University of Nevada, Reno
Computer Science and Engineering

Virtual Reality Physics Lab

Team 10:
Nicholas Bolling
Christopher Lewis
Andrew Munoz
Willem Zandbergen

Instructors
Sergiu Dascalu & Devrin Lee

External Advisors
Eelke Folmer & Benjamin Brown

November 16, 2017

1 Table of Contents

2 Abstract

The goal of Team 10's Virtual Reality Physics Lab is to utilize a new, revolutionary product for educational purposes. Virtual reality hardware, such as that in the HTC Vive, allows for personalized, immersive experiences which can address all forms of VARK (Visual, Aural, Read/write, Kinesthetic) learning. Very few environments outside of private tutoring have the same potential for an entertaining, educational experience. The VR Lab project will be filling this gap currently present in the virtual reality repertoire. The final product of the VR Lab will provide an immersive and educational experience into kinematic physics while promoting STEM (Science, Technology, Engineering, Mathematics) principles.

3 Introduction

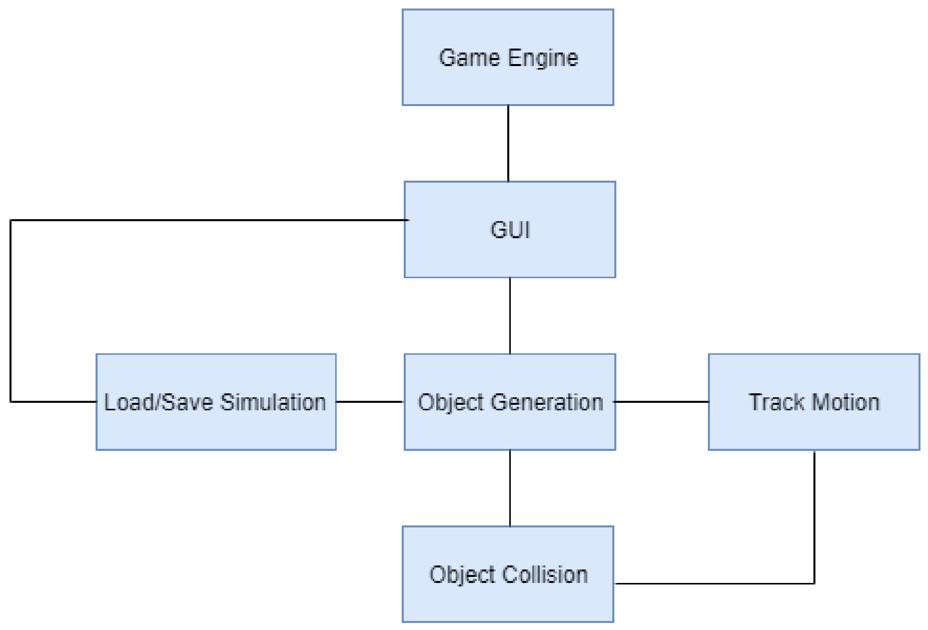
On the forefront of interactive games hardware are virtual reality headsets and their controllers. Products such as HTC's Vive, Oculus Rift, and Playstation VR provide immersive experiences that outperform what computer monitors can provide. It is with these virtual experiences that Team 10 will build an educational environment that can effectively use all forms of learning.

The commonly used forms of learning are shortened into the acronym "VARK" which stands for "Visual, Aural, Read/write, Kinematic." The typical classroom environment can at best address two of the four forms of learning. Classroom lectures are spoken to students who listen for important details while reading what a teacher has written on the subject and recording the details in notes for later use. Labs that are associated with classroom lectures address the remaining two forms of VARK learning. Students are given the opportunity to see what was being described in their lecture while interacting with the lecture material. Team 10's Virtual Reality Physics Lab will address all four forms of learning using a personalized experience in HTC Vive.

As mentioned, Team 10's Virtual Reality Physics Lab will incorporate all of the forms of learning usually restricted to two environments into one environment. This environment will be in the form of a "sandbox" environment. "Sandbox" environments encourage freedom of movement, where a user is left to their own devices to interact with various objects in the environment. Each object in the Virtual Reality Physics Lab will have information associated with the object that relays physics data relevant to the student's education. With the combination of an immersive "sandbox" experience and the information necessary to learning physics, students will leave the simulation having enjoyed learning about physics.

4 High-level and Medium-level Design

4.1 System Level Diagram



The graph above lays out a simple component-based high level design of the VR Physics Lab. Each component is essential to the system and the graph shows how each is associated with each other and the overall flow of the system.

4.1.1 Game Engine

The game engine is developed using the Unity game development platform. It runs the game and is connected to various classes including the graphical user interface (GUI).

4.1.2 GUI

The GUI is created through a class and is main interface the user will use before starting a simulation. It provides users with a visual space and acts as a tool for the user to view the different simulations available and once one is chosen the user is then placed into the simulation.

4.1.3 Object Generation

Once the user is placed into a simulation, objects are then generated for the user to interact with. Objects are generated through a class and then react to the user based on real physics, which is realized through the use of Unity assets.

4.1.4 Track Motion

Motion of the player and objects is tracked and captured using an HTC Vive Headset and controller. This will work through the use of two classes, a movement class and a controller class.

4.1.5 Object Collision
Through the use of classes, object collision will be detected. The data will be stored and processed in the object and physics classes and the data/score can be viewed in the GUI.

4.1.6 Load/Save Simulation
The Load/Save operation will be available to the user at any time during the playthrough. It can be accessed through a pause screen which will allow the user to save the game and then go back to the GUI. Loading the game can be accessed from the GUI as well as the in game pause screen.

4.2 Class Diagram
For class diagram, see Appendix A.

4.2.1 Classes
The classes listed below represent different functionality within the game. Each table lists the role of each class and how they are associated with each other.

Class	UserInterface
Role	Displays the information necessary for the player to complete the simulation.
Attributes	PlayerName gameDisplay:

Methods	main()
	RenderScreen()
	Buttons()

Class	Display
Role	The full screen the player sees when interacting with a simulation. Includes the objects present within the worldspace.
Attributes	entities height width

Methods	render()
	update()

Class	Objects
Role	Any physical, three dimensional object that is present in the worldspace.
Attributes	name Transform

Methods	generateObject()
	moveObject()

Class	Sound
Role	Sounds associated with objects and object physics.
Attributes	soundTitle object
Methods	playSound

Class	TriggerObjects
Role	Changes the transform of an object based on a trigger from the player or the worldspace.
Attributes	name forceData

Methods	GetPosition()
	GetRotation()
	GetScale()

Class	Physics
Role	The physical properties of an object.
Attributes	entity force mass acceleration

Methods	initialize()
	detectCollisions()
	computeForce()

Class	Movement
Role	The movement of objects in the worldspace which includes the player.
Attributes	entity player velocity

Methods	objectMove()
	playerMove()

Class	Score
Role	The player’s score for the simulation.
Attributes	scoreData playerData
Methods	displayScore()

Class	FileSystem
Role	Saves and loads a simulation that a player wishes to interact with.
Attributes	Filename playerData

Methods	saveGame()
	loadGame()

Class	GameState
Role	Records properties of the worldspace to be saved. Properties include player position in the worldspace, the number of objects in the worldspace, and the player's current score in the simulation.
Attributes	numObject position score

Methods	getPosition()
	getNumObjects()

Class	Player
Role	Holds information on the player currently in the worldspace.
Attributes	name score

Methods	player()
	play()

Class	Controller
Role	Interacts with the worldspace as the medium between player and simulation.
Attributes	ID player

Methods	controls()
	update()

Struct	Transform
Role	Holds the Position, Rotation, and Scale data for X, Y, and Z.
Attributes	Position X Position Y Position Z Rotation X Rotation Y Rotation Z Scale X Scale Y Scale Z
Methods	N/A

4.3 Methods

Each method is associated with a certain class and helps to make the game run efficiently and effectively. The methods are laid out in the table descriptions below and show which classes the method is related to as well as its main purpose within the system.

