

University of Nevada, Reno

CHYS_rcade: Virtual Reality Game Suite

A thesis submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering and the Honors Program

by

Miguel Teodoro D. Henares

Dr. Eelke Folmer, Thesis Advisor

Dr. Sergiu Dascalu, Thesis Advisor

May 2016

**UNIVERSITY
OF NEVADA
RENO**

THE HONORS PROGRAM

We recommend that the thesis
prepared under our supervision by

MIGUEL TEODORO D. HENARES

entitled

CHYS_rcade: Virtual Reality Game Suite

be accepted in partial fulfillment of the
requirements for the degree of

BACHELOR OF SCIENCE, COMPUTER SCIENCE AND ENGINEERING

Eelke Folmer, PhD., Thesis Advisor

Sergiu Dascalu, PhD., Thesis Advisor

Tamara Valentine, PhD., Director, Honors Program

May 2016

Abstract

Virtual reality is on the cutting edge of technology and will bring about many changes in communication, entertainment, and user interaction. As with all new technologies, the costs involved at this early stage of development can be too high for the average user. The Google Cardboard is an affordable and easily obtainable product that allows almost anyone to experience virtual reality. CHYS_rcade is a virtual arcade that utilizes this technology and allows users to play single and multiplayer arcade games in a virtual reality environment. Development for CHYS_rcade is done on Unity, which has a Google Cardboard SDK and can be deployed for iOS and Android devices simultaneously.

Acknowledgements

I'd like to first thank my fellow senior project group members Truman Chan, Jeff Soriano, and Douglas Yan for the countless hours of hard work that they put into this project and bringing our vision to reality. Thank you for your effective communication, willingness to pull all-nighters, flexibility with meeting times, and friendship. I'm very happy to have worked with an excellent group of individuals who have such a diverse set of skills.

I'd also like to thank my CS 426 professor, Sergiu Dascalu, and his teaching assistant, Benjamin Brown. They pushed our group to achieve higher and take this project above and beyond of what we had originally planned. Overseeing the development of our project from beginning to end, they played a pivotal role in helping it evolve by providing constructive feedback on the user experience and design structure of our program.

Lastly, I'd like to thank Dr. Eelke Folmer, our project advisor. I've worked extensively with Dr. Folmer over the last 3 years as his student, recitation leader, and research assistant. He provided our team with the basis for our project concept, gave us essential code to implement, and also supplied us with Google Cardboard headsets and Android phones that we needed for development.

Table of Contents

Abstract	i
Acknowledgement	ii
Table of Contents	iii
List of Tables	iv
List of Figures	v
List of Illustrations	vi
Introduction	1
Significance	1
Project Description	3
Requirements	4
Use Case Modeling	8
Analysis	11
Design	14
Conclusion	36
References	37

List of Tables

Table 1	6
Table 2	7
Table 3	9
Table 4	10
Table 5	30
Table 6	31
Table 7	32

List of Figures

Figure 1	8
Figure 2	11
Figure 3	11
Figure 4	12
Figure 5	13
Figure 6	14
Figure 7	15
Figure 8	16
Figure 9	17
Figure 10	18
Figure 11	29
Figure 12	30

List of Illustrations

Snapshot 1	33
Snapshot 2	33
Snapshot 3	34
Snapshot 4	34
Snapshot 5	34
Snapshot 6	35
Snapshot 7	35
Snapshot 8	35

Introduction

CHYS_rcade is a virtual reality gaming suite that brings an affordable yet immersive virtual reality gaming experience to the average gamer. The application is able to run on an Android or iOS device and utilizes the Google Cardboard to complete virtual reality experience. The experience puts the user in a realistic game lobby with a sleek design. Inside the lobby are interactive classic arcade machines that each contain a different game. CHYS_rcade was developed as a highly modularized system, where changes to any one subsystem are entirely independent from the rest of the project. This allows game developers to easily develop and add games to the system. CHYS_rcade currently comes preloaded with 4 games. One game is a first person something called Rocket Racer, another is a survival game called Maze Runner, the third game is a first person shooter called Attack of the CH_ubes, and the last game is a multiplayer shooter game called Pew Pew. The project applies several immersive features such as three-dimensional graphics, headtracking, and a walk-sensing feature provided by Dr. Eelke Folmer's lab.

Significance

As costs decrease, virtual reality is a market that is likely to explode in popularity fairly soon. Until then, most VR experiences are limited to those with the financial means to obtain the hardware and the technical expertise to create their own virtual experience. CHYS_rcade will provide an affordable, enjoyable, and easy to use system for people to experience virtual reality before the enterprise systems are released.

This project's innovation comes from its timing. VR might eventually become as commonplace as smartphones some day, but there aren't any current applications that

provide a VR experience that is as wholesome and immersive as CHYS_rcade. The project combines the novelty of VR systems with the familiarity of classic arcades. CHYS_rcade could be seen as a stepping stone for more creative and original virtual reality applications in the future, like training programs or communications systems.

Working on a cutting edge technology before it emerges into the market in a significant way is an enormous advantage for the team in terms of professional development. If the market for VR does expand as quickly as expected, then having this experience may prove invaluable. The highly modularized nature of CHYS_rcade also means that there is room for aggressive expansion. New games can easily be created and implemented, opening the possibility for an in-game marketplace for users to create and sell their own games.

CHYS_rcade is certainly not the first application to create games in a virtual reality. There are other applications that use the Google Cardboard, like the Google Street View application that lets users see Street View from Google Maps on their phone with the Google Cardboard. While this application does use the motion sensors that makes VR so immersive, it's not nearly as entertaining as a gaming application.

There are other VR platforms besides Google Cardboard, such as the Oculus Rift and Samsung Gear VR. However, both of these options are much more expensive than a Google Cardboard (\$599 and \$99, respectively, compared to ~\$25). The Oculus and Gear both require more hardware to run, while the Google Cardboard only needs a smartphone. The intention of CHYS_rcade is to make VR gaming as accessible as possible, making Google Cardboard the best choice for a platform.

Project Description

CHYS_rcade was created through the Unity 3D game engine. Many of the objects and 3D models were found online or imported from the Unity asset store. Scripts for the system were written in C#. The CHYS_rcade system can be broken down into different systems composed of different scenes. Each system is loosely coupled and can be easily changed without altering other parts of the program. The game lobby is considered to be its own system and each game is also its own system. The program utilizes the Google Cardboard. This allows users to explore CHYS_rcade by looking through the headset and maneuvering around the room using natural motions such as turning one's head or walking in place to move forward.

When users first begin CHYS_rcade they are greeted by a menu screen that allows them to either join a room or quit the application. Upon joining the room, the user is placed into the game's lobby. In the lobby are objects such as a pool table, grand piano, chairs, and even an upstairs area. The most important items in the room are the interactive arcade machines. Each machine is linked up to a game, and by clicking on the machines, the user is able to activate a menu screen that gives them the choice to either play the game or close the menu.

Rocket Racer is a first person infinite scrolling game. The objective is to obtain as many points as possible by surviving the environment. The user is essentially freefalling through space and must dodge incoming asteroids. The user is able to collect power ups which allow the player to gain temporary invincibility, earn extra lives, and shoot at the asteroids. The game ends once the user runs out of lives.

Ch_oin Runner is a game that utilizes CHYS_rcade's unique walk-sensing

feature. The player is able to navigate through a maze by physically walking in place to move forward and using headtracking to change directions. The game's objective is to earn a high score by surviving the maze's many moving monsters and collecting coins for extra points. The score increases as the player spends more time alive in the maze. As soon as the player is hit by a monster, the game ends.

Attack of the Ch_ubes is a shooter game. The player earns points by shooting at and destroying incoming cubes. Any cube that isn't destroyed hits the player and damages them. Different colored cubes have different strengths, speeds, and damages. The user is able to fully utilize the headtracking capabilities of the Cardboard.

