Department of Computer Science and Engineering  
The University of Texas at Arlington



Architecture Design Specification

Project: Virtual Reality Xplorer

Team Members:   
*Osuvaldo Ramos*

*Joseph Onwuchekwa*

*Sukuya Nakhaima*

*Chris Otterbine*

Last Updated: 12/7/2013 2:36 PM

# Table of Contents

[Table of Contents 2](#_Toc373953740)

[Document Revision History 4](#_Toc373953741)

[List of Figures 5](#_Toc373953742)

[List of Tables 6](#_Toc373953743)

[1. Introduction 7](#_Toc373953744)

[1.1 Product Concept 7](#_Toc373953745)

[1.2 Product Scope 7](#_Toc373953746)

[1.3 Key Requirements 8](#_Toc373953747)

[2. Meta Architecture 10](#_Toc373953748)

[2.1 Architectural Vision 10](#_Toc373953749)

[2.2 Guiding Principles 10](#_Toc373953750)

[2.3 Assumptions 11](#_Toc373953751)

[3. Architecture Overview 12](#_Toc373953752)

[3.1 Input Layer 12](#_Toc373953753)

[3.2 Processing Layer 12](#_Toc373953754)

[3.3 Storage Layer 13](#_Toc373953755)

[3.4 Output Layer 13](#_Toc373953756)

[3.5 Detailed Architecture 13](#_Toc373953757)

[4. Input Layer 14](#_Toc373953758)

[4.1 Description 14](#_Toc373953759)

[4.2 Purpose 14](#_Toc373953760)

[4.3 Function 14](#_Toc373953761)

[4.4 Dependencies 14](#_Toc373953762)

[4.5 Data 15](#_Toc373953763)

[4.6 Subsystems 15](#_Toc373953764)

[5. Processing Layer 17](#_Toc373953765)

[5.1 Description 17](#_Toc373953766)

[5.2 Purpose 17](#_Toc373953767)

[5.3 Function 18](#_Toc373953768)

[5.4 Dependencies 18](#_Toc373953769)

[5.5 Data 18](#_Toc373953770)

[5.6 Subsystems 18](#_Toc373953771)

[6. Output Layer 23](#_Toc373953772)

[6.1 Description 23](#_Toc373953773)

[6.2 Purpose 23](#_Toc373953774)

[6.3 Function 23](#_Toc373953775)

[6.4 Dependencies 23](#_Toc373953776)

[6.5 Data 24](#_Toc373953777)

[6.6 Subsystems 24](#_Toc373953778)

[7. Storage Layer 26](#_Toc373953779)

[7.1 Description 26](#_Toc373953780)

[7.2 Purpose 26](#_Toc373953781)

[7.3 Function 26](#_Toc373953782)

[7.4 Dependencies 27](#_Toc373953783)

[7.5 Data 27](#_Toc373953784)

[7.6 Subsystems 27](#_Toc373953785)

[8. Requirements Mapping 30](#_Toc373953786)

[9. Relationship Mapping 31](#_Toc373953787)

[9.1 Section Overview 31](#_Toc373953788)

[9.2 Data Flow Definition 32](#_Toc373953789)

[9.3 Producer-Consumer Relationship Matrix 33](#_Toc373953790)

[10. Testing Consideration 34](#_Toc373953791)

[10.1 Overview 34](#_Toc373953792)

[10.2 Input Layer 34](#_Toc373953793)

[10.3 Processing Layer 34](#_Toc373953794)

[10.4 Storage Layer 34](#_Toc373953795)

[10.5 Output Layer 34](#_Toc373953796)

[11. Operating System Dependencies 35](#_Toc373953797)

[11.1 Input Layer 35](#_Toc373953798)

[11.2 Processing Layer 35](#_Toc373953799)

[11.3 Storage Layer 35](#_Toc373953800)

[11.4 Output Layer 35](#_Toc373953801)

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Revision Date | Description | Rationale |
| 0.1 | 12/4/13 | Initial Integration |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# List of Figures