4.3.1

Name	Main
Description	The driving method.
Higher level unit	UserInterface
Input parameters	string
Output	void
Program subunits	None
Exceptions/Interruptions	Program Exit, Unexpected Crash
Additional comments	Main driver of the program, this will be running and calling other functions to update

4.3.2

Name	RenderScreen
Description	Displays the heads up display, objects, and worldspace to the player.
Higher level unit	UserInterface
Input parameters	Display
Output	void
Program subunits	None
Exceptions/Interruptions	None
Additional comments	RenderScreen is a main display component to the game and therefore does not rely other components.

4.3.3

Name	Buttons
Description	Displays interactive buttons on screen for players to select. Some of these buttons can instantiate objects for the player to use.
Higher level unit	UserInterface
Input parameters	Display
Output	void
Program subunits	None
Exceptions/Interruptions	None
Additional comments	This is part of the User Interface group

4.3.4

Name	render
Description	Renders the display on screen.
Higher level unit	Display
Input parameters	void
Output	void
Program subunits	None
Exceptions/Interruptions	None
Additional comments	Main function for displaying on-screen activity

4.3.5

Name	update
Description	Updates the game with additional calculations or transform changes.
Higher level unit	Display
Input parameters	N/A
Output	void
Program subunits	None
Exceptions/Interruptions	None
Additional comments	Updates the on-screen activity

4.3.6

Name	generateObject
Description	Instantiates an object for the player to use in the simulation.
Higher level unit	Objects
Input parameters	N/A
Output	bool
Program subunits	Physics, TriggerObjects
Exceptions/Interruptions	None
Additional comments	None

4.3.7

Name	moveObject
Description	Returns a boolian on movement of object
Higher level unit	Objects
Input parameters	N/A
Output	bool
Program subunits	None
Exceptions/Interruptions	None
Additional comments	This is to check for movement/displacement of an object

4.3.8

Name	GetPosition
Description	Returns the position of an object
Higher level unit	Objects, TriggerObjects
Input parameters	N/A
Output	int
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.3.9

Name	GetRotation
Description	Returns the rotation of an object.
Higher level unit	Objects, TriggerObjects
Input parameters	N/A
Output	int
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.3.10

Name	GetScale
Description	Returns the scale of an object.
Higher level unit	Objects, TriggerObjects
Input parameters	N/A
Output	int
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.3.11

Name	playSound
Description	Plays a sound associated with an object or action in the interface.
Higher level unit	Sound
Input parameters	object
Output	bool
Program subunits	None
Exceptions/Interruptions	None
Additional comments	Will return true or false depending on if the sound can be played

4.3.12

Name	displayScore
Description	Displays the current score for the player on the user interface.
Higher level unit	Score
Input parameters	int
Output	int
Program subunits	None
Exceptions/Interruptions	None
Additional comments	None

4.3.13

Name	objectMove
Description	Moves the specified object to position requested
Higher level unit	Physics, TriggerObjects, Movement
Input parameters	int Objects
Output	int
Program subunits	None
Exceptions/Interruptions	None
Additional comments	None

4.3.14

Name	playerMove
Description	Player movement through the worldspace.
Higher level unit	Player, Controller, Physics
Input parameters	int Player
Output	int
Program subunits	None
Exceptions/Interruptions	None
Additional comments	None

4.3.15

Name	initialize
Description	Initializes values for kinematic objects and their physical properties.
Higher level unit	Objects, Physics
Input parameters	None
Output	void
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.3.16

Name	detectCollisions
Description	Determines if an object has collided with the associated physics object.
Higher level unit	Objects, Physics
Input parameters	N/A
Output	void
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.3.17

Name	computeForce
Description	Calculates the appropriate force which moves the physics object.
Higher level unit	Objects, Physics
Input parameters	int int int Objects
Output	int
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.3.18

Name	saveGame
Description	Records data related to the simulation to a database for loading.
Higher level unit	Score, FileSystem
Input parameters	string Player&
Output	bool
Program subunits	GameState
Exceptions/Interruptions	Not enough filespace
Additional comments	None

4.3.19

Name	loadGame
Description	Loads saved data for a simulation.
Higher level unit	FileSystem
Input parameters	string Player&
Output	bool
Program subunits	GameState
Exceptions/Interruptions	Save File not found, No data to load, Corrupted Save
Additional comments	None

4.3.20

Name	getPlayerPosition
Description	Returns the position of the player
Higher level unit	FileSystem, GameState
Input parameters	N/A
Output	Float
Program subunits	None
Exceptions/Interruptions	None
Additional comments	None

4.3.21

Name	getNumObjects
Description	Returns the number of objects in the worldspace.
Higher level unit	GameState
Input parameters	N/A
Output	int
Program subunits	None
Exceptions/Interruptions	None
Additional comments	None

4.3.22

Name	Player
Description	Constructs the player and all associated values.
Higher level unit	GameState
Input parameters	N/A
Output	Player
Program subunits	Controller, Score
Exceptions/Interruptions	None
Additional comments	None

4.3.23

Name	play
Description	Enables player movement and interaction with the VR space
Higher level unit	Player
Input parameters	N/A
Output	void
Program subunits	GameState, Controller
Exceptions/Interruptions	None
Additional comments	None

4.3.24

Name	controls
Description	Interface between VR controller and GameEngine
Higher level unit	Player, Controller
Input parameters	string Player
Output	void
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.3.25

Name	update
Description	Updates the positions and interactions of the controllers.
Higher level unit	Player, Controller
Input parameters	None
Output	void
Program subunits	Movement
Exceptions/Interruptions	None
Additional comments	None

4.4 Database Tables

This database will hold the save and load data associated with each player.

4.4.1 Save/Load

Username	Sim

Username: The name associated with the player.

Sim: A list of simulations completed by the player.

4.4.2 Sim

Highscore	Objects

Highscore: The recorded high score for the player in a specific simulation.

Objects: A list of objects used to complete the simulation.

4.4.3 Object

Shape	Transform

Shape: Description of the object's shape

Transform: An associated list of the coordinates, orientation (rotation), and scale of the objects.

