University of Nevada, Reno
Computer Science and Engineering

Virtual Reality Physics Lab

Project Part 2: Revised Specification & Design

Team 10:
Nicholas Bolling
Christopher Lewis
Andrew Munoz
Willem Zandbergen

Instructors
Sergiu Dascalu & Devrin Lee

External Advisors
Eelke Folmer & Benjamin Brown

February 23, 2018

Table of Contents

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.

Recent Project Changes
The project is a Virtual Reality game in which the user will be placed into different simulations where they will be able to learn about different physics concepts while also being able to have a fun and immersive experience. Game elements will be using these learned concepts to complete an objective (main plan is target practice). Since completing project 1 of CS 426, there have been no recent project changes that have occured.

Updated Specification
Summary of Changes
The main change in the specification of the project as compared to the specification document for CS 425 is within the Technical Requirements for the project. Since we were able to achieve some of the Tier One Requirements from the previous document, the requirements needed an update to narrow down our goals for this Spring semester. Software should remain the same with Unity though with added assets we will be modifying to ensure proper integration with our VR project.

Updated Technical Requirements
Tier One Functional Requirements

Requirement	Description
FR1.1	The system will enable the user to point at an intended destination with a curved ray.
FR1.2	The application will have the ability to let the player freely move around in a small space without teleportation.
FR1.3	The program will have a splash screen with the game title.
FR1.4	The program will have an interactable start button that will initiate the simulation.
FR1.5	The system will allow the player to shoot a projectile from an object.
FR1.6	The system will allow for data about the objects to be displayed on an in-game canvas.
FR1.7	The system will have the ability to save user profiles.

FR1.8	The system will enable the ability to load the user's profile.

Tier Two Functional Requirements

Requirement	Description
FR2.1	The program will have a line of motion for the intended projectile.
FR2.2	The program will allow the user to be able to spawn objects in the environment.
FR2.3	The program will have a scoring system.
FR2.4	The application will allow the player to choose separate environments.
FR2.5	The application will have intended targets for the player to hit.
FR2.6	The application will have an interactable options screen.
FR2.7	The application will have different types of projectiles that the player can use.

Tier Three Functional Requirements

Requirement	Description
FR3.1	The system will allow the user to toggle a color-blind mode.
FR3.2	The system will allow for the user to load their last session from where they ended.
FR3.3	The program will have achievements for the player.
FR3.4	The program will have an experience points system.
FR3.5	The program will enable the player to be able to interact with an in-game scoreboard.
FR3.6	The application will have air resistance as part of the simulation.
FR3.7	The application will enable to ability to toggle air resistance.

Non Functional Requirements

Requirement	Description
NFR1	The program will be interfaced with the HTC-Vive.

NFR2	The program will be programmed using C#.
NFR3	The program will have sound through a wired headset.
NFR4	The system have version control through Git.
NFR5	The system will be interlaced with motion controllers.
NFR6	The application will be developed using Unity.
NFR7	The application will tested using Steam VR.

Updated Use Case Modeling

Use Case Diagram:

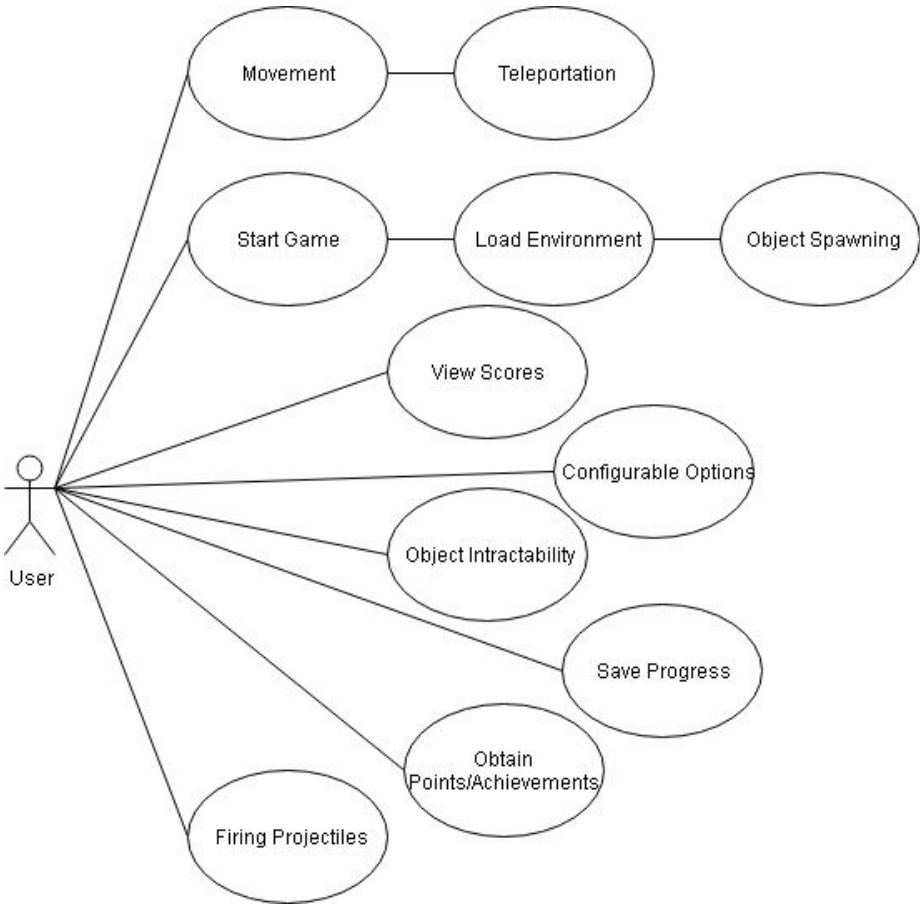


Fig. 1: The use case diagram shows the different use cases and how they relate to the user.

