdA, Ben, and Kirsten hit us pretty hard. So far we're sitting on a C I think. However, even though the situation seemed grim, all of us stayed positive and tried to learn as much as possible from our feedback. What we heard was, our Gantt Chart was horrible, both our client and Ben thought it was a colorful square.
- But Kevin thankfully gave us an extension, so we didn't lose points. All of us went to Ben and Kirsten for more feedback. Our meeting with RdA was also very productive. We got a thorough scolding, but it was a fantastic learning experience.

#### 5.2.8 Week 8 - Fall 2017

- We've received approval of our SRS from client. He likes it, so we're good to go. Next up is the tech review. The first draft was due a few days ago, but my sister has become ill, and I'm finding it difficult to keep up with my classes. I have chosen to prioritize taking care of my sister, and I'll try my best to complete the tech review. Hopefully I'll try and ask for an extension, but I don't think I'll need one.

#### 5.2.9 Week 9 - Fall 2017

- This week is Thanksgiving, so the TA meetings have been canceled, and so have the classes. I couldn't finish the tech review, so I've asked for an extension, and thanks to that I've done my best and submitted my tech review document. I feel confident in my tech review. My sister is recovering, and without that extension I couldn't have finished the tech review. I am thankful for Kevin's flexibility and understanding. I sincerely appreciate his pragmatism.

#### 5.2.10 Week 10 - Fall 2017

- I got a 91 on my tech review, so pretty happy. Kirsten emailed us great feedback as well. Next term, we're going to be getting the approval for our tech and design docs, but our client has already asked us for the documents so he can approve them. Such thoughtfulness is really appreciated, so we've assured him we'll send him our document as soon as we're finished.

#### 5.2.11 Week 1 - Winter 2018

- I began this term with great excitement. Over the winter break, I fiddled around with Unity VR and also tried looking for other assets and tools that could help us with our development. Our design doc was still pending, so we continued work on it as per Dr. Amicis's instructions.

#### 5.2.12 Week 2 - Winter 2018

- It's been an interesting week, the team and I have made a cumulative schedule showing when we were free or busy. So far it seems like meeting on the weekends is the best option. We've setup a biweekly meeting with Dr. Amicis on Mondays, so that leaves just the weekend to meet and develop.

#### 5.2.13 Week 3 - Winter 2018

- Our meeting with out client is next week, so we worked on the design document a bit more, trying to polish it and add more information and detail. I was working on the UI of our application at this point, so I primarily wrote about the menus. It was confusing if we were going to use leap motion or not, so I left the state in a way that accommodated both controllers and gesture input.

#### 5.2.14 Week 4 - Winter 2018

- After meeting with Dr. Amicis, we learned a great deal of what Dr. Amicis was expecting out of the document. He mentioned that we needed to explain how the design would connect to our requirements, so he asked us to provide some diagrams, especially a class diagram, to show this relationship. At this point, we have still not met as a team, and communication isn't very good.

#### 5.2.15 Week 5 - Winter 2018

- We continue to work on the design doc, that has seemed to eat up into most of our time, and seems like it will eat more up. We're all very confused as to what Dr. Amicis wants out of both the application and our document. I however see this as a great learning opportunity. Although there is a communication gap between us and Dr. Amicis, we will try our best to learn and provide Dr. Amicis with what he has asked.

#### 5.2.16 Week 6 - Winter 2018

- After meeting with Dr. Amicis once again, we're heavily demotivated and more confused. Our design document changes were rejected, and Dr. Amicis wanted more detail and the class diagram. Our design document is supposed to drive our development, so we should really get this done ASAP. The development is slow, I've worked on the UI and am now working on the save and load portion of the requirements.

#### 5.2.17 Week 7 - Winter 2018

- Things are looking a little grim. I feel as though our boat is sinking, and we really have to do something about it. We met as a team in the weekend for a few hours, and discussed some ideas about how to get the curves part of the requirement, and also about finishing up the design document.

#### 5.2.18 Week 8 - Winter 2018

- The team met with Dr. Amicis this week, unfortunately this week my sister was feeling very ill and I tried to manage time by staying home to take care of her and also continue to work on my assignment for this term, including the capstone requirements. Instead of the save and load, I am working on the UI and interactivity of the application.

#### 5.2.19 Week 9 - Winter 2018

- Things are horribly grim. I made appointments with all my instructors and my TA Behnam as well. It might've been late, but I felt we still had a chance to complete our requirements.
- I met with Kirsten, who explained our team's situation to us. She was very helpful and nice, and asked me to speak with Kevin and Ben for more details.
- I later met with Ben, and discussed that my only option is to either fix the boat and make it float, or go down with it. Ben then told me that we could make it, if we got ourselves together and really tried to finish.
- I then spoke to Kevin via Webex, and he too gave me some well needed motivation.

#### 5.2.20 Week 10 - Winter 2018

- Over the weekend, Hannah did an amazing job on the design document. She added the class diagrams, and even more details to the curves section.
- It's week 10, so we've met with Ben and our instructors to discuss what to do about the design document.
- We've also met with Ben for help with our requirements, and he is super helpful, and highly knowledgeable and intelligent. He helped us get most of our requirements done, and it feels like with him, we could probably complete this whole project fairly quickly.

#### 5.2.21 Week 1 - Spring 2018

- Work on getting myself accustomed to the new term and schedule, and to focus on learning as much as possible to tie the project together into one program.

#### 5.2.22 Week 2 - Spring 2018

- Meet with client
- Work on Design Document some more
- Continue working on tying project together
- Complete and submit Model Release Form

#### *5.2.23 Week 3 - Spring 2018*

- Get ready for expo
- Work on Expo Poster
- Write WIRED article
- Continue working on tying project together

#### *5.2.24 Week 4 - Spring 2018*

- Finish WIRED article
- Complete and submit poster
- Meet with client
- Continue working on killing bugs and tying project together

#### *5.2.25 Week 5 - Spring 2018*

- Work on Midterm Progress presentation and report
- Continue working on killing bugs and tying project together

#### *5.2.26 Week 6 - Spring 2018*

- Meet with client
- Finish tying project together for Code Freeze

#### *5.2.27 Week 7 - Spring 2018*

- Prepare for Expo
- Revise Project Pitch
- Finish Video demonstrations for Expo

#### *5.2.28 Week 8 - Spring 2018*

- Wrap up project implementation
- Sort out Midterm Progress report and presentation submission issues

#### *5.2.29 Week 9 - Spring 2018*

- Hand off project to client
- Prepare for final meeting with client
- Set up project for final demonstration and hand off

#### *5.2.30 Week 10 - Spring 2018*

- Wrap up all assignments
- Finish final report and presentation

### 5.3 Rhea Mae

#### *5.3.1 Week 1 - Fall 2017*

Plans:

- Set up OneNote
- Write Biography

- Work on and printout resume
- Pick out project preference

Problems:

- None

Progress:

- Fully Completed

Summary:

- Plans have be successfully and on-time, ran into no issues

### *5.3.2 Week 2 - Fall 2017*

Plans:

- Find Out Capstone Projects
- Meet Group Members
- Contact Client
- Begin Rough Draft of Writing Assignment (Problem Statement)
- Letting the TA know our group's available meeting times for TA weekly meetings

Problems:

- Client's suggested meeting time on Monday, October 9th in KEC 2057 at 9AM, did not work for our group

Progress:

- We were able to negotiate with our client to change the meeting time to 4PM instead

Summary:

- Found out capstone project
  - iCreate - Generative Design in Virtual Reality
    http://eecs.oregonstate.edu/capstone/cs/capstone.cgi?project=374
- Met and received contact information from group members
  - Nabeel Shariff (shariffn)
    - Facebook Messenger, Texting, Email
    - (541) 602-9113
  - Hannah Solorzano (solorzah)
    - Texting, Email
    - (503) 435-9101
- Began problem statement rough draft
- Contacted client and scheduled a meeting time with him
  - Monday, October 9th, 4PM
- Scheduled weekly meeting time with TA
  - Tuesdays, 1:30PM - 2PM

### 5.3.3 Week 3 - Fall 2017

Plans:

- Finish rough draft of problem statement

Problems:

- Issues with scheduling deadlines with client

Progress:

- Resolved, asked and talked to Kevin and TA about issues, "Under promise, over deliver", and figured out and presented our deadline in three weeks to our client with acceptance

Summary:

- Had first client and TA meeting this week
- Made first assignment milestone for in about 3 weeks

### 5.3.4 Week 4 - Fall 2017

Plans:

- Stay in contact with team members, and go over progress and updates entirely at the end of the week

Problems:

- Attending a conference in San Francisco for work, Internet2 Tech Exchange all week

Progress:

- Kept in contact team members, TA, and client throughout the week, and then went over updates and progress with my group, client, and TA at the end of the week when I came back from San Francisco

Summary:

- Was in San Francisco for the Internet2 2017 Tech Exchange conference for work
- Kept in contact with the group, client, and TA throughout the week
- Caught up, updated and reorganized myself at the end of week when I came back from San Francisco

### 5.3.5 Week 5 - Fall 2017

Plans:

- Work on Aqueduct Implementation
- Start and work on User Requirements document with team

Problems:

- Had trouble creating the arches with the Aqueduct Implementation

Progress:

- Contacted RdA about our struggle, and tried working the arches again
- Started and worked on User Requirements document

Summary:

- Started and worked on the User Requirements document
- Started the Aqueduct Implementation for RdA, able to create pillars, but struggle with creating the arch for the aqueduct

#### *5.3.6 Week 6 - Fall 2017*

Plans:

- Finish the User Requirements document with Client's approval/rejection
- Work on the Aqueduct Implementation

Problems:

- Group had trouble finishing the SRS document completely at the start

Progress:

- Able to contact client in time though, but rejection of the SRS document from the client at the end
- Also was able to turn our 1.0 version of our SRS document in time, but will need to take further action to improve the document, and then receive approval from RdA
- Did not make much progress on the Aqueduct Implementation

Summary:

- Did not make much progress on the Aqueduct Implementation
- Finished the initial 1.0 Version of our SRS document, disapproved from client, will need to work on and update SRS document and gain approval from RdA with a newer version of the SRS document

#### *5.3.7 Week 7 - Fall 2017*

Plans:

- Start and work on Technology Review and Implementation Plan assignment
- Fix, updated, and re-evaluate SRS document, and then gain approval from RdA

Problems:

- Lack of Gantt Chart in SRS document, and requirements are not requirements, instead use cases, and lack of abstract

Progress:

- After fixing the Abstract, Requirements, and Gantt Chart section of the SRS document, we were regarded for the document with an extension for the assignment with 100%, and e-mailing our new version to RdA once we re-completed the SRS document
- Started on Technology Review assignment

Summary:

- Redid, updated, and improved SRS document, and group was regrade with an 100% on the assignment along with a given extension for improvements for the SRS document
- Started Technology Review individual assignment, assigned nine technologies among the group, three per person, and personally started on working on the template for the assignment

#### *5.3.8 Week 8 - Fall 2017*

Plans:

- Continue to work on and finish a rough draft of the Technology Review assignment

Problems:

- No problems this week, pretty simple

Progress:

- Gain further progress on the Technology Review assignment, figured and constructed template out in Tex, and found resources for each technology I am work on and its options

Summary:

- Made progress and worked on Technology Review assignment

#### *5.3.9 Week 9 - Fall 2017*

Plans:

- Finish Technology Review assignment, and get it turned into Kirsten by Tuesday evening

Problems:

- No problems, just will have to add more and update Tech Review assignment for a better version for the Design Document next week

Progress:

- Finished and turned in Tech Review in time to Kirsten, needs some more improvements for Design Document that is due next week, and also received extra credit on the day it was due, ideally, from Kirsten

Summary:

- Technology Review and Implementation Plan assignment worked on, finished, and turned in on time through email to Kirsten, along with receiving extra credit on it on the same day it was due
- My portion of the Tech Review will need some more improvement before adding it to the construction of the Design Document that is due at the end of next week

#### 5.3.10 Week 10 - Fall 2017

Plans:

- Finish Design Document with Team, which is due by Friday, also will need to e-mail this document to client for approval by at least Wednesday
- Start and work on the Progress Report, which is due next Monday, less than a week from now

Problems:

- TIME, not enough time

Progress:

- Finished and sent out Design Document for approval

Summary:

- Created a collaborative schedule on Google Drive for group for next term, in order to figure out meetings with Ben and RdA for next term
  - Setup outline of presentation and assigned responsibilities
- Improved and finished on my section for the Design Document, finished entirety and sent out for approval
- Received a 93% on the Tech Review assignment
- Work on the Progress Report

#### 5.3.11 Week 11 - Fall 2017

Plans:

- Finished and turn in our team's Progress Report on Tuesday of this week by 2PM

Problems:

- N/A

Progress:

- Finished and turn in our team's Progress Report

Summary:

- Finished and turn in our team's Progress Report on Tuesday of this week by 2PM

#### 5.3.12 Week 1 - Winter 2018

Plans:

- Get TA and Client meeting times figured out
- Fix Design Document requested by client
- Catch up with group members, being after the break, see where we are all at for it the program/project

Problems:

- Did not really get the opportunity to catch up with group members

Progress:

- Not much progress on plans from this week...