PewPew is the only multiplayer game available on CHYS_rcade at the moment. It uses a client-server network structure to allow two or more users to join the same game room and play with each other. The game is a first-person shooter game where players simply try to win against their opponents by shooting at them, decreasing their health, and remaining as the last player standing.

Requirements

Requirements Workshop Summary

Our team met on March 3rd, 2016 at 12:00 PM at the Pennington Student Achievement Center. The people who were in attendance were Truman Chan, Miguel Henares, Douglas Yan, and Jeff Soriano. Truman was appointed as the secretary (he was in charge of making the minutes), Miguel was the leader of the meeting, Jeff was the researcher, and Douglas was the notetaker. The meeting took one hour to complete. The requirements found by the team are the following (ordered from highest priority to lowest):

1. Character movement (walking functionality, external controller)
2. User interaction with objects via head cursor (arcade machines, other user avatars, arcade room)
3. Single player game
4. Multiplayer game
5. Options menu for game settings
6. Create/Customize user avatar
7. Voice chat functionality
8. Create/Join arcade room
9. User login
10. Display scoreboard of games

The main focus of the meeting was to identify the requirements we wanted to fully incorporate. To do this, we looked at our old requirements from last semester and pared them down to what we felt was absolutely necessary to implement. After doing this process, we came up with the following list of requirements:

1. Character movement via walking functionality
2. User interaction with objects via head cursor (arcade machines, arcade room)
3. Include a single player game
4. Include a multiplayer game
5. Options menu that allows users to alter game settings

Over the course of the semester we devoted most of our time to implementing these five requirements. We worked on the other requirements only after these were complete and to our satisfaction.

System Requirements

Requirement	Priority	Description
R01	1	System will allow user to move using the head bobbing (walking in place) mechanism.
R02	1	System will allow users to interact with objects via the head cursor.
R03	1	System will provide a single player game.
R04	1	System will provide a multiplayer game where users can play games with other users in the same room.
R05	1	System will associate each user with an avatar.
R06	1	System will display games as arcade machines.
R07	1	System allows users to open VRCade menu and options.
R08	1	System will allow users to initiate gameplay by interacting with the displayed arcade machines.
R09	1	System will display a visible arcade room environment.
R10	2	System will allow users to talk to other users in the same room.
R11	2	System will allow users to hear the other users talking in the same room.
R12	3	System will allow users to see other users within the same arcade room.
R13	3	System will display a scoreboard of games won/lost.
R14	3	System will allow users to create and join arcade rooms.
R15	3	System will allow users to customize their avatars.

Table 1. Functional Requirements

Requirement	Description
NFR01	System will be programmed using C# on the Unity 3D Game Engine.
NFR02	System will run on iOS/Android mobile devices.
NFR03	System will use Google Cardboard to view the virtual reality environment.
NFR04	System will use Google Cardboard's magnet input for user interaction.
NFR05	System provides secure network connection between players.
NFR06	System is extensible to allow easy addition of games.
NFR07	System will be scalable to accommodate multiple users.
NFR08	System is processor efficient.
NFR09	System will use the smartphone battery efficiently.

Table 2. Non-functional Requirements

Use Case Modeling

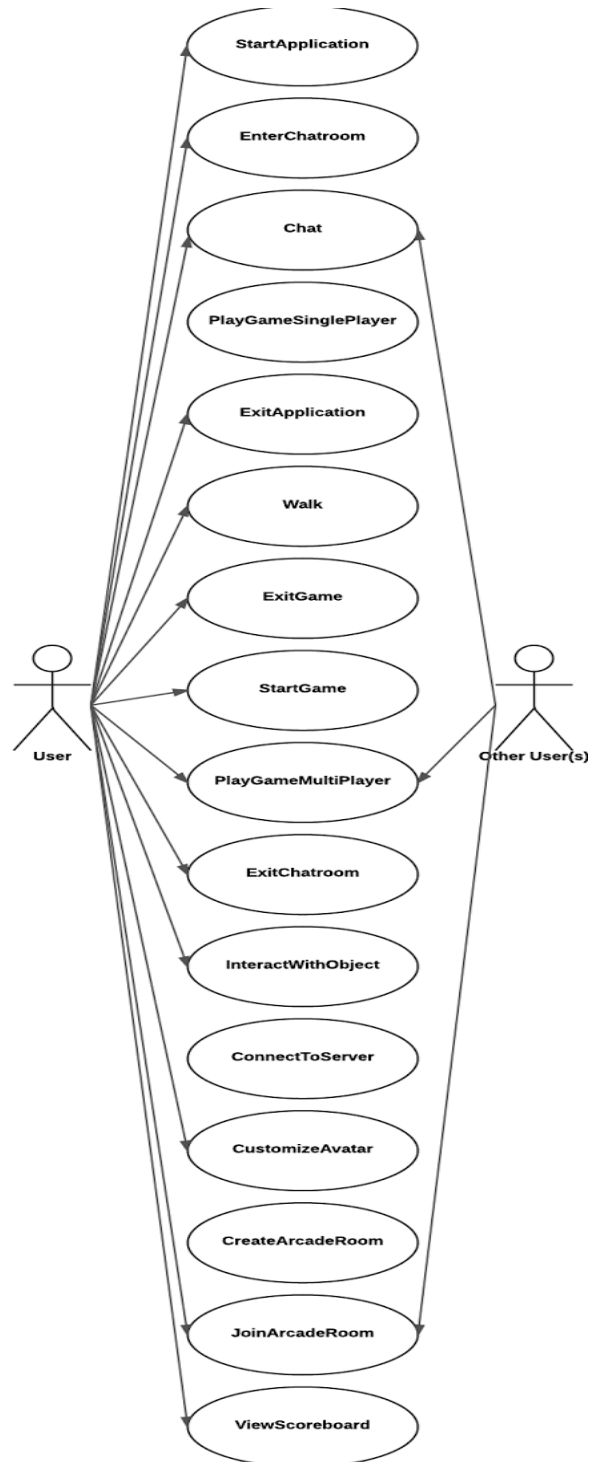


Figure 1. Use Case Diagram

Use Case ID	Use case name	Use case description
UC01	StartApplication	The user starts the application and places the Google Cardboard headset on the phone.
UC02	EnterChatroom	The user uses the main menu to join the chatroom and begin the virtual reality simulation.
UC03	Walk	The user uses the bobbing head mechanism to simulate walking in the virtual reality space, causing the user's avatar to move.
UC04	Chat	The user speaks out loud to transmit an audio message to nearby user avatars. In return, the user can hear their own audio replies.
UC05	StartGame	The user starts a game by interacting with an arcade machine asset inside the virtual reality space by using the magnet switch on the Google Cardboard headset.
UC06	PlayGameSinglePlayer	The user plays a single player game, finishes, then automatically returns to the main chat room.
UC07	PlayGameMultiPlayer	The user plays a multiplayer game, interacts with other user avatars, finishes, then automatically returns to the main chat room.
UC08	ExitGame	The user uses the options menu to exit the game currently being played. If it's a multiplayer game, the other users remain inside the game space.
UC09	ExitChatroom	The user uses the options menu to exit the chatroom and returns to the main menu.
UC10	ExitApplication	The user uses the main menu to exit the application and return to the phone's home screen.
UC11	InteractWithObject	The user uses the magnet slider to activate in-game objects (doors, arcade machines, menu screens, etc.)
UC12	ConnectToServer	The user can initiate multiplayer games by connecting to a remote server
UC13	CustomizeAvatar	The user can customize the color and features of his or her avatar using a series of menus
UC14	CreateArcadeRoom	The user can create his or her own room that other users can join
UC15	JoinArcadeRoom	The user can join a room that was created by another user
UC16	ViewScoreboard	The user can view the scoreboard which lists the highscores of all the players that played the various games