Detailed Use Cases:

Number	Name	Description

UC01	Teleportation	The user will be able to teleport to an intended destination. This will be done by pointing at the destination, pulling the trigger on the controller, and then teleporting to the intended destination.
UC02	Movement	The user will move within a fixed space without teleportation. This will be done by tracking the player's movement throughout the space/environment.
UC03	Start Game	From a splash screen a user will be able to access a specific simulation. Before the user starts the simulation a user can load their intended profile, check specific options for their simulation, and click on a start button to start the game. If the user's profile is not there the user can make a new profile.
UC04	View Scores	The user can view their scores to different activities in the game. For example, they will be able to see how much force they applied to certain object, such as a ball, and try to increase the amount of force applied to allow it to travel a farther distance and the user will receive a score.
UC05	Load Environment	The user will be able to load an environment with their profile. If the profile does not exist the user can create a new profile. The environment will be a catered toward a specific task or goal for the user to overcome.
UC06	Object Intractability	The user will be able to interact with placed objects in the environment, including shooting a projectile or interacting with a scoreboard. From there physics will be accurately represented on a canvas for the user to see.
UC07	Firing Projectiles	The user will be able to fire a projectile from a manipulable object toward an intended target. The data for that projectile will be processed and the projectiles line of motion will be displayed in real-time.
UC08	Save Progress	The user will be able to save their scores and progress on the different activities available. Saving will start once the player selects the save option and will then be followed by a screen that shows different users save files and asks the user which file they wish to save their game to.
UC09	Object Spawning	While in the simulation the user will be able to spawn given objects at will and place them in the environment. From there the user will be able to interact with those objects and display data about them on an in-game canvas.
UC10	Obtain Points and Achievements	During the simulation the user will be able to obtain points for specific actions. The points can be a score or experience toward further actions in the game. From there the user may obtain an achievement for their actions in the specific simulation.
UC11	Configurable options	From an in game options screen the user will be able to toggle specific enhancements. The enhancements will be toggling air resistance in the simulation and colorblind modes. Colorblind modes available will be: Protanopia, Deuteranopia, and Tritanopia.

Detailed Templates:

Use Case: Movement
ID: 2
Brief Description: The user will move within a fixed space without teleportation
Primary actors: User
Secondary actors: None
Preconditions: 1. Environment must exist for user to interact in.
Main flow 1. The use case starts once the user begins an activity. 2. The user controls the character and moves around the environment. 3. If a model is interacted with in the game 3.1 The model will react realistically based on real physics.
Post conditions: None.
Alternative flows: None.

Use Case: ViewScores
ID: 4
Brief Description: The user can view their scores to different activities in the game.
Primary actors: User
Secondary actors: None
Preconditions: 1. User must complete an activity to see score.
Main flow 1. The use case starts when the User begins an activity. 2. The User completes an activity. 2.1 Score is then displayed on screen. 3. If User views score from menu 3.1 Menu option shows user scores for each activity.
Post conditions: None.
Alternative flows: None.

Use Case: SaveProgress
ID: 8
Brief Description: The player saves their progress in the game to a file.
Primary actors: User
Secondary actors: None
Preconditions: 1. User must begin a new game before saving.
Main flow 1. The use case starts when the Player selects the save option from the menu. 2. The system displays a screen with different save options. 3. If the Player selects a save option 3.1 The system creates a save file that contains the Player's progress.
Post conditions: None.
Alternative flows: None.

Updated Traceability Matrix

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11
FR1.1	X	X				X					
FR1.2		X				X	X				
FR1.3		X							X		
FR1.4	X										
FR1.5	X										
FR1.6		X							X		
FR1.7			X								
FR1.8			X		X			X			
FR2.1				X			X				
FR2.2						X			X		
FR2.3			X		X			X			
FR2.4			X		X			X			
FR2.5							X				
FR2.6									X		
FR2.7			X		X			X		X	
FR2.9			X								X
FR3.1											X
FR3.2					X			X			
FR3.3				X						X	
FR3.4				X						X	
FR3.5				X						X	
FR3.6				X		X					
FR3.7											X

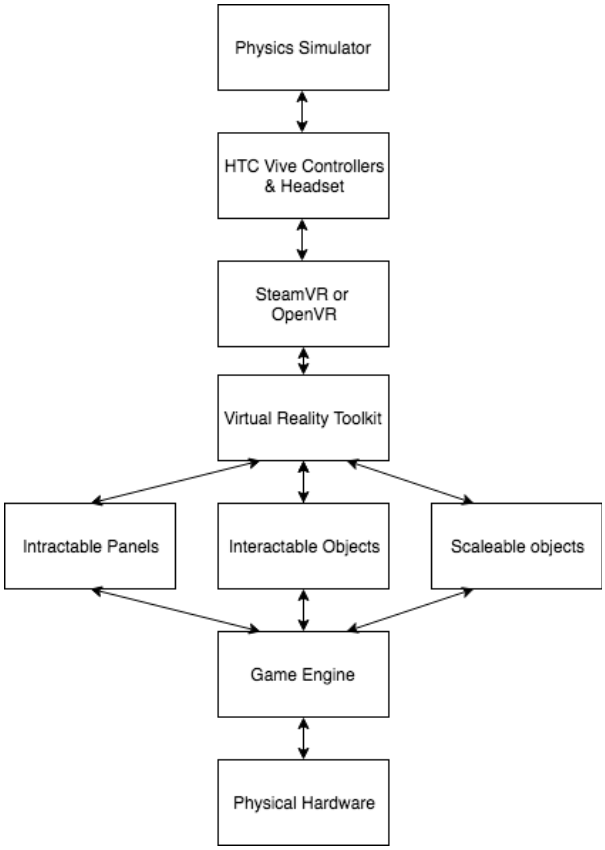
Updated Design

Summary of Changes

1. **Modeling -** Modeling the correct objects to then attempt texture application was important to ensure we could still spawn objects but also model our end game objects - Some, not all Models are done (many in beta).
2. **Textures** - We have figured out how to apply textures to our world objects in the way we intended, though it did take a bit longer to set up initially
3. **Main menu -** Being able to load into different levels and start the simulation/game is important feature to have for even a beta level program, for this there was a unity asset for selection using the controllers that could be utilized to improve menu interaction.
4. **Save/Load** - The team found a module (Chris to be specific) that has a save and load feature compatible with VR simulations/games that will be useful for implementing this feature, though it is still undergoing changes and proper implementation
5. **Design of other levels** - Design has been started on other levels to load that will provide a user with a sense of progression and achievement. This will be tied into game progression and main menu integration.
6. **Scores** - This is being implemented after the first level is fully textured and ready for game components, but there will be a HUD with the current score of the player in the level.