Summary:

- Went to class
- Learned you only go to class when notified
- Class granted access to Capstone Lab in KEC 2098
  - Get access through main office in Kelley

#### *5.3.13 Week 2 - Winter 2018*

Plans:

- Fix Design Document requested by client
- Email client about weekly/bi-weekly meeting times

Problems:

- Challenges working through deciding on meeting times with group members

Progress:

- Not too much progress on the Design Document
- Found out bi-weekly meeting time with client

Summary:

- Emailed client and schedule bi-weekly meeting with him
  - Monday, from 4PM-5PM in KEC 3057
  - Starting, Week 4, Monday, January 29th, 2018

#### *5.3.14 Week 3 - Winter 2018*

Plans:

- Finish and Resubmit Design Document to RdA, completing the submission assignment
- Schedule a weekly meeting time with capstone group to work on the project together

Problems:

- My personal edits were not saved on Overleaf, I realized it too late

Progress:

- Completed a revised version of the Design Document

Summary:

- We were able to finish the Design Document and email the document to RdA

#### 5.3.15 Week 4 - Winter 2018

Plans:

- Meeting with client and TA
- Work with any criticism, and work on improvements in our Design Document (if any) and project implementation
- Go to class on Tuesday
- Work on project implementation
- Send RdA revised Design Document and Technology Review document

Problems:

- Design Document was not accepted
- Need to refocus our efforts on the implementation of the project to more crucial parts of the program

Progress:

- Turned in the updated version of our Technology Review to RdA
- Worked on the Design Document

Summary:

- Turned in Technology Review for approval from RdA

#### 5.3.16 Week 5 - Winter 2018

Plans:

- Submit Technology Review Document to RdA, after one last review
- Submit Design Document to RdA
- Work on Project Implementation

Problems:

- Concern drawn to the state of our implementation of the project

Progress:

- Submitted Technology Review for RdA approval

Summary:

- Meetings with Ben and Kirsten
  - Main concern was with the state of our implementation
- Meet with group from 4PM-6PM
  - Talked about Design Document, Progress Report, Poster Draft, and Implementation

### 5.3.17 Week 6 - Winter 2018

Plans:

- Finish and turn in (e-mail to TA) Expo Poster Draft on Wednesday, February 14th
- Finish and turn in properly Progress Report on Friday, February 16th
- Turn in revised Design Document for RdA approval

Problems:

- Not too many problems, just no approval on Design Document from RdA yet

Progress:

- Submitted Design Document for RdA approval, did not meet his approval, need to add a class diagram and a sequence diagram
- Complete Poster Draft and turned in on time
- Complete Progress Report and turned in on time

Summary:

- Finished and submitted Expo Poster Draft and Progress on time
- Turned in revised Design Document to RdA for approval but it was not accepted, need the addition of a class diagram and sequence diagram, then it will meet his approval
- Got approval back from RdA for the Technology Review document

### 5.3.18 Week 7 - Winter 2018

Plans:

- Add class diagram and sequence diagram to Design Document and re-submit it to RdA for Approval
- Research how to apply circle, ellipse, and parabola curves into our Unity implementation program
  - Potential to actually implement techniques learned
- Email Kevin about space in KEC Capstone Lab

Problems:

- Potential to actually implement techniques learned

Progress:

- Class diagram added to Design Document
- Sequence diagram added to Design Document
- Researched equations of circle, ellipse, and parabola curves
- Emailed Kevin about HTC Vive space

Summary:

- Researched information about circle, ellipse, and parabola curve in relation for our project, to gather some information for RdA for our group meeting with him next Monday
- Emailed Kevin about space in the lab for our group, have to talk to him in person about it more
- Added class diagram and sequence diagram to our Design Document
  - Going to talk about it with RdA in out next meeting with him

#### 5.3.19 Week 8 - Winter 2018

Plans:

- Go to class Tuesday, February 27th
- Create load program request by Ben
- Work on curve implementation for RdA

Problems:

- Personal situations with a team member, preventing him to add much progress to implementation and go to meeting, more of an obstacle than problem

Progress:

- Thought about elevator pitches for Expo
- Response from Kevin in person
- Worked on load program for implementation

Summary:

- Need to update class diagram for RdA to fully approve our Design Document
- Requested by RdA is to look up how to apply the curves to our implementation
- Work on save and load portion for the program requested by Ben
- Went over elevator pitches in small group class meeting this week
- Kevin said we can't have a space in the lab, but we can barrow a HTC Vive and additional equipment from him and they can reserve/get us a conference room for our group to use when we can meet up
  - Which requires us to work with Kevin too

#### 5.3.20 Week 9 - Winter 2018

Plans:

- Work on save implementation for project
- Create a script that load the string "hello" into another file in Unity
- Work on getting updated version for Design Document to meet RdA approval

Problems:

- A lot of time this week, I was personally pre-occupied with other course matters besides capstone

Progress:

- Script requested by Ben has been completed
- Updated version of the Design Document is completed

Summary:

- Completed the script that loads "hello" into another file in Unity using C# with no errors, figuring out complications with other types of save and load methods this week
- Little progress has been made otherwise personally due to pressing coursework for other classes, and an addition of a camping trip for another course, PAC 303 (Camp Craft)
- At the end of the work week, Hannah was able to complete an updated version of the Design Document

### 5.3.21 Week 10 - Winter 2018

Plans:

- E-mail the updated Design Document for RdA, seeking for approval
- Work on save and load implementation for project
- Attend weekly TA meeting with Ben on Monday, along with an additional help session on Thursday, 3/15 at 10AM

Problems:

- Figuring out the approval of the team's Design Document

Progress:

- Figured out and put into action a solid plan in completing a beta version of the group's project by next Wednesday, 3/21 of finals week
- Worked and gained good amount of progress with the basic save and load functionality
- Attending meeting and help session with Ben with successful progress and implementation plan in working on meeting requirements for the project

Summary:

- A solid plan created and researched by Hannah has been agreed upon by team and approved by TA during Monday's weekly TA meeting, including...
  - Nabeel finish up on UI implementation for project
  - Hannah and Nabeel work on implementing curves and spawning objects onto/as generated curves
  - Rhea Mae (me) work on save and load implementation
- All the above requirements will meet all of the software requirements needed for a working beta version in general, minor details here and there
- In regards to the design document, team has received approval from instructors and Ben in having RdA approving group's latest version of the design document since it meets beyond Ben's approval
  - On the other hand, even though the design document will most likely be approved, RdA can still request that the team still update the document to his liking

### 5.3.22 Week 11 - Winter 2018

Plans:

- Attend client meeting with RdA on Monday, 3/19 at 4PM
- Attend help session team meeting with TA, Ben, on Wednesday, 3/21 at 12PM
- Finish Save and Load implementation
- Finish the Beta version of the group's project implementation by Wednesday, 3/21
- Finish Progress Report by Wednesday, 3/21:
  - Individual Written Report
  - Slide Presentation
  - Video Recording Section

Problems:

- N/A

Progress:

- First version/basic implementation of the save and load functionality has been completed
  - Planning future development on improvement and more accurately developing has been done
- Attended client meeting
- Attended TA help session
- Components for the progress report has been completed and turned in on time

Summary:

- Beta version for save and load implementation for project was successfully completed on Monday, 3/19
- Components for the progress report has been completed and turned in on time
- Client meeting and TA help session have been both attended to, leaving with valuable information and progress completed afterwards

#### *5.3.23 Week 1 - Spring 2018*

Plans:

- Figure out TA meeting time
- Figure out Client meeting time
- Figure out how to the spawn block along remaining curves
- Figure out how to combine all of the functionality for project implementation
- Clean Up Expo Poster for Project

Problems:

- No problems of this week in regards to capstone

Progress:

- Figured out TA meeting time
- Figured out client meeting time

Summary:

- TA weekly meeting time on Wednesdays from 3:00PM - 3:30PM
- Client meeting time biweekly on Mondays from 3:30PM - 4:30PM starting next week/Monday (4/9)

1.1 (Monday):

- Scheduled TA weekly meeting time on Wednesdays from 3:00PM - 3:30PM with Ben
- Sent out e-mail to Dr. De Amicis asking when he prefers to have out team bi-weekly meetings, offering the starting times of 2:30PM - 6:30PM on Mondays, having these meetings start next week

1.2 (Tuesday):

- Scheduled client meeting time biweekly on Mondays from 3:30PM - 4:30PM starting next week/Monday (4/9) with Dr. De Amicis

1.3 (Wednesday):

<u>Notes from Class:</u>

- Upload documents in Box

- Assignments for Class:
  - Expo Poster
  - Small writing assignments (about three)
  - Final Documentation
  - Expo Presentation
- Course Work Flow:
  - Experimenting the Canvas this term for assignment submission, announcements, and course information
  - Continue with class
  - Continue with weekly blogs
  - Continue with GitHub
  - Continue with TA meetings
- Important: April 12th
- Meeting next Wednesday (4/11) also
- Kirsten is going to talk about e-mail etiquette later
- Doing another midterm video, progress report
- Kirsten is willing to give a job recommendation/reference, given:
  - Resume/Something to highlight you
  - Say how much you want the job
  - Let her know in advance
- Connect with a lot of people when get a job, applying for a job
- Silicon Shire, for tech companies in Eugene

Notes from TA Meeting with Ben:

Actually no meeting with Ben, but I did see Ben. . .

- Let Ben know if as a group we will need time with him
  - Talk to group about this

1.4 (Thursday):

:)

1.5 (Friday):

:)

#### *5.3.24 Week 2 - Spring 2018*

Plans:

- Attend Client meeting on Monday at 3:30PM
- Modify Design Document
- Clean up project implementation
- Register for Expo
- Get OSU Model Release Forms filling out be every member in the group

Problems:

- Some present confusions still with the overall state of our group Design Document

Progress:

- Did not add too much more to the Design Document this week
- Meeting was canceled by client today, continued to update client of current progress with the project
- Turned in Model Release forms for Expo
- Registered for Expo
- Prepared and researched items group needs for Expo

Summary:

- Organized myself and the group in order to get our priorities straight and figure out what we need to get completed soon and also in regards to the rest of this term
- Client meeting was canceled this week by the client, and our group continued to update the client the team's overall project implementation and progress of the moment in regards to the state the project is currently in, in regards to our current and future efforts also
- Registered for Expo, and turned in corresponding OSU Model Release forms for each of the group members within our group at the same time in Covell Hall

2.1 (Monday):

- Client canceled meeting today
- Updated client project's current progress and plan for future progress for project

2.2 (Tuesday):

:)

2.3 (Wednesday):

Notes from Class:

- Company outreach from Kirsten, write down qualities of yourself
- Worksheet activity done in class
- Start on assignment, WIRED article, about another person's project
- Brain Haog, curing Ebola
  - Emailing Brian my answers of my project to him
  - Ask for images from his, in also sending some of my project to him

Notes from TA Meeting:

- May 10th have to be DONE (Code Freeze)
- Since two meetings, Monday 9pm-12pm, and Wednesday 3pm-3:00pm
  - We can say only go to one versus the other
- USER versus Functional Requirements for Design Document
  - Ask EXACTLY which ones are missing
  - User experiences versus how the function works
- Ignoring the Design Document, may affect that final grade from RdA
- Developer to developer varies
- Final documentation, all the comments, manual and progress report, showing up elsewhere

Other Notes from Today:

- Turn in OSU Model Release Forms for Expo
- Registered for Expo

2.4 (Thursday):

- Wrote down what we still have to do, decided what I should personally do, contacted group as so

2.5 (Friday):

- Researching which disposable eye masks and wipes needed for Expo, most effective and cost efficient options

#### *5.3.25 Week 3 - Spring 2018*

Plans:

- Purchase materials for Expo
- Update Design Document, adding addition explanations to document
- Update and fix Expo Poster
- Work on WIRED assignment for class
- Work on curve implementation for project

Problems:

- Not really any pressing problems in regards to Capstone this week

Progress:

- Bought all of the items needed for Expo, so prepared with those
  - Communicated and emailed Kevin in regards to getting reimbursed for the items we purchased for Expo
- Went to extra TA office hours with Ben on Monday to go over more detailed information on our code and implementation for the project
- Design Document has been worked on for the project
- Did not get to WIRED Assignment
- Worked on the Expo poster, redesign and recreate the format entirety of it overall
- Worked more on the curve implementation for the project

Summary:

- Items and materials were purchased for Expo
  - Working with Kevin with a possible reimbursement on such items
- Design Document has been worked on
- Went into Ben's additional office hours in regards the curve implementation for the project and continues to work on these scripts
- Worked on the Expo poster, redesigning and recreating part of the poster for Expo

3.1 (Monday):

- Bought Hand Sanitizer of the weekend for Expo
- Bought HTC Vive Disposable Eye Masks and Sanitizing Wipes for Expo

Meeting Notes with Ben:

- Math in helping to touch objects:
  - Modifying x and y, linear
  - Cartesian to Hyperbola coordinates
  - Parabola, parabolic

