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

Detailed Design Specification
CSE 4317: Senior Design II
Summer 2023

VR Nursing
Palliative Care VR

Cesar Cantu-Perez
Braddock Bresnahan
Nelson Lam
Carlos Cruz

Revision History

Revision	Date	Author(s)	Description
0.1	6.16.2023	CCP	document creation
0.2	6.21.2023	CCP, BB, NL, CC	complete draft

CONTENTS

1 Introduction **5**

2 System Overview **5**

3 Manager Layer Subsystems **6**
- 3.1 Layer Hardware . . . 6
- 3.2 Layer Operating System . . . 6
- 3.3 Layer Software Dependencies . . . 6
- 3.4 Scene Manager . . . 6
- 3.5 Patient Manager . . . 7
- 3.6 Event Manager . . . 7
- 3.7 Audio Manager . . . 8
- 3.8 Player Manager . . . 9

4 Game Object Layer Subsystems **11**
- 4.1 Layer Hardware . . . 11
- 4.2 Layer Operating System . . . 11
- 4.3 Layer Software Dependencies . . . 11
- 4.4 Medical Equipment . . . 11
- 4.5 Clipboard . . . 12
- 4.6 Patient . . . 12
- 4.7 House . . . 13
- 4.8 Furniture . . . 14
- 4.9 Player . . . 15

5 Enum Layer Subsystems **17**
- 5.1 Layer Hardware . . . 17
- 5.2 Layer Operating System . . . 17
- 5.3 Layer Software Dependencies . . . 17
- 5.4 Patient Health . . . 17
- 5.5 Heart Rate . . . 18
- 5.6 Blood Pressure . . . 18
- 5.7 Player Score . . . 19
- 5.8 Hospice . . . 20
- 5.9 Care . . . 20
- 5.10 Courtesy . . . 21

6 Data Object Layer Subsystems **23**
- 6.1 Layer Operating System . . . 23
- 6.2 Layer Software Dependencies . . . 23
- 6.3 Subsystem 1: Score Max . . . 23
- 6.4 Subsystem 2: Score Keeper . . . 24
- 6.5 Subsystem 3: Environment Status . . . 25
- 6.6 Subsystem 4: Patient Status . . . 25

7 Appendix A **27**

List of Figures

1 System architecture . . . 5
2 Scene manager diagram . . . 6
3 Patient manager diagram . . . 7
4 Event manager diagram . . . 8
5 Audio manager diagram . . . 8
6 Player manager diagram . . . 9
7 Medical equipment game object diagram . . . 11
8 Clipboard game object diagram . . . 12
9 Patient game object diagram . . . 13
10 House game object diagram . . . 14
11 Furniture game object diagram . . . 14
12 Player game object diagram . . . 15
13 Subsystem of Enum Layer: Patient Health . . . 17
14 Subsystem of Enum Layer: Heart Rate . . . 18
15 Subsystem of Enum Layer: Blood Pressure . . . 19
16 Subsystem of Enum Layer: Player Score . . . 19
17 Subsystem of Enum Layer: Hospice . . . 20
18 Subsystem of Enum Layer: Care . . . 21
19 Subsystem of Enum Layer: Courtesy . . . 21
20 Subsystem of Data Object Layer: Score Max . . . 23
21 Subsystem of Data Object Layer: Score Keeper . . . 24
22 Subsystem of Data Object Layer: Environment Status . . . 25
23 Subsystem of Data Object Layer: Patient Status . . . 26

List of Tables

1 Introduction

VR Palliative Care is a VR simulation game meant to be used by nursing student within the UTA system. The goal of this project is to give nursing student an immersive environment for learning how to deal with hospice care and different scenarios associated with it. Comprised of four scenarios, the program will take the user through the stages of cleaning, caring, and check ups with the patient. Currently the architectural and design documents can be found on the senior design blog website (https://blog.uta.edu/cseseniordesign/2022/12/12/vr-nursing-2/).

2 System Overview

The VR Palliative Care Simulation is made up four sections (layers). These four layers are "Managers", "Game Objects", "Enums", and "Data Objects". The Managers layer is the brains of the operation controlling how the game will be run. The Game Objects layer shows the types of game objects that will be used in the game. The Enums layer shows what metrics will be taken into account while the user is playing. And the Data Objects layer shows how the data will be collected and stored. The layers interact by sending messages/using each other to run the game.

Figure 1: System architecture

3 Manager Layer Subsystems

This section is describing the Managers section of the project. In this section, sub-managers which handle the flow of logic in the game project will be defined.

3.1 Layer Hardware

Oculus headset and controllers

3.2 Layer Operating System

None

3.3 Layer Software Dependencies

System is based in the Unity Game Engine and utilizes VRX.

3.4 Scene Manager

The scene manager is a system existing within the game project to handle the changing and saving of scene specific data. This manager allows the user to load and progress into existing scenes after some condition is met (ex: button to load scene is pressed).