Figure # Title Page #

3-1 High Level Architecture 13

3-2 Detailed Architecture 14

4-1 Input Layer 15

5-1 Processing Layer 18

6-1 Output Layer 24

7-1 Storage Layer 27

9-1 Relationship Mapping 32

# List of Tables

Table # Title Page #

1-1 Key Requirements 9

5-1 Physics Interlayer Interface 20

7-1 Project File Interlayer Interface 28

7-2 Save File Interlayer Interface 29

7-3 3D Assets Cache Interlayer Interface 29

7-4 Audio Assets Cache Interlayer Interface 30

7-5 Game Scripts Interlayer Interface 30

8-1 Requirements Mapping 31

9-1 Data Flow Definition 33

9-2 Producer-Consumer Relationship Matrix 34

# 1. Introduction

## Product Concept

The Virtual Reality Xplorer is an educational video game that will employ the Oculus Rift virtual reality device and an Xbox controller to immerse students in a virtual environment where they can learn and explore different topics from their curriculum. Students will be presented different topics and learn how to apply the knowledge they are learning in the virtual sandbox environment. Teachers will be able to see how each student performed in the different virtual environments.

The Virtual Reality Xplorer is a product designed to simulate an environment while simultaneously providing the user an entertaining and educational experience. The Virtual Reality Xplorer will be stored on a PC. The program will be launched from the operating system and display the main menu. Once the user starts a new game, they will be allowed to explore an open environment while the Virtual Reality Xplorer displays information using the heads up display. The user will also encounter intermittent puzzles or challenges.

The Virtual Reality Xplorer’s intended users will be 5th and 6th grade science students. The intended consumer will be 5th and 6th grade science teachers that want an alternative method of teaching a certain topic. Other audiences may consist of school districts or educational programs.

## Product Scope

The Virtual Reality XPlorer sets out to provide an educational and entertaining experience through virtual reality as to allow the user to gain more “sensory” knowledge.

The hardware component of this system accommodates the Oculus Rift, a piece of hardware needed for virtual reality, an Xbox 360 controller for lateral movement and a wireless headset for user freedom.

The user interface will feature a menu for selecting options such as new, save, quit, etc. When the user is given the opportunity to interact with the environment and objects, the software switches to a HUD (Heads-Up Display) to relay informative feedback to the user.

This product is designed for 5th and 6th graders, although the system is informative to all ages.

## Key Requirements

Table 1-1 Key Requirements

|  |  |  |
| --- | --- | --- |
| Number | Requirement | Description |
| 3.1 | The student will learn about matter | The student will learn about states of matter, properties of matter, boiling and freezing points of water, and mixtures by solving puzzles based on these topics. |
| 3.2 | The student will learn about force, motion, and energy | The student will learn about the uses of energy, electrical circuits and how they produce energy, and the effect of force on an object by solving puzzles based on these topics. |
| 3.3 | The student will learn about organisms and environments | The student will learn about the way organisms live and survive, food chains, ecosystem changes, carbon dioxide-oxygen cycle, adaptions, inherited traits and learned behaviors by solving puzzles based on these topics. |
| 3.4 | The student will learn about light reflection and refraction | The student will learn about light traveling in a straight line, light reflection off of objects, and refraction of objects by solving puzzles based on these topics. |
| 3.5 | Generate progress file | A teacher must be able to track the progress of his or her students and identify problem areas for each student. The information in the progress file will include how many attempts it took a user to solve a puzzle and how long it took them to solve a puzzle. |
| 3.9 | User must be able to configure controls | The user must have the option to setup how the input devices (Oculus Rift and Xbox controller) control the user’s character in the game. |
| 4.1 | Oculus Rift | The system must be packaged with an Oculus Rift itself. |
| 4.4 | Game Controller | The game controller must be a standard Xbox 360 controller. |
| 5.1 | Frame Rate | The Virtual Reality Xplorer provides smooth gameplay and animations as to not disrupt the user’s immersion. |
| 5.4 | Responsiveness | The Xplorer must implement proper responsive checks so as to not disorient the user too quickly. Ideal response time is 1/60 seconds. |
| 8.1 | Cross-Platform Compatibility | The Virtual Reality Xplorer must work with Windows and Mac OS. |
| 8.6 | Multiplayer | The Virtual Reality Xplorer must allow multiple users to interact in the same environment in real-time. |
| 8.9 | Online Patching | Bug fixes and patches must be delivered through an online update mechanism. |
| 8.10 | Downloadable Content | The user must be able to download additional content such as additional environments online. |

# ****2. Meta Architecture****

The Meta Architecture Section describes the guiding principles and key assumption that team VR-X will use when developing the Virtual Reality Xplorer. Five guiding principles will drive team VR-X when making future architecture design decisions. The Guiding Principles section outlines and describes each of the principles. The Assumptions section lists and key assumptions made by team VR-X that will affect the architecture design of the Virtual Reality Xplorer.