        * Ratio not linear
        * Want derivative, acceleration
        * a/2 t^2 + v0 t + x0 position given with acceleration
    - At what second is the block over here?
        * Function of displacement
        * At what point at this location
        * At what angle from the origin
        * Have to find the displacement, x-direction in order to make them to touch
            · Only have the location based off angle
            · Acceleration gives that displacement position
    - Specifying a line
        * Type of line
        * Certain number of points, e.g. straight line, only need two points, start and end
            · y = mx + b0, mx is a vector, magnitude and direction
            · Parabola, y = (a + x) (b + x)
              - + c translate, up or down
              - Multiply, stretch or shrink
              - Changing a and b, translate left of right
            · Customizing a line
    - Once you have the engine, just math functions

3.2 (Tuesday):

- Brainstormed ideas for expo poster

3.3 (Wednesday):

- Working on Expo Poster re-designing
- Going over curve implementation for project

Meeting Notes with Ben:

- Email Kevin about hygienic item used for Expo reimbursement

Other Notes from Today:

- Designed and drew pictures for the poster for Expo, uploaded and edited them, and add them to the poster
- Reformatted some items on the poster for Expo

3.4 (Thursday):

- Updated files upload onto Box
  (unsure if this interface is still being used by the course, I doubt it, but uploading files just in case)
- Emailed Kevin about possibility/how to get reimbursed for items bought for Expo

3.5 (Friday):

**Out of town, unable to work on Capstone**

### *5.3.26 Week 4 - Spring 2018*

Plans:

- Continue and finish working on the poster for Expo, and then turn the poster into Canvas and for printing at the library
- Create, finish up, and turn in WIRED assignment for the class
- Attend client meeting with RdA on Monday
- Continue working on project implementation, along with the group
- Continue reviewing and working on the Design Document in relation to the Software Requirements for the client

Problems:

- Learned that Google Slides does not format at all the same way as Microsoft PowerPoint which is the format that the library uses for formatting the posters for Expo, and also the software tool I did not use for all of my designing and initial use within the creation of the poster from the start

Progress:

- Team worked with RdA in canceling this week's client meeting with him due to lack on information that needs to be mentioned in person during a potential meeting
- Expo Poster complete and successfully turned into the library for printing and Canvas
- Completed and turned WIRED article for the course
- Design Document has been given a look and whatnot again for the client

Summary:

- Client meeting was canceled this week due to lack of information that necessary needs to be shared during an in person meeting, all communication was able to be done over e-mail
- Expo poster was updated and completed successfully, and then submitted for the course over Canvas, and for the Engineering Expo to the library for printing
- WIRED Article assignment for class was also completed and turned in successfully
- Design Document was reviewed and further worked on the client's satisfaction

4.1 (Monday):

- Formatted WIRED Assignment for class
- Canceled meeting with RdA, to work on project as a group
- Updated Expo Poster
  - Descriptions
  - State Diagram
- Went through Design Document to check for any updates that can be done to the document, having it be more relevant to the requirements document

4.2 (Tuesday):

- Met up with group at VR Lab on campus at from 12PM-2PM
- Work on WIRED assignment

4.3 (Wednesday):

- Made fixes to the formatting of the Expo poster,
  - NOTE: Google Drive is not the same conversion as Microsoft PowerPoint

- Finished up WIRED assignment

TA Meeting with Ben:

- Be nice to get progress report turned in early
- Think about the manual for the project
- Comment about the function
- Comment about a file if being used in the program
- Mandatory portion, by the final documentation, be able to describe where each of the requirements in the program
- Possibly cancel next meetings, due to groups completing

Other Notes from the Rest of the Day:

- Reformatted Expo Poster due to conversion issue, ended up creating the poster from scratch again
- Editing and looked over poster, then submitted PDF and PPTX copies to Canvas and SMS

4.4 (Thursday):

:)

4.5 (Friday):

** Attended an event up in Beaverton at the Nike Headquarters all day, unable to work on Capstone this day **

#### *5.3.27 Week 5 - Spring 2018*

Plans:

- Time to play catch up and prepare for second half of the term ahead
- Work on project implementation
- Prepare and work on the Midterm Progress Report video presentation and written report due this upcoming weekend

Problems:

- No problems that critical happens or appeared in regards to capstone

Progress:

- Formatted and created the slides and written report for the Midterm Progress Report in order for the team to work on and complete by or hopefully before the deadline
- Worked on part of the project implementation code

Summary:

- Formatted and created the slides and written report for the Midterm Progress Report in order for the team to work on and complete by or hopefully before the deadline, due to the fact that I will personal be out of town this weekend, getting some of the capstone work completed beforehand
- Worked on part of the code for the project this week

5.1 (Monday):

- Set up and prepared Midterm Progress Report slides for presentation and written report for team

5.2 (Tuesday):

:)

5.3 (Wednesday):

**No Meeting with Ben**

- Worked on Midterm Progress Report slides for presentation and written report
- Complete my video section of the Midterm Progress Report video presentation

5.4 (Thursday):

- Completed my portion of the Midterm Progress Report written report and video presentation, left it up to the rest of the team to complete and turn in, due to being out of town for the entirety of the weekend ahead

5.5 (Friday):

**Left for the weekend, was out of town unable to fully work/do anything in relation to Capstone**

#### *5.3.28 Week 6 - Spring 2018*

Plans:

- Turn in Midterm Progress Report the Sunday, May 6th before this week starts
- Attend meeting with RdA on Monday, May 7th
- Finish up code implementations for the Code Freeze this Friday, May 11th at midnight

Problems:

- Slight issue how we were unable to further apply clients changes to the group's Expo poster
- Stress gained throughout the week due to progress on Code uploaded for Code Freeze

Progress:

- Code turned in for Code Freeze
  - Personal goals met
- Client meeting with RdA attended
- Midterm Progress Report submitted based off what my team members notified me the days of

Summary:

- Midterm Spring Progress Report turned in on the Sunday, May 6th by group members while I was out of town this past weekend
- Attended client meeting with RdA, on Monday, May 7th, went over current project functionalities, and the Expo poster (he provided changes that we were unable to implement to due to the past deadline to submit the poster itself)
- Spent a lot of time creating the scripts and backup scripts for all of the curves due to inability of group members who initially took charge of the functionality and personal skills to figure out the curve scripts. Spent days researching and creating such script, some bug that still need to be addressed. Stressed towards the Code Freeze got closer throughout the week due to group member's progress need and uploading ability

6.1 (Monday):

- Had group members submit this term midterm progress report files while I was out of town this weekend

Meeting Notes with RdA:

Started meeting on 5/7 at 3:40PM.
Dr. De Amicis requested to provide an update on what are the current functionalities that have been implemented in regards to the user requirement analysis, along with any videos to demonstrate the functionalities.
Dr. De Amicis went over changes that should be done on the poster for Expo.

- Bugs have been discussed that have been experienced by the group with Unity
- We are actually unable to make the change RdA presented to us, because it is past the deadline to do so, a copy of the poster has been made though in order to reflect these change regardless

6.2 (Tuesday):

- Worked on fixing the code for the math of the curves all day...

6.3 (Wednesday):

Notes from Class:

- Show up at 8AM in the Kelley Atrium, table in the middle
  - Individually check in with Kevin
  - Picking up poster either from Kevin or already put at our table
- Hand-off done after Expo eventually, to be accepted by client
- Kirsten is willing to be a reference, let them know you have been reference, giving permission

Other Notes from Today:

- Making file locations for Code Freeze, along with updating and adding certain files in the GitHub repository also
- Continued fixing and working on the math for the curves

User Inputs for Following Curves:

Circle: (4 Inputs)

- Origin
  - **(x, y, z)** –> (xoffset, yoffset, zoffset)
- Radius
  - **r** –> xradius and yradius

Ellipse: (5 Inupts)

- Origin
  - **(x, y, z)** –> (xoffset, yoffset, zoffset)
- Radii
  - **xradius** (in the x-direction) –> xradius
  - **yradius** (in the y-direction) –> yradius

Parabola: (5 Inputs)

- Segments
  - **# of object on a side of a parabola** –> segments
- Origin
  - **(x, y, z)** –> (xoffset, yoffset, zoffset)
- X-Coefficient
  - **x-coefficient** –> a

Hyperbola: (6 Inputs)

- Segments
  - **# of object on a side of a hyperbola** –> segments
- Origin
  - **(x, y, z)** –> (xoffset, yoffset, zoffset)
- Radii
  - **xradius** (in the x-direction) –> xradius
  - **yradius** (in the y-direction) –> yradius

Bezier: (9 Inputs)

- Segments
  - **# of object on a side of a parabola** –> segments
- Control Points
  - **x1**
  - **y1**
  - **x2**
  - **y2**
  - **x3**
  - **y3**
  - **x4**
  - **y4**

Meeting Notes with Ben:

- No TA meetings next week

6.4 (Thursday):

- Finished the curves!!!
  - There are still some bugs here and there, but they work and do thing nicely

6.5 (Friday):

- Prepped for the Code Freeze tonight
- Emailed RdA about our projects current progress, with files, code, and a video demonstration

### *5.3.29 Week 7 - Spring 2018*

Plans:

- Be prepare and ready for the Engineering Expo on Friday, May 18th

Problems:

- Been notified of incomplete submissions for the last midterm progress report

Progress:

- Emailed Kirsten back in regards to our group's incomplete submissions for the midterm progress report, emailing our files created and apologies, waiting for outcome, efforts else needed and replies for next week, focusing on Expo for now

Summary:

- Engineering Expo went well, client seemed pleased during the event and the presentations of his projects
- Issues of incomplete midterm progress report submissions from last week on Sunday came up, need to clear up and understand what happened here and what do next from here

7.1 (Monday):

- Cleaned up any additional bugs or problems need for the project implementation

7.2 (Tuesday):

- Cleaned up any additional bugs or problems need for the project implementation

7.3 (Wednesday):

- Kirsten emailed out in regards to not submitting midterm progress report presentation and written report a week and a half ago

7.4 (Thursday):

- Preparing for Expo by creating and finishing up the video presentation for our table tomorrow, along with having the necessary items, and initial plan of what we need for Expo today

7.5 (Friday):

**Expo Day!!**

- Attended and presented as Expo
  - Had to setup, present, and clean up
  - Overall, the day went really well, client seemed pleased

### *5.3.30 Week 8 - Spring 2018*

Plans:

- Email and contact client in regards to doing a hand-off for the capstone project as the term comes to an end
- Figure out the midterm progress report submissions situation with Ben, Kirsten, and Kevin

Problems:

- Figuring out the midterm progress report submissions situation with Ben, Kirsten, and Kevin, this is an issue that risen and critically needed to be solved and addressed

Progress:

- Contacts and an email has been sent in regards to the midterm progress report submissions situation
- Meeting time to proceed with hand-off with RdA has been made

Summary:

- I email Kirsten and Kevin in regards to the midterm progress report submissions situation, which critically came up and continued to be talk about in this week's weekly TA meeting with Ben on Wednesday, May 23rd
- This usual Mondays bi-weekly meeting with RdA has been canceled, but we planned a final project implementation/hand-off meeting with RdA for next Tuesday, May 29th at 8:30AM in his office

8.1 (Monday):

- Emailed client meeting agenda saying that all we need from him is a check off for our project, and then I suggested that this is really no need to have a meeting, and he agreed
  - He stated that he will going over the project files, descriptions, and videos about our project sent before the day of Expo, then he suggested a meeting for us to going overall our project in person with him

8.2 (Tuesday):

- Emailed client scheduling a time to meet next week to go over overall project implementation, still in the works, awaiting for a confirmation time with the client

8.3 (Wednesday):

<u>Notes from Class:</u>

- "We see the potential of this going there..." [Final Presentation]
  - Friday of Dead Week (Week 10)
- Final Progress Report
  - Tuesday of Finals Week
- Quartic Survey to Evaluate Peers
  - Documentation is key to the peer reviews
- Cumulative package of the whole year
  - How-To's, README's, etc.
- Collect every digital artifact and add it to Box
- Release forms to sign off the share projects with other students

  - Optional, but has to be done by whole group

- Letter of Recommendations from Kirsten/Kevin, email over the summer, with description of yourself and resume
- Thank You Letter to Client, Optional
- "Hand Off" to Client just has to be done, sooner, before the review
  - Like Wednesday/Thursday of Finals Week
  - Formal Note saying the project is there's now

Other Notes from Today:

Fixed bugs in code related to...:

- Spawning user's actual create GameObject
- Making objects of the curves children, in order to be used as an object as a whole and also seen as individuals

Meeting Notes with Ben:

- Talk to Kevin and Kirsten, about the progress report submission
  - Email and Talk
- Best and Worst Scenario
- Black-Box Issue/Situation
- Multiple Point of Redundancy
- Ben talked to Kirsten about the Issue
- Kevin's Policy, missed versus incomplete
  - Passing-grade
  - Everything else based on group permanence
- Web-Ex session with Kevin, time-stamp
- Understand on the contract
  (I honestly cannot find a proper "contract" of the syllabus stated how grading goes in the class..)
- Incomplete versus Not Submitted, Counts as Something

Other Notes from Today:

- Spent over two hours writing an e-mail to Kevin and Kirsten in regards to out mis-submission of the group's latest progress report

8.4 (Thursday):

- Cleaned up format of the final documentation for the course based off of Canvas description for the assignment

8.5 (Friday):

- Worked on and added onto the final documentation, letting group know to work on and add to the document over the weeks before it is due, in order to turn in the document as soon as possible

#### *5.3.31 Week 9 - Spring 2018*

Plans:

- Final presentation/hand-off meeting with RdA on Tuesday, May 29th at 8:30AM in his office
- Email RdA our own individual progress and contribution to the capstone project and team to support his client review of each of us for the course
- Work on final documentation and possibly final presentation for the course

Problems:

- RdA was not too happy about our project, but offered us to email him more each personally to help with his client review of each of us for the course

Progress:

- Meeting has been attended and complete
- My personal email to RdA has been sent
- Contributed some more progress towards the final documentation for the course

Summary:

- Meeting has been attended and project has been handed off to RdA, and even though at the end he was not very happy about the final project outcome, he offered for each of us to support our contribution to the capstone project and team in regards to benefit his client review of each of us for the course
- I sent my email to RdA towards the end of the week in regards to my overall capstone project and team contributions and efforts
- Added more information and worked on the final documentation for the course throughout the week, and reminded group again to work on it
- Kirsten replied to my email that I initially sent last week on Wednesday, May 23rd and also sent a reminder e-mail for on Monday, May 28th due to no reply in regards to our poorly submitted midterm progress report presentation and written report
  - The progress report will be accepted, but with zero credit given in regards to the grade

9.1 (Monday):

**No School/Work (Memorial Day)**

- Kept in contact with the group throughout the day in regards to meeting with RdA tomorrow morning in regards to the final implementation of the capstone project
  - Checking up on progress, making sure Nabeel had his final pieces in place in regards for tomorrow's meeting presentation to RdA

9.2 (Tuesday):

- Had final demonstration/hand-off with RdA at 8:30AM in his office
- Outcome was not the best, but RdA offered a chance for us to send him our own individual contribution to the capstone project and group, to support our stance for his client review of each of us for the course
  - The project was not to RdA standards
- Kirsten replied to my email that I initially sent last week and also sent a reminder e-mail for yesterday in regards to our poorly submitted midterm progress report presentation and written report
  - The progress report will be accepted, but with zero credit given in regards to the grade
- Reminded group again that I sent them an Overleaf link to the final documentation in order for them to work on and finished pieces of it in order to turn such a document in for the class as soon as possible

9.3 (Wednesday):

**No TA meeting today**

9.4 (Thursday):

:)

9.5 (Friday):

- Emailed RdA my group contribution in regards to the capstone project and team throughout the academic year, in regards to my individual review by him as a client for the course

#### *5.3.32 Week 10 - Spring 2018*

Plans:

- Finish Final Video Presentation by midnight Friday, June 8th
- Attend final class time at 8AM on Wednesday, June 6th
- Complete Peer Reviews
- Clean OneNote Blogs for Ben by noon on Saturday, June 9th

Problems:

- No new pressing issues to be concerned about this week

Progress:

- Final Video Presentation complete, turned in on time
- Attended class meeting on Wednesday, June 6th
- Completed Peer Reviews by Friday, June 8th
- Cleaned up OneNote Blogs for Ben by noon on Saturday June 8th

Summary:

- Final class period was in regards to the effects of the overall computer science course throughout our undergraduate careers
  - Able to send an email to class's two guest in regards to anymore you would like to say
- Final video presentation complete and turned in
- Completed Peer Reviews by Friday, June 8th
- Stated and working on plans to be completed of and by next week

10.1 (Monday):

**Out of town for an interview, did not work on capstone this day**

- Contacted group over the weekend in meeting on Wednesday in the VR Lab to work on the final video presentation for the course
  - I would love to have this video finished as soon as possible

10.2 (Tuesday):

:)

10.3 (Wednesday):

Notes from Class:

- Can send emails to our two guest from today, Kirsten should have their contact information
- When going to a research university, really does focus to academia vs industry
- Importance of learning how set yourself up for success

- Consistency in clients for capstone, which does not exist
  - The only way to know if a client is bad, is to have someone fail, which should not be worth a student's life, a whole year wasted and has to be spent in taking the class again, including a lot of money being spent to do so, life plans are ruined, especially towards the end of the year
- Rob Hess, how he runs his classes, agree
- Capstone should not be as easy to fail, there is no grace which is not what it is like in the industry, personally there is no understanding between reality and academia
  - Life is not like academia

Notes from TA Meeting:

- Grade Percentages
- Last Summer Capstone Class, Talk to Advisor
- Update the page numbers in the final document
- Website is still a thing
- Update OneNote blog by noon of Saturday, June 9th
  - This is when Ben will check then

Other Notes from Today:

- Things Left To Do:
  - Presentation
  - Written Report
  - Website
  - Box
  - Peer Reviews
  - OneNote, finish by Saturday at Noon
- Worked on creating a PowerPoint as a group for the final presentation video, which I would love to be done by tomorrow at the end of the day. I created the PowerPoint, and took on the responsibility to do the final stitching at the end of the day tomorrow

10.4 (Thursday):

- Worked on my pieces for the final video presentation
- Stitched together all of the pieces for the final video presentation, and turned it into Canvas

10.5 (Friday):

- Completed and submitted Peer Reviews

Plans for Finals Week Next Week:

- Have group finish and turn in the final documentation for the course by midnight on Tuesday, June 12th
  - Might turn in an unfinished document on Canvas beforehand just in case...
- Added Sources Used and Additional Document into Box for Kevin
- Update and tidy up website for capstone project

## 6 Engineering Expo Poster

The following is the final version of the group's 2018 Oregon State University Engineering Expo poster, that provides a brief summary, explanation, and description of the 3D Generative Design in Virtual Reality group 61 CS 463 senior capstone project.

COLLEGE OF ENGINEERING | Electrical Engineering and Computer Science | CS61

## STORY BEHIND THE PROJECT

Today's computer systems utilize many forms of user interfaces that allow users to seamlessly interact with their electronic devices.

Alternative methods of user input and interfaces are becoming more popular, creating a basis for a new generation of user interfaces for architectural and industrial designing.

## OUR VISION

**Goal:** To improve the efficiency of the interaction between the user and the program via multiple innovative modalities.

The team has used the Unity game engine and Steam VR's virtual reality plugin to develop the program that puts the power of creation in your hands. Literally.

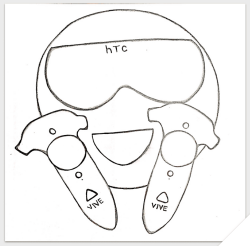
Simply pick up the HTC Vive controllers and headset to bring your imagination to life!

Oregon State University

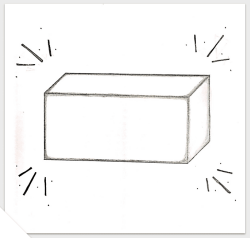
# DESIGNING IN VIRTUAL REALITY

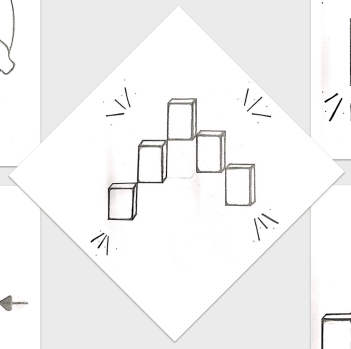
## Generative 3D Design in Architecture

1 HTC Vive Headset and Controllers

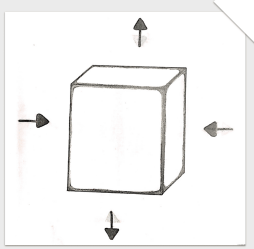



2 Generate 3D Object



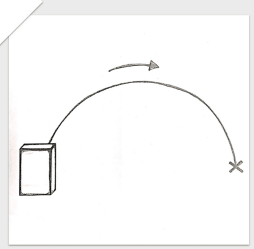


3 Resize 3D Object



5 Watch Structure Become [Virtual] Reality!

4 Draw Trajectory



## DESIGNING EXPLANATION

Power of Unity + Experience of VR + C# = An intuitive tool to create complex structures out of simple gestures and ideas

1) 3D objects can be spawned by selecting from the in-game menu via an HTC Vive controller.

2) Objects can be resized, combined, and altered according to the user's whim.

3) Curves can also be drawn mathematically, and then changed into various 3D structures.

4) Save and load objects and environments to come back to or continually cherish creations.

## WHAT HAPPENED IN THE END?

Usable HTC Vive Compatible VR Program!

Successes:

Program meets basic requirements of generating various 3D objects and creation of trajectories with mathematically curves, along with save and load functionalities.

Limitations:

User is unable to free-draw a curve, within a scene, but this addition is in the works for the future.

(Pictured from Left to Right)

**Nabeel Shariff, Hannah Solorzano, Rhea Mae Edwards**


**Raffaele de Amicis**

Associate Professor at Oregon State University, School of Electrical Engineering and Computer Science
***raffaele.deamicis@oregonstate.edu***

**Nabeel Shariff**

Computer Science Student focus in Business Entrepreneurship
***shariffn@oregonstate.edu***

**Hannah Solorzano**

Computer Science Student focus in Computer Graphics and Game Simulation
***solorzah@oregonstate.edu***

**Rhea Mae Edwards**

Computer Science Student focus in Computer Systems
***edwardrh@oregonstate.edu***

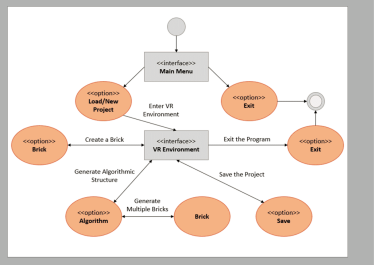




Figure 2. CS 463, Group 61, 3D Generative Design in Virtual Reality 2018 Oregon State University Engineering Expo Poster

# 7 Project Installation and Instructions

## 7.1 How the Project Works

### *7.1.1 Project's Structure*

The project's structure is composed of the following:

- iCreate Assets (this includes related prefabs, scenes, and scripts for the project)
- SteamVR plug in
- VRTK Asset

### *7.1.2 Theory of Operation*

The iCreate Program works by loading the user into the main menu scene, and then the user can navigate to the main scene where they can bring their creations into their virtual environment. In the main scene, the user has access to a main menu that allows them to generate their desired curves and objects by clicking on the respective button.

Additionally, the user can also precisely grab objects and scale or extrude them by using the grib buttons on the controller. Finally, the user can navigate through the environment via either teleporting or moving using the touchpad on the controller.

### *7.1.3 Class Diagram*

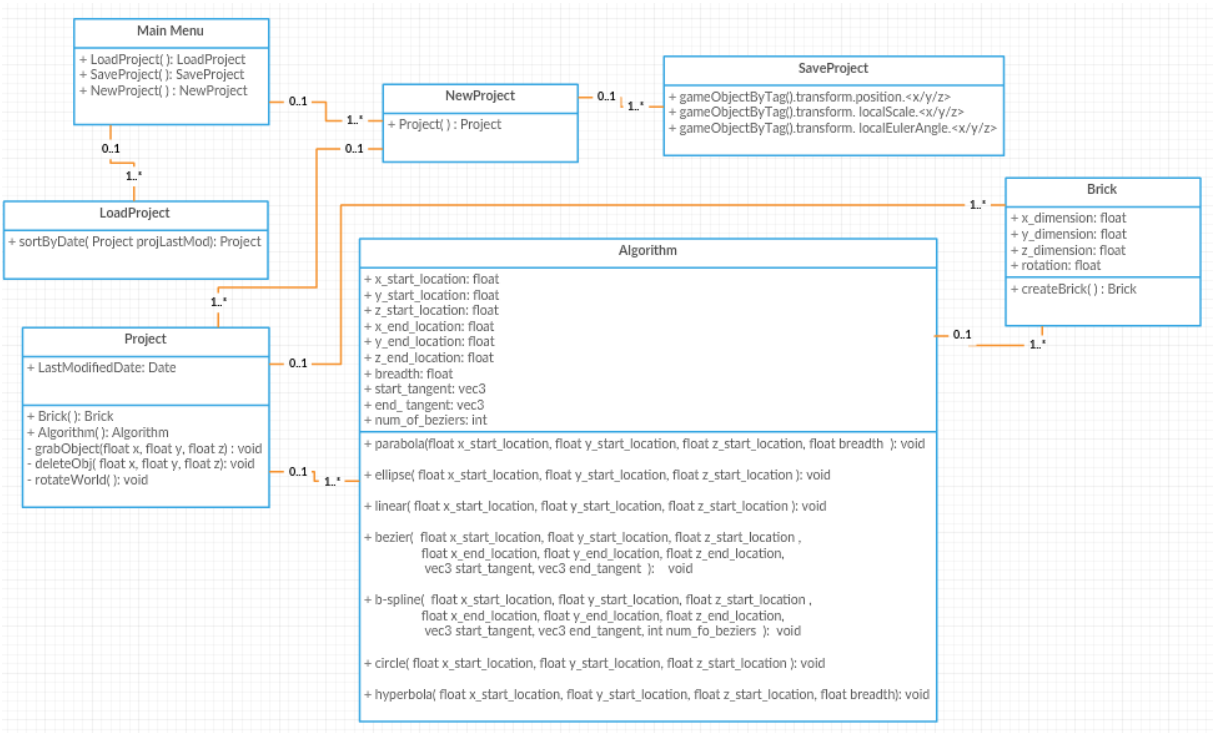



Figure 3. Class diagram that displays the methods available in iCreate, as well as the relationships between the different classes.