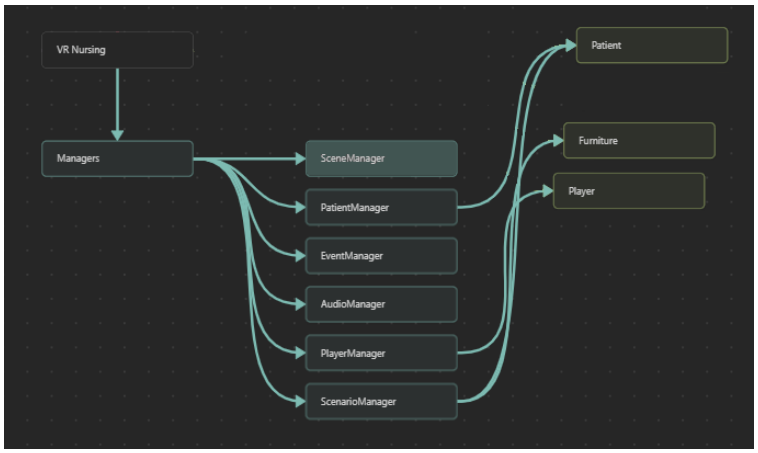

Figure 2: Scene manager diagram

3.4.1 Subsystem Hardware

None

3.4.2 Subsystem Operating System

None

3.4.3 Subsystem Software Dependencies

Unity Game Engine and VRX

3.4.4 Subsystem Programming Languages

C# Scripting language and standard Unity libraries

3.4.5 Subsystem Data Structures

Scripts will hold references to relevant gameObjects and events.

3.4.6 Subsystem Data Processing

This system will initiate the transitions of scenes via signals to relevant scripts.

3.5 Patient Manager

The manager controls and tracks the state of the patient throughout the progress of a scene. The patient's state will change from scene to scene and this manager will ensure the scene loads with the proper scenario conditions for the patient.

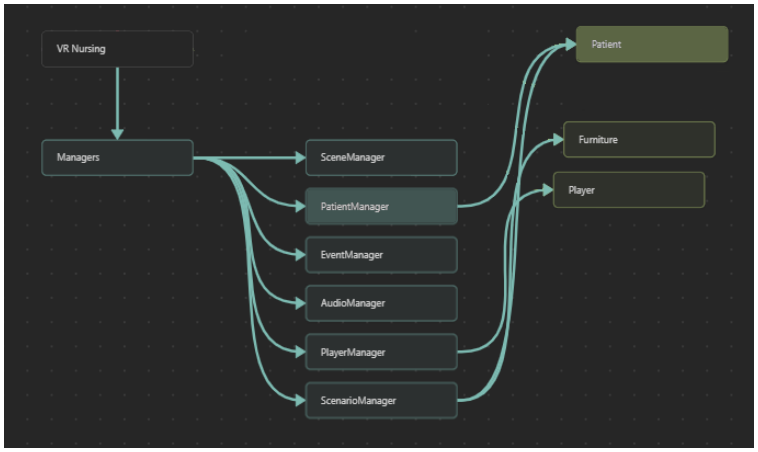

Figure 3: Patient manager diagram

3.5.1 Subsystem Hardware

None

3.5.2 Subsystem Operating System

None

3.5.3 Subsystem Software Dependencies

Unity Game Engine and VRX

3.5.4 Subsystem Programming Languages

C# Scripting language and standard Unity libraries

3.5.5 Subsystem Data Structures

Scripts will hold references to relevant gameObjects and events.

3.5.6 Subsystem Data Processing

This system will interact will gameObjects relevant to the operation of the patient.

3.6 Event Manager

The event manager handles any changes in a scene throughout the events of a scenario. This manager is responsible for triggering events once certain conditions are met.

3.6.1 Subsystem Hardware

None

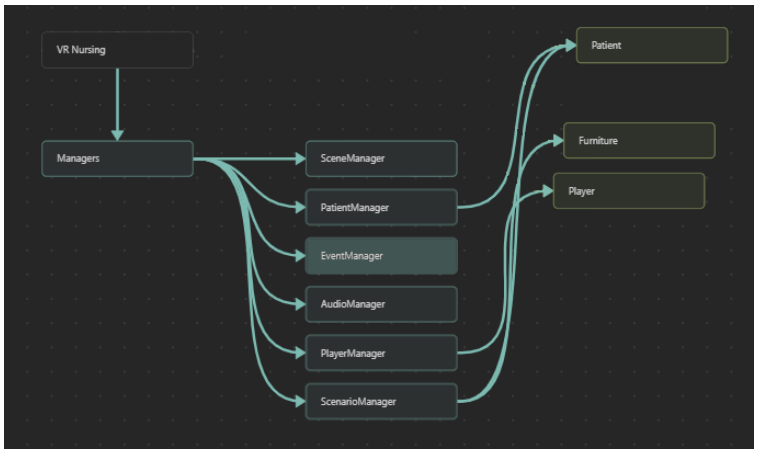

Figure 4: Event manager diagram

3.6.2 Subsystem Operating System

None

3.6.3 Subsystem Software Dependencies

Unity Game Engine and VRX

3.6.4 Subsystem Programming Languages

C# Scripting language and standard Unity libraries

3.6.5 Subsystem Data Structures

Scripts will hold references to relevant gameObjects and events.

3.6.6 Subsystem Data Processing

This system will send signals to initiate events in any scene.

3.7 Audio Manager

The audio manager will hold audio effect data and distribute them to the appropriate game objects.

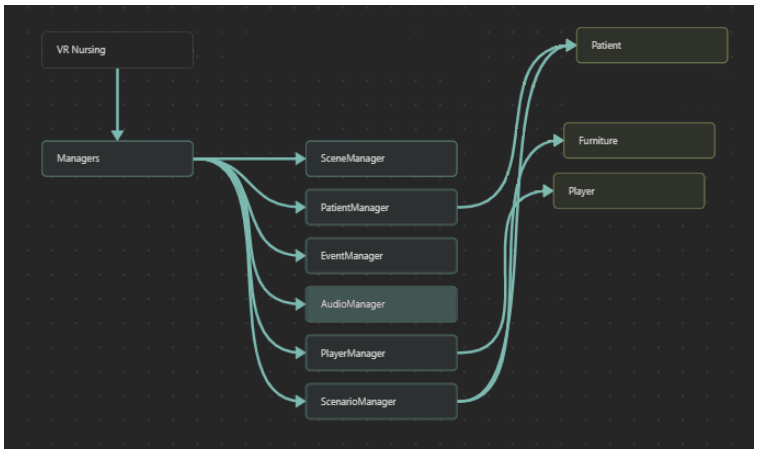

Figure 5: Audio manager diagram

3.7.1 Subsystem Hardware

None

3.7.2 Subsystem Operating System

None

3.7.3 Subsystem Software Dependencies

Unity Game Engine and VRX

3.7.4 Subsystem Programming Languages

C# Scripting language and standard Unity libraries

3.7.5 Subsystem Data Structures

Scripts will hold references to relevant gameObjects and events.

