PENNSTATE
1855

Virtual Reality---Texting While Driving

Jacob Wheeler: Major in SE
Nathan Christiansen: Major in SE
Nicholas Kapty: Major in SE

Course Instructor: Xiaocong Fan

Faculty Advisor: George Dudas

Industry Sponsor: Erie Insurance

Project Mentor: Matthew Panetta

IT Apprentice

Project Proposer: MaryJo Ingalls

Supervisor Enterprise Content Management

A capstone project report submitted to the faculty of

The Computer Science and Software Engineering Department

Penn State Erie, The Behrend College

April 2017
(Version 4.0)

Contents

1. Abstract

Erie Insurance currently works with its agents to help them display the dangers of distracted driving to their policyholders. This can often be very difficult for agents to do since the user is not able to experience the consequences of distracted driving for themselves in a safe way. In order to help solve this problem for the agents, we are creating a virtual reality experience to allow for better engagement between agents and the teen drivers they work with. This virtual reality experience will utilize the Unity 3D engine and the Google Cardboard SDK to give the policyholder different scenarios in which they will have to make decisions in which they will have to prevent the driver from texting. This virtual reality experience will help the policyholder to understand how they can influence dangerous driving activities as well as to help stop them. The overall goal of this virtual reality experience is to give young drivers a fun, memorable experience with their agent and to help encourage them not to drive while distracted.

2. Report Revision History

2.1 Changes in Version 1.5

In this version, we have made the changes recommended to us by our advisor. We have added a new user requirement and functional requirement detailing more information regarding the specific tasks that the AI driver should perform. The use case mapping diagram has updated as well. Along with that, we have changed the name of our use case "Begin Experience" to "Experience Loop" to make more sense. References are now available and are used in section 5 to further explain our exploratory studies.

2.2 Changes in Version 2.0

In this version, we have added our initial designs for the architecture, structure, interface, and behavior of the system. We have changed our architecture to the component-based architecture, which more accurately captures the way Unity objects build off each other to create the overall system. We have added and updated our requirements based off feedback from advisors and industry mentor. We have created test cases for our system, as well as the execution history. We have added the steps to set up the development environment and testing environment, build for the target platform, and install to the end user's device.

2.3 Changes in Version 2.5

In this version, we have made further changes recommended to us by our faculty advisor and industry mentor. We have modified the layout of the report in section 6.2 to better organize the descriptions that go with each individual image. We have also updated section 6.3 to be contained within one page for further formatting improvements.

2.4 Changes in Version 3.0

In this version, we have modified our requirements and descriptions to match the new direction that the project is taking. We are going with a less realistic and serious approach, and the new information reflects these changes. We have also edited the report so that it now reflects the desired report structure specified on the Capstone project management system.

2.5 Changes in Version 3.5

In this version, we have modified section 6.5 to better describe the new scenario system. Although scenarios are not as complex as they once were when using the old menu system, we are still referring to these gameplay elements as scenarios. System requirement SO-02-01 has been updated to state that Erie Insurance logos will be placed on "environmental objects" rather than on specific items that will most likely not be represented in the experience. Lastly, we have modified the execution report of Test Case TC-009 to include nothing about conversing with the driver.

2.6 Changes in Version 4.0

In this version, we have added to section 10.3 to describe the steps to allow the end user to run the application. We have also updated section 7.5, which outlines the driving system and the scenario system that we currently have in place. Most images have been updated in section 6.3 to demonstrate the most recent user interface. Section 6.5 has been updated to discuss the team's reasoning to add the convertible car. Section 9.3 has been updated to talk about the open issues that we are currently facing and their possible solutions.

3. Problem Statement

3.1 Business Background

Erie Insurance is a Fortune 500 insurance company employing thousands of people. Erie Insurance has been a figure in the insurance world for 90 years, and currently serves over 4 million customers in 13 states. They utilize and manage smaller agencies to deal directly with customers, selling them auto, home, life, and business insurance.

With the rise of technology, distracted driving has become more of a risk than ever before. As Erie Insurance invests in protecting people, they are taking the initiative in informing families about the dangers of driving while distracted.

3.2 Needs

Currently, it is very difficult to display the dangers of distracted driving to a younger generation in a way that engages them. Erie Insurance is seeking an innovative solution in order to solve this problem. The business sponsors of this project are interested in a product that allows for an open discussion on the dangers of texting and driving. While this project is not aimed at teaching policyholders to refrain from texting and driving, it will be used to help the agent share something fun while having that discussion with the young driver. The business sponsors need a solution to the problem of getting young drivers to remember this discussion. This project will also encourage policyholders to share their experiences with their agent by discussing the project with others.

3.3 Objectives

This project aims to utilize virtual reality technology to create an immersive experience that engages users of all ages. The application will be distributed to agents around Erie's footprint and will effectively capture the younger audience. The main objective of this project is to meet the needs of our industry partner, Erie Insurance, which are explained above. The business sponsors of the project need a way to help young drivers remember the dangers of distracted driving that have been explained to them by their Erie Insurance agent, which is a gap that this project aims to fill.

4. Requirements

4.1 User Requirements

4.1.1 Glossary of Relevant Domain Terminology

Virtual Reality (VR) – A simulation of a three-dimensional environment

Cardboard – Google's SDK created for smartphone devices

Headset – A head mounted device that displays virtual reality devices

Scene – A Unity scene is an aggregation of components that can be executed on its own

4.1.2 User Groups

User – Any person engaging in the experience

4.1.3 Functional Requirements

4.1.3.1 Project Scope (Use Case Diagram)

Figure 4.1 displays the system's use case diagram. This gives a layout of the main user interactions that can occur as they use the system. The VR – Texting and Driving application consists of two main use cases: Experience Loop and Solve Scenario. The user will navigate through the experience and solve scenarios which will result in a final outcome at the end of the experience.
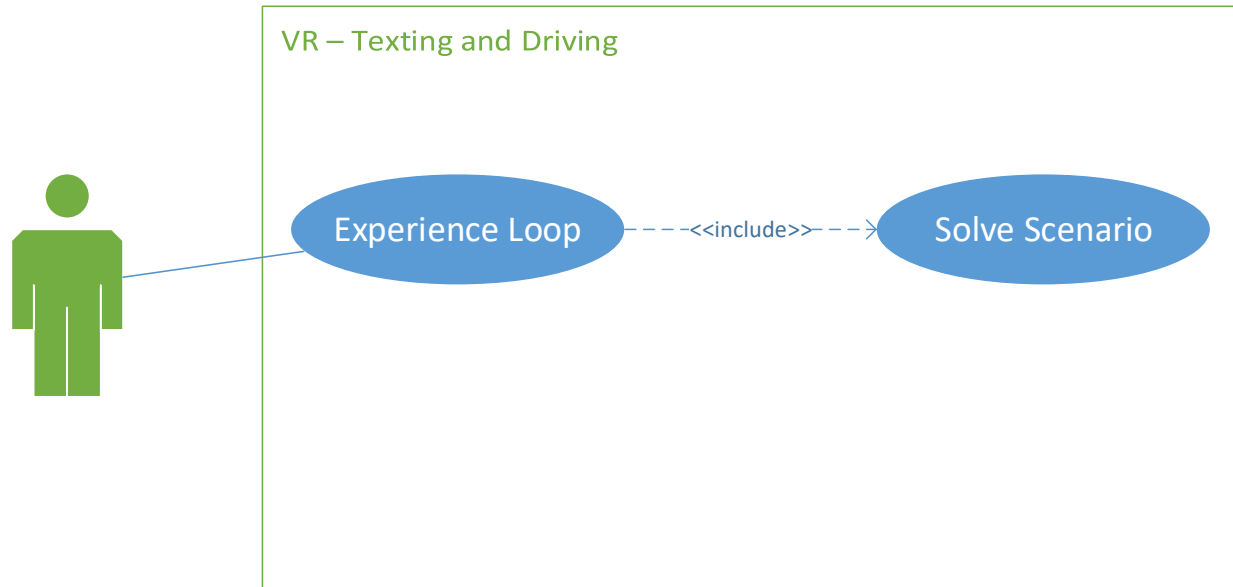

Figure 4.1 - Use Case Diagram

4.1.3.2   User Scenarios

Table 4.1 lists the details of the use cases that occur within the system. The use cases give an overview of the sequence of the interactions that occur with the user and the system. The scenario list consists of Experience Loop and Solve Scenario, which explained in greater detail in Table 4.2 and Table 4.3, respectively.

4.1.3.3   List of User Functional Requirements

User functional requirements describe functionality that the system should provide. The following will provide context to each of the system's user functional requirements.

Table 4.4 describes that the application should present a variety of different scenarios to the user. These scenarios should all feature a distracted driver. The user should be able to overcome the potential negative effects that may occur due to the carelessness of the driver.

Table 4.5 describes the main action of the user, which is to control the camera in the front passenger seat of the vehicle. The passenger will be in control of himself the entire time while the driver navigates through the various scenes.

Table 4.6 describes that the system should feature multiple outcomes that result from the actions of the driver influenced by the user. Similar to Table 4.4, there should be multiple types of scenarios as well as multiple types of outcomes resulting from those scenarios.

Table 4.7 describes that the user should be able to interact with the environment while within the experience loop. This may include things such as interacting with items in the car.

Table 4.8 describes that an AI will control the car driving. This AI will take the passenger's decisions into consideration while he engages in various tasks throughout the duration of the experience.

4.1.4   Non-functional Requirements

Non-functional requirements describe the constraints and quality of the functionalities, providing testable features and specifying restrictions.

4.1.4.1   Product: Usability Requirements

Usability requirements describe how easily a user interacts with the system.

4.1.4.2   Product: Performance Requirements

Performance requirements describe how well a system performs in terms of time and resource usage.

Table 4.9 describes that the system should run at an acceptable frame rate for virtual reality use. This is to prevent users from becoming ill during their time in the experience.

4.1.4.3   Product: Dependability/Security Requirements

Dependability/Security requirements describe the reliability and security concerns of the project.

4.1.4.4   Organizational: Development Requirements

Development requirements specify development practices and constraints.

Table 4.10 describes that the VR – Texting and Driving application should be designed with modern Android devices in mind. Virtual reality needs this to support our system and keep it running at an acceptable frame rate.

Table 4.11 explains that we developed the system for Google Cardboard. There are currently many VR headsets on the market, however, Erie Insurance would like to keep access to the application relatively cheap.

Table 4.12 states that the system must feature Erie Insurance logos throughout the duration of the experience. Erie Insurance would like to clearly show off their brand while keeping it within the context of the experience.

4.1.4.5 Organizational: Operational Requirements

Operational requirements describe conditions that a system must support.

4.1.4.6 Organizational: Environmental Requirements

Environmental requirements describe the look and feel of the system's interface.