### *7.1.4 State Diagram*

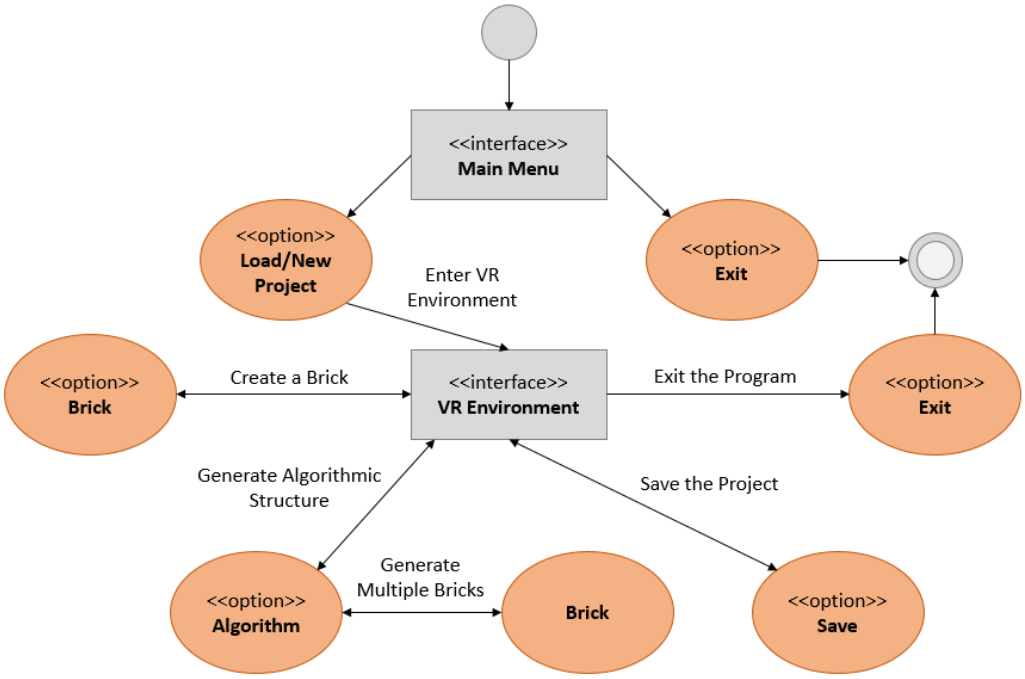



Figure 4. State diagram that discloses the various states and choices the user is presented with when using the iCreate program.

## 7.2 How to Install Software

To start, Unity and Steam VR need to be installed on the machine. To accomplish this, first download and install Unity latest release from their website:

`https://unity3d.com/get-unity/download`

Next, download and install Steam from their website:

`https://store.steampowered.com/about/`

Thirdly, download, install, and setup SteamVR through Steam. You can do this by following these two guides:

Pre-Installation Guide:

`https://support.steampowered.com/kb_article.php?ref=2001-UXCM-4439`

Installation Guide:

`https://support.steampowered.com/steamvr/HTC_Vive/`

Finally, download and extract the iCreate program via the following link:

http://web.engr.oregonstate.edu/~shariffn/iCreateStuff/iCreate%20Capstone%20Final%201.zip

### 7.3 How to Run Program

Firstly, open Unity and make sure it's running. Then, through Unity, open the iCreate project. Remember to select the main folder when choosing what project to open.

For a more detailed tutorial in getting started with Unity, you can refer to the following link:

https://docs.unity3d.com/Manual/GettingStarted.html

### 7.4 Equipment and Software

In order to run Unity and iCreate smoothly, the machine the software is being run on must meet the following minimum requirements:

GPU: Nvidia GeForce GTX 970, AMD Radeon R9 290 equivalent or better

CPU: Intel i5-4590, AMD FX 8350 equivalent or better

RAM: 4 GB or more

Video Output: HDMI 1.4, DisplayPort 1.2 or newer

USB Port: 1x USB 2.0 or better port

Operating System: Windows 7 SP1, Windows 8.1 or later, Windows 10

Specifically for iCreate, a Windows machine with an HTC Vive are required to properly run and experience the program.

### 7.5 Guide and Documentation

In this section, we have provided guides to download and install the VRTK asset the team has used for the VR implementation, as well as getting started with setting up SteamVR on Unity.

VRTK:

https://github.com/thestonefox/VRTK

SteamVR Unity Plugin:

https://assetstore.unity.com/packages/templates/systems/steamvr-plugin-32647

## 8 Technical Resources for Learning More

This section explains and provides more detail about the resources the team used to further their understanding of Unity and VR and to help build iCreate.

Firstly, the following tutorial helped us get started with setting up Unity and SteamVR:

https://www.raywenderlich.com/149239/htc-vive-tutorial-unity

Next, we used VRTK as our main tool/asset to help build the backbone of iCreate and its VR interactivity. The following link is a link that we used to help us get started with the asset, however there is a plethora of documentation for specific uses and scripts that were extremely useful to us:

`https://vrtoolkit.readme.io/docs/getting-started`

Thirdly, the following video was extremely useful in further facilitating the team's understanding of VRTK and its uses: Setting Up Doors with VRTK in Unity: Steam VR

`https://www.youtube.com/watch?v=iSj0X37KReI`

Finally, the most helpful person on campus that aided the team in their development process was Behnam Saeedi. With his help, the team was able to stay afloat and continue to learn and enjoy developing iCreate.

# 9 Team Experience

## 9.1 Hannah

*What technical information did you learn?*

During this project, I learned a lot about how VR programming works and how to scour documentation when looking to learn how to create a C# script or a feature of Unity. As I have had no prior experience with Unity, learning the basics of how to use it has taught me how to learn a new tool from scratch by asking colleagues and searching forums on the internet.

*What non-technical information did you learn?*

I learned how to document a program and what important documentation such as Design Documents and Software Specification documents should be formatted.

*What have you learned about project work?*

Communication and clarity are important things to have when working on a project as one person's misunderstanding of a concept can derail the entire project.

*What have you learned about project management?*

Learning how to divvy up the work into pieces that different members can work on is important. Being able to take a large project, segment it into sections, prioritizing features, and assigning the appropriate member to the task is important to the progression of the project.

*What have you learned about working in teams?*

Good communication amongst members is one of the most important things that a team can have. Without it, people are left without direction and things don't get done, or they get completed past their deadline.

*If you could do it all over, what would you do differently?*

I would ask questions earlier. In the Fall term when we were just learning about the details of the project, we would wait days or even weeks to follow up with our client to clarify what he meant or was asking for. Additionally, learning the process of using Unity early in the beginning would have been helpful as we spent a lot of time working on things that really did not benefit us, or help with the progression of the project.

## 9.2 Nabeel

*What technical information did you learn?*

The technical information I learned was in regards to writing scripts in c# for Unity. Specifically, learning how Unity handles behavior and how we as developers can manipulate that behavior via scripts was an interesting learning experience.

Apart from writing scripts, i learned a great deal of technical information regarding Unity as well. Unity itself is a robust engine capable of handling and managing behavior, especially through the addition of assets created by other developers. i learned the specifics of Unity and how to tie a project together in terms of scripts, assets, external tools, and other environment objects and models.

*What non-technical information did you learn?*

This project has taught me a lot of essential non-technical information. I've learned what it means to develop a project under stressful and tough circumstances, and how important teamwork is to everything. I've learned the hard way that no matter what, no matter how good one thinks they are, it is nearly impossible to make any progress without being proactive with your team. I've learned the importance of patience, but I've also learned that it is important to ask questions no matter how silly or irrelevant they are because clarity is integral to success. I've learned just how horrible and crippling any sort of ambiguity can be to a project.

I've also learned what it means to fail and correct a mistake continually, and to try to stay calm and focused and keep everything afloat. All in all, after learning from all my mistakes, I feel the non-technical information that I've learned will not only serve to help me become better as a computer scientist, but also a human being, so I can be a more helpful teammate in the future.

*What have you learned about project work?*

I've learned that regarding project work, clarity and teamwork are essential. Nothing can truly be accomplished alone, and for certain any ambiguity will simply kill any motivation and determination if it is not addressed immediately.

*What have you learned about project management?*

Regarding project management, I've learned just how important team communication is. There will always be some sort of gap between human beings, whether it is cultural or psychological, and that it is important to shorten this gap to facilitate further communication.

I've also learned that it is important to recognize team members' strengths and weaknesses and to delegate work accordingly to bring out the best from everyone to facilitate effectively and efficiency.

*What have you learned about working in teams?*

Regarding teams, I've learned communication is key. Any ambiguity that needs to be killed can easily be erased by communication. Additionally, cooperation is essential to get the best out of everyone so there is no hesitation to ask questions. I've also learned that fear is a huge factor that inhibits team performance, and even one person's fear about something can drag the whole team down. It is important to remain focused and to do what can and needs to be done, and to always remain proactive.

*If you could do it all over, what would you do differently?*

If I could do it all over again, I wouldn't hesitate with the client as much. I would be more proactive in finding out what exactly they wanted and how they could help facilitate our project and correct our mistakes.

Additionally, I myself would be more proactive in general. I'd get started with project much earlier, question the purpose and application of some of our project exercises, and consider using Unreal Engine more seriously over Unity.

Finally, I would try to be stronger as a student and team mate. I would try to be more calm, manage my time better, calculate for mistakes, learn and get better at scripting, and learn about Unity's finicky bugs much sooner.

In conclusion, I would try to be more proactive, communicate more, ask more questions, and remain adamant about the need for clarity concerning the project and team. And I would definitely want to be under Ben again.

### 9.3 Rhea Mae

*What technical information did you learn?*

The main concepts of technical information that I learned was in regards to writing the scripts for a virtual reality program in the programming language C# using the game engine Unity, in order for a user to interact with on a virtual reality headset being the HTC Vive for our group capstone project. I was unable to learn and earned experience in coding in the programming language C#, and building these types of scripts to work an corporate and working with an HTC Vive headset, bringing on challenges with certain command lines and functionalities being completely different from typical script coding techniques that I have implemented throughout my four years at Oregon State University.

*What non-technical information did you learn?*

Non-technical information that I learned this past year with this capstone experience were the effects that can cause stress when working with other or a client, not all necessarily because of who a person is, but the type of dynamic that can be created based on a variety of factors. I did not quite learn but instead enhanced my patience and professional skills together when it comes to communicating with others and in focusing on the main purpose and what is important, disregarding for a bit the fact on how others may treat you and act towards you.

*What have you learned about project work?*

One thing that I have learned about project work, is that it can be difficult to plan in the creation of project time-wise where the pieces of work required for a project to be created is unknown and unclear in the beginning of being requested. I also learned that it can be tough to solve a problem for a system that one may not even know how it works yet, especially with limited time, energy, and sources.

From previous experience, I have learned that previously planning out piece of work that needs to be completed for a project beforehand helps a lot in visually seeing what has been and needs to be done when working on a project. But with this capstone project, I have learned that it is rough to do so when the task at hand are not clear, and even with using one's sources, the tasks can still be unclear, which is where one may have to just try and see what work, but doing so takes up a lot time, where on a limited time-line, can be very stressful. I learned sometimes you just have to do what you can, and do your best and see what outcomes you come up with, working on the one that hopefully get you closer to your solution.

*What have you learned about project management?*

When is comes to project management, more in relation to this capstone project, is that sometimes you have to be put in a position even though such a position may not be a place where you do not want to be, but if you are able to take on such a responsibility and no one else would be able to, in order to increase the chance of success for the project at hand, then such an adaptation is necessary. Personally, I have greatly learned skills in managing a project team, such as taking the initiative to reach out to others when work has to get done, critical organizing work and individuals when time is greatly limited, and more. Overall, I have learned to take on a greater role than I was expecting in the beginning. In addition, communication, organization, and collaboration are all very important skills when it comes to managing a project efficiently.

*What have you learned about working in teams?*

In regards to working in teams, I have learned that there a variety people when put together, can create a very different type of team compared to another type of team. Variety is very important when it comes to teams. It is important to have people that have a variety of specialties, especially when it comes to a project with a short time-frame given the circumstances such a project may be given and considered in.

Also setting boundaries and rules in the beginning is very important when working in teams. There has to be a set of guidelines when working in a team, where some teams would need those guidelines more than other, and not only between the group members, but possibly even with a mentor/client to reduce the amount of confusion and conflict that may occur. Every team is different, and some teams work better together than other no matter what the project may be.