Table 3. Detailed Use Cases

	U C0 1	U C0 2	U C0 3	U C0 4	U C0 5	U C0 6	U C0 7	U C0 8	U C0 9	U C1 0	U C1 1	U C1 2	U C1 3	U C1 4	U C1 5	U C1 6	Tot al Use Ca ses
R01	x		x			x	x										3
R02		x			x	x	x	x			x	x	x				8
R03					x	x											2
R04					x		x										2
R05							x						x				2
R06					x	x	x				x						4
R07								x			x						2
R08					x	x	x				x						4
R09		x												x	x		3
R10		x		x		x	x		x	x				x	x		6
R11		x		x		x	x		x	x				x	x		6
R12		x		x		x	x		x	x				x	x		6
R13																x	1
R14														x	x		2
R15													x				1
Tota l Req uire men ts	1	5	1	3	5	8	9	2	3	3	4	1	3	5	5	1	

Table 4. Requirement Traceability Matrix

Analysis

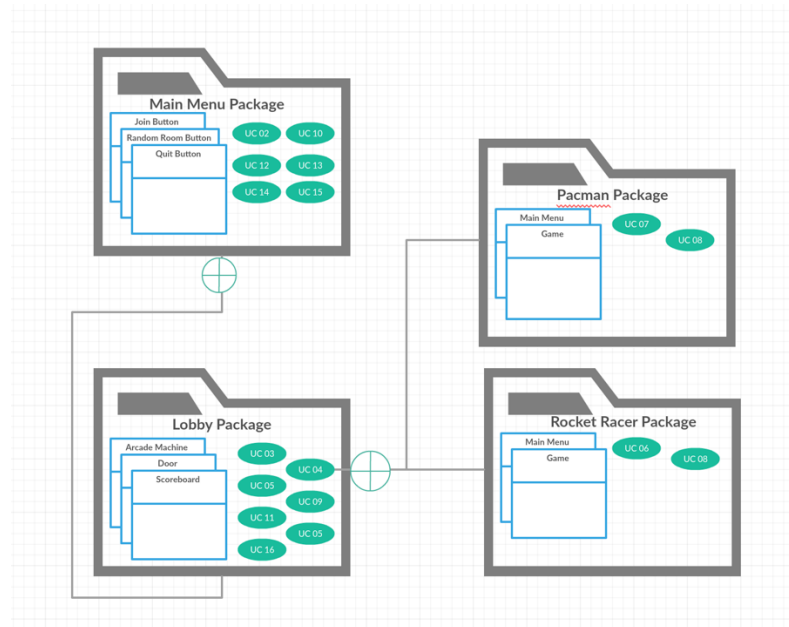


Figure 2. Analysis Package Diagram

The package diagram shows that the project has four analysis packages. The packages essentially represent separate rooms in the game and are loosely coupled modules.

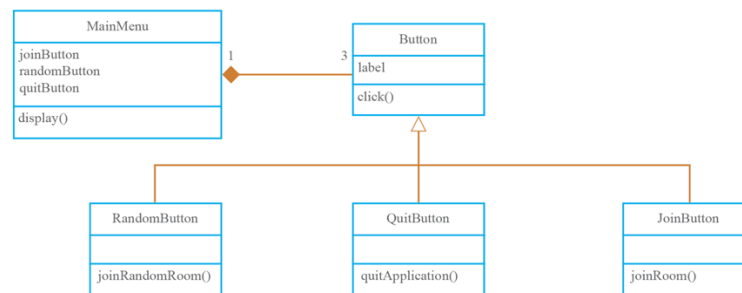


Figure 3. Main Menu Package

The main menu package is small and simple. It is the first room encountered by the user and displays the system's opening menu. The menu is a class composed of three buttons each with a different label and action associated with it. The JoinButton brings

the user to the main lobby. The QuitButton exits the application. The RandomButton puts the user in a random room.

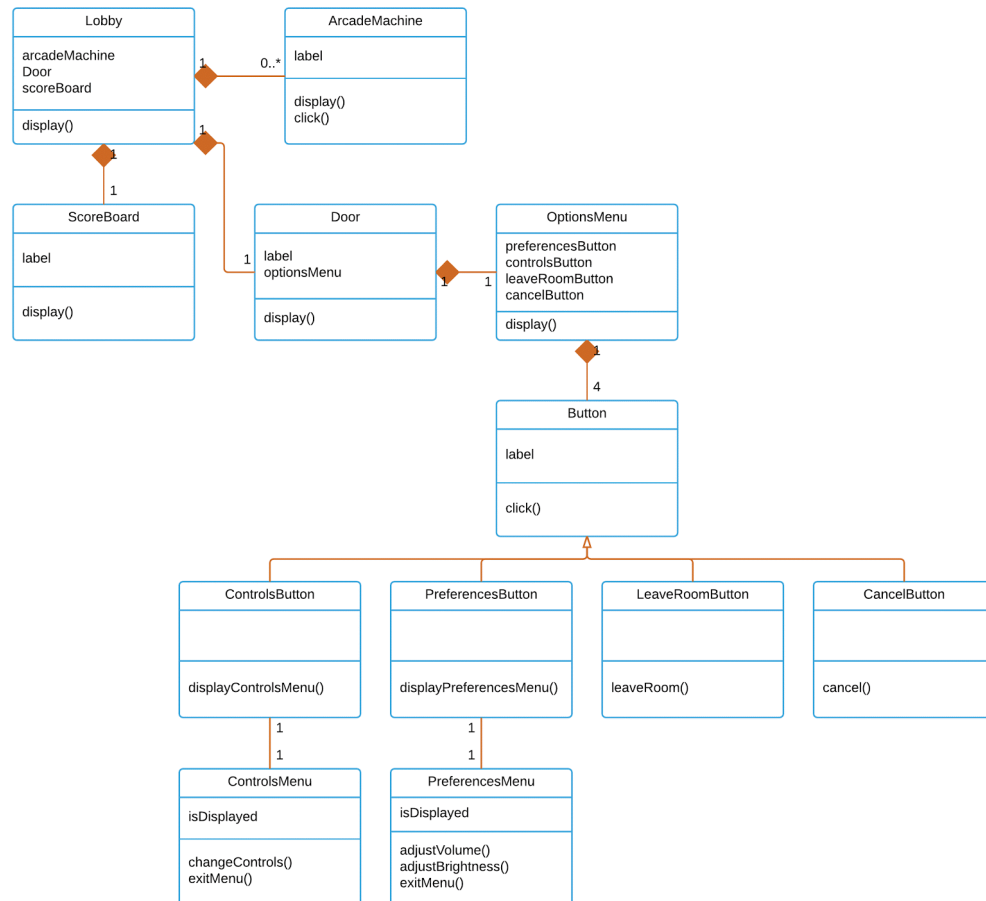


Figure 4. Lobby Package

The lobby is the room where users can interact with arcade machines, see the scoreboard, change settings, or leave the room. The `ArcadeMachine` lets the user start a game. The `ScoreBoard` lets the user see current high scores. The `Door` shows the `OptionsMenu`, which contains a `PreferencesButton`, `ControlsButton`, `LeaveRoomButton` and `CancelButton`. The `PreferencesButton` lets the user change the preferences. The

ControlsButton lets the user alter the controls to their liking. The LeaveRoomButton lets the user leave the room, and the CancelButton closes the OptionsMenu.