4.4.4 Transform

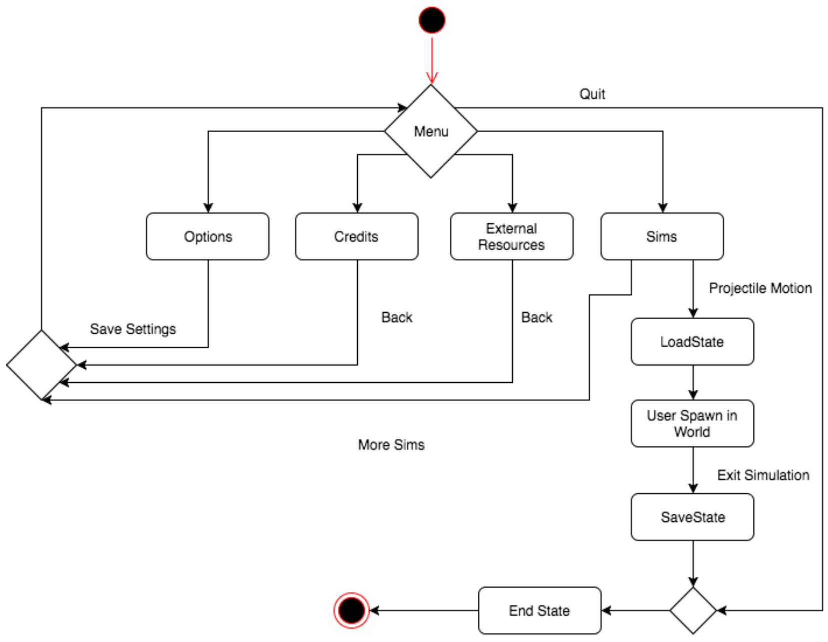
Position X	Position Y	Position Z	Rotation X	Rotation Y	Rotation Z	Scale X	Scale Y	Scale Z

Position X: The ‘x’ position of an object.
Position Y: The ‘y’ position of an object.
Position Z: The ‘z’ position of an object.
Rotation X: The ‘x’ rotation of an object.
Rotation Y: The ‘y’ rotation of an object.
Rotation Z: The ‘z’ rotation of an object.
Scale X: The ‘x’ scale of an object.
Scale Y: The ‘y’ scale of an object.
Scale Z: The ‘z’ scale of an object.

5 Detailed Design

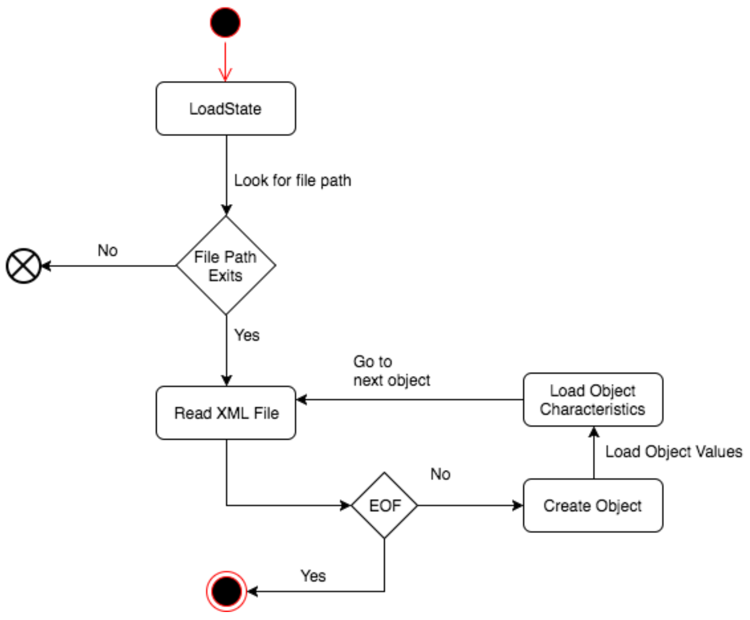
5.1 State Diagram (Entering Main Menu)

The following diagram specifies all the possible activities the user can take from the main menu. When the user chooses “Sims” they will be directed to a menu where they will choose between “More Sims” or “Projectile Motion”. Choosing “Projectile Motion” will continue to more use of the program which is the main game. When the game is over an “End State” will happen which is closing processes and initiating shutdown of the game.



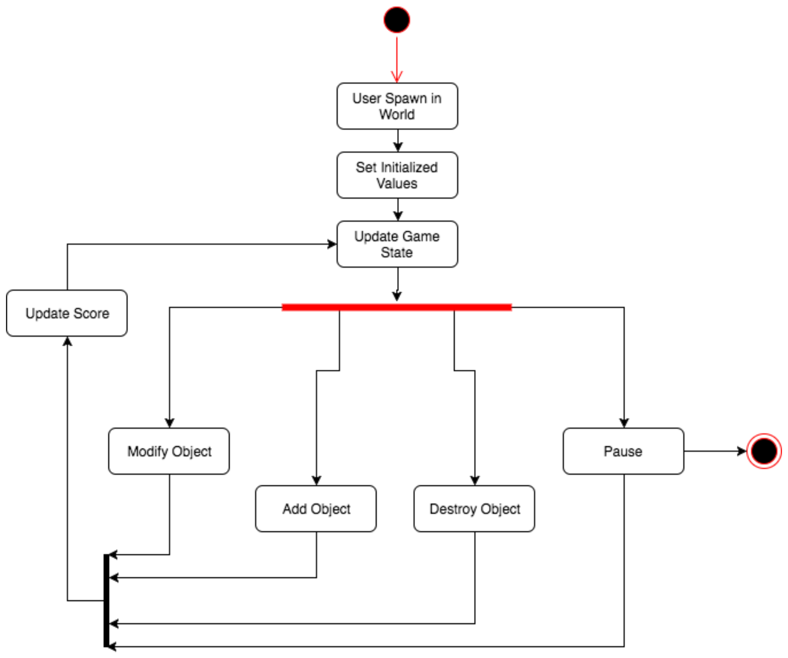
5.2 State Diagram (Entering Load State)

The following diagram describes what happens when the user starts an activity starting with a file path that includes saves. This includes everything that is needed before the actual simulation starts.



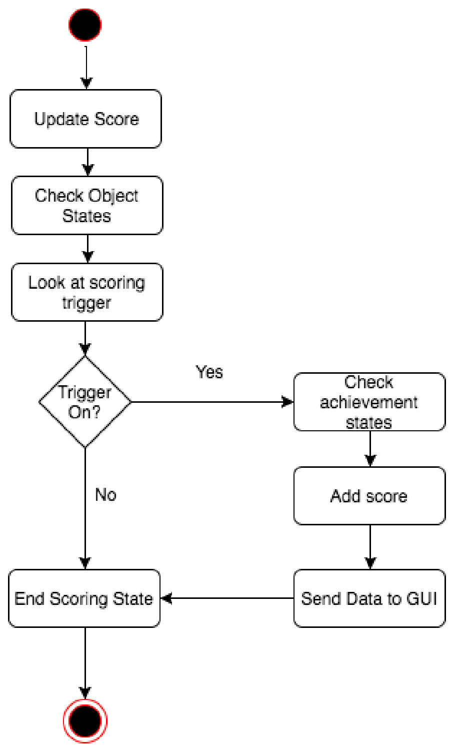
5.3 Activity Diagram (User Spawn in World)

Below is an activity diagram that specifies how the world is like in the game. At start the user will set the initialized values from Load State (See 5.2 Above) and the game state will update. From there the user can everything in the game from modifying an object to destroying an object. When they do an action the game updates allowing for further gameplay.