*If you could do it all over, what would you do differently?*

Even though I enjoyed learning and having the opportunity to learn more about programming a virtual reality program

in C# with a game engine, being Unity for this capstone project, if I could do it all over again, I would go back to the beginning, and choose a project that I not only would have interest in, but also a project that I have at least some more already previous knowledge in also in order to have a better chance in succeeding. I failed to realize the fact that learning new skills and software takes a lot of time and effort, where being a full time student in college and also individual working part-time during the year, can be extremely stressful and be negative to one's well being and health. But it is college, and we are all here to learn, even though all experiences may not be completely positive at times.

# APPENDIX A
# CLIENT NOTES

## A.1 Client Grade

This is the grade from the team of the project's client.

Rating: 2/20

## A.2 Overall Explanation

Present, but lack of understanding academically and culturally causes confusion and misdirection with such a limited time frame and knowledge for such a project given.

On a good note, our client attended most of our planned meetings with him, and answered e-mails, even though at times his responses or requests were not always clear. He offered to help us with our project, but when we went to go ask for help either in person or over e-mail, we were always quickly discouraged for asking in the first place.

As group we understood that our client struggled understanding the level of education and student standing our project was meant to be at. It felt like that our client thought of our group as a graduate project versus a undergraduate class project. The stresses, struggles, and the miss and lack of direction our group was put through, made a rough path as the year went on. We learned more in patience, roaming in the dark, struggling to always satisfying our client, versus honestly mostly learning a new technical skill in virtual reality, programming in C# within the gaming engine Unity for the virtual reality headset of an HTC Vive.

Capstone clients need to understand more on what being a client for an undergraduate computer science capstone project entails, understanding the group such a client has been given. Clients should not be setup for failure, but for success, and be given a learning experience that will be valuable to a student future endeavors. If the capstone course projects are meant for student to understand what having and working for a client is like in the industry, then all of those realities such as grace, should be given. In academics, if one rule is misunderstood or broken, no matter how hard one works, such efforts are overlooked, only the result is viewed, human actions are seen as minimal at times. Some improvement needs to be done.

# APPENDIX B
# ESSENTIAL CODE LISTINGS

## B.1 Creating and Spawning a Circle Curve

```
// Number of times object in placed on Circle
private int segments = 10;
// Width of the Object
public float objectWidth = 1;
// User Input Variables
public float xoffset = 0;
public float yoffset = 0;
public float zoffset = 0;
// Radius of the Circle
public float xradius = 4;
public float yradius = 4;
// List of Objects
public List<GameObject> cube;
// Some Variable
private LineRenderer line;
// User's GameObjects/objects
public GameObject parent_object;
// Public GameObject;
public GameObject main_object;

void Start () {

    // Having object spawn in front of user
    xoffset = Camera.main.transform.position.x;
    yoffset = Camera.main.transform.position.y;
    zoffset = Camera.main.transform.position.z + 6;

    // Calculating number of spaces an object will be placed at on circle
    segments = (int)(360 / (Mathf.Asin(objectWidth / (xradius)) * (180.0 / 3.1415)));

    // Setting up line components
    line = gameObject.GetComponent<LineRenderer>();
    line.positionCount = (segments + 1);
    line.useWorldSpace = false;

    // Calling function to place objects along circle
    CreatePoints();
}

void CreatePoints() {

    // Create a new list for objects
    cube = new List<GameObject>(segments);
    // Length of the list created above
    int list_length = 0;
    // Creating a GameObject for the object being used
    GameObject store_object;
    // Physical variables to use to define each object along the circle
    float x = 0f;
    float y = 0f;
    float z = 0f;
    // Initial angle for the object
    float angle = 20f;
```

```
    // Loop going through circle based off number of segments calculated earlier
    for (int i = 0; i < (segments + 1); i++) {

        // Calculating x and y of an object based off angle and radius of the circle
        x = Mathf.Sin(Mathf.Deg2Rad * angle) * xradius + xoffset;
        y = Mathf.Cos(Mathf.Deg2Rad * angle) * yradius + yoffset;

        // Setting position calculated
        line.SetPosition(i, new Vector3(x, y, z));

        // Placing the user's object along the curve
        store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                       as GameObject;

        // Inserting object to list
        cube.Insert(i, store_object);

        // Calculating next angle based off an additional segment calculated from earlier
        angle += (360f / segments);
    }

    // Counts the number of GameObjects in the list
    list_length = cube.Count;
    // Makes all of the elements in the list above children of the user's
       created GameObject previously
    for (int i = 0; i < list_length; i++) {
        cube[i].transform.parent = main_object.transform;
    }
}
```

This code explained above generates a circle curve by inputs given by default or by the user (which the code has to be further modified to do so), along with the created and modified game object of the user to place along the trajectory of the then calculated circle curve. The later generated structure of the circle curve of the user's game object is first placed by default in front of the user's camera by the offset values calculated in this script being six space in front of the user. Then from calculated segments being the number of times the user's game object is place along the circle curve, a for loop calculates each x and y position for a single game object taking into consideration the offset location.

The line renderer is also implemented to display a continuous line of the generated circle curve, and each placed game object along the circle curve is then stored as a child game object of an empty game object this script is attached to with the program, in order to use and view the curve as a single game object in a sense, along with the ability to modify single game objects of the circle curve structure itself afterwards.

In addition, to further explain the ability to modify this curve, is through the public float variables initially stated within this script. Since these are public variables the user can change these to their liking and manipulate this within a game environment when implemented successfully.

### B.2 Creating and Spawning an Ellipse Curve

```
// Equation for an Ellipse
// 1 = (((x-h)^2)/a^2) + (((y-v)^2)/b^2)

// Number of times object in placed on Ellipse
private int segments = 10;
// Width of the Object
public float objectWidth = 1;
// User Input Variables
public float xoffset = 0; // h of equation
```

```
public float yoffset = 0; // v of equation
public float zoffset = 0;
// Radius of the Ellipse
public float xradius = 4; // a of equation
public float yradius = 2; // b of equation
// List of Objects
public List<GameObject> cube;
// Some Variable
private LineRenderer line;
// User's GameObjects/objects
public GameObject parent_object;
// Public GameObject;
public GameObject main_object;

void Start () {

    // Having object spawn in front of user
    xoffset = Camera.main.transform.position.x;
    yoffset = Camera.main.transform.position.y;
    zoffset = Camera.main.transform.position.z + 6;

    // Calculating number of spaces an object will be placed at on Ellipse
    // segements = perimeter of Ellipse / objectWidth
    int segs = (int)(((2 * 3.1415) * Mathf.Sqrt(((xradius * xradius) +
               (yradius * yradius)) / 2)) / objectWidth);
    segments = (5*segs) / 4;

    // Setting up line components
    line = gameObject.GetComponent<LineRenderer>();
    line.positionCount = (segments + 1);
    line.useWorldSpace = false;

    // Calling function to place objects along Ellipse
    CreatePoints();
}

void CreatePoints() {

    // Create a new list for objects
    cube = new List<GameObject>(segments);
    // Length of the list created above
    int list_length = 0;
    // Creating a GameObject for the object being used
    GameObject store_object;
    // Physical variables to use to define each object along the Ellipse
    float x = 0f; // x of equation
    float y = 0f; // y of equation
    float z = 0f;
    // Initial angle for the object
    float angle = 20f;

    // Loop going through Ellipse based off number of segments calculated earlier
    for (int i = 0; i < (segments + 1); i++) {
        // Calculating x and y of an object based off angle and radii of the Ellipse
        // x = a*cos(angle)
        // y = b*sin(angle)
        x = xradius * Mathf.Cos(Mathf.Deg2Rad * angle) + xoffset;
        y = yradius * Mathf.Sin(Mathf.Deg2Rad * angle) + yoffset;
```

```
        // Setting position calculated
        line.SetPosition(i, new Vector3(x, y, z));

        // Placing the user's object along the curve
        store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                       as GameObject;

        // Inserting object to list
        cube.Insert(i, store_object);

        // Calculating next angle based off an additional segment calculated from earlier
        angle += (360f / segments);
    }

    // Counts the number of GameObjects in the list
    list_length = cube.Count;
    // Makes all of the elements in the list above children of the user's
       created GameObject previously
    for (int i = 0; i < list_length; i++) {
        cube[i].transform.parent = main_object.transform;
    }
}
```

This code explained above generates an ellipse curve by inputs given by default or by the user (which the code has to be further modified to do so), along with the created and modified game object of the user to place along the trajectory of the then calculated ellipse curve. The later generated structure of the ellipse curve of the user's game object is first placed by default in front of the user's camera by the offset values calculated in this script being six space in front of the user. Then from calculated segments being the number of times the user's game object is place along the ellipse curve, a for loop calculates each x and y position for a single game object taking into consideration the offset location.

The line renderer is also implemented to display a continuous line of the generated ellipse curve, and each placed game object along the ellipse curve is then stored as a child game object of an empty game object this script is attached to with the program, in order to use and view the curve as a single game object in a sense, along with the ability to modify single game objects of the ellipse curve structure itself afterwards.

In addition, to further explain the ability to modify this curve, is through the public float variables initially stated within this script. Since these are public variables the user can change these to their liking and manipulate this within a game environment when implemented successfully.

### B.3 Creating and Spawning a Parabola Curve

```
// Equation for an Parabola (Vertex Form)
// y = a(x-h)^2 + k

// Number of times object in placed on Parabola
public int segments = 10;
// Width of the Object
public float objectWidth = 1;
public float objectHeight = 1;
// User Input Variables
public float xoffset = 0; // h or x0 of equation
public float yoffset = 0; // v or y0 of equation
public float zoffset = 0;
// x-coefficient of the Parabola
public float a = 1; // a or p of equation
// List of Objects
public List<GameObject> cube;
// Some Variable
```

```
private LineRenderer line;
// User's GameObjects/objects
public GameObject parent_object;
// Public GameObject;
public GameObject main_object;

void Start () {

    // Having object spawn in front of user
    xoffset = Camera.main.transform.position.x;
    yoffset = Camera.main.transform.position.y;
    zoffset = Camera.main.transform.position.z + 6;

    // Setting up line components
    line = gameObject.GetComponent<LineRenderer>();
    line.positionCount = (segments + 1);
    line.useWorldSpace = false;

    // Calling function to place objects along Parabola
    CreatePoints();
}

void CreatePoints() {

    // Create a new list for objects
    cube = new List<GameObject>(segments);
    // Creating a GameObject for the object being used
    GameObject store_object;
    // Physical variables to use to define each object along the Parabola
    float x = 0f; // x of equation
    float y = 0f; // y of equation
    float z = 0f;
    // Starting angle for the object
    float angle = 20f;

    // Loop going through Parabola based off number of segments inputted
       by user initially
    for (int i = 0; i < (segments + 1); i++) {
        // Calculating x and y of an object based off equation for a Parabola
        // y = a(x-h)^2 + k
        // Inverse: x = sqrt((y-k) / a) + h
        y = y + objectHeight;
        x = (Mathf.Sqrt(y - yoffset) / a) + xoffset;

        // Setting position calculated
        line.SetPosition(i, new Vector3(x, y, z));

        // Placing the user's object along the curve
        store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                       as GameObject;

        // Inserting object to list
        cube.Insert(i, store_object);

        // Makes the element/GameObject above a child of the user's
           created GameObject previously
        cube[i].transform.parent = main_object.transform;

        // Graphing the other side of the Parabola
```

```
        /*************************************************************************/

        // Calculating left-side of the Parabola
        x = (-1) * (Mathf.Sqrt(y - yoffset) / a) + xoffset;

        // Setting position calculated
        line.SetPosition(i, new Vector3(x, y, z));

        // Placing the user's object along the curve
        store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                       as GameObject;

        // Inserting object to list
        cube.Insert(i, store_object);

        // Makes the element/GameObject above a child of the user's
           created GameObject previously
        cube[i].transform.parent = main_object.transform;

        /*************************************************************************/

        // Calculating next angle based off an additional segment calculated from earlier
        angle += (360f / segments);
    }
}
```

This code explained above generates a parabola curve by inputs given by default or by the user (which the code has to be further modified to do so), along with the created and modified game object of the user to place along the trajectory of the then calculated parabola curve. The later generated structure of the parabola curve of the user's game object is first placed by default in front of the user's camera by the offset values calculated in this script being six space in front of the user.

Then from the inputted dimensions for the parabola curve, a for loop with curve's equation calculates each x and y position for a single game object taking into consideration the offset location. In this implementation, the y position of the object along a parabola curve will only depend on the height of the object, where the x position is then later calculated by using the parabola equation based off the current y position initially calculated. There are two sides that needs to be considered when creating this parabola curve. So within the for loop, with the current y position, the opposite side of the parabola curve is calculated by calculating the negative value of the x position within this implementation.

The line renderer is also implemented to display a continuous line of the generated circle curve, and each placed game object along the circle curve is then stored as a child game object of an empty game object this script is attached to with the program, in order to use and view the curve as a single game object in a sense, along with the ability to modify single game objects of the circle curve structure itself afterwards.

In addition, to further explain the ability to modify this curve, is through the public float variables initially stated within this script. Since these are public variables the user can change these to their liking and manipulate this within a game environment when implemented successfully.