4.1.4.7 External: Safety/Security Requirements

Safety/Security requirements detail how the system will interact with other systems, and the security concerns of these interactions.

4.1.4.8 External: Cultural and Social Requirements

Cultural and social requirements describe how the system conforms to cultural and social expectations.

4.1.4.9 External: Political Requirements

Political requirements detail how the system will influence different sections of the company.

4.2 System Requirements

User requirements tend to be vague, so they are refined into system requirements. System requirements engineer and refine the user requirements into many detailed requirements that are much more descriptive and implementable.

4.2.1 Functional Requirements

4.2.1.1 List of System Functional Requirements

Table 4.13 describes that the system should feature two possible scenarios for every decision presented to the user. This further defines the user requirement for use in the system.

Table 4.14 further discusses how the user views a scenario. Each scenario will have its own respective trigger to begin the scenario.

Table 4.15 describes the view and input that the user has during the experience. The user will have a first-person view at all times, in which the user is able to move their head around to rotate the camera.

Table 4.16 explains that the user will be able to use on input device on the Google Cardboard, which is a single button. This button allows the user to interact with their environment, and will allow the user to make decisions for each scenario presented to them.

Table 4.17 explains how to solve the problem of the application beginning in the incorrect position. Due to the incorrect positioning, the user may not be facing towards the front of the car. To solve this, a double click system allows the user to re-center the camera to face the front of the car.

Table 4.18 gives a more detailed look into what outcomes will possible in regards to scenarios. Outcomes that can occur include falling objects, avoiding collisions, and moving off the path.

Table 4.19 describes what the user should be able to interact with in between scenarios. These things include grabbing a drink from the cup holder, opening and closing the window, and adjusting the radio.

Table 4.20 further details what the AI driver should do while driving the car. This includes texting while driving, however, he should stop texting when the user looks in his direction.

4.2.1.2   System Behavior

Figure 4.2 details the sequence of flow between user and system. The use cases are covered by the diagram, which provides an in-depth depiction of the use cases for the system.
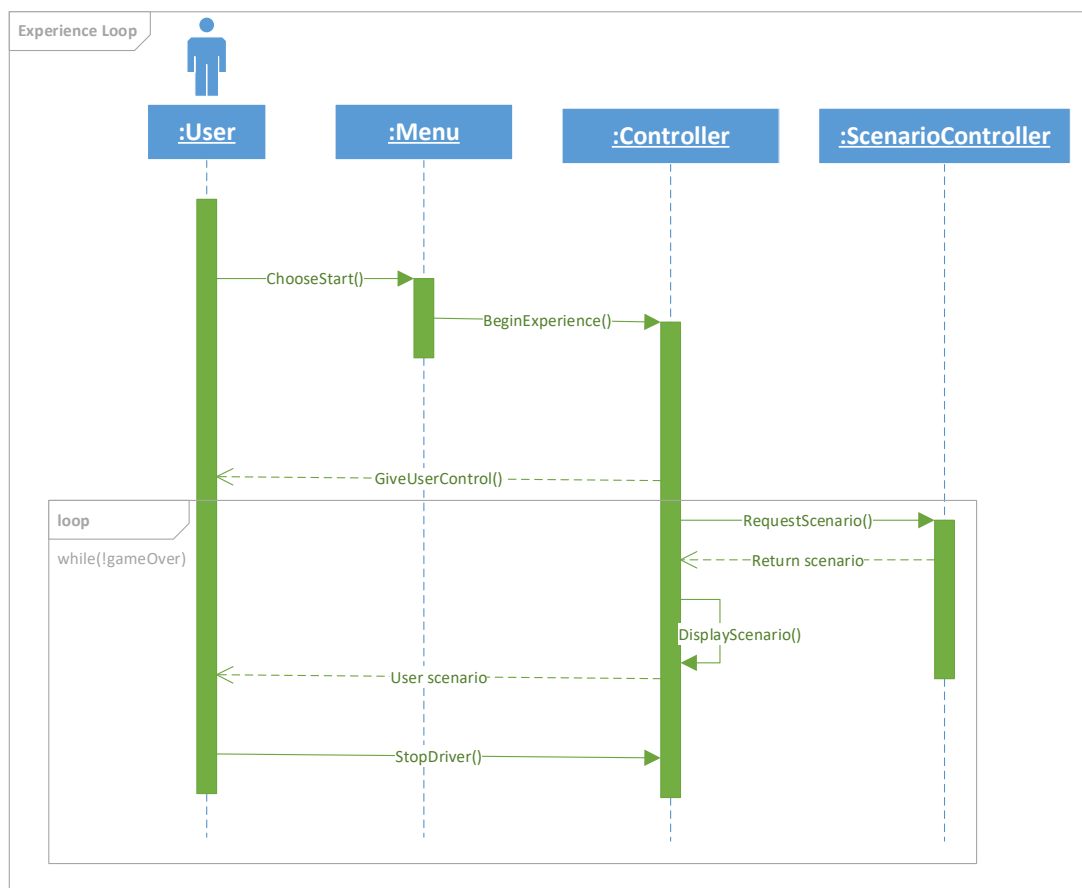
Figure 4.2 - Experience Loop Sequence

4.2.1.3   Data Requirements

N/A

4.2.2   Non-functional Requirements

4.2.2.1   Product: Usability Requirements

N/A

4.2.2.2   Product: Performance Requirements

Table 4.21 explains that the system should run at 30 frames per second. This will ensure a fairly smooth VR experience on a variety of hardware.

4.2.2.3   Product: Dependability/Security Requirements

N/A

4.2.2.4   Organizational: Development Requirements

Table 4.22 gives explicit detail as to which version of Android the application uses.

Table 4.23 discusses that the system will use features of the Google VR SDK to render the dual image to create the VR effect.

Table 4.24 explains in further detail how the environments feature Erie Insurance logos. Buildings, billboards, bumper stickers, and air fresheners may contain these logos.

4.2.2.5   Organizational: Operational Requirements

N/A

4.2.2.6   Organizational: Environmental Requirements

N/A

4.2.2.7   External: Safety/Security Requirements

N/A

4.2.2.8   External: Cultural and Social Requirements

N/A

4.2.2.9   External: Political Requirements

N/A

4.3   Requirements Trace Table

Table 4.25 gives a breakdown of the user requirements and system requirements mapping.

5. Exploratory Studies

5.1   Relevant Techniques

We will be using the Unity 3D game engine to create our application. We have chosen this engine because of its C# scripting, large community, and because it allows us to create an immersive VR experience very quickly. Along with Unity 3D, we will be using the Google VR SDK for Unity to adapt our project for VR use [6]. We also plan to take advantage of the Unity Asset Store to collect models, animations, and scripts to allow us to focus on implementing the requested features and not worry about having to create all of our assets from scratch. Within the Asset Store exists an important package called Unity Test Tools [4]. Unity Test Tools allows us various ways of testing including unit tests, integration tests, and assertion component to make sure our work is as bug free as possible. All of these technologies working together will allow us to create an experience that puts the user into the middle of a seemingly dangerous situation.

5.2   Relevant Packages/Products

The main products and packages we will be using include Unity 3D, Google VR SDK, a variety of assets from the Unity Asset Store, the Android SDK to build from within the Unity engine, Unity Test Tools to complete our application testing, Visual Studio for writing C# scripts, and potentially more as we move forward.

5.3   Broader Impacts

This virtual reality experience has the potential to help minimize distracted driving by providing a system to insurance agents that will help them better communicate with young drivers when explaining the dangers that come with distracted driving. Minimizing distracted driving means that there will be less accidents, less injuries, and less deaths because of distracted driving. Since the application runs on the Android operating system, which is used by millions of people every day, this application has the potential to reach a large number of drivers and passengers.

6. System Design

6.1 Architectural Design

The system will be using a component-based architectural design, which emphasizes the creation of components, which other components reuse to create a scene. Multiple scenes are sequenced together to create the overall system. Unity objects are a component that is self-contained, meaning that it can run on its own inside a scene. As objects are defined, they can be used in other objects to create large components that are combined to create complex scenes. Figure 6.1 shows our high-level architecture, which is consisting of a starting interface *GameObject* that has a composition with itself to allow the components to have other components that make it up.
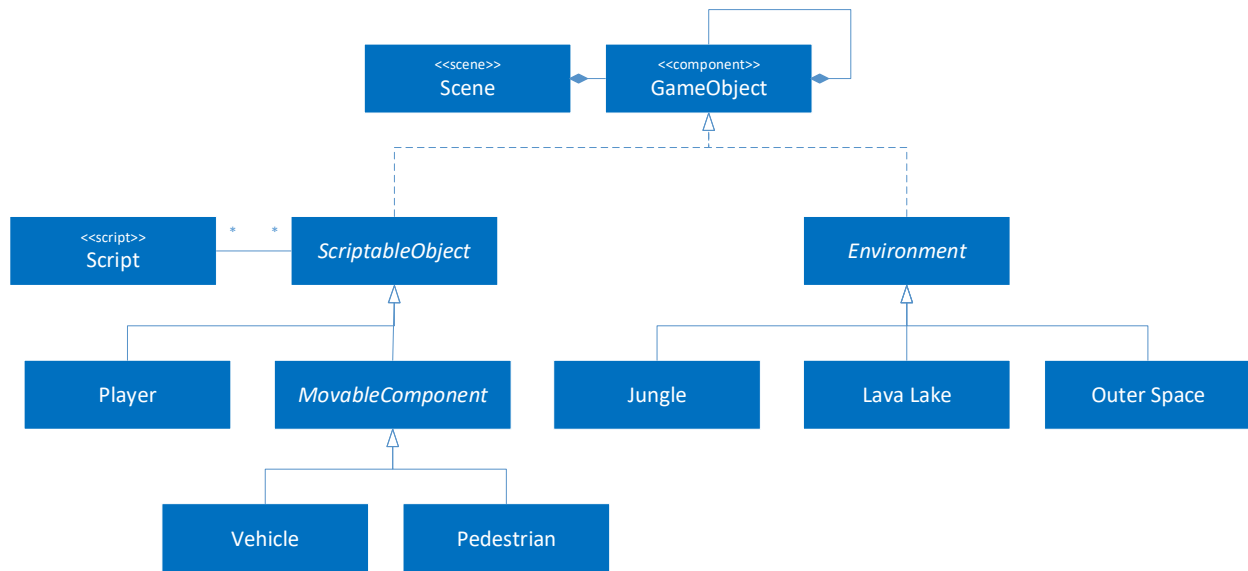
Figure 6.1 - Architectural Design

6.2   Structural Design

The structural diagram provides the detailed components that are defined in the architectural design. The basic components are refined into each individual component that create the overall layout of the Unity scene.