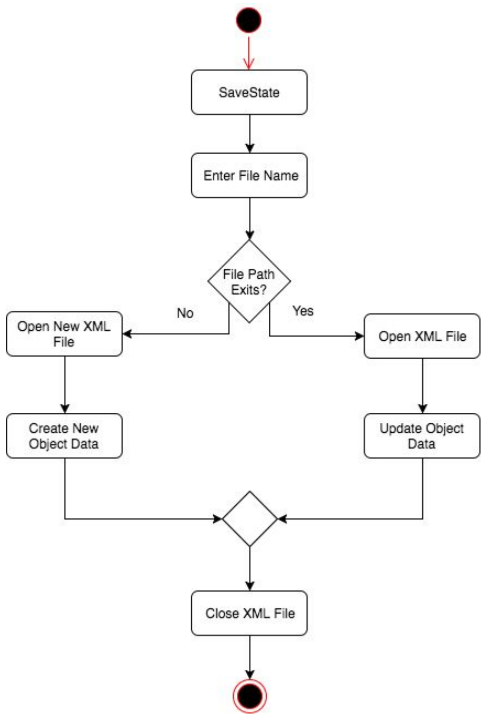
5.4 State Diagram (Update Score)

The following diagram describes the process of how the game is doing the score. When the score is updated, scoring must be checked if there is scoring on or off. This will affect whether or not score will be updated in game.

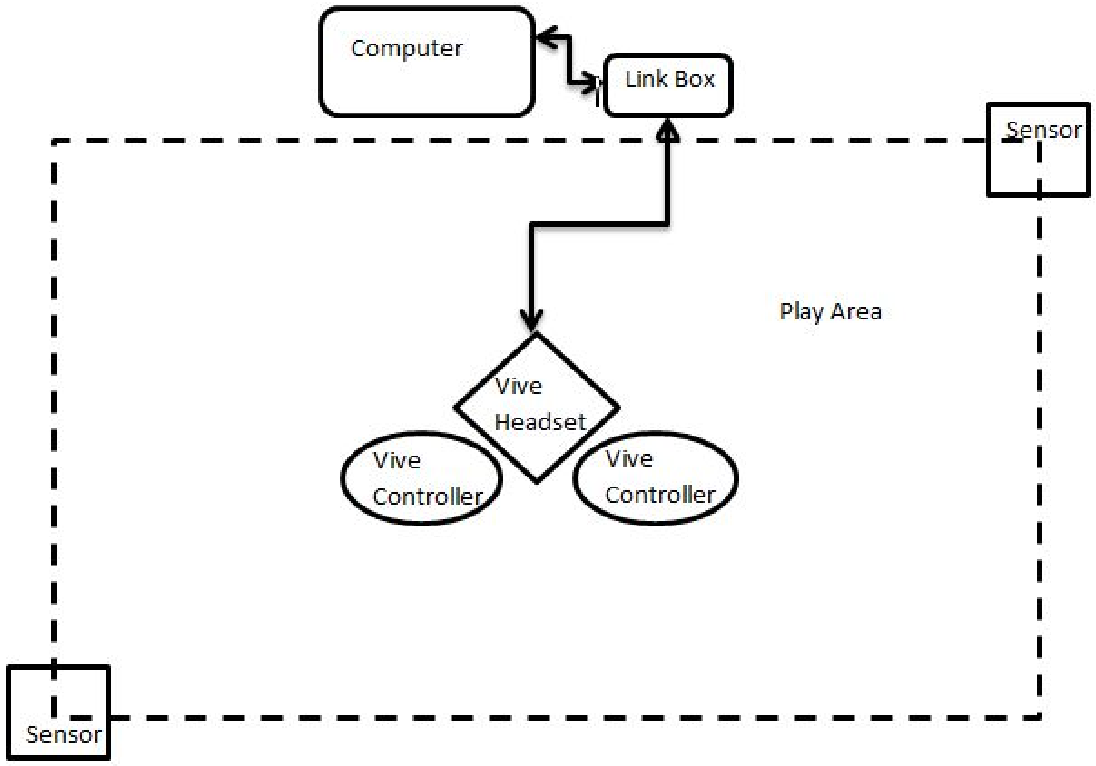


5.5 State Diagram (SaveState)

The following diagram describes the process that happens when the user quits the game. They user will be asked to enter a file name. Which leads to a designated XML file so when the user loads their game the Load State (See 5.2 above) will be able to pull the data. When the saving is done the XML file closes.



6 Initial Hardware Design



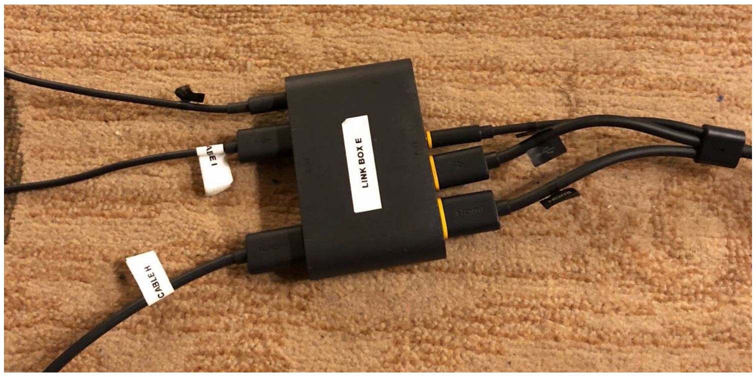
This diagram depicts the hardware view of the HTC Vive VR setup. The play area is virtually created, shown by the dotted lines. On the corners of the play area are the sensors to pickup headset and controller movement. In the center is the Vive headset and controllers which will be used to interface with the cyberspace. The Headset is wired with a audio, hdmi and usb that plugs into the link box, which in turn connects usb and hdmi to the computer where the simulation is run.

- HTC Vive Headset - This is the VR headset that will be showing the 3D world to the user.

- HTC Vive Controllers - These controllers will be used in conjunction with the VR headset to interface with the 3D environment.