### B.4 Creating and Spawning a Hyperbola Curve

```
// Equation for an Hyperbola (Vertex Form)
// 1 = (((x-h)^2)/a^2) - (((y-k)^2)/b^2)

// Number of times object in placed on Hyperbola
public int segments = 10;
// Width of the Object
public float objectWidth = 1;
// User Input Variables
```

```
public float xoffset = 0; // h of equation
public float yoffset = 0; // v or k of equation
public float zoffset = 0;
// Asymptotes of the Hyperbola
public float xradius = 1; // a of equation
public float yradius = 2; // b of equation
// List of Objects
public List<GameObject> cube;
// Some Variable
private LineRenderer line;
// User's GameObjects/objects
public GameObject parent_object;
// Public GameObject;
public GameObject main_object;

void Start () {

    // Having object spawn in front of user
    xoffset = Camera.main.transform.position.x;
    yoffset = Camera.main.transform.position.y;
    zoffset = Camera.main.transform.position.z + 6;

    // Setting up line components
    line = gameObject.GetComponent<LineRenderer>();
    line.positionCount = (segments + 1);
    line.useWorldSpace = false;

    // Calling function to place objects along Hyperbola
    CreatePoints();
}

void CreatePoints() {

    // Create a new list for objects
    cube = new List<GameObject>(segments);
    // Creating a GameObject for the object being used
    GameObject store_object;
    // Physical variables to use to define each object along the Hyperbola
    float x = 0f; // x of equation
    float y = 0f; // y of equation
    float z = 0f;
    // Starting angle for the object
    float angle = 20f;

    // Loop going through Hyperbola based off number of segments inputted
       by user earlier
    for (int i = 0; i < (segments + 1); i++) {

        /*************************************************************************/
        // Calculating Curve 1, Side 1 of Hyperbola
        /*************************************************************************/

        // Calculating x and y of an object based off the equation for a Hyperbola
        // Solving for y in respects to x
        // y = sqrt(((((x-h)^2) / a^2) - 1) * b^2) + k
        x = x + objectWidth;
        y = Mathf.Sqrt((((Mathf.Pow((x - xoffset), 2)) / Mathf.Pow(xradius, 2)) - 1) /
            Mathf.Pow(yradius, 2)) + yoffset;
```

```
// Setting position calculated
line.SetPosition(i, new Vector3(x, y, z));

// Placing the user's object along the curve
store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
               as GameObject;

// Inserting object to list
cube.Insert(i, store_object);

// Makes the element/GameObject above a child of the user's
   created GameObject previously
cube[i].transform.parent = main_object.transform;

// Calculating Curve 1, Side 2 of Hyperbola
/*************************************************************************/

// Calculating lower half of this side of the Hyperbola
y = y * (-1);

// Setting position calculated
line.SetPosition(i, new Vector3(x, y, z));

// Placing the user's object along the curve
store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
               as GameObject;

// Inserting object to list
cube.Insert(i, store_object);

// Makes the element/GameObject above a child of the user's
   created GameObject previously
cube[i].transform.parent = main_object.transform;

// Converting y-value back to positive value
y = y * (-1);

/*************************************************************************/
// Calculating Curve 2, Side 1 of Hyperbola
/*************************************************************************/

// Calculating x and y of an object based off the equations for a Hyperbola
// Solving for y in respects to x
// y = sqrt(((((x-h)^2) / a^2) - 1) * b^2) + k
x = (-1) * x;
y = Mathf.Sqrt((((Mathf.Pow((x - xoffset), 2)) / Mathf.Pow(xradius, 2)) - 1) /
    Mathf.Pow(yradius, 2)) + yoffset;

// Setting position calculated
line.SetPosition(i, new Vector3(x, y, z));

// Placing the user's object along the curve
store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
               as GameObject;

// Inserting object to list
cube.Insert(i, store_object);

// Makes the element/GameObject above a child of the user's
```

```
            created GameObject previously
        cube[i].transform.parent = main_object.transform;

        // Calculating Curve 2, Side 2 of Hyperbola
        /*************************************************************************/

        // Calculating lower half of this side of the Hyperbola
        y = y * (-1);

        // Setting position calculated
        line.SetPosition(i, new Vector3(x, y, z));

        // Placing the user's object along the curve
        store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                       as GameObject;

        // Inserting object to list
        cube.Insert(i, store_object);

        // Makes the element/GameObject above a child of the user's
            created GameObject previously
        cube[i].transform.parent = main_object.transform;

        // Converting y-value back to positive value
        y = y * (-1);

        /*************************************************************************/

        // Converting x-value back to positive value
        x = (-1) * x;

        // Calculating next angle based off an additional segment calculated from earlier
        angle += (360f / segments);
    }
}
```

This code explained above generates a hyperbola curve by inputs given by default or by the user (which the code has to be further modified to do so), along with the created and modified game object of the user to place along the trajectory of the then calculated hyperbola curve. The later generated structure of the hyperbola curve of the user's game object is first placed by default in front of the user's camera by the offset values calculated in this script being six space in front of the user.

Then from the inputted dimensions for the hyperbola curve, a for loop with curve's equation calculates each x and y position for a single game object taking into consideration the offset location. In this implementation, the x position of the object along a parabola curve will only depend on the width of the object, where the y position is then later calculated by using the hyperbola equation based off the current x position initially calculated. There are two sides of two curves that needs to be considered when creating this hyperbola curve. So within the for loop, the current y position is also read on the opposite side of the hyperbola curve given the current x position, completing one curve of the total hyperbola curve. Then the second curve is calculated along with the second curve's sides with a similar process, considering the opposite x and y positions of in comparison in world space, which is further explain within the code above.

The line renderer is also implemented to display a continuous line of the generated circle curve, and each placed game object along the circle curve is then stored as a child game object of an empty game object this script is attached to with the program, in order to use and view the curve as a single game object in a sense, along with the ability to modify single game objects of the circle curve structure itself afterwards.

In addition, to further explain the ability to modify this curve, is through the public float variables initially stated within this script. Since these are public variables the user can change these to their liking and manipulate this within a game

environment when implemented successfully.

### B.5 Creating and Spawning a Bezier Curve

```
// Number of times object in placed on Bezier Curve
public int segments = 10;
// Width of the Object
public float objectWidth = 1;
// User Input Variables, Points of Bezier Curve
public float x1 = -5; // x1 of curve
public float y1 = -1; // y1 of curve
public float x2 = -1; // x2 of curve
public float y2 = 5; // y2 of curve
public float x3 = 1; // x3 of curve
public float y3 = -5; // y3 of curve
public float x4 = 5; // x4 of curve
public float y4 = 1; // y4 of curve
// List of Objects
public List<GameObject> cube;
// Some Variable
private LineRenderer line;
// User's GameObjects/objects
public GameObject parent_object;
// Public GameObject;
public GameObject main_object;

void Start () {

    // Setting up line components
    line = gameObject.GetComponent<LineRenderer>();
    line.positionCount = (segments + 1);
    line.useWorldSpace = false;

    // Calling function to place objects along Bezier Curve
    CreatePoints();
}

void CreatePoints() {

    // Create a new list for objects
    cube = new List<GameObject>(segments);
    // Length of the list created above
    int list_length = 0;
    // Creating a GameObject for the object being used
    GameObject store_object;
    // Physical variables to use to define each object along the Bezier Curve
    float x = 0f; // x of equation
    float y = 0f; // y of equation
    float z = 0f;
    // Initial t for the object
    float t = 0f;

    // Loop going through Bezier Curve based off number of segments inputted
       by user earlier
    for (int i = 0; i < (segments + 1); i++) {

        // Calculating t variable for Bezier curve equations
        t = i / (float)segments;
```

```
        // Calculating x and y of an object based off the four points and t
           of the Bezier Curve
        // x = (1-t)^3*x1 + 3(1-t)^2*t*x2 + 3(1-t)*t^2*x3 + t^3*x4, where 0 < t< 1
        // y = (1-t)^3*y1 + 3(1-t)^2*t*y2 + 3(1-t)*t^2*y3 + t^3*y4, where 0 < t < 1
        x = (Mathf.Pow((1 - t), 3) * x1) + (3 * Mathf.Pow((1 - t), 2) * t * x2) +
            (3 * (1 - t) * Mathf.Pow(t, 2) * x3) + (Mathf.Pow(t, 3) * x4);
        y = (Mathf.Pow((1 - t), 3) * y1) + (3 * Mathf.Pow((1 - t), 2) * t * y2) +
            (3 * (1 - t) * Mathf.Pow(t, 2) * y3) + (Mathf.Pow(t, 3) * y4);

        // Setting position calculated
        line.SetPosition(i, new Vector3(x, y, z));

        // Placing the user's object along the curve
        store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                       as GameObject;

        // Inserting object to list
        cube.Insert(i, store_object);
    }

    // Counts the number of GameObjects in the list
    list_length = cube.Count;
    // Makes all of the elements in the list above children of the user's
       created GameObject previously
    for (int i = 0; i < list_length; i++) {
        cube[i].transform.parent = main_object.transform;
    }
}
```

This code explained above generates a Bezier curve by inputs given by default or by the user (which the code has to be further modified to do so), along with the created and modified game object of the user to place along the trajectory of the then calculated Bezier curve. The later generated structure of the Bezier curve of the user's game object is first placed by default in front of the user's camera by the offset values calculated in this script being six space in front of the user.

Then from the inputted dimensions for the Bezier curve, a for loop with curve's equation calculates each x and y position for a single game object taking into consideration the offset location. The script is given a variable t considering the number of segments inputted by the user, indicating how many instances of their object they would liked spawn along their wanted Bezier curve. The characteristics of a Bezier curve includes the input of four points which is given by default in this implementation and can also be given by the user to specify. The equations to calculate a Bezier curve takes in all of the needed set values to calculate the x and y positions for each object of the user.

The line renderer is also implemented to display a continuous line of the generated circle curve, and each placed game object along the circle curve is then stored as a child game object of an empty game object this script is attached to with the program, in order to use and view the curve as a single game object in a sense, along with the ability to modify single game objects of the circle curve structure itself afterwards.

In addition, to further explain the ability to modify this curve, is through the public float variables initially stated within this script. Since these are public variables the user can change these to their liking and manipulate this within a game environment when implemented successfully.