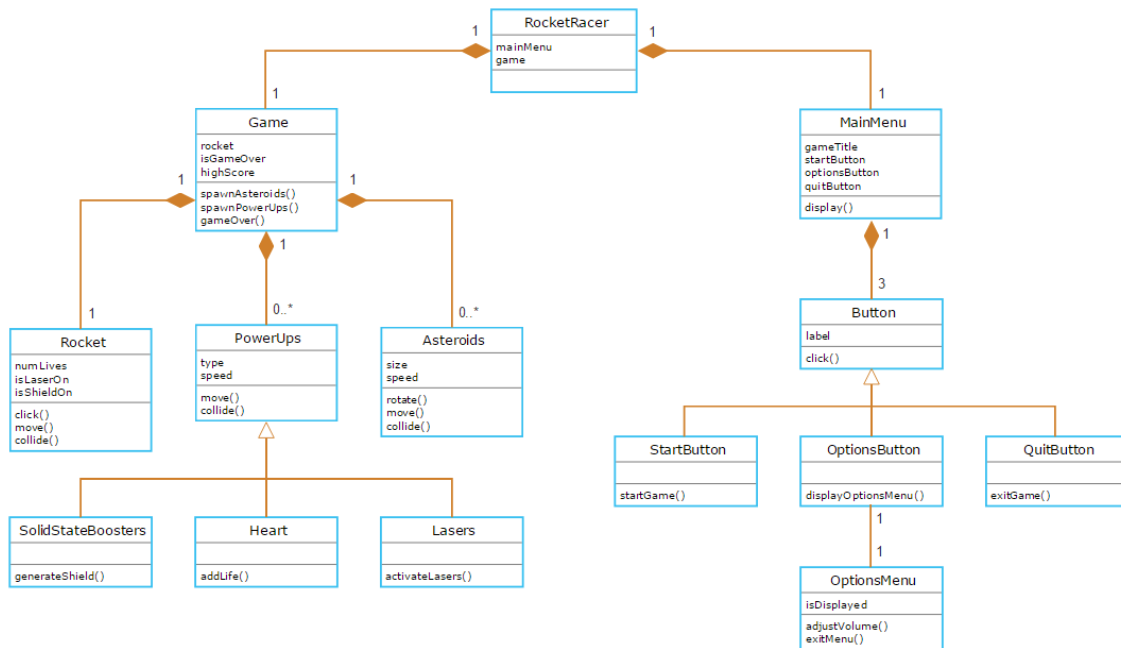


Figure 5. Rocket Racer Package

Rocket Racer is one of the arcade games that the user can play. The class RocketRacer consists of a MainMenu and the Game itself. MainMenu contains three buttons: StartButton, OptionsButton, and QuitButton. StartButton starts the game, OptionsButton brings up the options menu, and QuitButton quits the game. Game contains three objects: Rocket, PowerUps, and Asteroids. Rocket is controlled by the player and moves with tilt motions. Clicking the headset fires a gun. Powerups are objects that can be collected by the player when it collides with Rocket. Asteroids are what the player tries to avoid and they take away one life when it collides with Rocket.

Design

Architectural Design

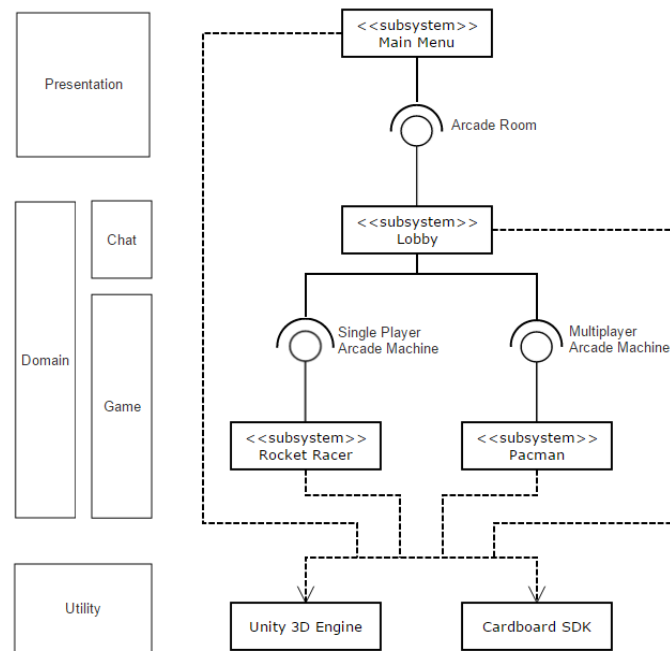


Figure 6. High-level Structure

The application is composed of a few subsystems: Main Menu, Lobby, and a subsystem for each game. The Main Menu subsystem is located within the presentation layer. It encapsulates components of the GUI, such as buttons, and can be reused for different applications and problem domains. The Lobby subsystem is part of the domain and chat layers. The domain layer consists of subsystems specific to the CHYS_rcade problem and business domains. The Lobby subsystem consists of components of the arcade room: players, arcade machines, etc. The game subsystems are located within the game layer. The game layer is specific to the gameplay aspect of the application, which includes both single and multiplayer games. The game layer is intended to allow games to be easily

added and removed from the application. The utility layer is composed of resources required for the application to function: Unity 3D Engine and Cardboard SDK.

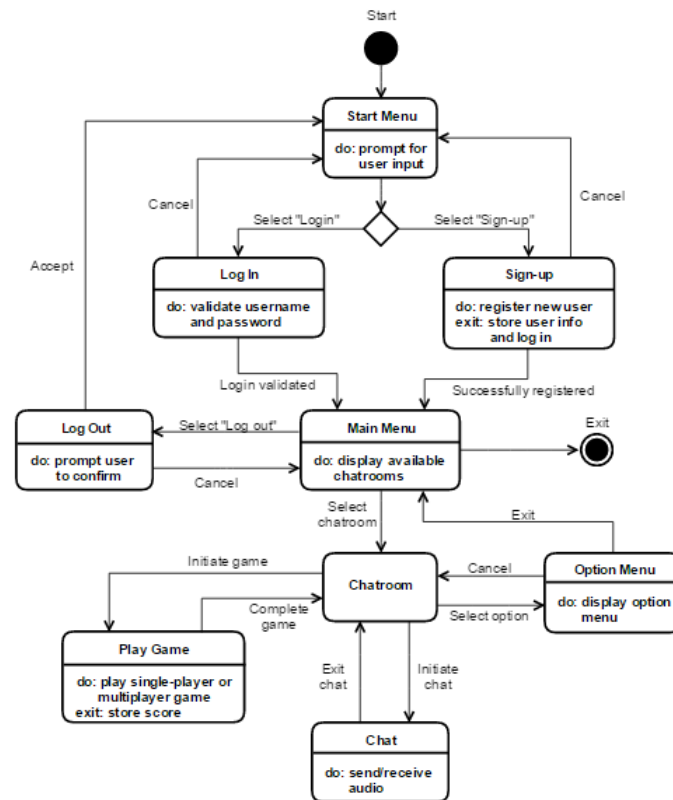


Figure 7. High-level Behavioral Diagram

The state chart shows the high level behavioral diagram for the CHYS_rcade system. The nodes Start Menu, Log in, Log Out, Main Menu, Sign-up, Option Menu, and Play Game are action nodes that represent different actions users can take throughout the application. The Chatroom node represents an object accessible through some of the actions. The connecting arrows show the different functions and events that occur to cause one action to lead to another.

Class Diagrams

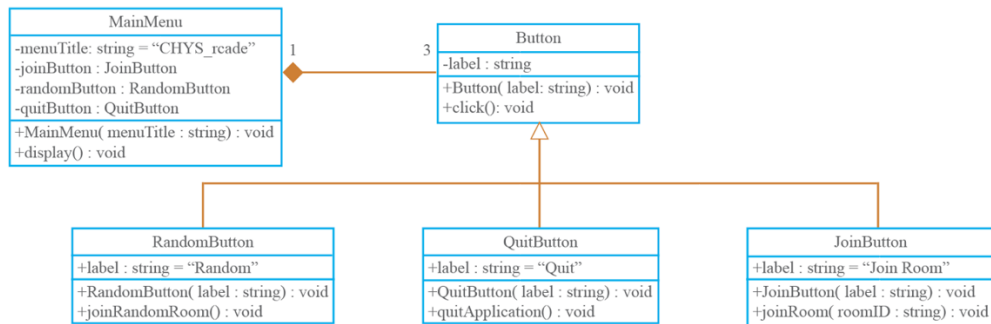


Figure 8. Main Menu Class Diagram

The Main Menu is the initial splash screen that the user sees when launching CHYS_rcade. The MainMenu consists of three Button objects. These objects are the RandomButton, QuitButton and JoinButton. RandomButton puts the user in a random room. QuitButton exits from the application. JoinButton puts the user in a specific room based on the room ID.