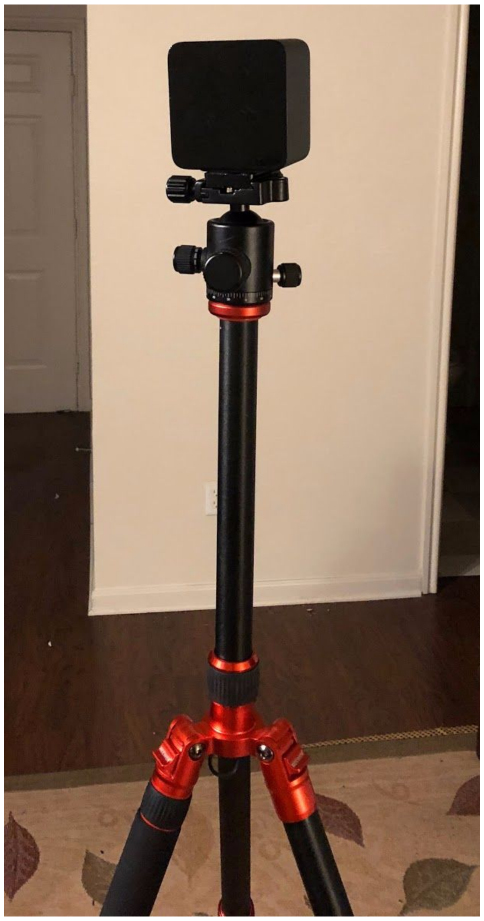
- HTC Vive Sensors - These sensors track the movement, position, and orientation of the VR headset and controllers.

- HTC Vive Link Box - This is the box that will plug into the cables from the headset, and into the computer to provide sensor data.



- Computer - A computer with powerful and capable hardware must be used to run the VR simulation.

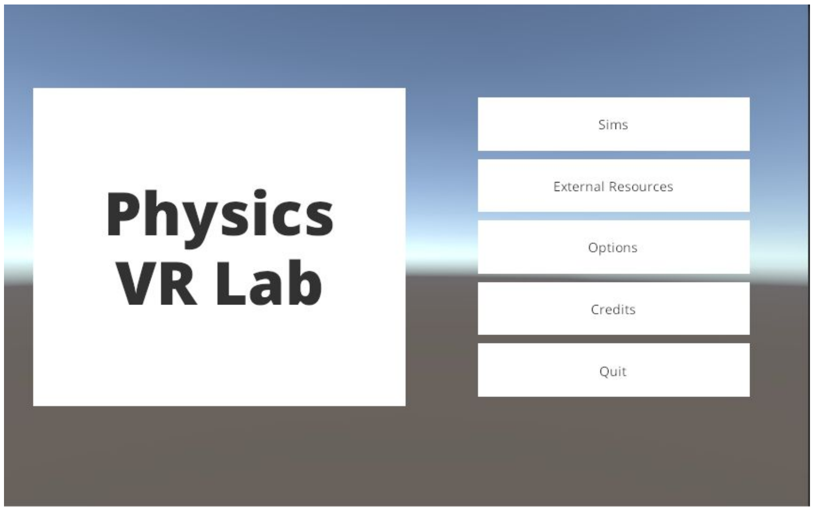
- Tripods (optional - not needed if Sensors are mounted) - These tripods help create the play area by providing boundaries and hold the HTC Vive sensors in position on the corners.



7 User Interface Design

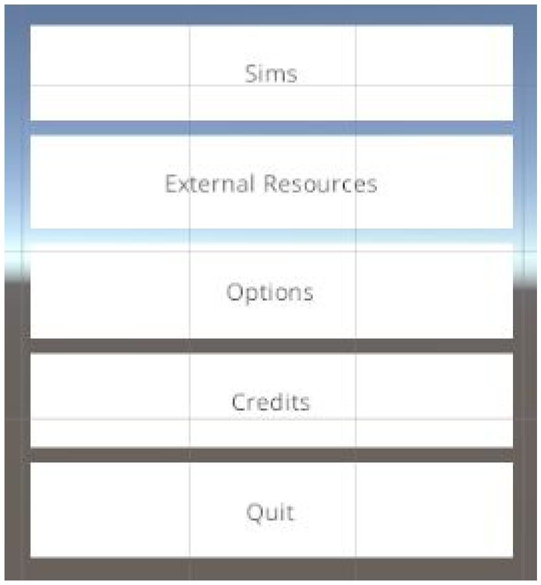
7.1.1 Main Menu

When the game starts the user will be directed to the main menu of the game which has multiple choices. The choices include the following: Sims, External Resources, Options, Credits, and Quit.



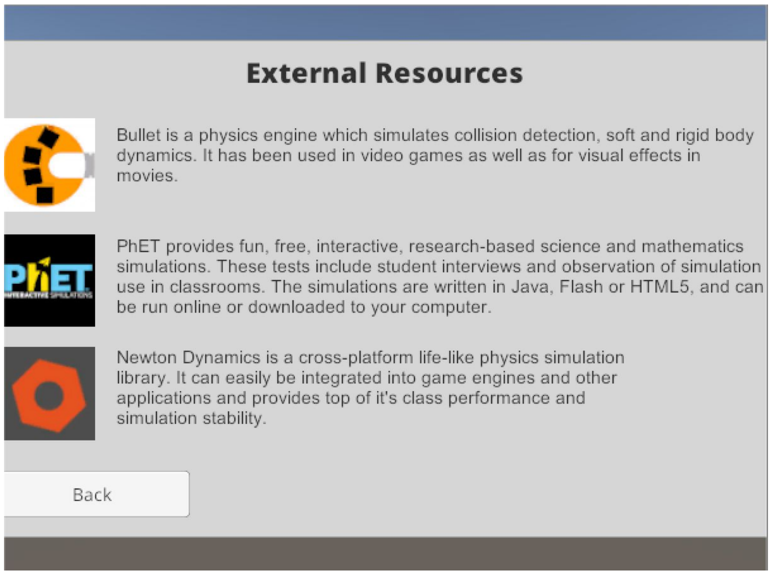
7.1.2 Main Menu Choices

These are the choices the player can choose from. Sims brings the player to a sub-menu which holds choices to what simulations the user may want to play. External Resources brings up a panel that shows possible other places the user may want to visit to learn more. Options allows the user to manipulate options before starting the game. Credits holds every team member's name and photo along with external advisors. The final choice is quit which closes the application.



7.2 External Resources

When the user chooses External Resources from the main menu a panel opens up which provides three places the user can visit: Bullet Physics, PhET Simulations, and Newton Dynamics. Along with each of these is a short description and their logo. The user can go back to the main menu by pressing "Back".