Updated High Level & 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.

Physics Simulator
The physics simulator is the game that will be developed using the Unity 3D engine.

Vive Controllers & Headset
The Vive controllers and headset are the user handled pieces of the system. When the user interacts with the headset and controllers that will decide what happens in the simulation.

SteamVR or OpenVR
SteamVR or OpenVR is an SDK that helps keep track of movement of the player and sends that to the headset.

Virtual Reality Toolkit
Virtual Reality Toolkit is a manager and SDK for VR. It comes with tools that will work with SteamVR and OpenVR directly.

Interactable Panels
Interactable panels will act as a GUI. This will enable to user to know what is happening at any time and will allow them to transition between scenes using interactable panels

Interactable Objects
Interactable objects make the game playable since the user will be interacting with objects to put in the physics environment.

Scalable Objects
Scalable objects make the game feel like there is variety to it since the user will be interacting with objects and being able to scale them will add some form of engagement.

Game Engine
The game engine will be managing all the data thats happening in the game. Any sort of hardware or software decision will make its way through the game engine.

Physical Hardware
The physical hardware will be the hardware the game will be ran on. This is at one end of the pipeline because it is the start of how the game will be ran. It starts from the hardware and makes its way to the software (the physics game).

Class Diagram
For class diagram, see Appendix A.

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

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.

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

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Database Tables

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

Save/Load

Username	Sim

Username: The name associated with the player.

Sim: A list of simulations completed by the player.

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.

Object

Shape	Transform

Shape: Description of the object's shape

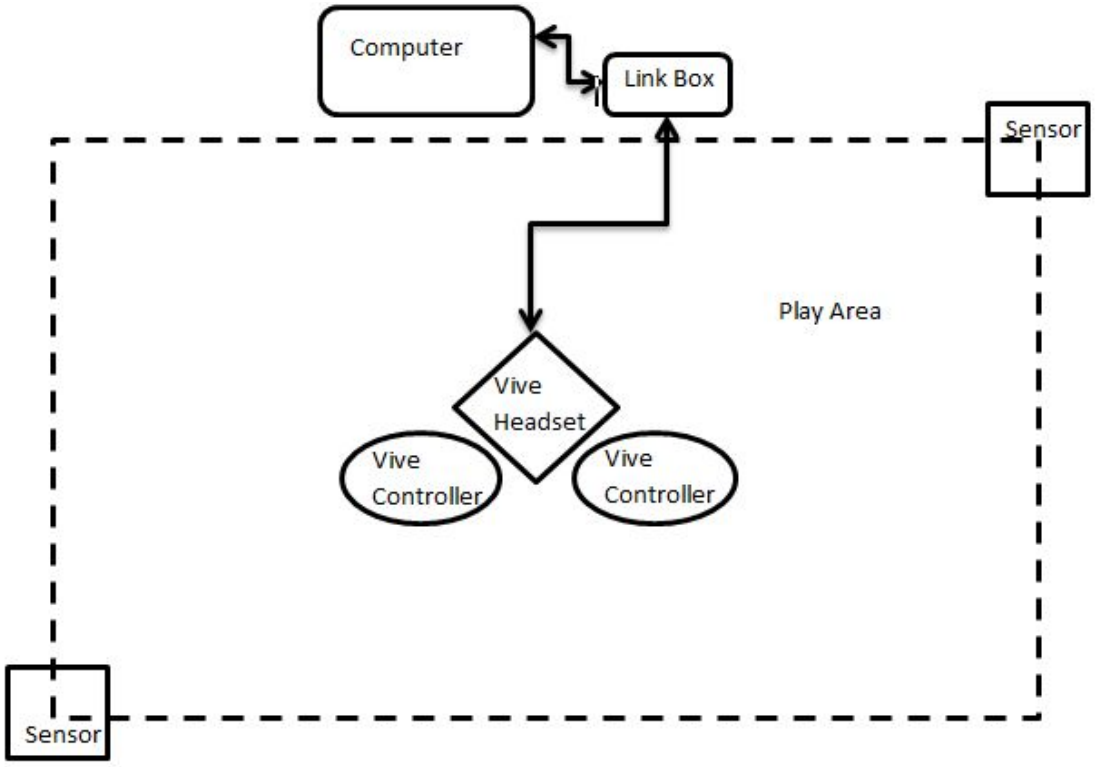
Transform: An associated list of the coordinates, orientation (rotation), and scale of the objects.

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.

Updated Hardware Design

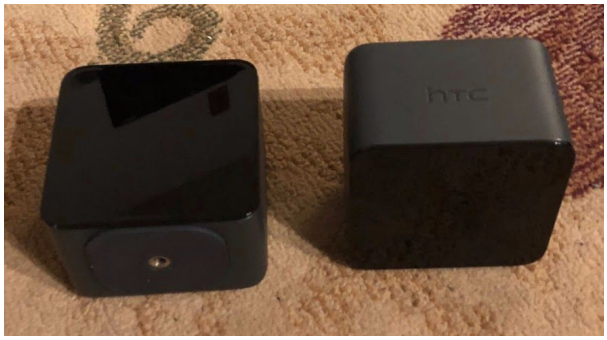


This diagram depicts the hardware view of the HTC Vive VR setup. The play area is virtually created, shown by the dotted lines. On the corners of the play area are the sensors to pickup headset and controller movement. In the center is the Vive headset and controllers which will be used to interface 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.