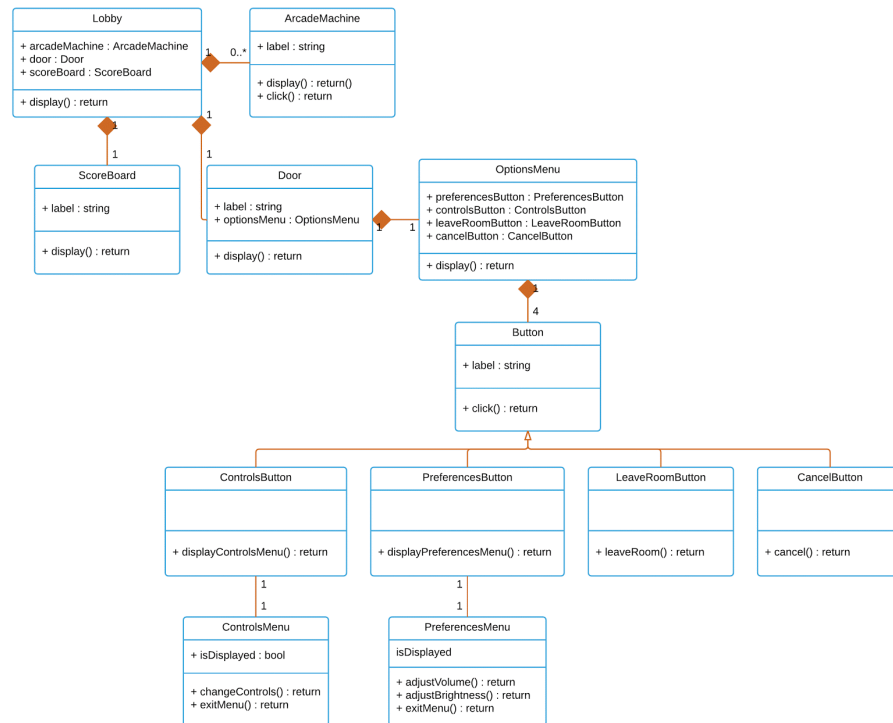


Figure 9. Lobby Class Diagram

The lobby is the room where users can interact with arcade machines, see the scoreboard, change settings, or leave the room. The ArcadeMachine lets the user start a game. The ScoreBoard lets the user see current high scores. The Door shows the OptionsMenu, which contains a PreferencesButton, ControlsButton, LeaveRoomButton and CancelButton. The PreferencesButton lets the user change the preferences. The ControlsButton lets the user alter the controls to their liking. The LeaveRoomButton lets the user leave the room, and the CancelButton closes the OptionsMenu.

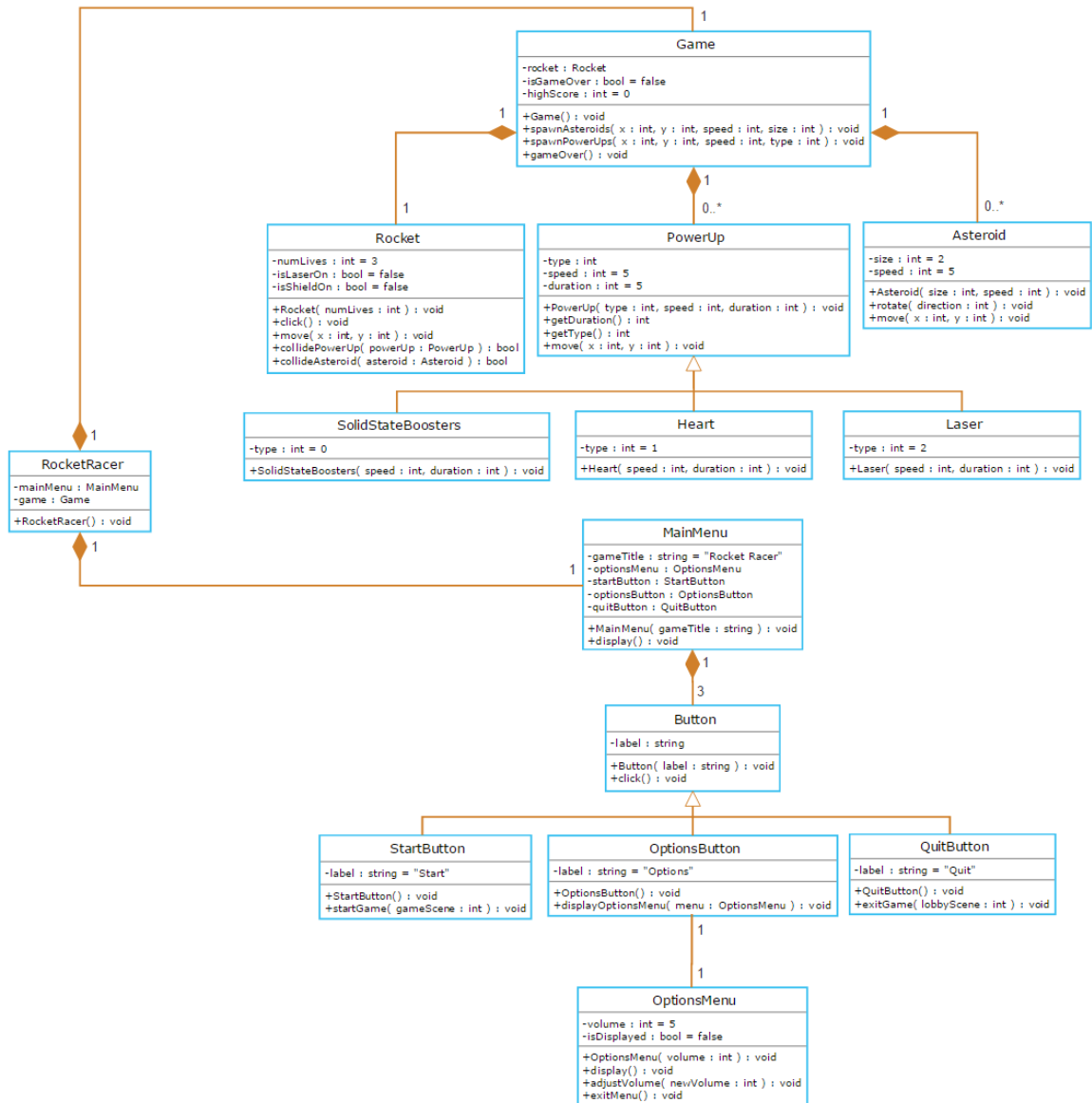


Figure 10. Rocket Racer Class Diagram

Rocket Racer is one of the arcade games that the user can play. The class RocketRacer consists of a MainMenu and the Game itself. MainMenu contains three buttons: StartButton, OptionsButton, and QuitButton. StartButton starts the game, OptionsButton brings up the options menu, and QuitButton quits the game. Game contains three objects: Rocket, PowerUps, and Asteroids. Rocket is controlled by the player and moves with tilt

motions. Clicking the headset fires a gun. Powerups are objects that can be collected by the player when it collides with Rocket. Asteroids are what the player tries to avoid and they take away one life when it collides with Rocket.