## 2.1 Architectural Vision

The architectural design for the Virtual Reality Xplorer consists of four layers. The layers are an input layer, processing layer, storage layer, and a control layer. Input will be received from the Oculus Rift and Xbox controller. The processing layer handles the processing of the inputs and renders the scene in the Virtual Reality Xplorer accordingly. The game engine will be located in the processing layer. The storage layer will store files such as the save file and the student progress file. The output layer will handle sending the audio and visuals to the correct devices.

## 2.2 Guiding Principles

2.2.1 Reliability

Since the Virtual Reality Xplorer is a video game, reliability is an important principle in a sense that the Virtual Reality Xplorer has no delay or latency when a user presses a button on the Xbox controller and expects an instant response on screen. This means the architecture design needs to be as thin as possible to prevent data from being sent through a large amount of layers.

2.2.2 Portability

The Virtual Reality Xplorer will be portable between computers that use different hardware. The Virtual Reality Xplorer should work independently of the computer hardware. To achieve this, the architecture design must not contain any data flows that communicate with hardware directly. Modules will be abstracted to communicate with the drivers for sound and graphics as well as the Oculus SDK and the Xbox controller driver.

2.2.3 Modularity

The Virtual Reality Xplorer will allow additional content such as game stages. To ease integration, the architecture must have as little points of contact as possible. There should be low coupling between the files for game stages and the modules that process them.

2.2.4 Ease of Use

The Virtual Reality Xplorer must be intuitive and easy to use. A user that has little to no knowledge of video games should still be able to understand how to use the Virtual Reality Xplorer. This is especially important since the target audience users are 5th grade students.

## 2.3 Assumptions

2.3.1 Game Engine

Team VR-X will use an existing game engine to assist in the development of the Virtual Reality Xplorer. The choice has been narrowed down to two game engines, Unity 3D and Unreal Engine. Both engines support the Oculus Rift as well as the Xbox controller.

2.3.2 Device Drivers

Drivers for devices such as graphics and sound cards will already exist. The Virtual Reality Xplorer will just need to interface with the drivers.

2.3.3 Operating System

The primary operating system for the Virtual Reality Xplorer will be Windows 7 minimum.

2.3.4 3D and Sound Assets

The game engine will be able to accept multiple file formats for assets regardless of the editor used to create them.

# 3. Architecture Overview

The architecture for the Virtual Reality Xplorer will consist of four layers. The layers are the input layer, processing layer, storage layer, and output layer. The architecture was developed while following the guiding principles.

Figure 3-1 High Level Architecture



## 3.1 Input Layer

The responsibility of the input layer is to gather the input data that comes from the external devices the Oculus Rift and Xbox controller. The Oculus SDK will poll the head movements from the Oculus rift and translate the movements to data the processing layer will recognize. The Xbox controller driver will listen to button events that come from the Xbox controller and translate the events to data the processing layer will recognize.

## 3.2 Processing Layer

The processing layer will be responsible for processing what the game will be doing. The game engine will reside in the processing layer. The layer will receive data from the input layer and calculate your position in the 3D environment, if you are interacting with an object, and if you are colliding with objects. The processing layer will continue to then render the frame and sound at that instant and gather the image and audio and synchronize them. The processing layer will also generate the statistics for the progress file and the state for the save file.

## 3.3 Storage Layer

The storage layer contains all the files that will be included with the Virtual Reality Xplorer. This is where the progress files and save files will be located. The source game scripts, 3D assets, and audio assets will also be stored here. This layer will communicate with the processing layer to render the textures of objects and sound in the virtual environment.

## 3.4 Output Layer

The output layer is responsible for outputting the images and sounds to the correct devices. The layer will receive data from the processing layer and utilize the device drivers for the Oculus Rift, graphics card, and sound card to output the sound and graphics.

## 3.5 Detailed Architecture

Figure 3-2 Detailed Architecture



# 4. Input Layer

Figure 4-1 Input Layer



## 4.1 Description

The Input Layer provides a platform to gather input from the user through the Oculus Rift and the Xbox controller, and transport this input to their respective subsystem. The subsystems then send this input to the input controller in the processing layer for further processing.