External Resources

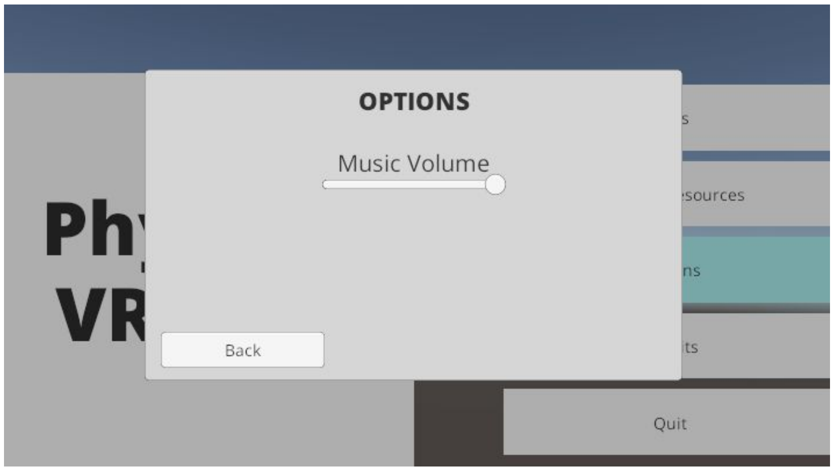
Bullet is a physics engine which simulates collision detection, soft and rigid body dynamics. It has been used in video games as well as for visual effects in movies.

PhET provides fun, free, interactive, research-based science and mathematics simulations. These tests include student interviews and observation of simulation use in classrooms. The simulations are written in Java, Flash or HTML5, and can be run online or downloaded to your computer.

Newton Dynamics is a cross-platform life-like physics simulation library. It can easily be integrated into game engines and other applications and provides top of it's class performance and simulation stability.

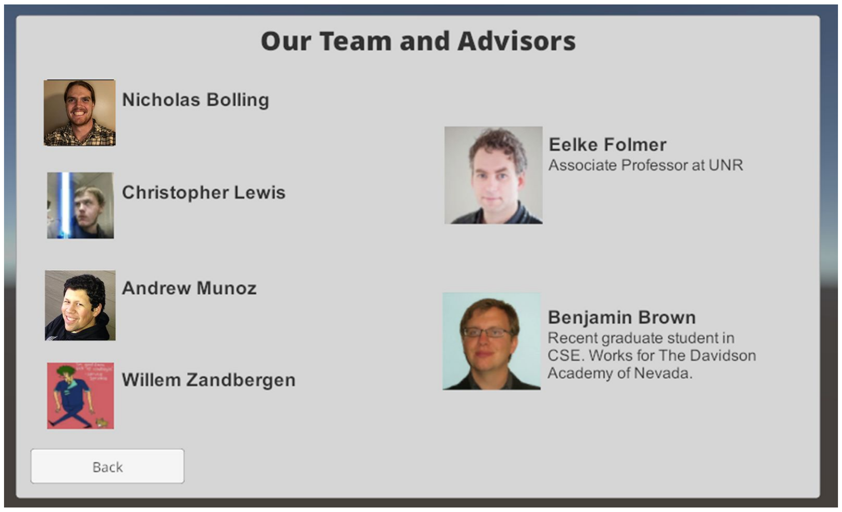
7.3 Options

When the user chooses “Options” a sub-menu opens up where only one option lies. This option is a music volume slider that changes volume. When the user slides it left the game music gets quieter. When the user is done they press back and the changes are saved.



7.4 Credits

When the user chooses “Credits” a panel opens up and on the left a list of names (with pictures) of all the members of the project in the game. On the right there are two individuals (with pictures) who are external advisors of the project. When user wants to go back to the main menu they click “Back”.



Our Team and Advisors

Nicholas Bolling

Christopher Lewis

Andrew Munoz

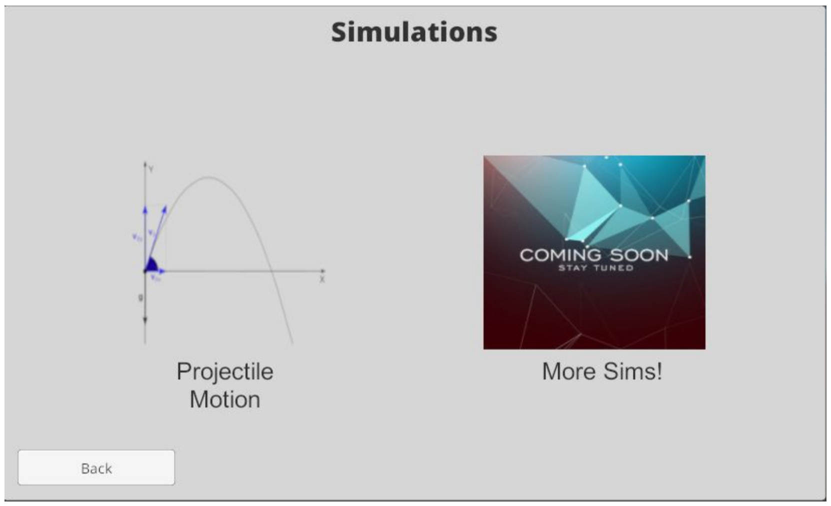
Willem Zandbergen

Eelke Folmer
Associate Professor at UNR

Benjamin Brown
Recent graduate student in CSE. Works for The Davidson Academy of Nevada.

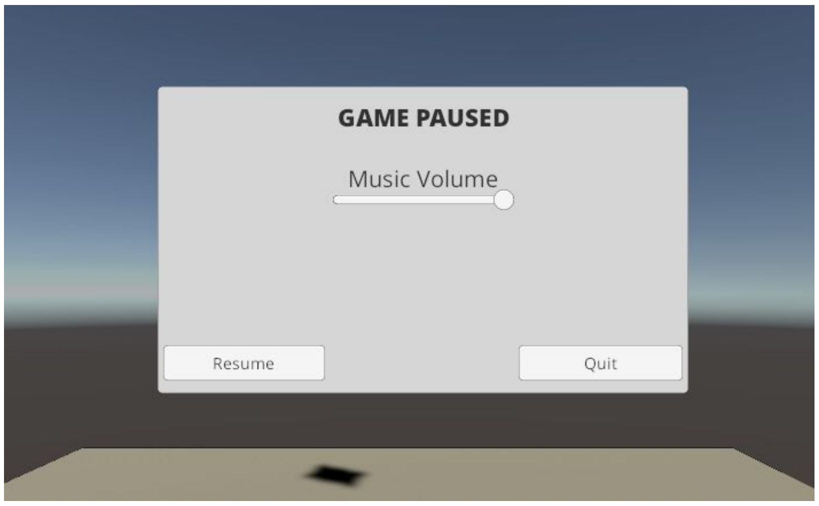
7.5 Sims

When the user chooses "Sims" a sub-menu shows up with two choices which are Projectile Motion and More Sims. When the user chooses "Projectile Motion" they will be redirected to the load their file and if they choose "More Sims" they will be redirected to the main menu since it is currently in development. If the user clicks "Back" they will be taken back to main menu.