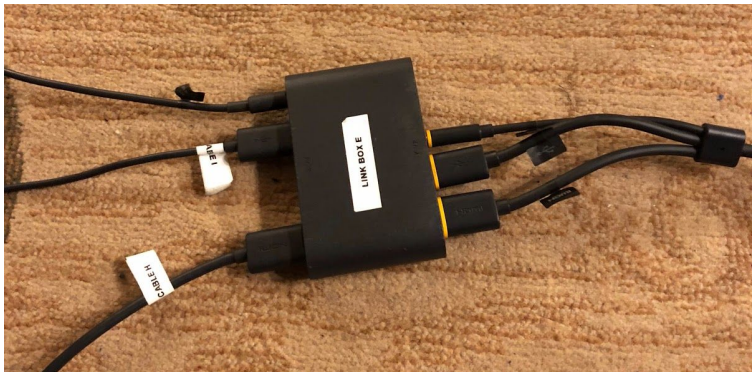
- HTC Vive Headset - This is the VR headset that will be showing the 3D world to the user.

- 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.

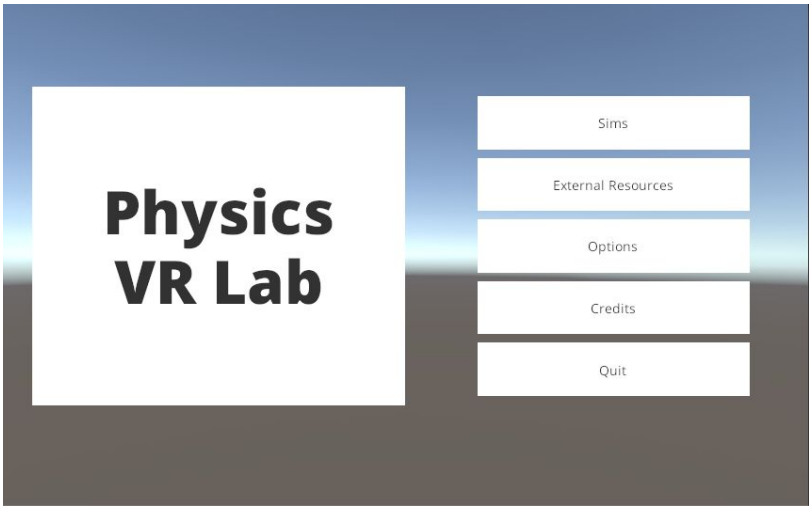

Data Structures:

1. **Linked list**
    a. We will need a linked list for objects that have added properties or attachments
2. **Queue**
    a. We need to be able to load items into the projectile cannon and have them queue properly to fire
3. **Stack**
    a. For enemies/Targets we will need a stack of entities that will be able to be created/modified/destroyed
        i. Potentially a Vector can be used for this as well but for now, stack implementation seemed like a good initial start
4. **XML Files**
    a. For saving and loading data for the users. This will make for easy file transfer protocol between systems.

Updated User Interface Design

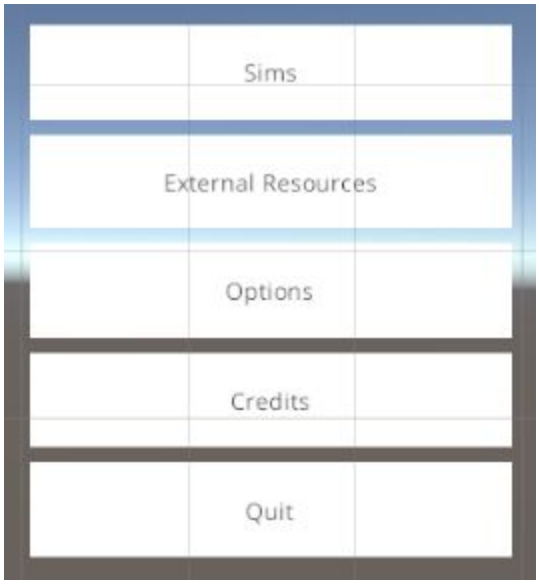
User Interface Design

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.



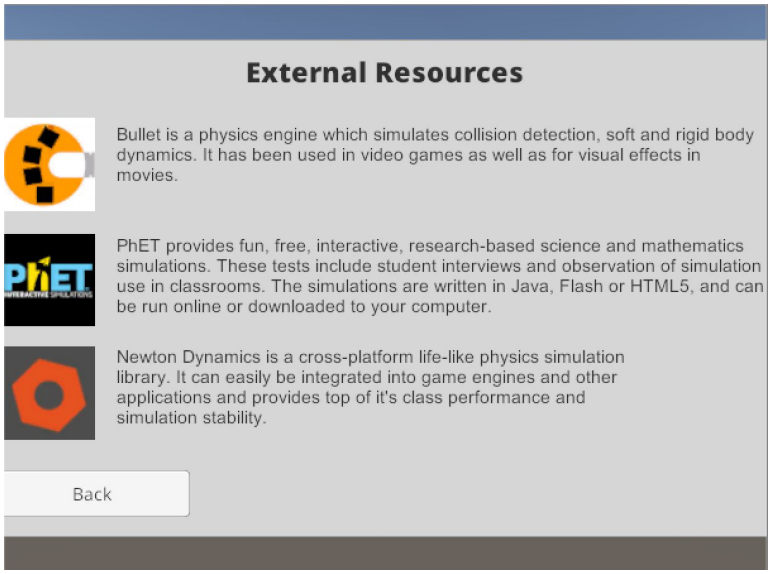
Main Menu Choices

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.