## 4.2 Purpose

The purpose of the Input layer is to capture head motion events from the Oculus Rift and keystroke events from the Xbox controller, and then transport this event to the Input controller in the processing layer.

## 4.3 Function

The Input layer is solely responsible for retrieving input from the user and providing the hardware interface to the VR-X system for the user. It is also the layer were all drivers to the input devices (Oculus Rift and Xbox controller) will be housed and connected to the system.

## 4.4 Dependencies

The Input layer is dependent on the input devices such as Oculus Rift and the Xbox controller. It is also dependent on the user’s input and the computer where the software is being run, but it is not dependent on any other layer.

## 4.5 Data

The data in this layer comprises of the keystrokes, head motion events which are rotational angles (x, z, y axis) and button-clicked events. The events will be captured by event handlers (magnetic sensors) that will translate this input to a format readable by the input controller in the processing layer.

## 4.6 Subsystems

4.6.1 Oculus SDK

#### **4.6.1.1 Description**

This subsystem is responsible for acquiring any user input from the Oculus Rift. These inputs are rotational angles about the x, y, z coordinates, which are generated from head motion events.

**4.6.1.2 Function**

This subsystem is responsible for acquiring any user input from the Oculus Rift. These inputs are rotational angles about the x, y, z coordinates, which are generated from head motion events.

**4.6.1.3 Interlayer Interface(s)**

None

**4.6.1.4 Public Interface(s)**

None

4.6.2 Save File

#### **4.6.2.1 Description**

This subsystem is responsible for retrieving user input from the Xbox controller. These inputs are keystrokes and button pressed events which will be translated by this subsystem to a format readable by the input controller in the processing layer.

**4.6.2.2 Function**

Since the Xbox controller is the Human Interface Device (HID) which provides the user the ability to navigate throughout the system and interact with the objects in the game, the Xbox controller driver makes sure that the user’s inputs(button pressed events and keystrokes) are being supplied to the input controller in real time (milliseconds) to improve user experience.

**4.6.2.3 Interlayer Interface(s)**

None

**4.6.2.4 Public Interface(s)**

None

# 5. Processing Layer

Figure 5-1 Processing Layer



## 5.1 Description

This Layer includes the game engine which is responsible for performing most of the processing of data throughout the entire system. Some of the processing includes the rendering of graphics for adequate visual display, the loading and saving of games dynamically, and will be the single point of contact with the Storage layer (database). There are a total of seven subsystems in this layer; each subsystem is located in the game engine except the State controller which retrieves information from the Load/Save controller in order to save the user’s progress to the database.

## 5.2 Purpose

The purpose of the Processing layer is to handle any heavy-weight processing of data throughout the game play. This layer could be perceived as the core of the system because it is located at the median point of the system, relating the Input Layer to the Output layer. It also contains the game engine which is the mechanism that supplies the virtual real-time interaction in the Virtual Reality Xplorer.

## 5.3 Function

The Processing layer is responsible for retrieving input from the Input layer, interacting with the database to store and retrieve data, render image and sound for quality gaming experience. This layer also ensures adequate syncing of images and sound in real time.

## 5.4 Dependencies

The Processing layer is dependent on the input layer in order to acquire the input needed since the processing layer never interacts with the hardware. It’s also dependent on the Storage layer because it retrieves data and saves data to the Storage layer.

## 5.5 Data

The data retrieved and processed in this layer consist of input from the Oculus SDK and the Xbox Controller Driver. Other data in this layer consist of generated statistics from the State controller, images (including graphic models) and sound from the Storage layer, and the current state of the game at specified instance.

## 5.6 Subsystems

5.6.1 Input Controller

#### **5.6.1.1 Description**

This subsystem retrieves input from the Oculus SDK and the Xbox Controller Driver in the Input layer and delegates the appropriate subsystem to process those inputs.

**5.6.1.2 Function**

The input controller manages the data retrieved from the Input layer in order to distribute it to the appropriate subsystem to be processed.

**5.6.1.3 Interlayer Interface(s)**

None

**5.6.1.4 Public Interface(s)**

None

5.6.2 Physics

#### **5.6.2.1 Description**

This subsystem is responsible for ensuring a virtual realistic real-time environment for the system. This subsystem is constantly controlling the objects in the game which is collected from the Game Scripts in the Storage layer. The physics layer also mitigates the appropriate data to the following subsystem respectively. These subsystems are Graphics Processor and Sound Processor which are subsystem of the Processing layer and not a subsystem inter-layer of the Physics subsystem.