Program Units

Main Menu Classes

MainMenu

The MainMenu is the splash screen that the user sees when they first turn on the application

mainMenu()	Constructor for the RocketRacer class. Instantiates Game object and MainMenu object.	Author: Douglas Yan Input: None Output: None Exceptions: None
display()	This function will initialize the objects that the user will see inside the Lobby.	Author: Douglas Yan Input: None Output: None Exceptions: None

Button

Button is a template that will be instantiated by the various types of buttons employed within CHYS_Arcade. The user will be able to click on these buttons by using the magnetic slider on the Google Cardboard device

click()	This function is called when the user looks at the button and clicks on it. The implementations of Button will handle this function differently.	Author: Jeff Soriano Input: None Output: None Exceptions: None
---------	--	---

RandomButton

RandomButton is a button instantiated from the Button template. It brings the user to a random lobby.

joinRandomRoom()	This function brings the user to a random lobby.	Author: Douglas Yan Input: None Output: None Exceptions: None
------------------	--	--

JoinButton

JoinButton is a button instantiated from the Button template. It takes in a RoomID and uses it to put the user in a specific room.

joinRoom()	This function takes in a roomID as a string and sends the user to that room.	Author: Douglas Yan Input: roomID: string Output: None Exceptions: None
------------	--	--

QuitButton

QuitButton is a button instantiated from the Button template. It closes the application.

quitApplication()	This function closes the application.	Author: Douglas Yan Input: None Output: None Exceptions: None
-------------------	---------------------------------------	--

Lobby Classes

Lobby

The Lobby will be the location where the player is initialized when entering CHYS_Arcade. Inside the Lobby, the user will see instantiations of ArcadeMachine, Door, and ScoreBoard, and will be allowed to move and interact with the objects.

display()	This function will initialize the objects that the user will see inside the Lobby.	Author: Jeff Soriano Input: None Output: None Exceptions: None
-----------	--	---

ArcadeMachine

The ArcadeMachine is what the user will interact with in order to play the games within CHYS_Arcade.

click()	This function is called when the user looks at the ArcadeMachine and uses the magnetic switch on the Google Cardboard. After the function is called, the user will be transported to the game that he or she wishes to play.	Author: Jeff Soriano Input: None Output: None Exceptions: None
---------	--	---

Door - The Door is a visual representation of what the user will want to interact with in order to leave the Lobby. It will call the OptionsMenu when the user walks through it.

display()	This function initializes the visual display of the door and is called when the display() function of Lobby is called.	Author: Jeff Soriano Input: None Output: None Exceptions: None
-----------	--	---

ScoreBoard

The ScoreBoard displays all the high scores that have been achieved by the players of CHYS_Arcade

OptionsMenu

The OptionsMenu displays a colorful menu display that the user can interact with to change the various settings of CHYS_Arcade

display()	This function initializes the visual display of the menu and is called when the user walks through the Door.	Author: Jeff Soriano Input: None Output: None Exceptions: None
-----------	--	---

PreferenceButton

The PreferenceButton will reveal the preferences section of the OptionsMenu, where the user can manipulate visual and audio settings when click() is called.

displayPreferencesMenu()	This will display the preferences menu and its options to the user	Author: Jeff Soriano Input: None Output: None Exceptions: None
--------------------------	--	---

PreferenceMenu

The PreferenceMenu will reveal two sliders that will allow the user to adjust the brightness and the volume.

adjustVolume()	This will change the volume value based on where the volume slider is located.	Author: Jeff Soriano Input: None Output: None Exceptions: None
adjustBrightness()	This will change the brightness value based on where the brightness slider is located	Author: Jeff Soriano Input: None Output: None Exceptions: None

ControlsButton

The ControlsButton will reveal the controls section of the OptionsMenu, where the user can manipulate the controls of CHYS_Arcade when click() is called.

displayControlsMenu()	This will display the controls menu and its options to the user.	Author: Jeff Soriano Input: None Output: None Exceptions: None
-----------------------	--	---

ControlsMenu

The ControlsMenu will reveal various buttons, sliders, and options that represent the various controls of the game. Manipulating these things will change the controls upon leaving the menu.

changeControls()	This will set the controls based on what the user has selected on the menu screen.	Author: Jeff Soriano Input: None Output: None Exceptions: None
------------------	--	---

LeaveRoomButton

The LeaveRoomButton will transport the user back to the MainMenu screen when click() is called.

CancelButton

The CancelButton will remove the visual representation of the OptionsMenu from the screen, allowing the user to interact with CHYS_Arcade normally.

Rocket Racer Classes

RocketRacer

The RocketRacer class contains the main menu scene and game scene, and is responsible for instantiating the MainMenu and Game objects.

RocketRacer()	Constructor for the RocketRacer class. Instantiates Game object and MainMenu object.	Author: Truman Chan Input: None Output: None Exceptions: None
---------------	--	--

Game

The Game class is the game controller that manages the overall game play, rocket (player), and spawning of asteroids and power-ups.

Game()	Constructor for the Game class. Instantiates Rocket object for player. Sets default values for highscore to zero and isGameOver boolean flag to false.	Author: Truman Chan Input: None Output: None Exceptions: None
spawnAsteroids()	Function generates waves of asteroids objects that the rocket player must	Author: Truman Chan

	avoid. The position of the asteroid is determined by the x and y inputs. The speed and size are determined by the function parameters of the same name.	Input: x:int, y:int, speed:int, size:int Output: None Exceptions: None
spawnPowerUps()	Function generates PowerUp objects that the rocket player can collide with to activate special powers. The position and speed of the PowerUp is determined by the input parameters. The type parameter determines whether the PowerUp is a Heart, Laser, or SolidStateBoosters object.	Author: Truman Chan Input: x:int, y:int, speed:int, type:int Output: None Exceptions: None
gameOver()	Function sets isGameOver flag to true and ends game. The function displays the game score and high score for the game.	Author: Truman Chan Input: None Output: None Exceptions: None

Rocket

The Rocket class is the player controller that manages the rocket object movement and actions within Rocket Racer. The user moves the rocket by tilting the Google Cardboard. If the laser power-up is activated, clicking the screen will shoot lasers that destroy asteroid objects. The Rocket class also manages collisions with asteroids and power-ups.

Rocket()	Constructor for the Rocket class. Sets default values for numLives to 3. Sets boolean flags for power ups to false (isLaserOn, isShieldOn).	Author: Truman Chan Input: numLives:int Output: None Exceptions: None
click()	Function is called when user uses Google Cardboard trigger. If laser power up is activated, rocket shoots lasers that can destroy asteroids.	Author: Truman Chan Input: None Output: None Exceptions: None
move()	Function moves the rocket object to the x and y coordinates input parameters.	Author: Truman Chan Input: x:int, y:int Output: None

		Exceptions: None
collidePowerUp()	Function activates rocket ability based on PowerUp type (SolidStateBoosters, Heart, Lasers). Input accepts PowerUp object that collides with rocket. Returns true if rocket collides with object.	Author: Truman Chan Input: powerUp:PowerUp Output: bool Exceptions: None
collideAsteroid()	Function decreases player life if rocket collides with asteroid. Input accepts asteroid object that collides with rocket. Returns true if collision occurs.	Author: Truman Chan Input: asteroid:Asteroid Output: bool Exceptions: None

PowerUp

The PowerUp class is the base class for power-up objects in Rocket Racer. Power-ups grant players special abilities for a specified duration, such as invulnerability, lasers, and extra lives.

PowerUp()	Constructor for PowerUp class. Sets value for PowerUp type, speed, and duration of PowerUp after rocket collides with object.	Author: Truman Chan Input: type:int, speed:int, duration:int Output: None Exceptions: None
getDuration()	Returns integer value for PowerUp duration in seconds.	Author: Truman Chan Input: None Output: int Exceptions: None
getType()	Returns integer value of PowerUp type.	Author: Truman Chan Input: None Output: int Exceptions: None
move()	Function moves the PowerUp object to the x and y coordinates input parameters.	Author: Truman Chan Input: x:int, y:int Output: None Exceptions: None

SolidStateBoosters

The SolidStateBoosters class is a child class of PowerUps that grants players invulnerability for a specified duration. Rocket collisions with asteroids does not reduce the number of lives of the player.

SolidStateBoosters()	Constructor for SolidStateBoosters class. Sets values for object speed and PowerUp duration. Sets type member to default value of 0.	Author: Truman Chan Input: speed:int, duration:int Output: None Exceptions: None
----------------------	--	--

Heart

The Heart class is a child class of PowerUps that grants players an extra life. The heart power-up does not have any effect if a player has the maximum amount of lives.