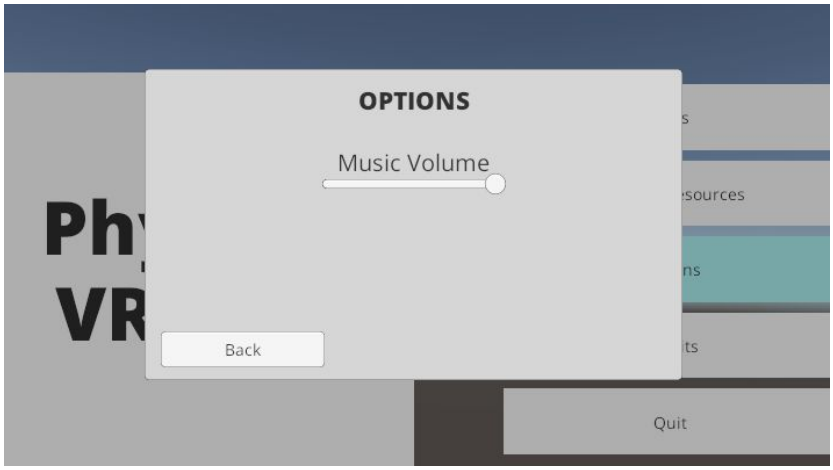
External Resources

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".



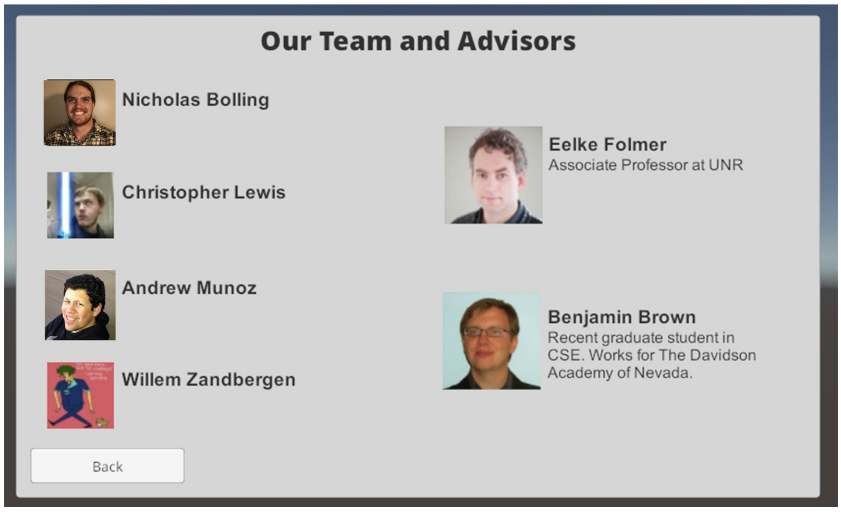
Options

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.



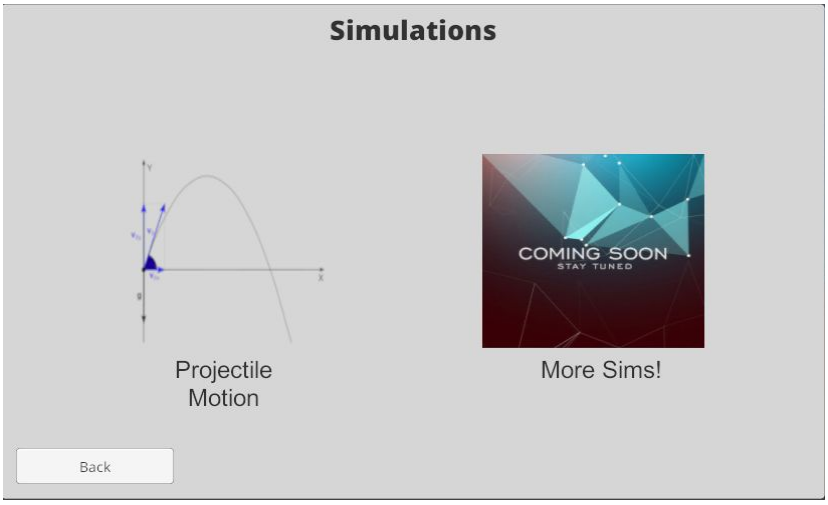
Credits

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”.



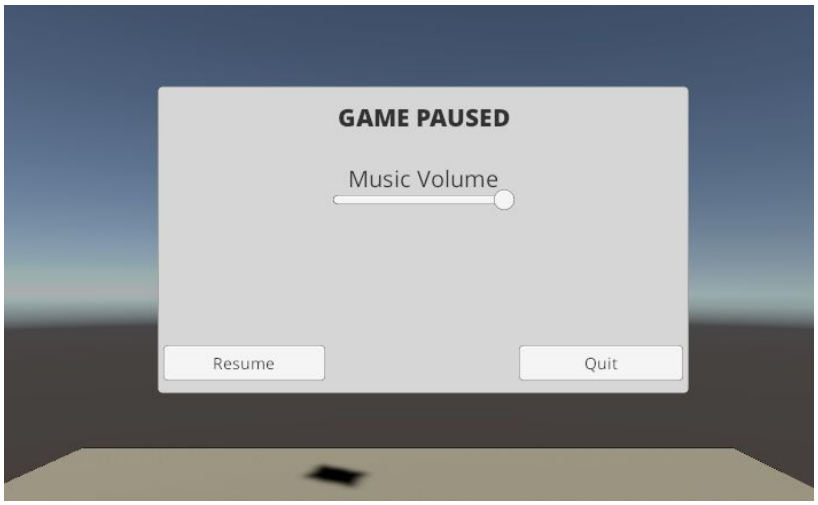
Sims

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.