**5.6.2.2 Function**

The Physics module calculates the position of the player (user), and objects in the virtual environment during game play. This subsystem also handles all interactions to objects by the user or other objects, which comprises of detecting collision (when two objects collide), simulation of rigid; fluid and soft body dynamics. Functions such as oncollision() are called to perform a specific action based on the collision. The Physics module is also vital during a game play, it is responsible for ensuring corresponding behaviors of different objects like making an object slide on an ice surface, on prevent objects from falling through solid surface. It is also task with ensuring real time response after an object interaction in order to prevent deteriorating frame rate.

**5.6.2.3 Interlayer Interface(s)**

Table 5-1 Physics Interlayer Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| Linecast | Returns true if there is any collider intersecting the line between start and end | Vectors | Bool |
| OnCollisionEnter | Event | Performs a specified action after collision occur | None |

**5.6.2.4 Public Interface(s)**

None

5.6.3 Graphics Processor

#### **5.6.3.1 Description**

The Graphic processor is responsible for using the game engine’s memory to create high quality image within a short latency. Basic shapes such as polygons, lines, circles and rectangles are mapped to more sophisticated objects in the 3D environment to create good quality models.

**5.6.3.2 Function**

The key purpose of the Graphic processor is to render high quality image within a short latency. The Graphic processor retrieves these 3D models from the 3D Assets in the Storage layer.

**5.6.3.3 Interlayer Interface(s)**

None

**5.6.3.4 Public Interface(s)**

None

5.6.4 Sound Processor

#### **5.6.4.1 Description**

The Sound processor is crucial for producing quality audio to match the game environment.

**5.6.4.2 Function**

This subsystem ensures that the accurate sound is mapped to the appropriate object during an event in the game play. The Sound processor is also responsible for supply adequate audio during dialogs and scenery in the game. This subsystem acquires the audio files from the Audio Assets subsystem in the Storage layer. This subsystem is also tasked with modifying sounds, adding echoes and pitch modulation to give the user a Virtual realistic environment.

**5.6.4.3 Interlayer Interface(s)**

None

**5.6.4.4 Public Interface(s)**

None

5.6.5 Load/Save Controller

#### **5.6.5.1 Description**

The purpose of the Load/Save controller is to properly obtain and save files to the Storage layer. The files will include states of particular game stages, cinematic, and game scripts.

**5.6.5.2 Function**

The Load/Save controller retrieves saved files and loads them to the game engine for processing. It transfers some of the data to the output controller in order for the user to view the saved content.

**5.6.5.3 Interlayer Interface(s)**

None

**5.6.5.4 Public Interface(s)**

None

5.6.6 State Controller

#### **5.6.6.1 Description**

This subsystem is responsible for generating statistics from the Virtual Reality Xplorer. This information will then be sent to the Progress File in the storage layer.

**5.6.6.2 Function**

The State Controller is responsible for getting information about how many times a user has attempted a puzzle, how long the user has been on a certain stage, etc. After the statistics have been generated they will be sent to the Progress File where they will be written in a readable format.

**5.6.6.3 Interlayer Interface(s)**

None

**5.6.6.4 Public Interface(s)**

None

5.6.7 Output Controller

#### **5.6.7.1 Description**

This subsystem is responsible for getting the rendered objects of the image and sound and sending them to the output layer.

**5.6.7.2 Function**

The Output Controller will receive the image and sound objects for the Graphics Processor and Sound processor respectively. The Output Controller will proceed to synchronize the image with the sound ensuring that they coincide. Finally, the Output controller will then send the object to the output layer to be sent to the correct devices.

**5.6.7.3 Interlayer Interface(s)**

None

**5.6.7.4 Public Interface(s)**

None

# 6. Output Layer

Figure 6-1 Output Layer



## 6.1 Description

The Output layer takes the sole role of displaying the Output to the user. These outputs are mainly images, videos, game visuals and sound which will be displayed on the output devices mentioned later in the subsystem of this section.

## 6.2 Purpose

The purpose of this layer is to provide the user a visually appealing and immersive feel of the system by displaying the game content on the output devices. Data is retrieved from the output controller in the Processing layer and it is properly dispersed to its appropriate driver in the Output layer, so this Layer ensures that those data are properly propagated to their respective output devices.