Heart()	Constructor for Heart class. Sets values for object speed and PowerUp duration. Sets type member to default value of 1.	Author: Truman Chan Input: speed:int, duration:int Output: None Exceptions: None
---------	---	--

Laser

The Laser class is a child class of PowerUps that grants players the ability to shoot lasers for a specified duration. The lasers that the rocket shoots destroys asteroids upon collision.

Laser()	Constructor for Laser class. Sets values for object speed and PowerUp duration. Sets type member to default value of 2.	Author: Truman Chan Input: speed:int, duration:int Output: None Exceptions: None
---------	---	--

Asteroid

The Asteroid class is the main obstacle in Rocket Racer. Asteroid collisions with the

rocket reduces the player's life.

Asteroid()	Constructor for Asteroid class. Sets values for object size and speed.	Author: Truman Chan Input: size:int, speed:int Output: None Exceptions: None
rotate()	Function rotates Asteroid based on direction input (positive integer = clockwise, negative integer = counter-clockwise).	Author: Truman Chan Input: direction:int Output: None Exceptions: None
move()	Function moves Asteroid to the x and y coordinates input parameters.	Author: Truman Chan Input: x:int, y:int Output: None Exceptions: None

MainMenu

The MainMenu class is the menu scene where players can initiate the game, change game options, or quit the game.

MainMenu()	Constructor for MainMenu class. Instantiates buttons and options menu.	Author: Truman Chan Input: gameTitle:string Output: None Exceptions: None
display()	Function renders MainMenu and buttons onto game screen.	Author: Truman Chan Input: None Output: None Exceptions: None

Button

The Button class is the base class for button objects in Rocket Racer. Buttons are interactable objects that have special functionality based on the type of button.

Button()	Constructor for Button class. Input string defines the text displayed on the button.	Author: Truman Chan Input: label:string Output: None Exceptions: None
----------	--	--

click()	Function is called when user interacts with button. Action is defined by type of button.	Author: Truman Chan Input: None Output: None Exceptions: None
---------	--	--

StartButton

The StartButton class is a child class of Button that initiates the gameplay.

StartButton()	Constructor for StartButton class. Sets label to default value of “Start”.	Author: Truman Chan Input: None Output: None Exceptions: None
startGame()	Function is called when user interacts with button. Starts game and changes game scene to parameter value.	Author: Truman Chan Input: gameScene:int Output: None Exceptions: None

OptionsButton

The OptionsButton class is a child class of Button that displays the options menu.

OptionsButton()	Constructor for StartButton class. Sets label to default value of “Options”.	Author: Truman Chan Input: None Output: None Exceptions: None
displayOptionsMenu()	Function is called when user interacts with button. Calls display() function of OptionsMenu object, which renders the options menu.	Author: Truman Chan Input: menu:OptionsMenu Output: None Exceptions: None

OptionsMenu

The OptionsMenu class allows players to change the game settings.

OptionsMenu()	Constructor for OptionsMenu class. Sets default volume value to 5 and isDisplayed boolean flag to false.	Author: Truman Chan Input: volume:int Output: None Exceptions: None
---------------	--	--

display()	Function renders options menu onto game screen and sets isDisplayed boolean flag to true.	Author: Truman Chan Input: None Output: None Exceptions: None
adjustVolume()	Function sets volume to newVolume parameter value.	Author: Truman Chan Input: newVolume:int Output: None Exceptions: None
exitMenu()	Function closes options menu and sets isDisplayed boolean flag to false.	Author: Truman Chan Input: None Output: None Exceptions: None

QuitButton

The QuitButton class is a child class of Button that closes the game and returns the player back to the lobby.

QuitButton()	Constructor for QuitButton class. Sets label to default value of “Quit”.	Author: Truman Chan Input: None Output: None Exceptions: None
exitGame()	Function closes RocketRacer game and returns player to scene specified by lobbyScene parameter.	Author: Truman Chan Input: lobbyScene:int Output: None Exceptions: None

Detailed Design



Figure 11. Gameplay Activity Diagram

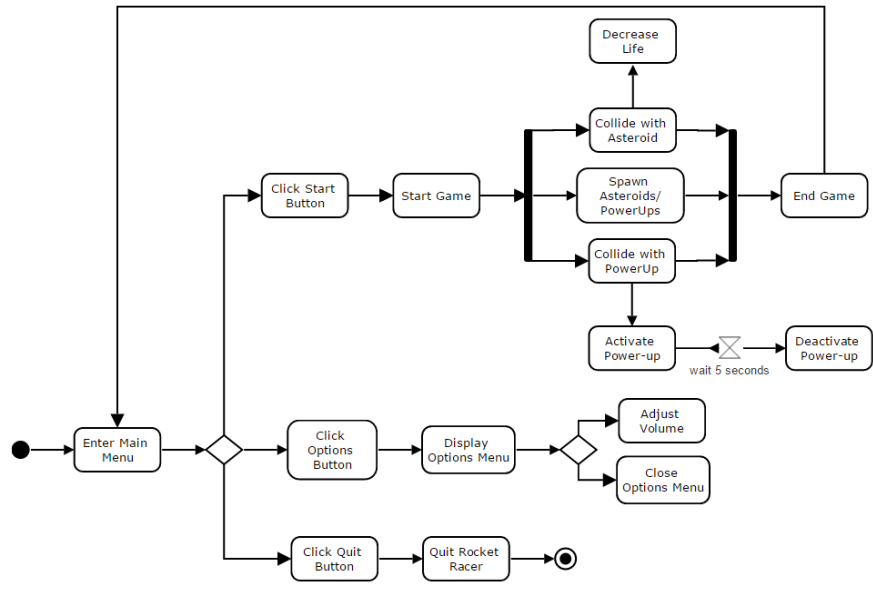


Figure 12. Rocket Racer Activity Diagram

Data Design

MainMenu	
Variable (type)	Description
menuTitle (string)	This string is the title of the object MainMenu.
label (string)	This string is the label for each button.
joinButton (button)	This button allows the user to join a room based on the roomID.
randomButton (button)	This button puts the user in a random room.
quitButton (button)	This button quits the application.

Table 5. MainMenu data design

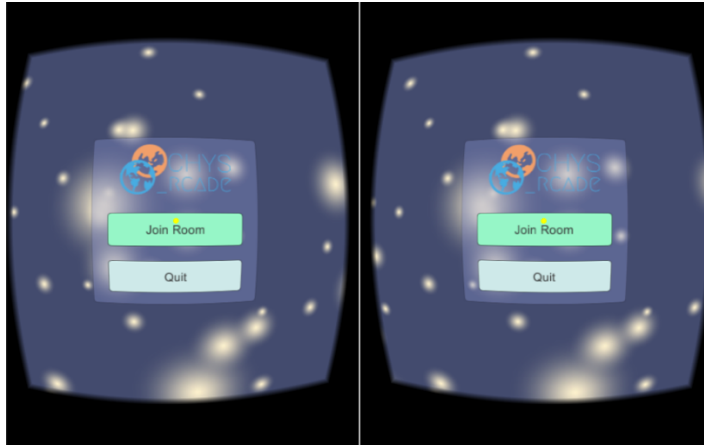
Lobby	
Variable (type)	Description
arcadeMachine (ArcadeMachine)	The arcadeMachine variable is an instantiation of the ArcadeMachine object and points to a game for the user to play.
label (string)	This string labels each button
optionsMenu (OptionsMenu)	The optionsMenu variable is an instantiation of the OptionsMenu object and is a menu for the user to interact with.
preferencesButton (button)	This button opens the preferences menu.
controlsButton (button)	This button opens the controls menu.
leaveRoomButton (button)	This button goes back to the MainMenu object.
cancelButton (button)	This button closes the controls or preferences menu.
isDisplayed (bool)	This bool tells whether a menu is active or not.