Figure 6.2 represents the Scenes package within the structural diagram. This package will contain each scene within the experience and show how they connect to each other. This package also contains the main GameObject interface that all other components will be inheriting from throughout the system.
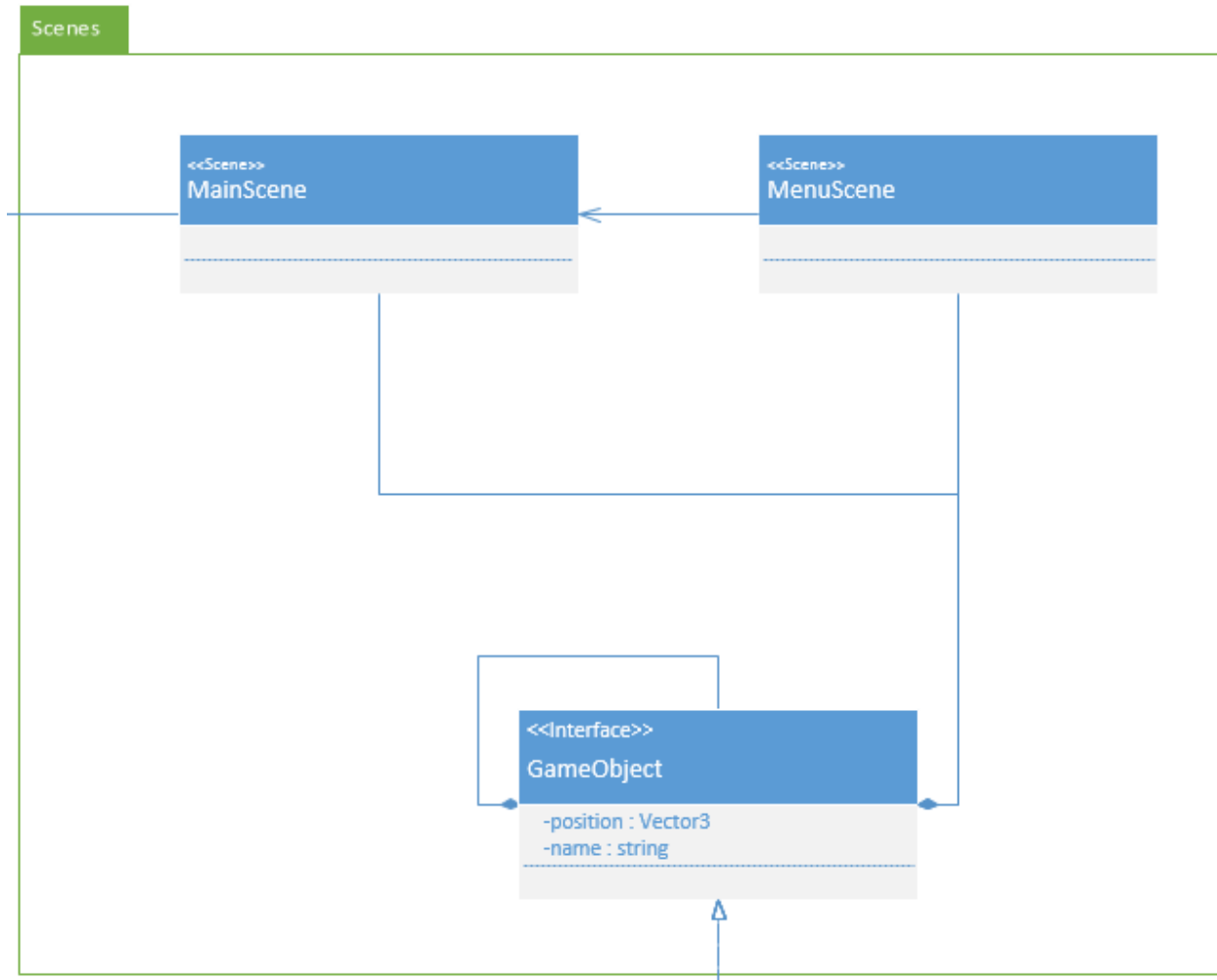
Figure 6.2 - Structural Diagram (Scenes)

Figure 6.3 is the package representing the player. The player is essentially a camera component using a player controller script to allow the user to move their head around to view and interact with what is happening in the scene. The player controller is able to perform actions with the environment such as reentering the user, which is shown below.
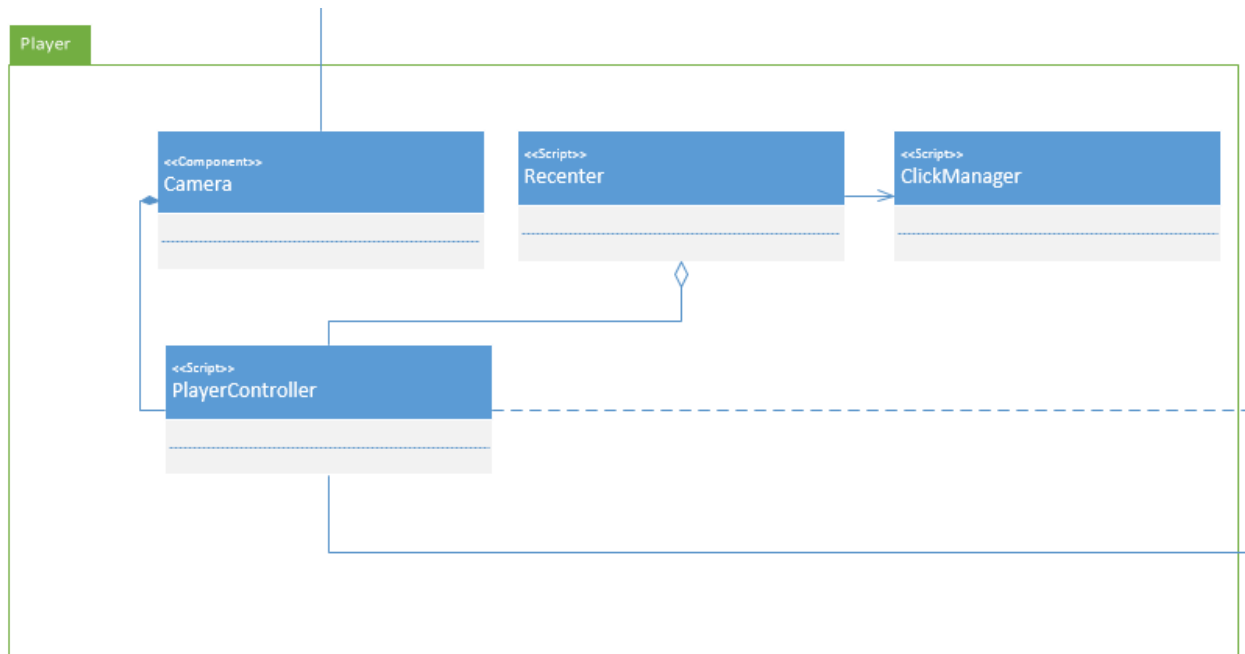
Figure 6.3 - Structural Diagram (Player)

Figure 6.4 shows many of the components that come from Google's VR SDK. As mentioned above, the player is a camera that is able to interact with the environment. To do this, the camera utilizes components, interfaces, and scripts in this package. This package allows components to be set as either objects causing interactions to happen or allows components to be the object that is interacted with.
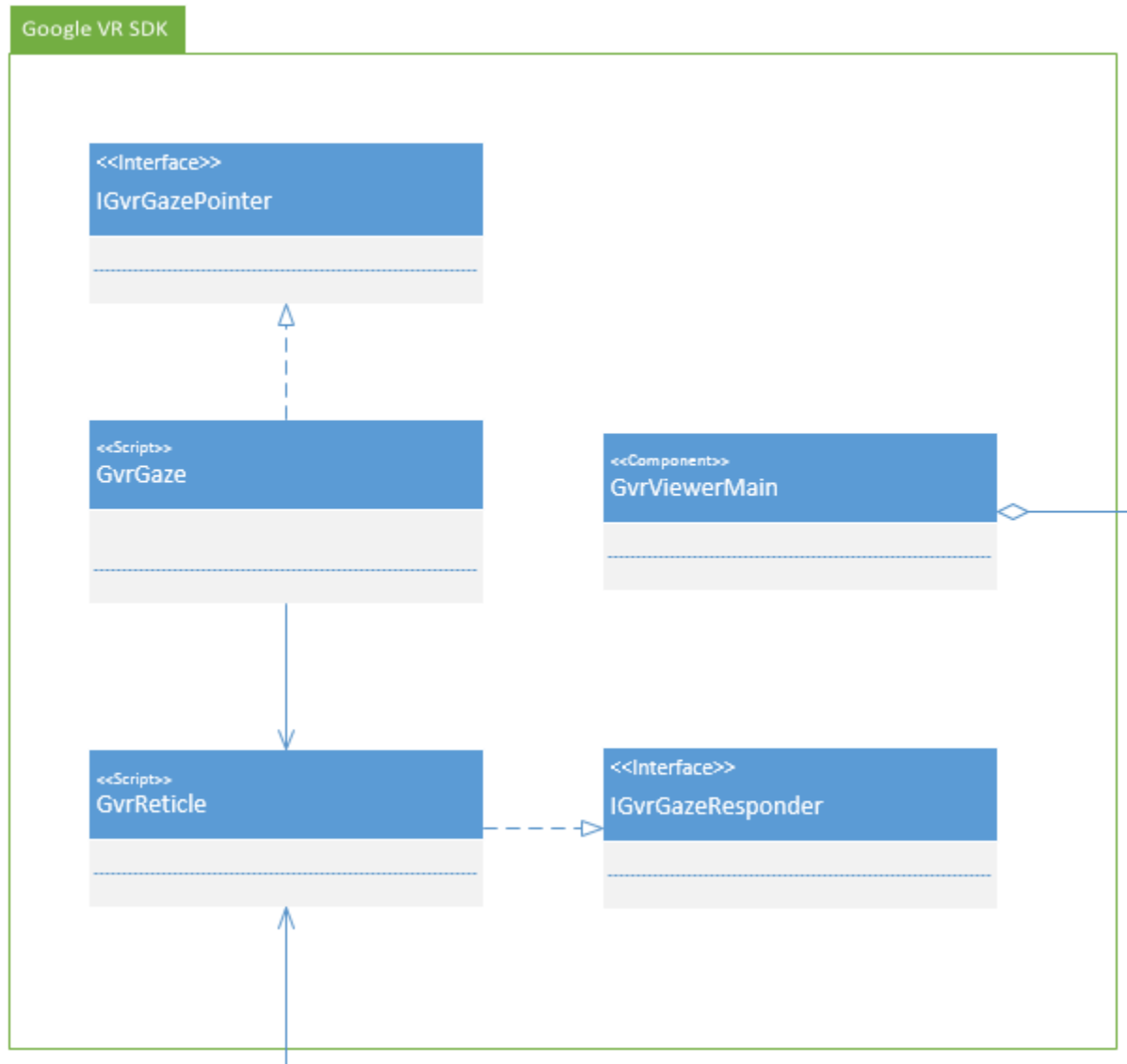
Figure 6.4 - Structural Diagram (Google VR SDK)