## 6.3 Function

This layer is responsible for providing the graphics and sound to the user.

## 6.4 Dependencies

The Output layer is dependent on the processing layer for the visual and sound outputs.

## 6.5 Data

The data in this layer consist of the sound outputs, and the visual outputs.

## 6.6 Subsystems

6.6.1 Graphics Driver

#### **6.6.1.1 Description**

The purpose of the Graphics driver is to properly distribute the visual output to the appropriate output devices which is the monitor and the Oculus Rift. The output device then displays the content to the user.

**6.6.1.2 Function**

Graphics driver is responsible for acquiring the visual data from the output controller in the processing layer, and displaying of the data through their respective output devices.

**6.6.1.3 Interlayer Interface(s)**

None

**6.6.1.4 Public Interface(s)**

None

6.6.2 Oculus SDK

#### **6.6.2.1 Description**

The Oculus SDK is responsible for acquiring the visual output from the Graphics driver and transporting this data to the Oculus Rift to be displayed which will give the user a 3D virtual reality view.

**6.6.2.2 Function**

The function of the Oculus SDK is to appropriately transfer the visual output to the Oculus Rift to be displayed. The subsystem is also task with maintenance of the connection of the Oculus Rift to the system.

**6.6.2.3 Interlayer Interface(s)**

None

**6.6.2.4 Public Interface(s)**

None

6.6.3 Sound Driver

#### **6.6.3.1 Description**

The sound driver provides the appropriate sound output to the headphones simultaneously with the video display on the Oculus Rift. This provides the user dynamic audio during game play.

**6.6.3.2 Function**

The function of the sound driver is to transfer sound output to the headphones to ensure the user receives real-time audio. This subsystem is also responsible for ensuring the headphones are connected to the system throughout the game play to ensure adequate audio output.

**6.6.3.3 Interlayer Interface(s)**

None

**6.6.3.4 Public Interface(s)**

None

# 7. Storage Layer

Figure 7-1 Storage Layer



## 7.1 Description

The Storage layer contains a database that stores the state of the game from the last time the user quit the game as well as all saves that the user has created through the Save feature.

## 7.2 Purpose

The Storage layer stores all relevant data in the database.

## 7.3 Function

When the user saves a game, a file is created in the database that allows for retrieval when the user wants to resume. When the user exits the virtual reality simulation and returns to the main menu, the game creates an “autosave” in which the Storage layer stores and holds the state of the game by default, allowing the user to resume from where the individual left off. The Storage Layer is also responsible for holding all game assets and code that allows the game to work at a fundamental level.

## 7.4 Dependencies

The Storage layer depends on the Processing Controller for sending/receiving data.

## 7.5 Data

Data processed will be saved games and resume states. The Processing Layer will also receive data from the subsystems Game Scripts and 3D Assets.

## 7.6 Subsystems

7.6.1 Progress File

#### **7.6.1.1 Description**

The Progress File Controller will act as a “state holder” that allows a user to resume the game from where the individual left off via the “Resume” option on the main menu.

**7.6.1.2 Function**

The Program Controller receives and holds the state of the game via the State Controller.

**7.6.1.3 Interlayer Interface(s)**

Table 7-1 Progress File Interlayer Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| getState | Gets the state of the game. | Data from the State Controller. | None. |

**7.6.1.4 Public Interface(s)**

None

7.6.2 Save File

#### **7.6.2.1 Description**

Stores the state of the game from using the “Save” feature.

**7.6.2.2 Function**

The Save controller gets the save file from the Load/Save controller and stores the file in the database allowing the user to retrieve and load the file via the “Load” feature.

**7.6.2.3 Interlayer Interface(s)**

Table 7-2 Save File Interlayer Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| getFile | Gets the save file from the game. | Data from the Load/Save Controller. | None. |
| loadFile | Loads the file. | A save file already stored. | None. |

**7.6.2.4 Public Interface(s)**

None

7.6.3 3D Assets Cache

#### **7.6.3.1 Description**

A cache that holds the data representing 3D models and environments.

**7.6.3.2 Function**

The 3D Assets Cache sends data to the graphics processor in order to process/render them.