### B.6 Creating and Spawning a B-Spline Curve

```
// Number of times object in placed on a B-Spline Curve
public int segments = 7;
// Width of the Object
public float objectWidth = 1;
```

```
// User Input Variables, Points of B-Spline Curve 1
public float x1 = -4; // x1 of curve
public float y1 = -3; // y1 of curve
public float x2 = -7; // x2 of curve
public float y2 = -1; // y2 of curve
public float x3 = -5; // x3 of curve
public float y3 = 4; // y3 of curve
public float x4 = -3; // x4 of curve
public float y4 = 6; // y4 of curve
public float x5 = -1; // x5 of curve
public float y5 = 7; // y5 of curve
public float x6 = 1; // x6 of curve
public float y6 = 4; // y6 of curve
public float x7 = 3; // x7 of curve
public float y7 = 6; // y7 of curve
public float x8 = 5; // x8 of curve
public float y8 = 7; // y8 of curve
public float x9 = 3; // x9 of curve
public float y9 = 9; // y9 of curve
public float x10 = 1; // x10 of curve
public float y10 = 11; // y10 of curve
public float x11 = 2; // x11 of curve
public float y11 = 13; // y11 of curve
public float x12 = 8; // x12 of curve
public float y12 = 10; // y12 of curve
public float x13 = 7; // x13 of curve
public float y13 = 7; // y13 of curve
// List of Objects
public List<GameObject> cube;
// Some Variable
private LineRenderer line;
// User's GameObjects/objects
public GameObject parent_object;
// Public GameObject;
public GameObject main_object;

void Start () {

    // Setting up line components
    line = gameObject.GetComponent<LineRenderer>();
    line.positionCount = (segments + 1);
    line.useWorldSpace = false;

    // Calling function to place objects along B-Spline Curve
    CreatePoints();
}

void CreatePoints() {

    // Create a new list for objects
    cube = new List<GameObject>(segments);
    // Length of the list created above
    int list_length = 0;
    // Creating a GameObject for the object being used
    GameObject store_object;
    // Physical variables to use to define each object along the B-Spline Curve
    float x = 0f; // x of equation
    float y = 0f; // y of equation
    float z = 0f;
```

```
// Initial t for the object
float t = 0f;

// Loop going through B-Spline Curve based off number of segments inputted
   by user earlier

/*****************************************************************************/
// Calculating Curve 1 of B-Spline Curve
/*****************************************************************************/
for (int i = 0; i < (segments + 1); i++) {

    // Calculating t variable for B-Spline curve equation
    t = i / (float)segments;

    // Calculating x and y of an object based off the four points and t
       of the B-Spline Curve
    // x = (1-t)^3*x1 + 3(1-t)^2*t*x2 + 3(1-t)*t^2*x3 + t^3*x4, where 0 < t < 1
    // y = (1-t)^3*y1 + 3(1-t)^2*t*y2 + 3(1-t)*t^2*y3 + t^3*y4, where 0 < t < 1
    x = (Mathf.Pow((1 - t), 3) * x1) + (3 * Mathf.Pow((1 - t), 2) * t * x2) +
        (3 * (1 - t) * Mathf.Pow(t, 2) * x3) + (Mathf.Pow(t, 3) * x4);
    y = (Mathf.Pow((1 - t), 3) * y1) + (3 * Mathf.Pow((1 - t), 2) * t * y2) +
        (3 * (1 - t) * Mathf.Pow(t, 2) * y3) + (Mathf.Pow(t, 3) * y4);

    // Setting position calculated
    line.SetPosition(i, new Vector3(x, y, z));

    // Placing the user's object along the curve
    store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                   as GameObject;

    // Inserting object to list
    cube.Insert(i, store_object);
}

// Counts the number of GameObjects in the list
list_length = cube.Count;
// Makes all of the elements in the list above children of the user's
   created GameObject previously
for (int i = 0; i < list_length; i++) {
    cube[i].transform.parent = main_object.transform;
}

/*****************************************************************************/
// Calculating Curve 2 of B-Spline Curve
/*****************************************************************************/
for (int i = 0; i < (segments + 1); i++) {

    // Calculating t variable for B-Spline curve equation
    t = i / (float)segments;

    // Calculating x and y of an object based off the four points and t
       of the B-Spline Curve
    // x = (1-t)^3*x1 + 3(1-t)^2*t*x2 + 3(1-t)*t^2*x3 + t^3*x4, where 0 < t < 1
    // y = (1-t)^3*y1 + 3(1-t)^2*t*y2 + 3(1-t)*t^2*y3 + t^3*y4, where 0 < t < 1
    x = (Mathf.Pow((1 - t), 3) * x4) + (3 * Mathf.Pow((1 - t), 2) * t * x5) +
        (3 * (1 - t) * Mathf.Pow(t, 2) * x6) + (Mathf.Pow(t, 3) * x7);
    y = (Mathf.Pow((1 - t), 3) * y4) + (3 * Mathf.Pow((1 - t), 2) * t * y5) +
        (3 * (1 - t) * Mathf.Pow(t, 2) * y6) + (Mathf.Pow(t, 3) * y7);
```

```
    // Setting position calculated
    line.SetPosition(i, new Vector3(x, y, z));

    // Placing the user's object along the curve
    store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                   as GameObject;

    // Inserting object to list
    cube.Insert(i, store_object);
}

// Counts the number of GameObjects in the list
list_length = cube.Count;
// Makes all of the elements in the list above children of the user's
   created GameObject previously
for (int i = 0; i < list_length; i++) {
    cube[i].transform.parent = main_object.transform;
}

/*****************************************************************************/
// Calculating Curve 3 of B-Spline Curve
/*****************************************************************************/
for (int i = 0; i < (segments + 1); i++) {

    // Calculating t variable for B-Spline curve equation
    t = i / (float)segments;

    // Calculating x and y of an object based off the four points and t
       of the B-Spline Curve
    // x = (1-t)^3*x1 + 3(1-t)^2*t*x2 + 3(1-t)*t^2*x3 + t^3*x4, where 0 < t < 1
    // y = (1-t)^3*y1 + 3(1-t)^2*t*y2 + 3(1-t)*t^2*y3 + t^3*y4, where 0 < t < 1
    x = (Mathf.Pow((1 - t), 3) * x7) + (3 * Mathf.Pow((1 - t), 2) * t * x8) +
        (3 * (1 - t) * Mathf.Pow(t, 2) * x9) + (Mathf.Pow(t, 3) * x10);
    y = (Mathf.Pow((1 - t), 3) * y7) + (3 * Mathf.Pow((1 - t), 2) * t * y8) +
        (3 * (1 - t) * Mathf.Pow(t, 2) * y9) + (Mathf.Pow(t, 3) * y10);

    // Setting position calculated
    line.SetPosition(i, new Vector3(x, y, z));

    // Placing the user's object along the curve
    store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                   as GameObject;

    // Inserting object to list
    cube.Insert(i, store_object);
}

// Counts the number of GameObjects in the list
list_length = cube.Count;
// Makes all of the elements in the list above children of the user's
   created GameObject previously
for (int i = 0; i < list_length; i++) {
    cube[i].transform.parent = main_object.transform;
}

/*****************************************************************************/
// Calculating Curve 4 of B-Spline Curve
/*****************************************************************************/
for (int i = 0; i < (segments + 1); i++) {
```

```
        // Calculating t variable for B-Spline curve equation
        t = i / (float)segments;

        // Calculating x and y of an object based off the four points and t
           of the B-Spline Curve
        // x = (1-t)^3*x1 + 3(1-t)^2*t*x2 + 3(1-t)*t^2*x3 + t^3*x4, where 0 < t < 1
        // y = (1-t)^3*y1 + 3(1-t)^2*t*y2 + 3(1-t)*t^2*y3 + t^3*y4, where 0 < t < 1
        x = (Mathf.Pow((1 - t), 3) * x10) + (3 * Mathf.Pow((1 - t), 2) * t * x11) +
            (3 * (1 - t) * Mathf.Pow(t, 2) * x12) + (Mathf.Pow(t, 3) * x13);
        y = (Mathf.Pow((1 - t), 3) * y10) + (3 * Mathf.Pow((1 - t), 2) * t * y11) +
            (3 * (1 - t) * Mathf.Pow(t, 2) * y12) + (Mathf.Pow(t, 3) * y13);

        // Setting position calculated
        line.SetPosition(i, new Vector3(x, y, z));

        // Placing the user's object along the curve
        store_object = Instantiate(parent_object, new Vector3(x, y, z), Quaternion.identity)
                       as GameObject;

        // Inserting object to list
        cube.Insert(i, store_object);
    }

    // Counts the number of GameObjects in the list
    list_length = cube.Count;
    // Makes all of the elements in the list above children of the user's
       created GameObject previously
    for (int i = 0; i < list_length; i++) {
        cube[i].transform.parent = main_object.transform;
    }
}
```

This code explained above generates a B-Spline curve by inputs given by default or by the user (which the code has to be further modified to do so), along with the created and modified game object of the user to place along the trajectory of the then calculated B-Spline curve. The later generated structure of the B-Spline curve of the user's game object is first placed by default in front of the user's camera by the offset values calculated in this script being six space in front of the user.

Then from the inputted dimensions for the B-Spline curve, a for loop with curve's equation calculates each x and y position for a single game object taking into consideration the offset location. The script is given a variable t considering the number of segments inputted by the user, indicating how many instances of their object they would liked spawn along each of the four sections of their B-Spline curve. The characteristics of a B-Spline curve includes the input of four points for four sections of the B-Spline curve at minimum, which is given by default in this implementation and can also be given by the user to specify, representing fluidity. The equations to calculate a B-Spline curve takes in all of the needed set values to calculate the x and y positions for each object of the user.

The line renderer is also implemented to display a continuous line of the generated circle curve, and each placed game object along the circle curve is then stored as a child game object of an empty game object this script is attached to with the program, in order to use and view the curve as a single game object in a sense, along with the ability to modify single game objects of the circle curve structure itself afterwards.

In addition, to further explain the ability to modify this curve, is through the public float variables initially stated within this script. Since these are public variables the user can change these to their liking and manipulate this within a game environment when implemented successfully.

### B.7 Saving GameObject Characteristics (Save Functionality)

```
public GameObject blockList;
string s = "";
string p1, p2, p3;
string sc1, sc2, sc3;
string r1, r2, r3;
string type;

    // Loops through all of the GameObjects in the list of blocks
    // Finds and Stores each GameObject's position, scaling, rotation, and type
    foreach (Transform myChild in blockList.GetComponentsInChildren<Transform>()) {
        s = string.Concat(s, "(");
        p1 = (myChild.position.x).ToString();
        p2 = (myChild.position.y).ToString();
        p3 = (myChild.position.z).ToString();
        s = string.Concat(s, p1, ",", p2, ",", p3, ",");
        sc1 = (myChild.localScale.x).ToString();
        sc2 = (myChild.localScale.y).ToString();
        sc3 = (myChild.localScale.z).ToString();
        s = string.Concat(s, sc1, ",", sc2, ",", sc3, ",");
        r1 = (myChild.eulerAngles.x).ToString();
        r2 = (myChild.eulerAngles.y).ToString();
        r3 = (myChild.eulerAngles.z).ToString();
        s = string.Concat(s, r1, ",", r2, ",", r3, ",");
        type = "T";
        s = string.Concat(s, type, ")\n");
    }
```

This script displays the execution method used to store float and string values into an ongoing string later to be stored in a file of game objects being saved within game scene. For now, the string value to indicate the type of game object being saved is valued at T for all game objects with this current save implementation, which will be fixed to be more appropriate for use in later development.

### B.8 Splits a String into Usable Descriptive Components (Load Functionality)

```
string textIn = "";
string[] object_list;
ArrayList character_list = new ArrayList();

    // Tokenizes GameObjects described in the string from the file by newline (\n)
    // Adds objects in a string array, object_list
    object_list = textIn.Split('\n');

    foreach (string gameobj in object_list) {
        string obj;

        // Ignoring the first and last character on each line
           (the opening and closing parantheses (( and )))
        // Puts a GameObject in a variable called obj for useage
        obj = gameobj.Replace("(", "");
        obj = obj.Replace(")", "");

        // Tokenize each component in each of the GameObjects by comma (,)
        // Takes a GameObject's components and stores them in a string array (item_list)
        string[] item_list;
        item_list = obj.Split(',');
```

```
        // Inputs each component in an ongoing Arraylist called character_list
        foreach (string item in item_list) {
            character_list.Add(item);
        }
    }
```

Above, this piece of code represents the implementation of splitting up the saved string back into small usable pieces to use to load game objects back into a game scene. The extra characters that were used in order to save the string consisting of information of saved game objects in an orderly fashion, is now stripped off and removed in order to extract values to be properly used, being stored in an arraylist within the process.

### B.9 Creating and Loading GameObjects (Load Functionality)

```
public List<GameObject> spawn_list;
ArrayList character_list = new ArrayList();
int count;
int spawn_index = 0;
GameObject tempObject;
spawn_list = new List<GameObject>();
float po1, po2, po3;
float sca1, sca2, sca3;
float ro1, ro2, ro3;

    // Create/Spawn a GameObject(s) based of the components in the
       Arraylist (character_list)

    // (character_list[0], character_list[1], character_list[2], character_list[3],
        character_list[4], character_list[5], character_list[6],
        character_list[7], character_list[8], character_list[9])
        and so on/repeat in the same list

    // (float, float, float, float, float, float, float, float, float, string)
    // (p1, p2, p3, sc1, sc2, sc3, r1, r2, r3, type)
    // (object, object, object, object, object, object, object, object, object, object)

    // For Loop Algorithm:
    // For every set of 10 elements in character_list AKA every one object
    // - Check which type the object is, every 10 steps,
         first occurrence: character_list[9] and so on..
    // - Covert strings/objects back into floats, except for the type,
         as the following occur:
    // - Get the x, y, z position of object
    // - Get the x, y, z scale of object
    // - Get the x, y, z rotation of object
    // - Spawn the object into the game scene

    count = character_list.Count;

    for (int i = 0; i < count - 1; i++) {
        // i += 9; at the end every time an object is spawned

        string tempType = Convert.ToString(character_list[i + 9]);

        if (tempType == "T") {
            po1 = Convert.ToSingle(character_list[i]);
            po2 = Convert.ToSingle(character_list[i + 1]);
```

```
            po3 = Convert.ToSingle(character_list[i + 2]);
            sca1 = Convert.ToSingle(character_list[i + 3]);
            sca2 = Convert.ToSingle(character_list[i + 4]);
            sca3 = Convert.ToSingle(character_list[i + 5]);
            ro1 = Convert.ToSingle(character_list[i + 6]);
            ro2 = Convert.ToSingle(character_list[i + 7]);
            ro3 = Convert.ToSingle(character_list[i + 8]);

            tempObject = GameObject.CreatePrimitive(PrimitiveType.Cube);

            tempObject.transform.position = new Vector3(po1, po2, po3);
            tempObject.transform.localScale = new Vector3(sca1, sca2, sca3);
            tempObject.transform.eulerAngles = new Vector3(ro1, ro2, ro3);

            spawn_list.Insert(spawn_index, tempObject);
            spawn_index += 1;

            i += 9;
        }
        else {
            Debug.Log("Unable to create a GameObject.");
            i += 9;
        }
    }
```

The code above illustrates the creation and spawning of game objects back into a game scene. The for loop implemented for this extraction and spawning functionality, iterates through the specified arraylist. Depending on the type of game object being read, will depend which game object is loaded back into the game scene. Note that with this current implementation, cubes are only being generated which the same string value T that is inputted for all of the previous game objects saved. Then based off where certain float values are previous systematically placed within the arraylist, will then determine which float values are which to later be used in creating game objects being loaded back into a game scene.

If a game object for some reason cannot be properly loaded back into a game scene, the program will currently only output an error message to the debug console log, and then move on until the loop has finished iterating through the arraylist.

# APPENDIX C
# PROJECT PHOTOS