Figure 6.5 shows the MovableComponent package which consists of all components that will be moving in some way during the execution of the program. This package includes pedestrians (people, animals, etc.) and vehicles. The package also contains the scripts that these components will rely on to perform their movement and coordination.
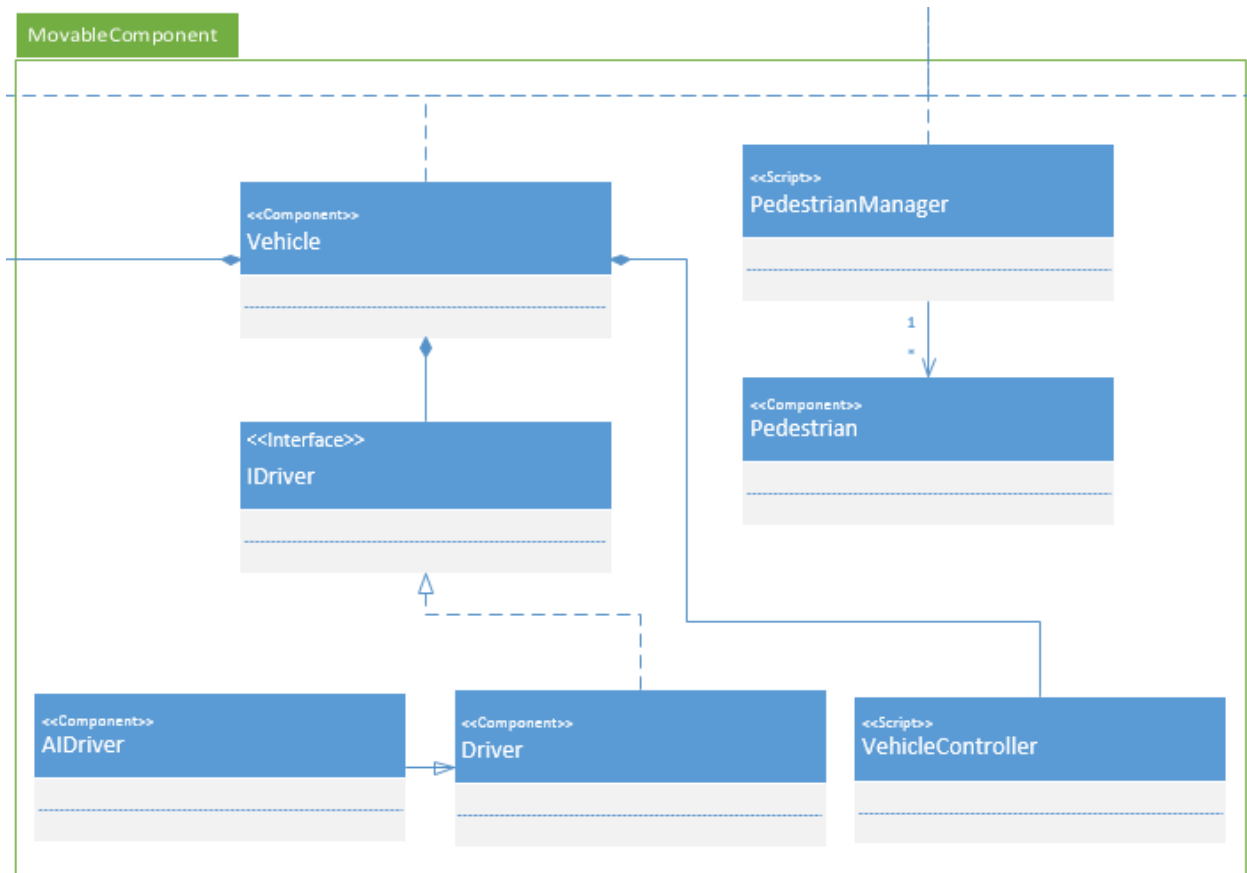
Figure 6.5 - Structural Diagram (MovableComponent)

Figure 6.6 shows the EnvironmentalObjects package which contains objects that are non-moving and exist in the environment such as plants, buildings, and roadways. The hierarchy below demonstrates how full environments will be made up of smaller components such as what was listed previously.
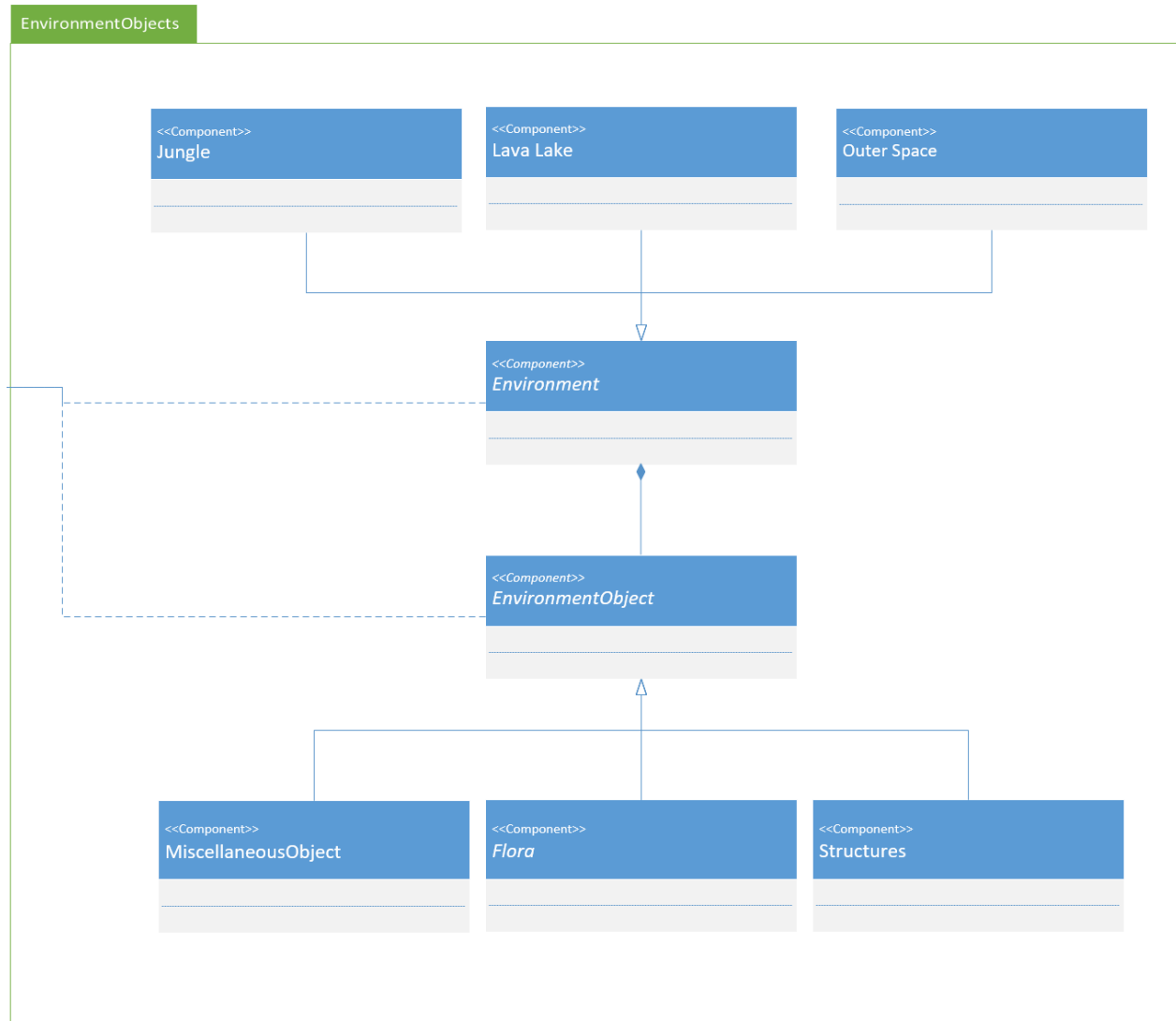
Figure 6.6 - Structural Diagram (EnvironmentObjects)

6.3   User Interface Design

Our user interface design is built around the technologies we are implementing. Virtual reality has a defined structure of displaying an image on two separate screens with logical angles that simulate what eyes see. With that, we are trying to create a very realistic depiction of riding in the car with a friend while the friend engages in dangerous activities. Google's SDK has provided many useful assets that have helped create the menu screens and input management to allow the player to control the experience. Figures 6.7 shows the user interface of the start menu. This is the first thing the user will see when starting the application. Figure 6.8 shows the view of the driver when the user is looking at him. This is a key view that the user will frequently look at. Figure 6.9 shows the tutorial level, which is designed to let the user get familiar with the goal and controlling the system. Figure 6.10 shows the first level of the experience, which is a jungle environment. Figure 6.11 displays the second level, a lava lake in a mountain. Figure 6.12 shows the final level, which is in outer space.
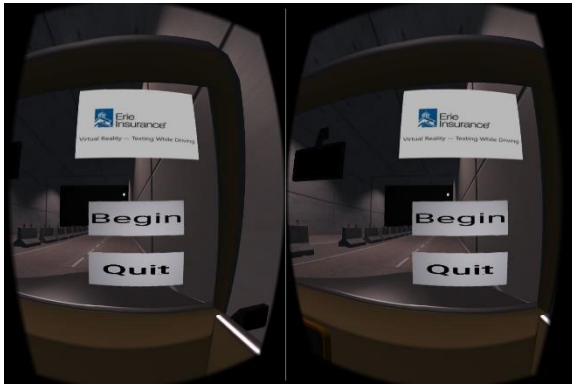
Figure 6.7 - Start Menu View

Figure 6.8 - Driver

Figure 6.9 - Tutorial
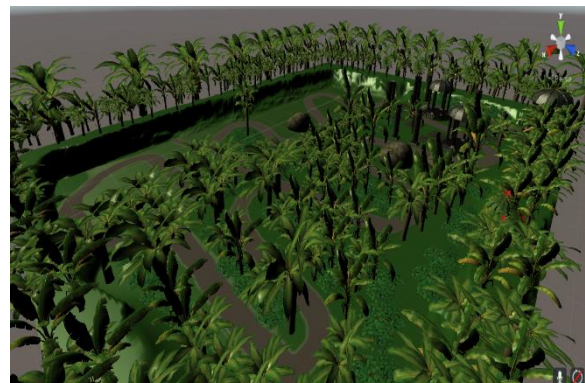
Figure 6.10 - Jungle