3.7.6 Subsystem Data Processing

This system will interact will audioSources in any given scene.

3.8 Player Manager

The player manager controls the player's movement and XR input.

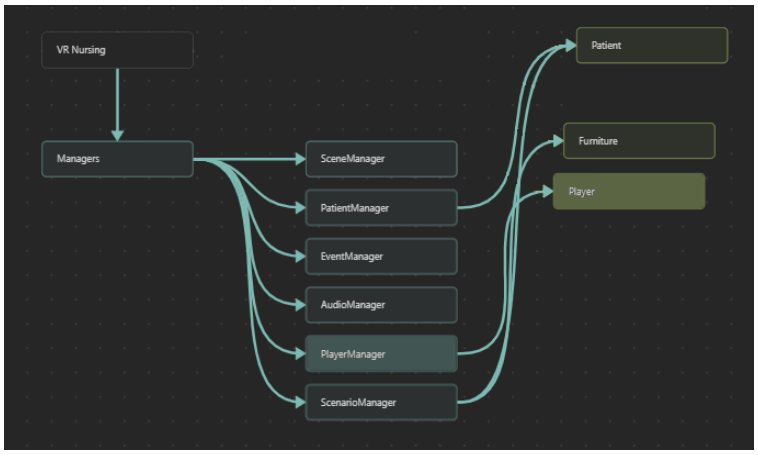

Figure 6: Player manager diagram

3.8.1 Subsystem Hardware

None

3.8.2 Subsystem Operating System

None

3.8.3 Subsystem Software Dependencies

Unity Game Engine and VRX

3.8.4 Subsystem Programming Languages

C# Scripting language and standard Unity libraries

3.8.5 Subsystem Data Structures

Scripts will hold references to relevant gameObjects and events.

3.8.6 Subsystem Data Processing

This system will interact will gameObjects relevant to the operation of the player.

4 Game Object Layer Subsystems

4.1 Layer Hardware

Oculus headset and controllers

4.2 Layer Operating System

None

4.3 Layer Software Dependencies

The game is based in Unity and uses VRX as the interface between user and game.

4.4 Medical Equipment

This object class is for all the intractable medical equipment the player can use. This includes things like the stethoscope, the IV system, the re-breather, morphine syringes, ect. These objects can be fully interacted with.

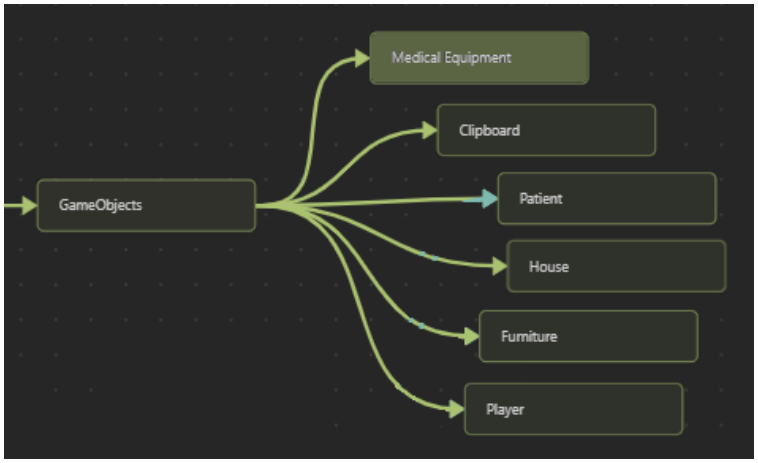

Figure 7: Medical equipment game object diagram

4.4.1 Subsystem Hardware

None

4.4.2 Subsystem Operating System

None

4.4.3 Subsystem Software Dependencies

We will be using Unity as our game engine. So, this game object class will literally be a GameObject. We will be using the assets provided to us by our predecessor teams.

4.4.4 Subsystem Programming Languages

Unity uses C# as it's scripting language.

4.4.5 Subsystem Data Structures

The objects will be sending and receiving messages from the managers. The messages will be sent via script events.

4.4.6 Subsystem Data Processing

The objects will use triggers to detect events and other interactions by the user and send the appropriate messages to the managers. They can also receive messages from the managers to perform actions (like playing a heartbeat sound in the stethoscope).

4.5 Clipboard

The clipboard is an object the user can interact with to navigate the scenes and change options in game. Think of it like a main menu but in physical form.