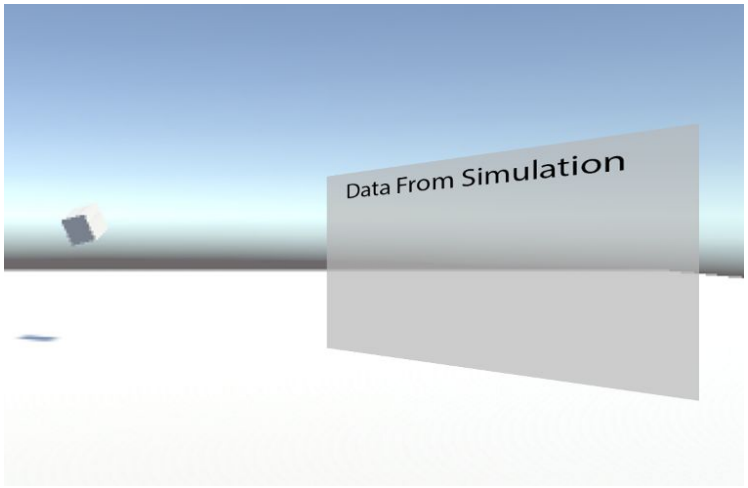
In-game pause

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.



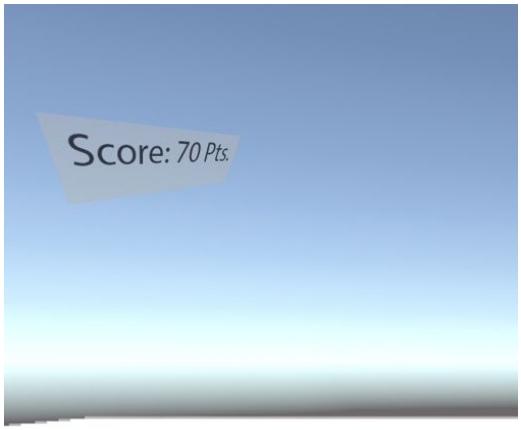
Sims

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.



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.

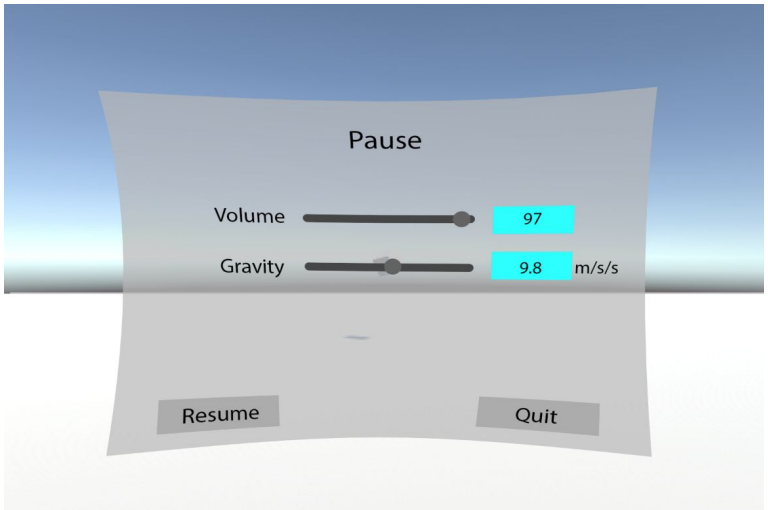


Achievements

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.

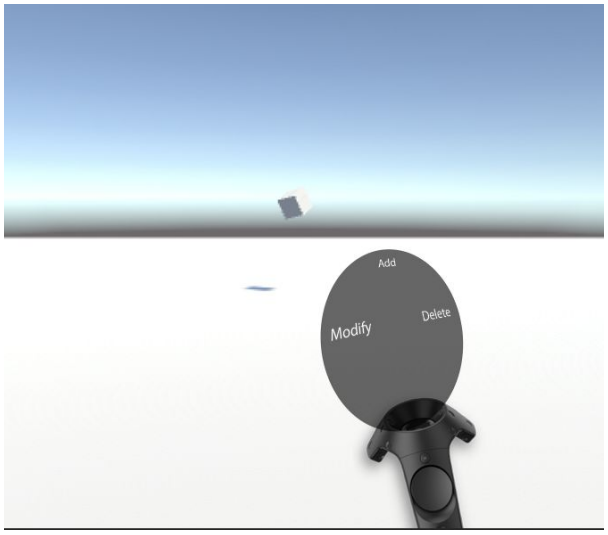
Pause

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.



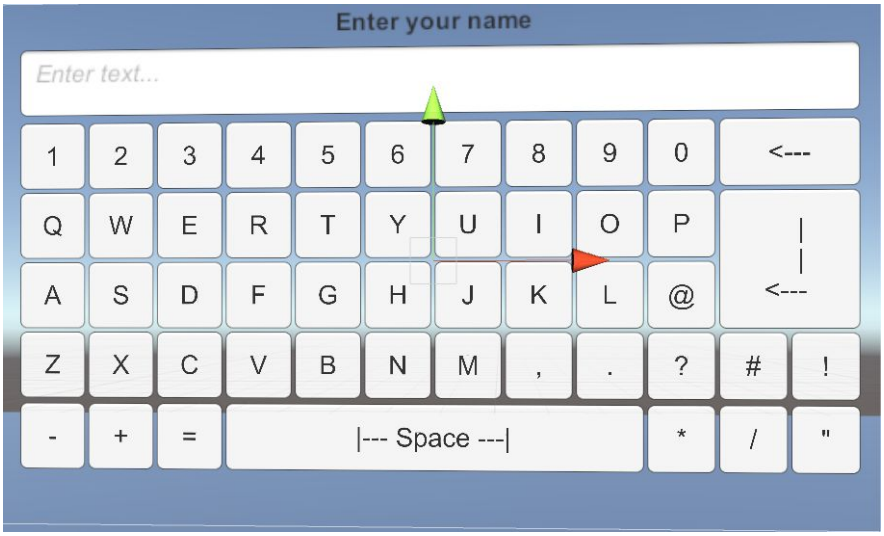
Controller Window

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.



Name Entry

When the user is in the simulation they will have the option to enter in a profile name for themselves, using the controller input. This will save with their saved game and then loaded profile.