Figure 6.11 – Lava Lake
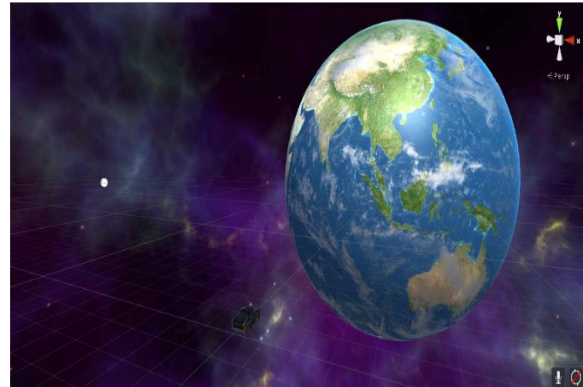
Figure 6.12 – Outer Space

6.4 Behavioral Design

In Figure 6.13, the behavior of the system is displayed. The activity diagram shows the flow of the experience and gives the steps required to succeed in the system, as well as the fail state requirements.
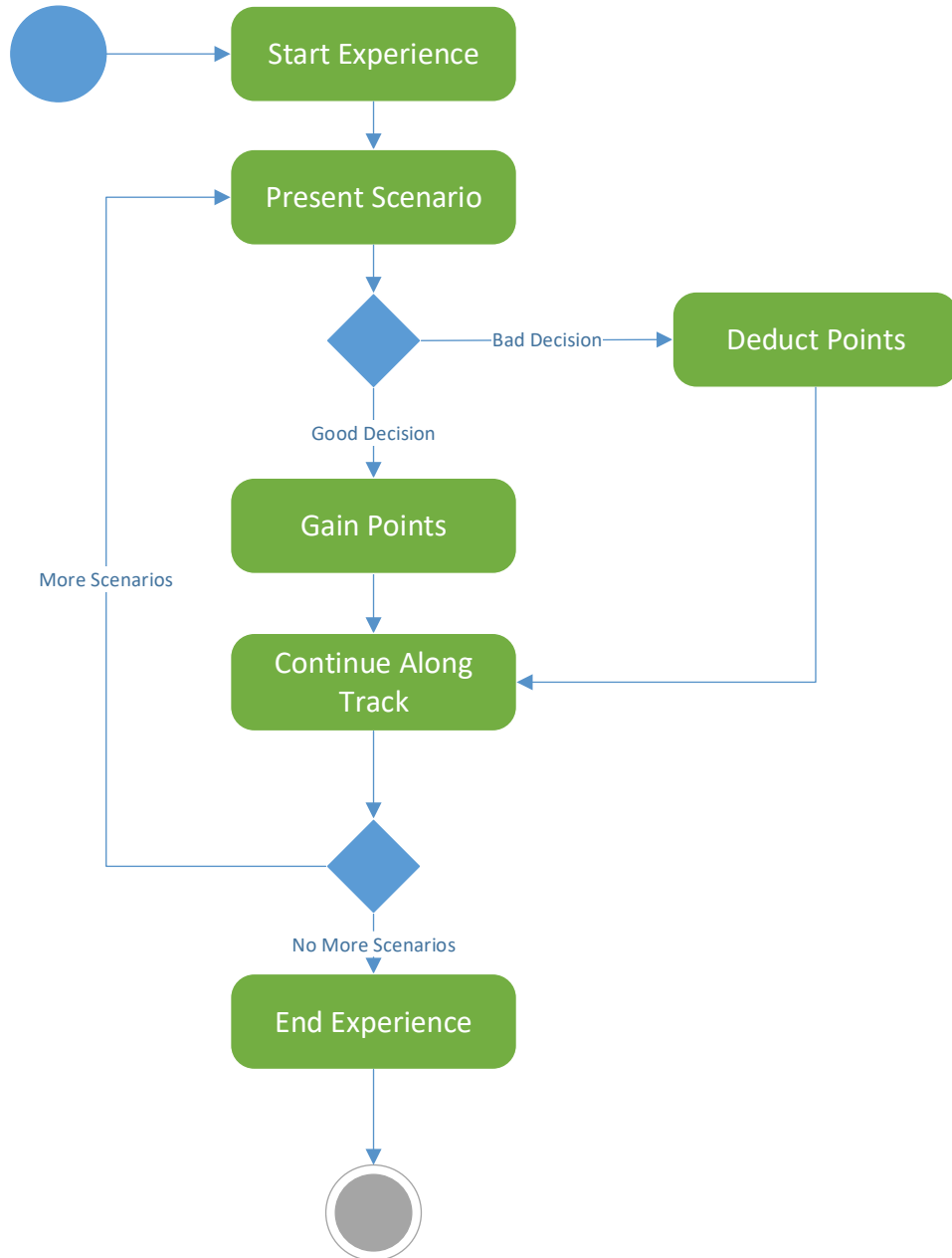
Figure 6.13 - Activity Diagram

6.5 Design Alternatives & Design Rationale

With our project, we are using Unity to create an experience that can run on mobile devices. Unity is designed with the component-based architecture in mind, and the way objects are implemented is based around that concept. Initially we looked into MVC, which is similar to our current design. However, each component in Unity essentially has its own model, view, and controller. The design would be complicated, and would not be as accurate as the component-based architecture. When designing environments, we started with a realist approach, having a highway, suburban, and city section. We were later asked by our business sponsors to take a less realistic approach. We have changed the environments into a jungle, a lava lake, and outer space. We still plan to feature our core systems in the game which includes the driver texting on his phone, interacting with the driver to solve scenarios, and creating a driving path that the user will be able to experience during each individual scene. The scenario system that was originally implemented into the project will continue to be a core feature of the system. However, scenarios will now have only two available choices that the user can make. The user can successfully complete a scenario by paying attention to the environment and looking over in the direction of the driver to make sure he is paying attention at a critical moment. The user can also choose to be completely oblivious as to whether the driver is paying attention or not and simply not look in the direction of the driver. The user will fail the scenario and receive no points if he/she chooses to make the latter decision.

As a team, we have decided to replace the existing car model with a convertible car. Although this car is less realistic than our previous car model, we feel that it fits fairly well with our system's new, cartoony look and feel. The convertible car allows the user to see much more of their environment. This change gives the user the feeling of being in a more open space, which should decrease the chances of a user feeling sick while playing in this virtual environment. It also allows the team to add point orbs in more interesting locations since the user can also look up and see the sky rather than the roof of a car.

7. System Implementation

7.1 Programming Languages & Tools

We are implementing our project using Unity, which takes advantage of C# for creating scripts. Unity provides an IDE called MonoDevelop, however we are using Microsoft Visual Studio, which can be used instead of MonoDevelop.

7.2 Coding Conventions

We will adhere to the coding conventions designed around Unity development as well as Microsoft's C# conventions. We will also be following Unity best practice for component design, which will help improve maintainability and performance.

7.3 Code Version Control

As with all projects being worked on by multiple personnel, version control is very important for the efficiency of our workflow. We will be using a combination of Git and Unity SmartMerge for our version control which will handle branching and merge conflicts. We will be hosting our repository in GitHub.

7.4 Implementation Alternatives & Decision Rationale

One alternative development tool we could have used instead of Unity is Unreal Engine. Unreal is another game engine that is widely available and features mobile development and also has Google Cardboard SDK support. With Unreal we would also be developing using C++ instead of C#. Our team decided to use Unity over Unreal because we are all more familiar with C# and virtual reality development is more popular with Unity, so the documentation and resources available will be better defined. Erie Insurance has stated that they are aware of the terms of service with Unity and has agreed to allow us to proceed with Unity development.

7.5 Analysis of Key Algorithms

Our two key algorithms are the driving system and the scenario system. These components work together to form what is essentially the majority of the system. The driving algorithm handles the controlling of the car, which has the ability to follow the path that we have predefined in the development environment. While driving the user is able to overcome scenarios, which is done by the work of our scenario system. The scenario system is composed of triggers, environmental objects, and the driver. This system is able to be reused, thus allowing us to easily create new scenarios.

8. System Testing

8.1 Test Automation Framework

Our project is developed following the test-driven development methodology. In section 8, we will be covering the tests designed for our application as we continue developing it. In order for our system requirements to be verified, there will be tests created for each one to ensure correct implementation.

8.1.1 Steps for Installing Test Framework

Our tests are designed using Unity Test Tools, which is an asset that allows assertions on Unity objects and scripts to verify that everything is working correctly. In order to install the testing framework, all that needs to be done is download Unity Test Tools from the Unity Asset Store and add it to an existing project.

8.1.2 Steps for Running Test Cases

In order to run a test case, the test case must be opened in Unity Test Tools. From there, the tests can be run or modified to specified settings.

8.2 Test Case Design

8.2.1 Test Suites

Test suites are collection of test cases that test related functions of a program. In our project, we have separated tests into scenario interaction, environment interaction, and system performance.

Table 8.2.1 defines the tests belonging to scenario interactions. These will test all of the features of the application that require user input to continue.

Table 8.2.2 defines the tests belonging to environment interactions. These represent situations that involve objects interacting in the environment, outside of the user's control.

Table 8.2.3 contains tests that test for the system performing at expected standards for VR.

8.2.2 Unit Test Cases

Unit test cases test all parts of an individual unit within a system or subsystem. Table 8.2.4 describes the test for resetting the camera to the center upon double clicks. Table 8.2.5 shows the test for allowing the user to change camera rotation in the application.

8.2.3 Integration Test Cases

System tests covers major system functionalities, and tests specific system requirements. Table 8.2.6 is the test for allowing the user to interact with the environment by clicking objects.

8.2.4 System Test Cases

Integration test cases test the connection between the units of a system or subsystem. Table 8.2.7 shows the test for scenario triggers in the environment. Table 8.2.8 shows the test for failing a scenario due to not preventing the driver from texting.

8.2.5 Acceptance Test Cases

These test cases reflect user requirements. The tests verify that specific requirements are working as planned for the user. They ensure that the system covers the most important requirements provided by the end user. Table 8.2.9 shows the test for the AI driver to be distracted during scenarios. Table 8.2.10 shows the test for the hardware requirement to run the application. Table 8.2.11 shows the test for the virtual reality representation of the application on the user's screen. Table 8.2.12 shows the test for the frame rate performance of the system, which provides a smooth virtual experience for the user.

8.3 Test Case Execution Report

The test case execution reports outline the steps taken to execute a given test case. They also provide the status of the test and any defects that will prevent the test from passing.

8.3.1 Unit Testing Report

Table 8.3.1 shows the execution steps for resetting the camera. The test is successful as we have implemented the feature.

Table 8.3.3 shows the execution steps for interacting with objects in the environment.

8.3.2 Integration Testing Report

Table 8.3.4 shows the execution steps for triggering a scenario.