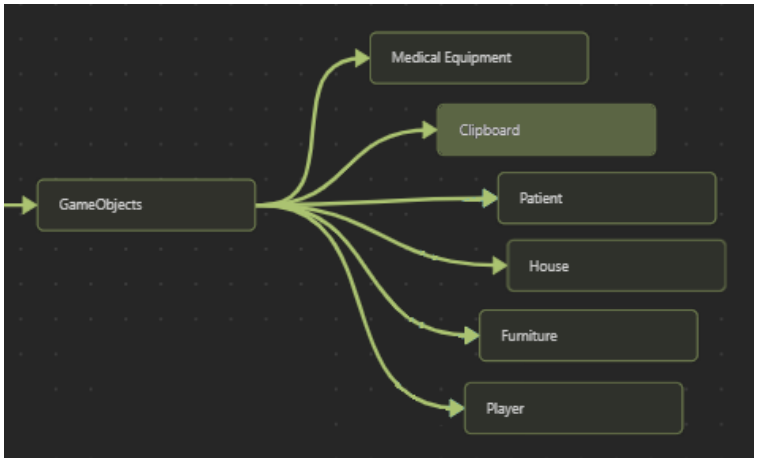

Figure 8: Clipboard game object diagram

4.5.1 Subsystem Hardware

None

4.5.2 Subsystem Operating System

None

4.5.3 Subsystem Software Dependencies

We will be using Unity as our game engine. So, this game object class will literally be a GameObject. We will have to find an appropriate clipboard asset to use.

4.5.4 Subsystem Programming Languages

Unity uses C# as it's scripting language.

4.5.5 Subsystem Data Structures

The object will be sending and receiving messages from the managers. The messages will be sent via script events.

4.5.6 Subsystem Data Processing

The objects will use triggers to detect user interaction and send the appropriate messages to the managers.

4.6 Patient

The patient is an object that the player will be able to interact with. It's exact interactions are described in the Manager Layer.

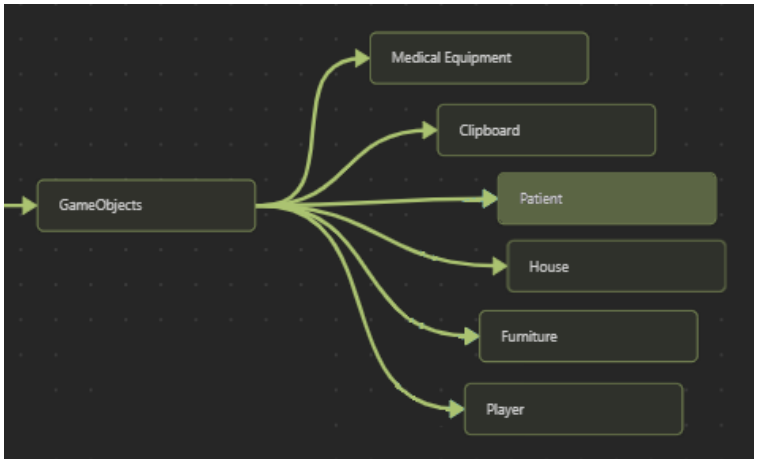

Figure 9: Patient game object diagram

4.6.1 Subsystem Hardware

None

4.6.2 Subsystem Operating System

None

4.6.3 Subsystem Software Dependencies

We will be using Unity as our game engine. So, this game object class will literally be a GameObject. We will be using the assets provided to us by our predecessor teams.

4.6.4 Subsystem Programming Languages

Unity uses C# as it's scripting language.

4.6.5 Subsystem Data Structures

The object will be sending and receiving messages from the managers. The messages will be sent via script events.

4.6.6 Subsystem Data Processing

The object will use triggers to detect events and other interactions by the user and send the appropriate messages to the managers. It can also receive messages from the managers to perform actions.

4.7 House

This object class is for the environment of the game. This includes things like the house, the background, and any other environmental settings.

4.7.1 Subsystem Hardware

None

4.7.2 Subsystem Operating System

None

4.7.3 Subsystem Software Dependencies

We will be using Unity as our game engine. So, this game object class will literally be a GameObject. We will be using the assets provided to us by our predecessor teams.

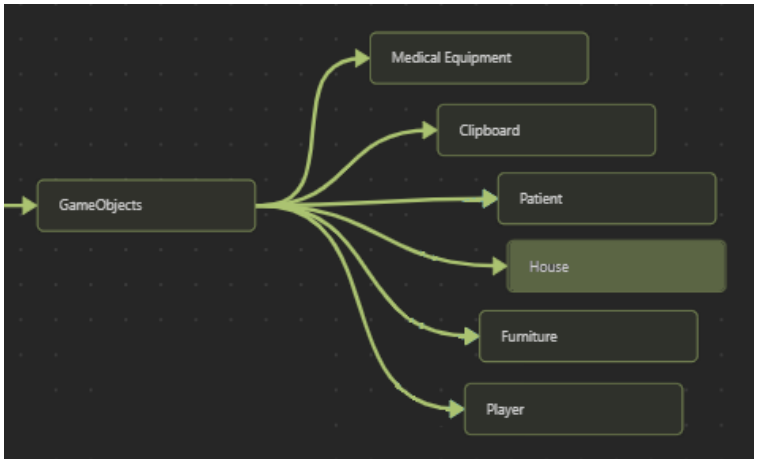

Figure 10: House game object diagram

4.7.4 Subsystem Programming Languages

This object class does not depend on a programming language, as it can be handled easily in the Unity editor.

4.7.5 Subsystem Data Structures

There are no data structures for this subsystem.

4.7.6 Subsystem Data Processing

There is no need for processing in this object class.

4.8 Furniture

This is an object class that populates the house and office to make it look more lived in. Some furniture will be intractable, like using the sink to wash hands, but cannot be moved by the player. Doors can be opened by the player, but by trigger instead of physics system interaction. This class of object includes, house furniture, office furniture, doors, plants, sinks, cabinets, light fixtures, etc.