Table 6. Lobby data design

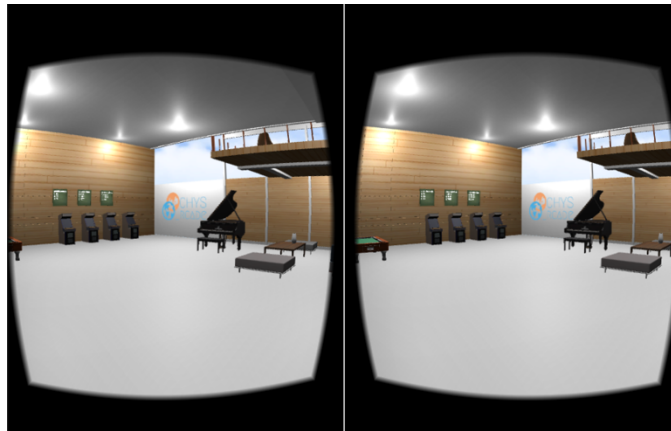
RocketRacer	
Variable (type)	Description
mainMenu (MainMenu)	This object is the main menu for the game and allows the user to start the game.
game (Game)	This is the game object that includes all the code for the game itself.
rocket (Rocket)	This object is the player avatar that they control in the game.
isGameOver (bool)	Boolean that represents whether or not the game is over.
highScore (int)	Integer that shows the high score.
numLives (int)	Integer that shows the number of lives the player has left.
isLaserOn (bool)	Boolean that represents whether or not the laser is active.
isShieldOn (bool)	Boolean that represents whether or not the shield is active.
type (int)	Integer that represents the type of power-up (boosters, heart, or laser).
speed (int)	Integer that represents the speed of a moving object.
duration (int)	Integer that represents how long a power-up is active.
size (int)	Integer that represents the size of an asteroid.
gameTitle (string)	String that holds the title of the game for the mainMenu.
optionsMenu (OptionsMenu)	Object that points to the options menu.
startButton (button)	A button that starts the game.
optionsButton (button)	A button that opens the options menu.
quitButton (button)	Quits the game and puts the user back in lobby.
label (string)	String that holds the label for each button.
volume (int)	Integer that represents the volume.
isDisplayed (bool)	Boolean that represents whether or not the options menu is displayed.

Table 7. RocketRacer data design

User Interface Design



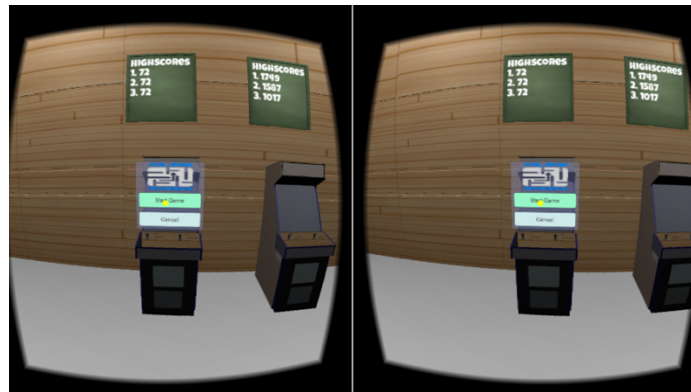
Snapshot 1. The user has the ability to join a room to enter the lobby or quit the application.



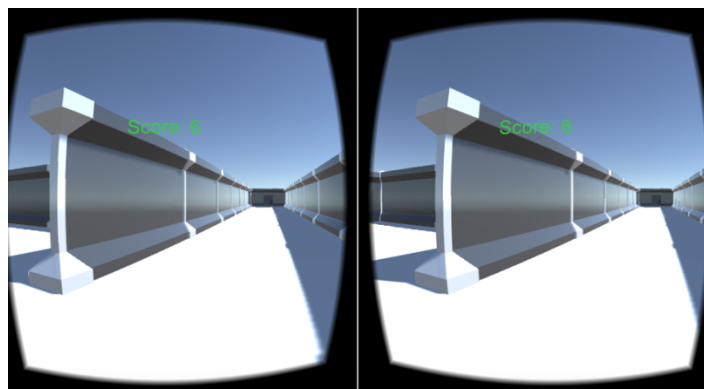
Snapshot 2. The first person point-of-view of the lobby in CHYS_rcade. This demonstration contains 4 arcade machines, in which the user can select to start Rocket Racer, Ch_oin Runner, Attack of the Ch_ubes, and Pewpew.



Snapshot 3. The Google Cardboard is a small cardboard box that the user can use with a smartphone to replicate a virtual reality headset.



Snapshot 4. Users can look around and interact with different objects in the game. To specify which object they are able to directly interact with, a yellow cursor appears on the screen.



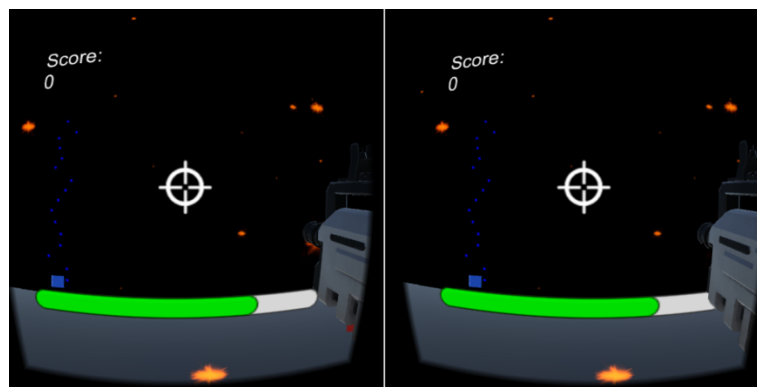
Snapshot 5. This snapshot shows the gameplay of Ch_oin Runner.



Snapshot 6. Walking up to and interacting with an arcade machine initiates the beginning of a game. This example shows the main menu screen for Rocket Racer.



Snapshot 7. This snapshot shows the gameplay of Rocket Racer. The user moves the rocket left and right by tilting the Google Cardboard. Clicking the trigger allows the user to shoot lasers and destroy the asteroids if the laser power-up is activated.



Snapshot 8. This snapshot shows the gameplay of Attack of the Chubes.

Conclusion

My team was able to develop a viable virtual reality game suite over the course of two semesters. The project shows the broad knowledge that each of the team members has gained through the pursuit of his degree. CHYS_rcade provides an immersive entertainment experience using easily obtainable materials. It is able to deliver next-generation gaming at an affordable rate.

The development of the project involved a number of intensive processes. The team refined the goals and purpose of our project by defining our functional and nonfunctional requirements as well as coming up with use cases to utilize for testing the program. Analysis of the early stages of our system enabled us to better structure the high-level portions of CHYS_rcade. Lastly, by following a strict design process for our low level elements, we were able to create a structurally sound program.

The project showcases the importance of the user interface and experience of programs. It also reveals the direction in which the video game industry is heading by demonstrating the up and coming technologies that are being developed by computer programmers and engineers. Our hopes are that the project will be able to catalyze the arrival of virtual reality entertainment to the home.

References

- [1] Wexelblat, Alan. 2014. Virtual Reality: Applications and Explorations. Academic Press.
- [2] F.P., Brooks. 1999. What's Real About Virtual Reality.
<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=799723>
- [3] Vince, John. 1995. Virtual Reality Systems. ACM Press/Addison-Wesley Publ. Co., New York, NY, USA.
- [4] Waldo, Jim. 2008. Scaling in Games & Virtual Worlds. *Queue* 6, 7 (November 2008), 10-16.
- [5] Zyda, Michael. 2005. From Visual Simulation to Virtual Reality to Games.
<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1510565&tag=1>
- [6] Tang Ying, Shetty, Sachin. 2011. Adaptive Virtual Reality Game System For Personalized Problem-Based Learning.
<<http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=5874957>>
- [7] Xiaoli, Guo, Li, Feng, Hong, Liu. 2010. Research on the Virtual Reality Simulation Engine. <<http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=5473359>>