8.3.3 System Testing Report

Table 8.3.2 shows the execution steps for moving the camera around in the environment.

Table 8.3.5 shows the execution steps for failing a scenario.

8.3.4 Acceptance Testing Report

Table 8.3.6 shows the execution steps for AI driver actions.

Table 8.3.7 shows the execution steps for running on mobile hardware.

Table 8.3.8 shows the execution steps for stereoscopic rendering.

Table 8.3.9 shows the execution steps for system frame rate.

9. Challenges & Open Issues

9.1 Challenges Faced in Requirements Engineering

We had trouble dealing with somewhat vague requirements provided by the industry sponsor, and were faced with the task of continuous meetings in order to get a clear understanding of the sponsor's needs in regard to the system.

9.2 Challenges Faced in System Development

Our first issue we faced was configuring version control to work with our system. Git alone does not work for Unity projects, and scenes are stored in binary files, so if a scene was worked on concurrently, it would not be able to merge. The documentation was confusing, and we failed to set it up properly a few times. We also had trouble with incompatible versions between Unity and the Google VR SDK. The SDK we originally had was out of date. Additionally, another challenge we faced was getting the driver into the car and being able to make him move in a realistic manner.

Another issue we faced was a changing direction of the project. When demoing progress to our business sponsors, they liked what we had. However, they wanted us to take a less realistic approach to everything and gamify the system more. Once we discussed our plan and modified the requirements everyone seemed positive about the new approach.

9.3 Open Issues & Ideas for Solutions

We currently have two existing issues: 1.) We cannot collect point orbs while the car is completely stopped and 2.) messages in the tutorial either cannot be clicked at all or can only be clicked on the left or right edges.

An idea that came up to attempt to fix issue the point orb collection issue would be to set the speed of the car to a very small number rather than zero. The reason for this is because the team thinks that the gaze module we are using may need a non-zero speed set to function correctly. This potential solution may also fix the second issue as well, since they are both using the gaze module.

10. System Manuals

10.1 Instructions for System Development

In order to develop the application, the environment must be set up. After the required steps are completed, the project must be opened in Unity. From there, any part of the system can be modified.

10.1.1 How to Set Up Development Environment

In order to develop the application, the developer must have Unity 5.4.1f1 installed as well as Git in order to pull from the repository. Once pulled, opening the project in Unity will allow for additional development.

10.1.2 Notes on System Further Extensions

10.2 Instructions for System Deployment

Steps to build and export to Android:

1. Select File > Build Settings
2. Select the platform as Android, then switch platform
3. Select Player Settings, in the resolution and presentation tab, select landscape left and use the 32-bit display buffer
4. Select Other Settings, change minimum API level to be Android 4.4 KitKat (API level 19)
5. Select Build to create APK

10.2.1 Platform Requirements

In order to build and deploy the application, Unity is required. Along with that, the Android SDK and Java SDK must be installed as well.

10.2.2 System Installation

To install on Android, the APK must be downloaded. After downloading, it can be installed and run.

10.3 Instructions for System End Users

The end user will be able to run the .apk file that is outputted by Unity. As mentioned in the system requirements, it is strongly recommended that the user runs the application on a Samsung Galaxy S5 or other comparable hardware. It is also recommended that the user is running the latest version of the Android operating system. Along with that, the user will need a Google Cardboard headset that has at least one button to be able to interact with the system.

Once the previous requirements are met, the user can download the. apk file from a source of Erie Insurance's choosing. The user will need to install the .apk file on to their Android phone. Once the user has done this, it is as simple as launching the application and placing the phone into the Google Cardboard headset.

11. Conclusion

11.1 Achievement

11.2 Lessons Learned

11.3 Acknowledgment

12. References

[1] MSDN, C# Programmer's Reference, Accessed on 10/21/2016

https://msdn.microsoft.com/en-us/library/618ayhy6(v=vs.71).aspx

[2] Unity, Unity Scripting Reference, Accessed on 10/21/2016

https://docs.unity3d.com/ScriptReference/

[3] Unity, Unity Manual, Accessed on 10/21/2016

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

[4] Unity, Unity Test Tools, Accessed on 10/21/2016

https://unity3d.com/learn/tutorials/topics/production/unity-test-tools

[5] Unity, Unity Community, Accessed on 10/21/2016

https://forum.unity3d.com/

[6] Google, Google VR SDK for Unity, Accessed on 10/21/2016

https://developers.google.com/vr/unity/

Table 4.1. Use Case Index Table

Project Name: Virtual Reality---Texting While Driving				
Use Case ID	**Use Case Name**	**Level**	**Author**	**Version**
UC-002	Experience Loop	Primary task	Nathan Christiansen	0.7
UC-003	Solve Scenario	Subfunction	Nathan Christiansen	0.4
Acknowledgment: Generated from the CapStone process management system ©2015				

Table 4.2. Use Case UC-002

Project Name:	
Use Case ID:	UC-002
Use Case Name:	Experience Loop
User Goal:	Experience the experience
Scope:	VR - Texting while Driving
Level:	Primary task
Relevant User Reqs:	UF-B,UF-C
Relevant System Reqs:	SF-B-01,SF-B-02,SF-C-01
Primary Actor:	User
Precondition:	The application is running and on the main menu
Minimal Guarantee:	The user enters the experience
Success Guarantee:	The user finishes the experience
Trigger:	User selects start experience on the main menu

	Step	Actions
Success Scenario:	1	The user selects start experience on the main menu
	2	The system begins the experience
	3	The user gains control of the passenger
	4	The user SOLVES SCENARIO
	5	The system continues until the next threshold
	6	The system repeats step 4-5 until the user completes the experience
	7	The system displays a results screen to the user

Extensions:	Branching Scenarios

Acknowledgment: Generated from the CapStone process management system ©2015

Table 4.3. Use Case UC-003

Project Name:	
Use Case ID:	UC-003
Use Case Name:	Solve Scenario
User Goal:	The user makes choices to solve a scenario
Scope:	VR - Texting While Driving
Level:	Subfunction
Relevant User Reqs:	UF-A
Relevant System Reqs:	SF-A-01
Primary Actor:	User
Precondition:	The user is in the experience and has not failed
Minimal Guarantee:	The default solution is chosen
Success Guarantee:	The user's solution is chosen
Trigger:	The user reaches a scenario threshold

Success Scenario:	Step	Actions
	1	The user reaches a scenario threshold
	2	The system presents a scenario involving a dangerous situation
	3	The user selects a solution presented by the scenario
	4	The system enters a success state for the scenario

Extensions:	Branching Scenarios	
3A	**Condition:** The user selects an incorrect solution or does not enter within the alloted time	
	Step	Actions
	1	The system enters a fail state for the scenario

Acknowledgment: Generated from the CapStone process management system ©2015

Table 4.4. User Functional Requirements: UF-A

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UF-A		Type	Functional	Non-Functional
Creation:	Sep 16 2016 12:51 PM		User	☒	☐
Modification:	Sep 30 2016 03:07 PM		System	☐	☐
Description:	The application should present various scenarios that display a distracted driver, and give the user the ability to overcome the potential negative outcome.				
Priority:	✔ Highest	High	Medium	Low	Lowest
This Req. is Refined Into:	SF-A-01, SF-A-02				
Justify why UF-A can be completely covered by SF-A-01, SF-A-02	SF-A-01 describes how many choices the user will be able to choose from to affect their outcome.				
Traceability:	Use cases cf.	UC-003			
	Test cases cf.	TC-002			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.5. User Functional Requirements: UF-B

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UF-B		Type	Functional	Non-Functional
Creation:	Sep 16 2016 01:05 PM		User	☒	☐
Modification:	Sep 30 2016 03:06 PM		System	☐	☐
Description:	The user should control a passenger in a vehicle driven by a person engaging in dangerous activities.				
Priority:	✔ Highest	High	Medium	Low	Lowest
This Req. is Refined Into:	SF-B-01, SF-B-02, SF-B-03				
Justify why UF-B can be completely covered by SF-B-01, SF-B-02, SF-B-03	SF-B-01 specifies how the user will be able to control a passenger. SF-B-02 specifies how the user will be able to input commands.				
Traceability:	Use cases cf.	UC-002			
	Test cases cf.	TC-001, TC-004, TC-005			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.6. User Functional Requirements: UF-C

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UF-C		Type	Functional	Non-Functional
Creation:	Sep 21 2016 02:59 PM		User	☒	☐
Modification:	Oct 18 2016 08:25 AM		System	☐	☐
Description:	The system should feature multiple outcomes that can occur due to the driver being distracted.				
Priority:	✔ Highest	High	Medium	Low	Lowest
This Req. is Refined Into:	SF-C-01				
Justify why UF-C can be completely covered by SF-C-01	SF-C-01 specifies how many outcomes the system will provide and gives detail about each.				
Traceability:	Use cases cf.	UC-002			
	Test cases cf.	TC-006			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.7. User Functional Requirements: UF-E

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UF-E		Type	Functional	Non-Functional
Creation:	Feb 16 2017 08:11 AM		User	☒	☐
Modification:	Feb 16 2017 08:19 AM		System	☐	☐
Description:	There should be orbs around each level that the user can collect to gain additional points.				
Priority:	Highest	✔ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-E-01				
Justify why UF-E can be completely covered by SF-E-01	SF-E-01 specifies what the orbs should look like and describes their use.				
Traceability:	Use cases cf.	Yet to be completed in use case worksheet!			
	Test cases cf.	Yet to be completed in test case worksheet!			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.8. User Functional Requirements: UF-F

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UF-F		Type	Functional	Non-Functional
Creation:	Oct 21 2016 12:12 PM		User	☒	☐
Modification:	Oct 21 2016 12:17 PM		System	☐	☐
Description:	The driver should be controlled by an AI and should engage in various tasks.				
Priority:	Highest	✔ High	Medium	Low	Lowest
This Req. is Refined Into:	SF-F-01				
Justify why UF-F can be completely covered by SF-F-01	Specifies what the AI will perform during the experience.				
Traceability:	Use cases cf.	Yet to be completed in use case worksheet!			
	Test cases cf.	TC-009			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.9. User NonFunctional Requirements: UP-01

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UP-01		Type	Functional	Non-Functional
Creation:	Sep 26 2016 03:06 PM		User	☐	☒
Modification:	Sep 30 2016 03:02 PM		System	☐	☐
Description:	The system should run at an acceptable frame rate suitable for virtual reality use.		Product (sub-type below)		
			Performance Requirements		
Priority:	Highest	High	✔ Medium	Low	Lowest
This Req. is Refined Into:	SP-01-01				
Justify why UP-01 can be completely covered by SP-01-01	Specifies what the acceptable frame rate the system should run at.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-012			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.10. User NonFunctional Requirements: UO-01