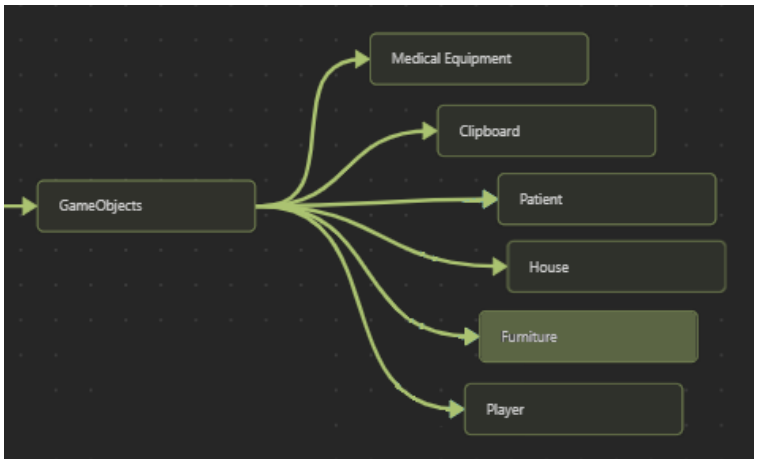

Figure 11: Furniture game object diagram

4.8.1 Subsystem Hardware

None

4.8.2 Subsystem Operating System

None

4.8.3 Subsystem Software Dependencies

We will be using Unity as our game engine. So, this game object class will literally be a GameObject. We will be using the assets provided to us by our predecessor teams.

4.8.4 Subsystem Programming Languages

This object class does not depend on a programming language, as it can be handled easily in the Unity editor.

4.8.5 Subsystem Data Structures

Most furniture do not need any data structures. However, for the doors, sinks, and grill, they will need to keep what state they are in. For the door, it needs to keep track of whether it is open or closed. For the sink, it needs to keep track if it is turned on or off. And for the grill, it needs to keep track whether it is burning or not.

4.8.6 Subsystem Data Processing

The object will use triggers to detect events and other interactions by the user and send the appropriate messages to the managers. It can also receive messages from the managers to perform actions.

4.9 Player

This is an object class that will be responsible for facilitating user interaction with the game world.

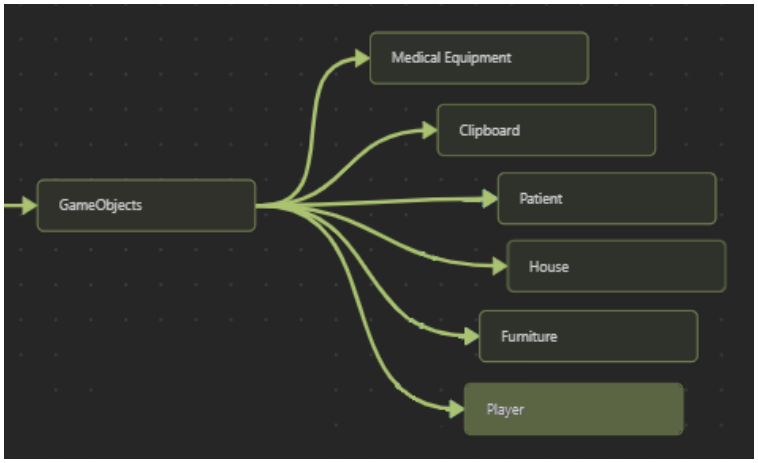

Figure 12: Player game object diagram

4.9.1 Subsystem Hardware

Oculus headset and controllers

4.9.2 Subsystem Operating System

None

4.9.3 Subsystem Software Dependencies

This player object will interface with VRX framework to allow the user to control and interact with the world.

4.9.4 Subsystem Programming Languages

Unity uses C# as it's scripting language.

4.9.5 Subsystem Data Structures

The object will be sending and receiving messages from the managers. The messages will be sent via script events.

4.9.6 Subsystem Data Processing

The object will interface with the VRX framework and use triggers to detect events and other interactions by the user and send the appropriate messages to the managers.

5 Enum Layer Subsystems

The Enum layer a set of classes with subclasses in each one. All of the classes in the Enum layer are used to quantify and store performance values for the user to determine how well they have done in the simulation. It is made up of the Patient Health and Player Score subsystems. The Patient Health subsystem includes the Heart Rate and Blood Pressure subsystems. The Player Score subsystem is made up of the Care, Hospice, and Courtesy subsystem.

5.1 Layer Hardware

The software processes for the Enum layer are all running on the Meta Quest 2 headset.

5.2 Layer Operating System

The operating system used for this layer is the Android OS. This is because the Meta Quest 2 runs on the Android OS.

5.3 Layer Software Dependencies

The Enum layer uses the System.Collections Unity library and it also uses the UnityEngine.

5.4 Patient Health

Patient Health is a class that stores 2 variables that indicate the patient's heart rate and blood pressure. This class is used to determine the correct response for the dialogue options.

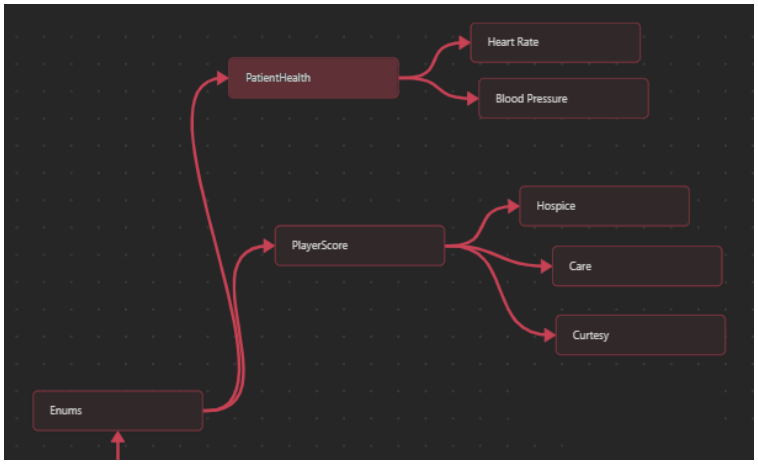

Figure 13: Subsystem of Enum Layer: Patient Health

5.4.1 Subsystem Operating System

Android OS

5.4.2 Subsystem Software Dependencies

The Patient Health class depends on the System.Collections and UnityEngine.