**7.6.3.3 Interlayer Interface(s)**

Table 7-3 3D Assets Cache Interlayer Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| send3DAsset | Sends data to the Graphics Processor. | None. | None. |

**7.6.3.4 Public Interface(s)**

None

7.6.4 Audio Assets Cache

#### **7.6.4.1 Description**

A cache that holds all audio files that will be used in the game.

**7.6.4.2 Function**

The Audio Assets Cache sends audio files to be processed by the Sound Processor in order for the game to produce sound.

**7.6.4.3 Interlayer Interface(s)**

Table 7-4 Audio Assets Cache Interlayer Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| sendAudioFile | Sends audio data to Sound Processor. | None. | None. |

**7.6.4.4 Public Interface(s)**

None

7.6.5 Game Scripts

#### **7.6.5.1 Description**

A cache that stores all source code for the game as well as the game engine (Unreal Engine, Unity, etc).

**7.6.5.2 Function**

The Game Scripts subsystem sends scripts containing source code to the Physics subsystem in order to render into the 3D environment.

**7.6.5.3 Interlayer Interface(s)**

Table 7-5 Game Scripts Interlayer Interface

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Description | Information Required | Information Returned |
| sendScripts | Sends scripts to the Physics subsystem. | None. | None. |

**7.6.5.4 Public Interface(s)**

None

# 8. Requirements Mapping

Table 8-1 Requirements Mapping

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Requirement | Input Layer | Processing Layer | Storage Layer | Output Layer |
| 3.1 | The student will learn about matter |  | X | X |  |
| 3.2 | The student will learn about force, motion, and energy |  | X | X |  |
| 3.3 | The student will learn about organisms and environments |  | X | X |  |
| 3.4 | The student will learn about light reflection and refraction |  | X | X |  |
| 3.5 | Generate progress file |  | X | X | X |
| 3.9 | User must be able to configure controls | X | X | X |  |
| 4.1 | Oculus Rift | X |  |  | X |
| 4.4 | Game Controller | X |  |  | X |
| 5.1 | Frame Rate |  | X |  | X |
| 5.4 | Responsiveness |  | X |  | X |
| 8.1 | Cross-Platform Compatibility |  | X |  |  |
| 8.6 | Multiplayer |  | X |  |  |
| 8.9 | Online Patching |  | X |  |  |
| 8.10 | Downloadable Content |  | X |  |  |

# 9. Relationship Mapping

Figure 9-1 Relationship Mapping



## 9.1 Section Overview

This section goes into detail on the data flows throughout the architectural design. Sections coded orange represent the Input layer, sections coded green represent the Processing layer, sections coded blue represent the storage layer, and sections coded dark gray represent the Output layer.

## 9.2 Data Flow Definition

* Note 1: S12 denotes the flow from the Save File to the Load/Save Controller. P12 denotes the flow from the Load/Save Controller to the Save File.
* Note 2: Flow 6 is an alternate (if/else) flow-path and needs to be taken before entering the Physics subsystem.

Table 9-1 Data Flow Definition

|  |  |  |
| --- | --- | --- |
|  | Data Flow | Data Flow Descriptions |
| Input Layer | 1 | Oculus Rift input into computer. |
| 2 | Xbox controller input into computer. |
| 3 | Oculus SDK being processed by Input Controller. |
| 4 | Xbox controller driver processed by Input Controller. |
| Storage Layer | 7 | Translates source code for the Physics subsystem to render. |
| 10 | 3D assets processed by the Graphics Processor subsystem to render. |
| 11 | Audio files/assets sent to Sound Processor system for processing. |
| S12 | Allows access to save files via command from the Load/State Controller. |
| Processing Layer | 5 | Inputs from the Controller can be translated to the simulation via the Physics subsystem. Flow 6 needs to be taken before entering the Physics subsystem. |
| 6 | Xbox input selects a file from the Load/Save controller. |
| 8 | 3D rendering of the Physics subsystem is sent to the Graphics Processor subsystem. |
| 9 | Audio is rendered to the Sound Processing subsystem. |
| P12 | Load/Save processor gets a save file from the Save File subsystem. |
| 13 | Graphics Processor subsystem sends data to the Output Processor to prepare it for output. |
| 14 | Sound Processor subsystem sends data to the Output Processor to prepare it for output. |
| 15 | Loads the game *IF* Path 6 is taken. |
| 16 | Sends data to the Graphics Driver for displaying. |
| 17 | Sends data to the Sound Driver for audio. |
| 18 | State Controller subsystem gets the state of the game from the Progress File subsystem. |
| Output Layer | 19 | Sends the data to the Oculus SDK to prepare for displaying visual. |
| 20 | Outputs display to the Oculus Rift. |
| 21 | Outputs display to a monitor. |
| 22 | Outputs audio to headset. |