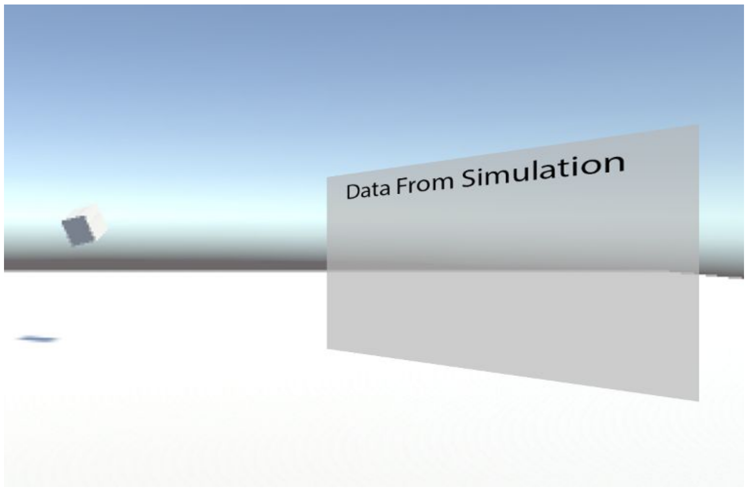
7.6 In-game pause

When in the game the user can go to the pause game screen indicated by "Game Paused" on the top. From there they can currently change the volume of the game and press resume or they can quit which takes the user back to the main menu. When the user quits the game data is saved into an XML file.



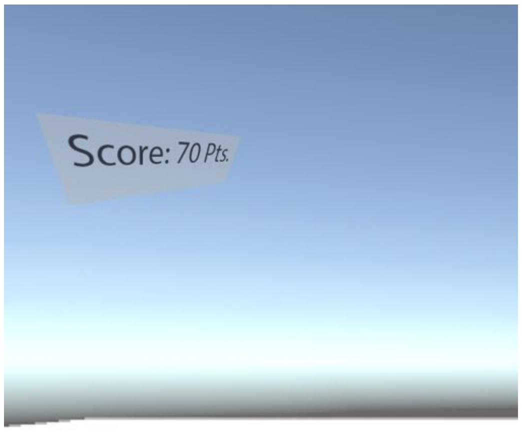
7.7 Sims

When in the game the user will be interacting with objects in various types of ways. When they decide to interact with an object data will be displayed on an in game canvas which will be labeled “Data From Simulation”. The data will be valuable to the user since the user will be able to do actions in the game and see data on the objects they interacted with.



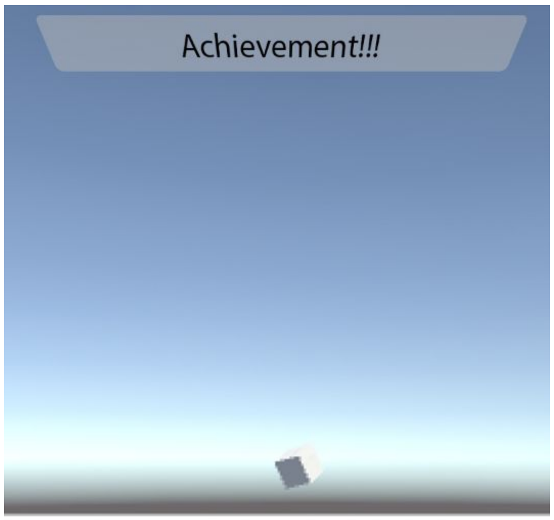
7.8 Score

When in game the user will be getting a score depending on whether the score is on or off. The score is tallied based on certain actions in the game. Depending on certain simulations (i.e Projectile Motion) the user will be indirectly tasked with completing a goal like shooting a target in the distance. The more shots it takes for the user to hit a target the score will be impacted.



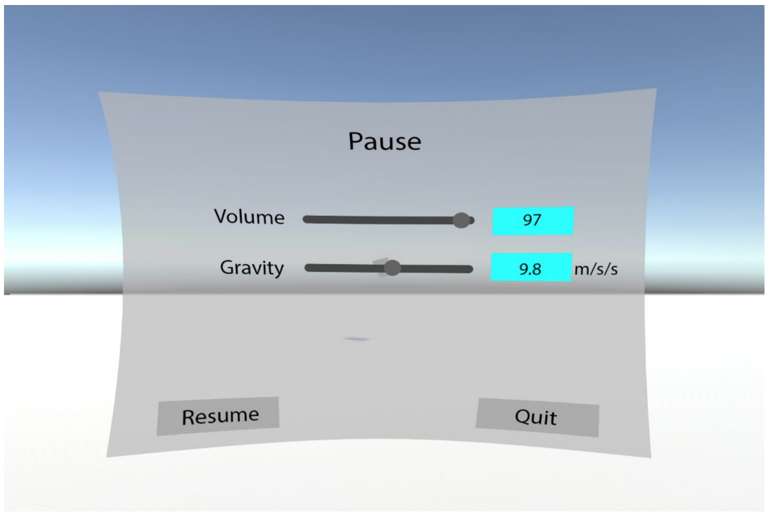
7.9 Achievements

Like scoring when the user does certain actions an achievement can be unlocked. The achievement banner is on the top portion of the screen and will stream with text when something special is done. An example of this would be like in the "Projectile Motion" application if they manage to destroy the target in under N (defined by a set number) shots.



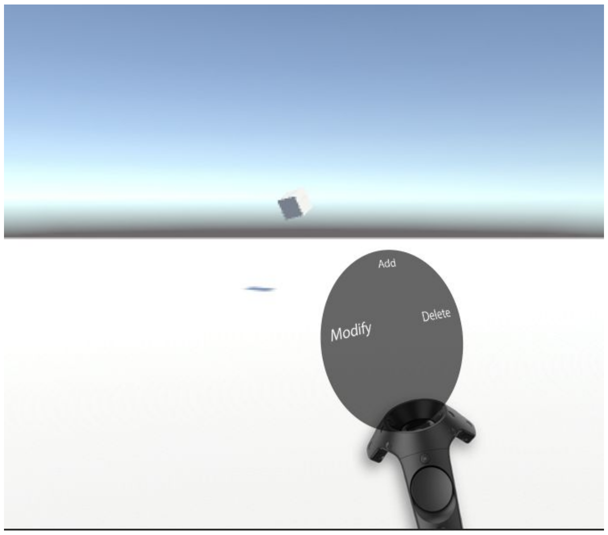
7.10 Pause

Below is what the game will look like after some adjustments have been made from the current in-game pause menu. The window will curve around the user and allow for more options like "Gravity" to be changed in the simulation. Changing things like gravity can affect how objects interact with each other in the simulation and will lead to further advancement.



7.11 Controller Window

When the user is in the simulation they will have access to: destroying objects, modifying objects, or adding objects. When they touch the right controller's thumbpad a submenu will pop-up will options to destroy, modify, or add. How the game will know what option the user is choosing the controller menu will be based on the position of the users thumb on the pad. The top part of the pad will indicate add, the left part will indicate modify, and left will indicate delete. Whichever option the user is currently on will be highlighted.