5.4.3 Subsystem Programming Languages

This subsystem uses C#.

5.4.4 Subsystem Data Structures

The Start method starts the checking for the patient health and the Update method checks the patients health after every frame.

5.5 Heart Rate

Heart Rate is a subsystem of Patient Health that stores the Heart rate of the patient in order for the Patient Health class to function.

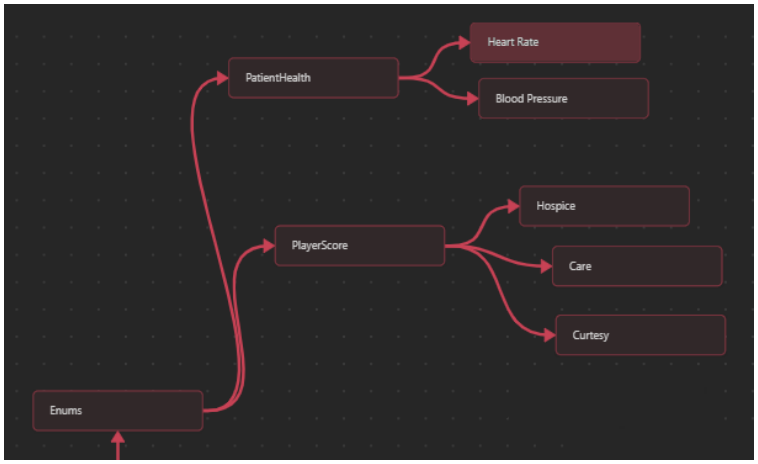

Figure 14: Subsystem of Enum Layer: Heart Rate

5.5.1 Subsystem Operating System

Android OS

5.5.2 Subsystem Software Dependencies

The Patient Health class depends on the System.Collections and UnityEngine

5.5.3 Subsystem Programming Languages

This subsystem uses C#.

5.5.4 Subsystem Data Structures

The Start method starts the checking for the patient health and the Update method checks the patients health after every frame.

5.6 Blood Pressure

Blood Pressure is a subsystem of Patient Health that stores the blood pressure of the patient in order for the Patient Health class to function.

5.6.1 Subsystem Operating System

Android OS

5.6.2 Subsystem Software Dependencies

The Patient Health class depends on the System.Collections and UnityEngine

5.6.3 Subsystem Programming Languages

This subsystem uses C#.

5.6.4 Subsystem Data Structures

The Start method starts the checking for the patient health and the Update method checks the patients health after every frame.

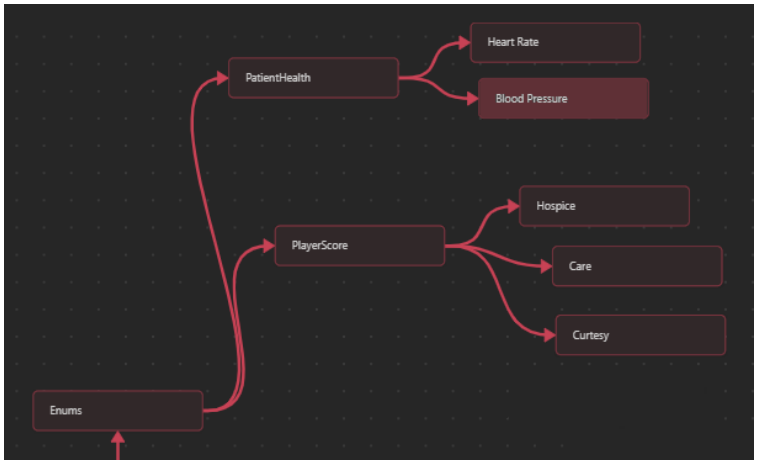

Figure 15: Subsystem of Enum Layer: Blood Pressure

5.7 Player Score

Player Score is a class that contains the Care, Hospice and Courtesy values used to judge the player at the end of the simulation.

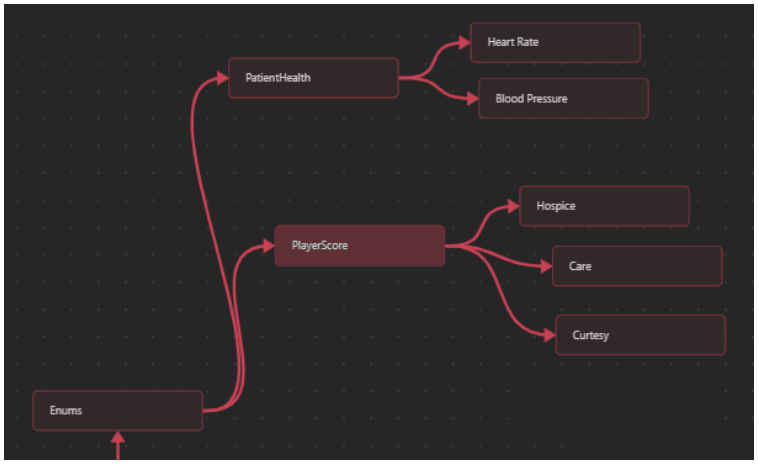

Figure 16: Subsystem of Enum Layer: Player Score

5.7.1 Subsystem Operating System

Android OS

5.7.2 Subsystem Software Dependencies

The Player Score class depends on the System.Collections and UnityEngine. It is also dependent on its three subclasses: Hospice, Care, and Courtesy.