Updated Glossary of Terms

- Augmented Reality - The natural world is overlaid with a layer of digital content, typically through glass.
- Avatar (Actor) - The Virtual representation of the player/controller in the VR environment.
- Controller - An interface device similar to a mouse or keyboard that allows for buttons, typically joysticks, and toggles to interface with the computer.
- Cyberspace - A computer synthesized reality. Often a computer synthesized 3-D space. See also: virtual reality.
- 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.
- Field of View - The angle in degrees of the visual field displayed to the user.
- Game Engine - The underlying framework and software behind a computer game or video game.
- Haptics - Physical feedback from interacting with an object, usually integrated into the controller
- 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.
- Head Tracking - Monitoring the position and orientation of the head through specialized devices called Head Trackers.
- Heads-up Display - A display device or interface that allows for graphics to be superimposed on the users view of the world.
- Immersion - The perception of being physically present in a virtual or non-present world.
- Kinethesis - Sensations derived from muscles, tendons and joints and stimulated by movement and tension.
- Latency - A delay between user input and system response, typically user motion and tracker system response in Virtual Reality Environments. Delayed response time.
- Position Sensor - A tracking device that provides information about its location and/or orientation.
- 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.
- 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.
- Spatial Navigation - Self-orientation and locomotion in virtual worlds.
- Tactile Feedback - Simulates touch and feel of objects in virtual reality.
- 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.
- Virtual Reality Environment - The whole space inside the Virtual Reality, essentially the “world” in which you are in while using Virtual Reality.

Engineerings Standards & Technologies

#	Name	Description	Use in Project
1	C# Coding Conventions	C# Programming Guide that is provided by Microsoft	Our project is C# based project so knowing the common core standards of C# is vital.
2	Unity Physics Best Practice Guide	A unity guide for the best practices when using physics in a game.	Our project is strongly tied with physics because we are developing a physics game.
3	VR Reality Design Standards and Best Practices	A guide developed by Oculus on how to design a VR game and the best choices when developing for VR.	Our physics game is in VR so to have knowledge of what is good practice is well needed.
4	Unity Programming Standards	Coding Standard for Unity	Our project uses Unity and to understand programming in Unity and the standards for the engine will be essential. Knowing what Unity says is best will pave a nice path for the project since it is built in Unity.
5	Java Programming Standards	Java Programming Guide that is provided by Oracle	Although our project is not being made using Java, our project does use a very similar language called C#. Since both languages are very similar is best to understand the best practices behind Java.
6	Handbook of Human Factors and Ergonomics (Chapters 42-50)[1]	Section of the handbook providing a comprehensive look at Human-Computer Interaction (HCI).	The physics game highly relies on HCI. This portion of the handbook will be beneficial since it provides insight on HCI

[1]

			and some studies on it.
7	CryEngine C# Programming Standard	Coding Standard for CryEngine.	Since our project is using C# looking at CryEngine standards is helpful. The insight provided from CryEngine will be useful since they go into full detail the parts of C# programming. Our scripting contains C# code.
8	ACM Engineering Code of Ethics	ACM's full software engineering code of ethics (COE) and professional practice.	The COE and professional practice guide is essential for our project. This is essential because we are developing software and if we do not abide by the the COE and professional practice we are being unprofessional software engineers.

Updated List of References

- **<u>"Problem-domain" book</u>**
  - **Virtual Reality**

    by Steven M. LaValle, University of Illinois
    - This book goes over many aspects of Virtual reality and explains the many interactions between our bodies and the VR simulation. It has both anatomy, general science, and programming advice/suggestions that help guide our program.
  - **The VR Book, Human-Centered Design for Virtual Reality.**

    by Jason Jerarld, Ph.D.
    - This book provides an in-depth look at the human element of virtual reality and the main principles of creating the best VR experiences rather than just a technical implementation.
  - **Unity Virtual Reality Projects**

    by Jonathan Linowes
    - This book helps readers to build their own VR games or applications. It provides step by step instructions for creating virtual reality environments and games.
  - **Learning Virtual Reality:**
    Developing Immersive Experiences and Applications for Desktop, Web, and Mobile

    By Tony Parisi
    - This book helps you to get a better understanding of UI design, 3D graphics and Unity 3D. It also provides valuable information for building apps for Oculus or Samsung Gear VR, as well as browser based applications using WebVR and WebGL.

- **Project reference articles**
  - **The NICE project: Narrative, Immersive, Constructionist/Collaborative Environments for Learning in Virtual Reality**
    https://www.evl.uic.edu/tile/NICE/NICE/PAPERS/EDMEDIA/edmedia.paper.html
    - This paper describes and discusses the NICE project, an immersive learning environment for children implemented in the CAVE and related multi-user virtual reality technologies. The NICE project provides an engaging setting where children construct and cultivate simple virtual ecosystems, collaborate via networks with other remotely-located children, and create stories from their interactions in the real and virtual world.

  - **Learning in Virtual Reality** https://eric.ed.gov/?id=ED359950
    - The essence of the computer revolution is yet to come, for computers are essentially generators of realities. Virtual reality (VR) is the next step in the evolutionary path; the user is placed inside the image and becomes a participant within the computational space.
  - **A Conceptual Basis for Educational Applications of Virtual Reality**
    http://www.hitl.washington.edu/research/education/winn/winn-paper.html~
    - This paper discusses the potential value of VR to education. It does so in the light of research conducted at the Human Interface Technology Laboratory at the University of Washington and on the basis of recent developments in cognitive theory that are relevant to human learning.The case is made that immersive VR offers very different kinds of experience than those students normally encounter in school. The psychological processes that become active in immersive VR are very similar to the psychological processes that operate when people construct knowledge through interaction with objects and events in the real world.
  - **Virtual Reality Simulations in Physics Education :**
    http://imej.wfu.edu/articles/2001/2/02/index.asp?referer=www.clickfind.com.au
    - A virtual reality physics simulation (VRPS) is an educational tool using a virtual reality interface that brings together a 3D model of real apparatus and a virtual visualization of physical situations in an interactive manner. VRPS enhances students' understanding by providing a degree of reality unattainable in a traditional two-dimensional interface, creating a sensory-rich interactive learning environment. In this paper, we present a computer-based virtual reality simulation that helps students to learn physics concepts such as wave propagation, ray optics, relative velocity, electric machines, etc. at the level of high school or college physics.