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UO-01		Type	Functional	Non-Functional
Creation:	Sep 16 2016 12:56 PM		User	☐	☒
Modification:	Sep 30 2016 03:03 PM		System	☐	☐
Description:	The application should be developed for modern Android devices.		Organizational (sub-type below)		
			Development Requirements		
Priority:	✔ Highest	High	Medium	Low	Lowest
This Req. is Refined Into:	SO-01-01				
Justify why UO-01 can be completely covered by SO-01-01	SO-01-01 specifies a modern Android operating system and device for eventual app deployment.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-010			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.11. User NonFunctional Requirements: UO-02

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	UO-02		Type	Functional	Non-Functional
Creation:	Sep 16 2016 12:58 PM		User	☐	☒
Modification:	Sep 30 2016 03:01 PM		System	☐	☐
Description:	The application should be developed for cardboard VR use.		Organizational (sub-type below)		
			Development Requirements		
Priority:	✔ Highest	High	Medium	Low	Lowest
This Req. is Refined Into:	SO-02-01				
Justify why UO-02 can be completely covered by SO-02-01	Specifies the SDK and method of displaying virtual reality applications.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-011			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.12. User NonFunctional Requirements: UO-03

Project Name:	Virtual Reality---Texting While Driving			Type	Functional	Non-Functional
Requirement #:	UO-03					
Creation:	Sep 16 2016 01:04 PM			User	☐	☒
Modification:	Sep 30 2016 03:01 PM			System	☐	☐
Description:	The application must feature ERIE Insurance branded paraphernalia advertising the company throughout.			Organizational (sub-type below)		
				Development Requirements		
Priority:	Highest	✔ High	Medium	Low		Lowest
This Req. is Refined Into:	SO-03-01					
Justify why UO-03 can be completely covered by SO-03-01	Specifies objects to be textured with ERIE Insurance textures.					
Traceability:	Use cases cf.	N/A				
	Test cases cf.	Yet to be completed in test case worksheet!				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.13. System Functional Requirements: SF-A-01

Project Name:	Virtual Reality---Texting While Driving			Type	Functional	Non-Functional
Requirement #:	SF-A-01					
Creation:	Sep 23 2016 01:00 PM			User	☐	☐
Modification:	Apr 03 2017 08:34 AM			System	☒	☐
Description:	The system should provide two possible solutions for every decision presented.					
Priority:	Highest	✔ High	Medium	Low		Lowest
This Req. is Engineered From:	UF-A					
Justify why meeting SF-A-01 can contribute to the fulfilment of UF-A	Specifies how the user can overcome each situation.					
Traceability:	Use cases cf.	UC-003				
	Test cases cf.	Yet to be completed in test case worksheet!				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.14. System Functional Requirements: SF-A-02

Project Name:	Virtual Reality---Texting While Driving					
Requirement #:	SF-A-02			Type	Functional	Non-Functional
Creation:	Nov 09 2016 02:24 PM			User	☐	☐
Modification:	Nov 09 2016 02:25 PM			System	☒	☐
Description:	When the car passes a trigger, a scenario should be presented.					
Priority:	✔ Highest	High	Medium	Low		Lowest
This Req. is Engineered From:	UF-A					
Justify why meeting SF-A-02 can contribute to the fulfilment of UF-A	Handles the beginning of individual scenarios.					
Traceability:	Use cases cf.	Yet to be completed in use case worksheet!				
	Test cases cf.	TC-002				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.15. System Functional Requirements: SF-B-01

Project Name:	Virtual Reality---Texting While Driving					
Requirement #:	SF-B-01			Type	Functional	Non-Functional
Creation:	Sep 23 2016 12:54 PM			User	☐	☐
Modification:	Sep 27 2016 10:55 AM			System	☒	☐
Description:	The user should have a first person perspective during the experience, and can use motion inputs to position the camera.					
Priority:	✔ Highest	High	Medium	Low		Lowest
This Req. is Engineered From:	UF-B					
Justify why meeting SF-B-01 can contribute to the fulfilment of UF-B	Specifies the inputs the user can use to control the passenger.					
Traceability:	Use cases cf.	UC-002				
	Test cases cf.	TC-004				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.16. System Functional Requirements: SF-B-02

Project Name:	Virtual Reality---Texting While Driving					
Requirement #:	SF-B-02		Type	Functional	Non-Functional	
Creation:	Sep 27 2016 10:55 AM		User	☐	☐	
Modification:	Sep 27 2016 10:56 AM		System	☒	☐	
Description:	The user will use the button on the cardboard headset to interact with objects in the environment, and select choices during scenarios					
Priority:	✔ Highest	High	Medium	Low		Lowest
This Req. is Engineered From:	UF-B					
Justify why meeting SF-B-02 can contribute to the fulfilment of UF-B	Specifies input the user has during control					
Traceability:	Use cases cf.	UC-002				
	Test cases cf.	TC-005				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.17. System Functional Requirements: SF-B-03

Project Name:	Virtual Reality---Texting While Driving					
Requirement #:	SF-B-03		Type	Functional	Non-Functional	
Creation:	Nov 09 2016 01:58 PM		User	☐	☐	
Modification:	Nov 09 2016 02:00 PM		System	☒	☐	
Description:	Double clicking the input button will re-center the camera to the front of the car					
Priority:	Highest	✔ High	Medium	Low		Lowest
This Req. is Engineered From:	UF-B					
Justify why meeting SF-B-03 can contribute to the fulfilment of UF-B	This requirement aids the user in their control of the passenger as it allows an easy way to refocus to the front of the car					
Traceability:	Use cases cf.	Yet to be completed in use case worksheet!				
	Test cases cf.	TC-001				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.18. System Functional Requirements: SF-C-01

Project Name:	Virtual Reality---Texting While Driving					
Requirement #:	SF-C-01			Type	Functional	Non-Functional
Creation:	Sep 23 2016 12:59 PM			User	☐	☐
Modification:	Feb 14 2017 12:35 PM			System	☒	☐
Description:	The system should include various types of outcomes that can occur within the environment, including avoiding falling objects, avoiding collisions with other objects, and dealing with going off the path.					
Priority:	Highest	✔ High	Medium	Low		Lowest
This Req. is Engineered From:	UF-C					
Justify why meeting SF-C-01 can contribute to the fulfilment of UF-C	Specifies the different situations and outcomes that the user is presented with.					
Traceability:	Use cases cf.	UC-002				
	Test cases cf.	TC-006				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.19. System Functional Requirements: SF-E-01

Project Name:	Virtual Reality---Texting While Driving					
Requirement #:	SF-E-01			Type	Functional	Non-Functional
Creation:	Feb 16 2017 08:13 AM			User	☐	☐
Modification:	Feb 16 2017 08:16 AM			System	☒	☐
Description:	There should be white, glowing orbs featuring the Erie Insurance logo placed around each level which the user can collect by focusing on them to gain an additional amount of points.					
Priority:	Highest	✔ High	Medium	Low		Lowest
This Req. is Engineered From:	UF-E					
Justify why meeting SF-E-01 can contribute to the fulfilment of UF-E	This requirement further defines UF-E by providing more specific detail.					
Traceability:	Use cases cf.	Yet to be completed in use case worksheet!				
	Test cases cf.	Yet to be completed in test case worksheet!				
Acknowledgment	*Generated from the CapStone Process Management System ©2015*					

Table 4.20. System Functional Requirements: SF-F-01

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	SF-F-01		Type	Functional	Non-Functional
Creation:	Oct 21 2016 12:13 PM		User	☐	☐
Modification:	Feb 08 2017 04:13 PM		System	☒	☐
Description:	The driver AI should text while driving, and should stop texting when the user looks in the direction of the driver.				
Priority:	Highest	✔ High	Medium	Low	Lowest
This Req. is Engineered From:	UF-F				
Justify why meeting SF-F-01 can contribute to the fulfilment of UF-F	Specifies exactly what the driver's AI will do during the experience.				
Traceability:	Use cases cf.	Yet to be completed in use case worksheet!			
	Test cases cf.	TC-009			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.21. System NonFunctional Requirements: SP-01-01

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	SP-01-01		Type	Functional	Non-Functional
Creation:	Sep 30 2016 02:54 PM		User	☐	☐
Modification:	Sep 30 2016 02:55 PM		System	☐	☒
Description:	The application should run at a minimum of 30 frames per second.		Product (sub-type below)		
			Performance Requirements		
Priority:	Highest	✔ High	Medium	Low	Lowest
This Req. is Engineered From:	UP-01				
Justify why meeting SP-01-01 can contribute to the fulfilment of UP-01	Specifies the minimum fps that the experience should perform at.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-012			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.22. System NonFunctional Requirements: SO-01-01

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	SO-01-01		Type	Functional	Non-Functional
Creation:	Sep 26 2016 02:59 PM		User	☐	☐
Modification:	Sep 26 2016 03:02 PM		System	☐	☒
Description:	The system should be targeted for Android 5.1.1 "Lollipop" for phones with hardware specifications of the Samsung S5 and up		Organizational (sub-type below)		
			Development Requirements		
Priority:	Highest	✔ High	Medium	Low	Lowest
This Req. is Engineered From:	UO-01				
Justify why meeting SO-01-01 can contribute to the fulfilment of UO-01	Specifies the OS version and hardware requirements				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-010			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.23. System NonFunctional Requirements: SO-02-01

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	SO-02-01		Type	Functional	Non-Functional
Creation:	Sep 30 2016 01:03 PM		User	☐	☐
Modification:	Sep 30 2016 01:05 PM		System	☐	☒
Description:	The system will utilize the Google VR SDK to display two images through the cardboard.		Organizational (sub-type below)		
			Development Requirements		
Priority:	✔ Highest	High	Medium	Low	Lowest
This Req. is Engineered From:	UO-02				
Justify why meeting SO-02-01 can contribute to the fulfilment of UO-02	The system will provide a VR experience that is designed around the cardboard.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	TC-011			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.24. System NonFunctional Requirements: SO-03-01

Project Name:	Virtual Reality---Texting While Driving				
Requirement #:	SO-03-01		**Type**	**Functional**	**Non-Functional**
Creation:	Sep 30 2016 01:00 PM		**User**	☐	☐
Modification:	Mar 13 2017 02:24 PM		**System**	☐	☒
Description:	Erie Insurance logos will be placed on objects within the environment.		Organizational (sub-type below)		
			Development Requirements		
Priority:	Highest	High	✔ Medium	Low	Lowest
This Req. is Engineered From:		UO-03			
Justify why meeting SO-03-01 can contribute to the fulfilment of UO-03	Erie Insurance will be represented within the experience.				
Traceability:	Use cases cf.	N/A			
	Test cases cf.	Yet to be completed in test case worksheet!			
Acknowledgment	*Generated from the CapStone Process Management System ©2015*				