5.7.3 Subsystem Programming Languages

This subsystem uses C#.

5.7.4 Subsystem Data Structures

The Start method starts the checking Hospice, Care, and Courtesy subsystems for the player score and the Update method checks the subsystems after every frame.

5.8 Hospice

The Hospice subsystem is used to store the points value for the object interactions related to scenarios 3 and 4. The points are scored based on the objects interacted and the dialogue options chosen.

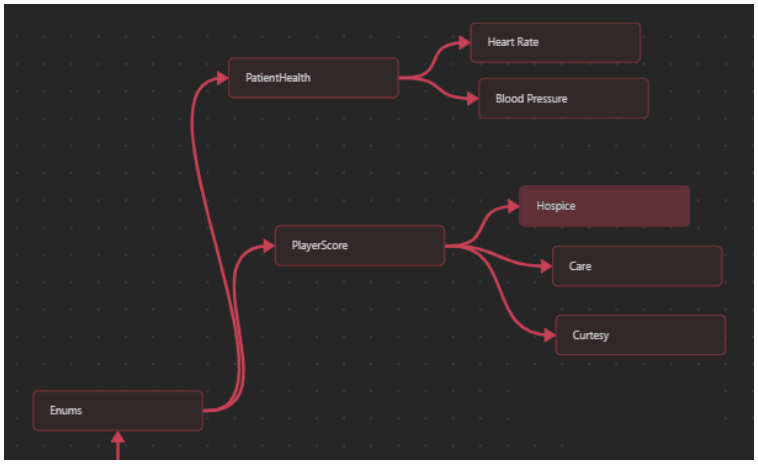

Figure 17: Subsystem of Enum Layer: Hospice

5.8.1 Subsystem Operating System

Android OS

5.8.2 Subsystem Software Dependencies

Hospice depends on the System.Collections and UnityEngine. It is also dependent on the dialogue manager and dialogue system event scripts.

5.8.3 Subsystem Programming Languages

This subsystem uses C#.

5.8.4 Subsystem Data Structures

The add points method is called when certain objects have been interacted with in scenarios 3 and 4.

5.9 Care

The Care subsystem is used to store the points attributed to the quality of care the nurse provides to the patient. This includes the order of test administered and knowing how to properly administer the medication.

5.9.1 Subsystem Operating System

Android OS

5.9.2 Subsystem Software Dependencies

Care depends on the System.Collections and UnityEngine. It is also dependent on the dialogue manager and dialogue system event scripts.

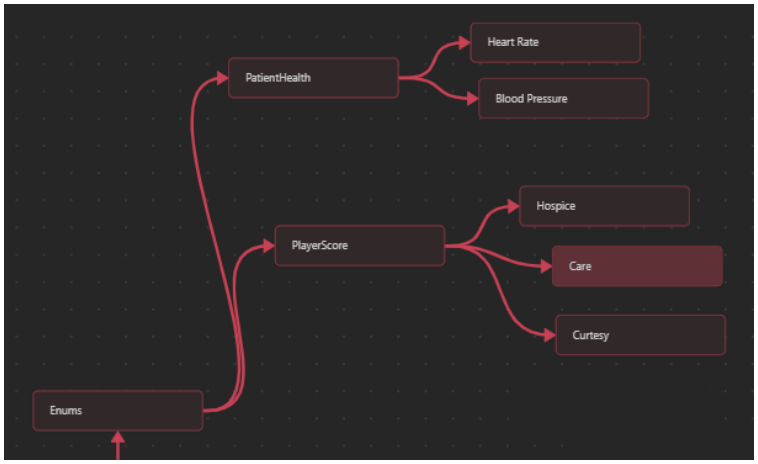

Figure 18: Subsystem of Enum Layer: Care

5.9.3 Subsystem Programming Languages

This subsystem uses C#.

5.9.4 Subsystem Data Structures

The add points method is called when certain objects have been interacted with.

5.10 Courtesy

Courtesy is used to store the points score for the dialogue with other people and the conversation with the patient.

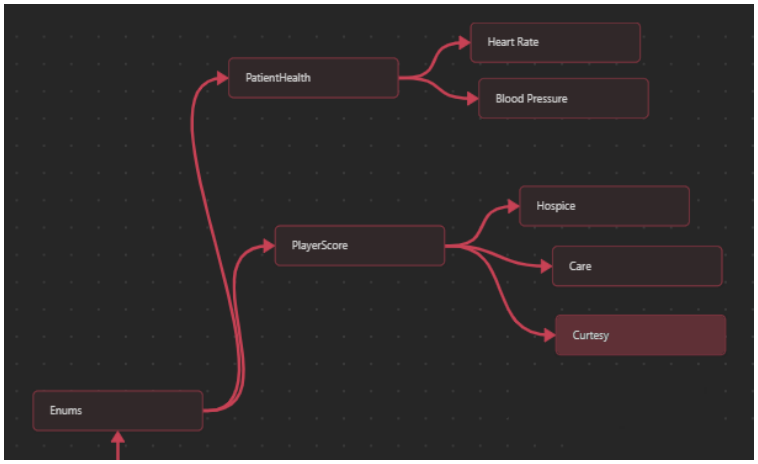

Figure 19: Subsystem of Enum Layer: Courtesy

5.10.1 Subsystem Operating System

Android OS

5.10.2 Subsystem Software Dependencies

Courtesy depends on the System.Collections and UnityEngine. It is also dependent on the dialogue manager and dialogue system event scripts.