- **Project related websites with useful resources**
  - **Lynda.com** - AMAZING resource that we already pay for in our tuition, and we as a university have a subscription to. Has a ton of Unity related courses, and even courses in general programming (C, C++, C#, ect) that will be really useful.
  - **Unity 3d Asset Store** (https://www.assetstore.unity3d.com/en/) - Easily the most helpful website other than Lynda, here we can purchase already made and professionally (and community) developed assets like objects, models, scripts, etc. We are always on the lookout for modules that can be added and modified to help ease our scripting.

Contributions of Team Members

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

Working hours:

Andrew: 3.5 hrs. + 10 hrs (Missed hours from before)

Creating Models and Applying textures, Recent Project Changes, Updated Specification

Chris: 7 hrs. + 10 hrs (Missed hours from before)

Testing new teleportation methods, Medium and High Level Design,Engineering Standards and Technologies, Appendix A

Nick: 2.3 hrs.

General Editing/rewording/formatting, Ensuring proper VR testing load, Updated references, Updated Hardware Design components and diagram, Glossary of Terms, Updated Summary of changes, UI description

Will: 1 hrs.

Cover page, Table of Contents, Abstract, and Contributions of Team Members

Appendix A
Class Diagram:

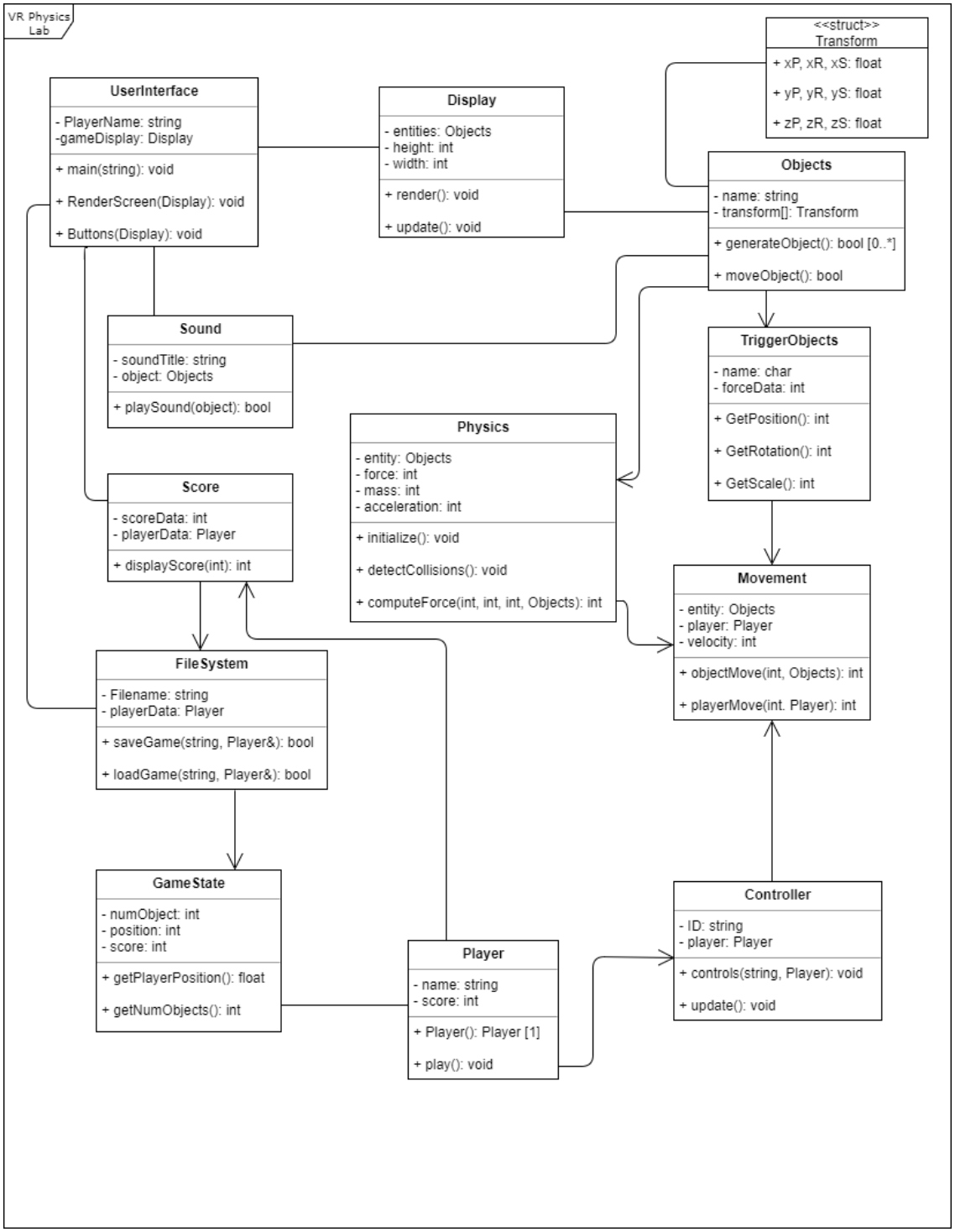


Fig A: The figure above details the classes within the VR Physics Lab and how they relate and associate with one another.

Cited Materials

1) Salvendy, Gavriel. “Handbook of Human Factors and Ergonomics” Pages 1179-1406. Accessed Feb 21, 2017. Web. https://goo.gl/ZSZfsV