## 9.3 Producer-Consumer Relationship Matrix

Table 9-2 Producer-Consumer Relationship Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Consumer** | | | | | | | | | | | | | | | | |
| Oculus SDK (input) | Xbox Cont. Driver | Input Controller | Physics | State Controller | Graphics Processor | Sound Processor | Load/Save Cont. | Output Controller | Progress File | Game Scripts | Audio Assets | Save File | 3D Assets | Sound Driver | Graphics Driver | Oculus SDK (output) |
| **Producer** | Oculus SDK (input) |  |  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Xbox Controller Driver |  |  | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Input Controller |  |  |  | 5 |  |  |  | 6 |  |  |  |  |  |  |  |  |  |
| Physics |  |  |  |  |  | 8 | 9 |  |  |  |  |  |  |  |  |  |  |
| State Controller |  |  |  |  |  |  |  |  |  | 18 |  |  |  |  |  |  |  |
| Graphics Processor |  |  |  |  |  |  |  |  | 13 |  |  |  |  |  |  |  |  |
| Sound Processor |  |  |  |  |  |  |  |  | 14 |  |  |  |  |  |  |  |  |
| Load/Save Controller |  |  |  |  |  |  |  |  | 15 |  |  |  | 12 |  |  |  |  |
| Output Controller |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 17 | 16 |  |
| Progress File |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Game Scripts |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Audio Assets |  |  |  |  |  |  | 11 |  |  |  |  |  |  |  |  |  |  |
| Save File |  |  |  |  |  |  |  | 12 |  |  |  |  |  |  |  |  |  |
| 3D Assets |  |  |  |  |  | 10 |  |  |  |  |  |  |  |  |  |  |  |
| Sound Driver |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Graphics Driver |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 19 |
| Oculus SDK (output) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# 10. Testing Consideration

## 10.1 Overview

This section is dedicated to ensuring that each layer will be validated as established in the overall architecture. Each layer will be checked to ensure each is real-time responsiveness, portable, intuitive, and reliable.

## 10.2 Input Layer

The Input Layer is responsible for managing all input required by the system. The Input Layer is also the layer in which data begins to flow throughout the overall architecture. Validating the Input Layer requires that any device or subsystem of the layer be properly tested to ensure that flow of information begins at these devices/subsystems. Specifically input flow enters the computer.

## 10.3 Processing Layer

The Processing Control Layer controls the flow of information in and out of this layer. To ensure proper validation, three subsystems will need to be tested. First, the Input Controller needs to be tested to properly receive data from the Input Layer. Second, the Output Controller needs proper configuring in order to send data to the Output Layer. Lastly, the State and Load/Save Controllers need to properly send data to the Storage layer.

## 10.4 Storage Layer

Validating the establishment of the Storage Layer first requires that the subsystems “Save File” and “Progress File” be tested. Reliability is an important aspect for testing the subsystems “Save File” and “Programs File”. Further testing will be done to ensure that the Storage Layer can hold the assets needed to create an immersive environment.

## 10.5 Output Layer

The Output Layer will be validated through the Graphics Driver subsystem and the Sound Driver subsystem, particularly the Wireless Headphone device and the Oculus Rift itself through the Graphics Driver. Real-Time Responsiveness, Portability, and Reliability will be checked in this layer.

# 11. Operating System Dependencies

## 11.1 Input Layer

The input layer will receive input from the Oculus Rift and the Xbox controller. Drivers for multiple operating systems exist for the Oculus Rift but this project will focus on the Windows operating system because the Xbox controller works as a plug-n-play device.

## 11.2 Processing Layer

The processing layer will use a game engine such as Unity or Unreal. These engines are cross-platform but team VR-X will be developing on a Windows platform.

## 11.3 Storage Layer

File output for save games and student progress is cross-platform. Graphics and audio resources will be stored in a format that is also cross-platform.

## 11.4 Output Layer

The only output device is the Oculus Rift and audio. Both of these output streams can be processed on any platform.