8 Glossary of Terms

1. Augmented Reality - The natural world is overlaid with a layer of digital content, typically through glass.
2. Avatar (Actor) - The Virtual representation of the player/controller in the VR environment.
3. Collision - an impact of one or more moving objects with another moving or stationary object.
4. Controller - An interface device similar to a mouse or keyboard that allows for buttons, typically joysticks, and toggles to interface with the computer.
5. Cyberspace - A computer synthesized reality. Often a computer synthesized 3-D space. See also: virtual reality.
6. Feedback - An output device that transmits pressure, force or vibrations to provide the VR participant with the sense of resisting force, typically to weight or inertia. This is in contrast to tactile feedback, which simulates sensation applied to the skin.
7. Field of View - The angle in degrees of the visual field displayed to the user.
8. Game Engine - The underlying framework and software behind a computer game or video game.
9. Generate (Generation) - the production of something, more specifically in computer science virtual reality and games, the production of an object in cyberspace.
10. GUI - Graphical User Interface - A software interface that allows the user to interact with electronical components though visual indicators and graphical icons instead of text or command line.

11. Haptics - Physical feedback from interacting with an object, usually integrated into the controller
12. Head Mounted Display (VR Headset) - A helmet or set of goggles that have small screens in front of each eye to generate images in three-dimensional space. This is typically combined with a Head Tracker to accurately display the view of the object.
13. Head Tracking - Monitoring the position and orientation of the head through specialized devices called Head Trackers.
14. Heads-up Display - A display device or interface that allows for graphics to be superimposed on the users view of the world.
15. Immersion - The perception of being physically present in a virtual or non-present world.
16. Kinethesis - Sensations derived from muscles, tendons and joints and stimulated by movement and tension.
17. Latency - A delay between user input and system response, typically user motion and tracker system response in Virtual Reality Environments. Delayed response time.
18. Play Area - The physical area in which the virtual world is simulated, this is drawn at time of starting SteamVR and tracked using the VR sensors.
19. Position Sensor - A tracking device that provides information about its location and/or orientation.
20. Sandbox game - A game type where there are no rules or objectives, the user can create and destroy at will, similar to a sandbox in real life.
21. Simulation - the imitation of the operation of a real-world process or system over time. In Virtual Reality, this refers to the simulation of “real life” or the simulation of a virtual world for the user to interact with.
22. Simulator Sickness (VR Sickness) - An unpleasant feeling that can include disorientation, nausea, and headaches when using a simulator or Virtual Reality. Similar to generic Motion Sickness.
23. Spatial Navigation - Self-orientation and locomotion in virtual worlds.
24. Tactile Feedback - Simulates touch and feel of objects in virtual reality.
25. Virtual Reality - A computer system that creates an artificial world in which a user has the ability to navigate through and interact with objects similar to the natural world.
26. Virtual Reality Environment - The whole space inside the Virtual Reality, essentially the “world” in which you are in while using Virtual Reality.

9 Contributions of Team Members

Listed below are the total working hours each team member contributed to this project as assessed by the team's project lead. Listed below these hours are the specific parts each member worked on for this project specification. Included in these times is the time spent as a group editing and formatting the final iteration of this paper.

Working hours:

Andrew:	11 hrs.
	High level and medium level design
Chris:	15 hrs.
	Detailed design, high level and medium level design, and user interface design
Nick:	7 hrs.
	Initial hardware design, glossary updates, and high level and medium level design
Will:	5 hrs.
	Cover page, table of contents, abstract, introduction, contributions of team members, and high level and medium level design

Appendix A
Class Diagram:

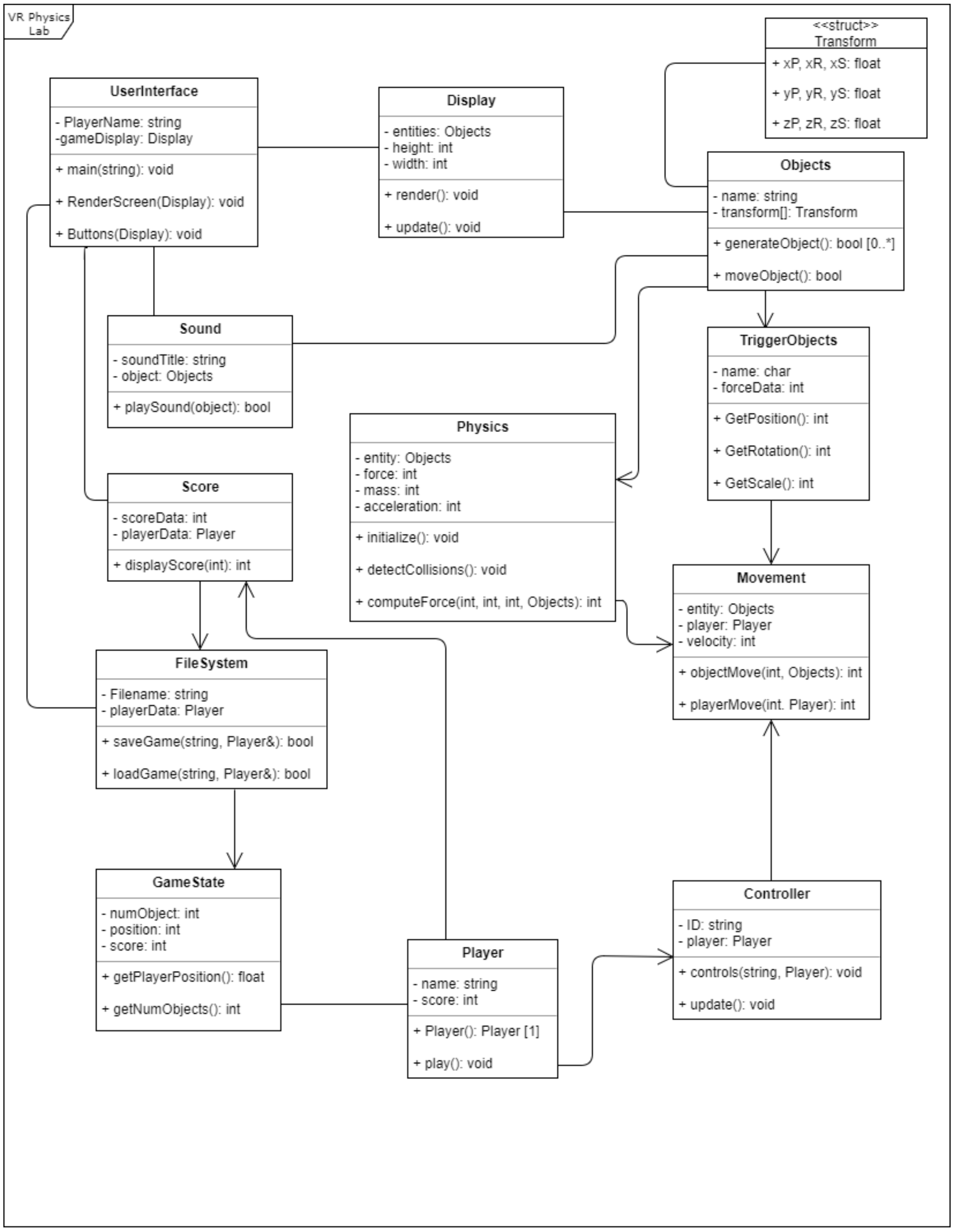


Fig A: The figure above details the classes within the VR Physics Lab and how they relate and associate with one another.