Table 4.25. Mapping from user requirements to system requirements

Project Name: Virtual Reality---Texting While Driving			
User Requirements		**System Requirements**	
Req ID	**Description**	**Req ID**	**Description**
UF-A	The application should present various scenarios that display a distracted driver, and give the user the ability to overcome the potential negative outcome.	SF-A-01	The system should provide two possible solutions for every decision presented.
		SF-A-02	When the car passes a trigger, a scenario should be presented.
UF-B	The user should control a passenger in a vehicle driven by a person engaging in dangerous activities.	SF-B-01	The user should have a first person perspective during the experience, and can use motion inputs to position the camera.
		SF-B-02	The user will use the button on the cardboard headset to interact with objects in the environment, and select choices during scenarios
		SF-B-03	Double clicking the input button will re-center the camera to the front of the car
UF-C	The system should feature multiple outcomes that can occur due to the driver being distracted.	SF-C-01	The system should include various types of outcomes that can occur within the environment, including avoiding falling objects, avoiding collisions with other objects, and dealing with going off the path.
UF-E	There should be orbs around each level that the user can collect to gain additional points.	SF-E-01	There should be white, glowing orbs featuring the Erie Insurance logo placed around each level which the user can collect by focusing on them to gain an additional amount of points.
UF-F	The driver should be controlled by an AI and should engage in various tasks.	SF-F-01	The driver AI should text while driving, and should stop texting when the user looks in the direction of the driver.
UO-01	The application should be developed for modern Android devices.	SO-01-01	The system should be targeted for Android 5.1.1 "Lollipop" for phones with hardware specifications of the Samsung S5 and up
UO-02	The application should be developed for cardboard VR use.	SO-02-01	The system will utilize the Google VR SDK to display two images through the cardboard.

UO-03	The application must feature ERIE Insurance branded paraphernalia advertising the company throughout.	SO-03-01	Erie Insurance logos will be placed on objects within the environment.
UP-01	The system should run at an acceptable frame rate suitable for virtual reality use.	SP-01-01	The application should run at a minimum of 30 frames per second.
Acknowledgment: Generated from the CapStone process management system ©2015			

Table 8.2.1. Test Suite TS-001: Scenario Interaction

TS-001: Scenario Interaction

Test Case ID	Test Stage	Test Case Description
TC-002	System	Scenario Triggers
TC-004	Unit	Camera Rotation
TC-006	System	Bad Scenario Outcomes

Table 8.2.2. Test Suite TS-002: Environment Interaction

TS-002: Environment Interaction

Test Case ID	Test Stage	Test Case Description
TC-001	Unit	Camera Recentering
TC-005	Integration	Environment Interaction
TC-009	Acceptance	AI Driver Actions

Table 8.2.3. Test Suite TS-003: System Performance

TS-003: System Performance

Test Case ID	Test Stage	Test Case Description
TC-010	Acceptance	Hardware Validation
TC-011	Acceptance	Double Image VR Display
TC-012	Acceptance	System Frame Rate

Table 8.2.4. Test Case TC-001

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-002: Environment Interaction	
Test Case ID	TC-001 (Unit Test)	
What To Test	Camera Recentering	
Test Data Input	Google Cardboard Input Button 2x	
Expected Result	Camera resets to the default view looking out of the windshield of the vehicle upon two rapid clicks of the input button.	
Traceability	Relevant User Req.(s)	UF-B
	Relevant System Req.(s)	SF-B-03
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.5. Test Case TC-004

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-001: Scenario Interaction	
Test Case ID	TC-004 (Unit Test)	
What To Test	Camera Rotation	
Test Data Input	Moving Google Cardboard	
Expected Result	Rotating the phone moves the camera in the experience.	
Traceability	Relevant User Req.(s)	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.6. Test Case TC-005

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-002: Environment Interaction	
Test Case ID	TC-005 (Integration Test)	
What To Test	Environment Interaction	
Test Data Input	Google Cardboard Input	
Expected Result	Looking at orbs in the environment will collect them.	
Traceability	Relevant User Req.(s)	UF-B
	Relevant System Req.(s)	SF-B-02
	Relevant Use Case(s)	UC-002, UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.7. Test Case TC-002

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-001: Scenario Interaction	
Test Case ID	TC-002 (System Test)	
What To Test	Scenario Triggers	
Test Data Input	Google Cardboard	
Expected Result	A scenario is presented to the user upon reaching a trigger in the environment.	
Traceability	Relevant User Req.(s)	UF-A
	Relevant System Req.(s)	SF-A-02
	Relevant Use Case(s)	UC-002, UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.8. Test Case TC-006

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-001: Scenario Interaction	
Test Case ID	TC-006 (System Test)	
What To Test	Bad Scenario Outcomes	
Test Data Input		
Expected Result	Failing to prevent the driver from texting during a scenario will lead to the failure of that scenario.	
Traceability	Relevant User Req.(s)	UF-C
	Relevant System Req.(s)	SF-C-01
	Relevant Use Case(s)	UC-003
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.9. Test Case TC-009

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-002: Environment Interaction	
Test Case ID	TC-009 (Acceptance Test)	
What To Test	AI Driver Actions	
Test Data Input	Google Cardboard	
Expected Result	The AI driver drives and gets distracted when scenarios are triggered through texting/not paying attention to the road.	
Traceability	Relevant User Req.(s)	UF-F
	Relevant System Req.(s)	SF-F-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.10. Test Case TC-010

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-003: System Performance	
Test Case ID	TC-010 (Acceptance Test)	
What To Test	Hardware Validation	
Test Data Input		
Expected Result	The system runs smoothly on hardware specifications of the Samsung S5 and up.	
Traceability	Relevant User Req.(s)	UO-01
	Relevant System Req.(s)	SO-01-01
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.11. Test Case TC-011

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-003: System Performance	
Test Case ID	TC-011 (Acceptance Test)	
What To Test	Double Image VR Display	
Test Data Input	Test Data Input	
Expected Result	Two images should be displayed for use with the Google Cardboard.	
Traceability	Relevant User Req.(s)	UO-02
	Relevant System Req.(s)	SO-02-01
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.2.12. Test Case TC-012

Project Name:	Virtual Reality---Texting While Driving	
Test Suite	TS-003: System Performance	
Test Case ID	TC-012 (Acceptance Test)	
What To Test	System Frame Rate	
Test Data Input	The app	
Expected Result	The system runs at or above 30 frames per second when viewed through on a mobile device through a Google Cardboard.	
Traceability	Relevant User Req.(s)	UP-01
	Relevant System Req.(s)	SP-01-01
	Relevant Use Case(s)	
Acknowledgment: Generated from the CapStone process management system ©2015		

Table 8.3.1. Execution Report of Test Case TC-001

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-001					
Testing Tools Used:	Android, Google Cardboard, Unity Remote 5					
Testing Type:	Function coverage					
Execution Steps:	1	Begin the experience				
	2	Turn the camera in some direction away from the default view				
	3	Quickly double tap the Cardboard input button				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	Double tapping does nothing	Fail	Not yet implemented	10/10/2016 by Jake Wheeler
2	Nick Kapty	11/15/2016	Double tapping recenters the camera	Pass		
Execution Summary:	Upon implementation, the feature works as intended.					
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.2. Execution Report of Test Case TC-004

Project Name:	**Virtual Reality---Texting While Driving**					
Test Case ID:	TC-004					
Testing Tools Used:	Unity Test Tools					
Testing Type:	Function coverage					
Execution Steps:	1	Begin the experience				
	2	Rotate Cardboard in any direction				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	Camera moves around	Pass		
Execution Summary:	The feature works as intended					
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.3. Execution Report of Test Case TC-005

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-005					
Testing Tools Used:	Unity Test Tools					
Testing Type:	Agile (automated) testing					
Execution Steps:	1	Begin the experience				
	2	Look at an interactive environment object				
	3	Click on object if reticule expands				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	Object does not move	Fail	Not yet implemented	
2	Nate Christiansen	2/8/2017	Window rolls down, driver looks up	Pass		
Execution Summary:						
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.4. Execution Report of Test Case TC-002

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-002					
Testing Tools Used:	Unity Test Tools					
Testing Type:	Agile (automated) testing					
Execution Steps:	1 Begin the experience					
	2 Allow the car to proceed to a predefined trigger point					
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	No scenario presented	Fail	Not yet implemented	1/14/2017 by Jake Wheeler
2	Nate Christiansen	2/8/2017	A scenario is presented passing the trigger	Pass		
Execution Summary:						
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.5. Execution Report of Test Case TC-006

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-006					
Testing Tools Used:	Unity Test Tools					
Testing Type:	Agile (automated) testing					
Execution Steps:	1	Begin the experience				
	2	Wait for the vehicle to move to the first scenario				
	3	Do not nudge driver in time				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	No outcomes occur	Fail	Not yet implemented	
2	Nick Kapty	4/8/2017	Car does not stop or branch and proceeds to hazard	Pass		
Execution Summary:						
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.6. Execution Report of Test Case TC-009

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-009					
Testing Tools Used:	Unity Test Tools					
Testing Type:	Functional testing					
Execution Steps:	1	Begin the experience				
	2	Look at driver to trigger animations				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	Driver does not animate or interact with the user in any way	Fail	Not Implemented	
2	Nick Kapty	4/8/2017	Driver looks up when nudged	Pass		
Execution Summary:						
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.7. Execution Report of Test Case TC-010

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-010					
Testing Tools Used:						
Testing Type:	Function coverage					
Execution Steps:	1	Build the application in Unity				
	2	Export to and attempt to launch the app on an Android phone				
	3	Begin the experience				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	App launches successfully	Pass		
Execution Summary:	The app was able to launch on a phone of comparable hardware to the Samsung S5 successfully.					
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.8. Execution Report of Test Case TC-011

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-011					
Testing Tools Used:						
Testing Type:	Function coverage					
Execution Steps:	1	Export built app to an Android phone				
	2	Launch app				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	The app displayed with a binocular view	Pass		
Execution Summary:	The app ran correctly with a binocular view using the Google VR SDK.					
Acknowledgment: Generated from the CapStone process management system ©2015						

Table 8.3.9. Execution Report of Test Case TC-012

Project Name:	Virtual Reality---Texting While Driving					
Test Case ID:	TC-012					
Testing Tools Used:						
Testing Type:	Functional testing					
Execution Steps:	1	Export the built app to an Android phone				
	2	Launch the app				
	3	Begin the experience				
	4	Monitor the FPS throughout the experience				
Test Execution Records:						
#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	Nick Kapty	11/9/2016	FPS unknown	Fail	Not yet implemented	
2	Nick Kapty	4/8/2017	FPS is able to be tracked	Pass		
Execution Summary:						
Acknowledgment: Generated from the CapStone process management system ©2015						