5.10.3 Subsystem Programming Languages

This subsystem uses C#.

5.10.4 Subsystem Data Structures

The add points method is called when certain objects have been interacted with.

6 Data Object Layer Subsystems

The Data Object Layer is a crucial component in the software architecture that facilitates seamless communication and management of data objects and entities between the application and the underlying database. It serves as an intermediary, enabling the storage, retrieval, and updating of objects such as scores, environments, and patients during scenario play-through by the user. With its various subsystems and components: data encapsulation, data access, data validation, persistence management, and scenario progression tracking, the Data Object Layer ensures efficient handling and synchronization of data, allowing the application to effectively manage and present completion grades and scenario statuses to the user. By providing a structured and organized approach to data management, the Data Object Layer plays a vital role in maintaining the integrity and accuracy of the application's data objects and entities.

6.1 Layer Operating System

According to the Oculus Quest 2, utilized during development, it is compatible with many operating systems for Unity development: Windows, MacOS, and Linux. However, for optimal performance, it will require the Windows OS.

6.2 Layer Software Dependencies

The layer will be required to have updated libraries for validation and verification over objects and entities within this layer.

6.3 Subsystem 1: Score Max

The purpose of the Score Max subsystem is to provide a representation of the maximum score a user can achieve within a specific scenario. This subsystem is designed to facilitate the comparison of the user's achieved score with the maximum possible score in order to assess their performance within the scenario.

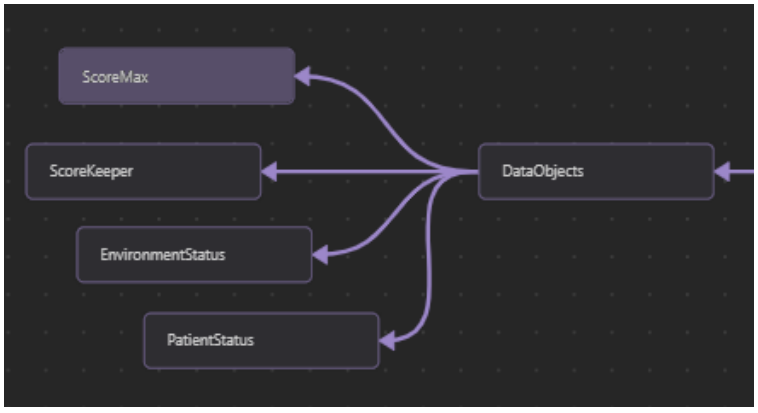

Figure 20: Subsystem of Data Object Layer: Score Max

6.3.1 Subsystem Operating System

It is compatible with many operating systems for Unity development: Windows, MacOS, and Linux. However, for optimal performance, it will require the Windows OS.

6.3.2 Subsystem Software Dependencies

The primary dependency for developing the subsystem in Unity is the Unity game engine itself. Unity provides a comprehensive set of tools and features for development, including support for scripting,

rendering, and user interface.

6.3.3 Subsystem Programming Languages

Unity uses C# as its language.

6.3.4 Subsystem Data Structures

It will be encapsulated in a class to hold and store total tasks within a scenario and hold the maximum score.

6.3.5 Subsystem Data Processing

The data will count and increment maximum score by iterating through specific tasks within a scenario and adding its point to achieve maximum score.

6.4 Subsystem 2: Score Keeper

The purpose of the Score Keeper subsystem is track user's task corresponding to a list of task needed to be completed by user in order to assess their performance within the scenario.

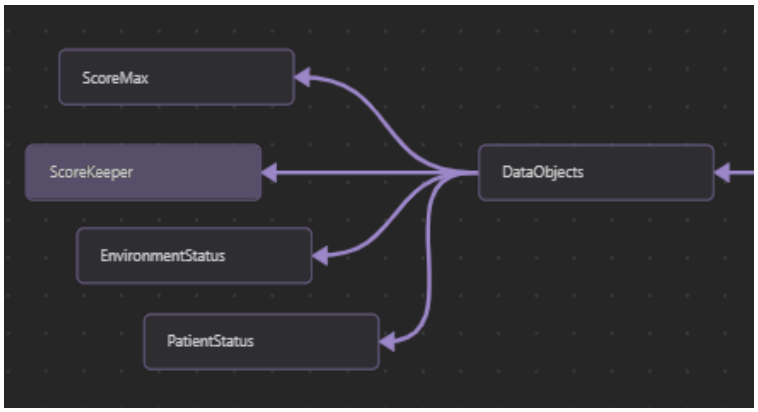

Figure 21: Subsystem of Data Object Layer: Score Keeper

6.4.1 Subsystem Operating System

It is compatible with many operating systems for Unity development: Windows, MacOS, and Linux. However, for optimal performance, it will require the Windows OS.

6.4.2 Subsystem Software Dependencies

The primary dependency for developing the subsystem in Unity is the Unity game engine itself. Unity provides a comprehensive set of tools and features for development, including support for scripting, rendering, and user interface.

6.4.3 Subsystem Programming Languages

Unity uses C# as its language.

6.4.4 Subsystem Data Structures

It will be encapsulated in a class to hold and store total tasks within a scenario and increment the score within this class to hold the total score the user has achieve within the scenario.

6.4.5 Subsystem Data Processing

The user upon completion of a task, the task will be utilized by Score Keeper to hold the user's current point by adding the object task to Score Keeper.

6.5 Subsystem 3: Environment Status

The purpose of the Environment Status subsystem is to represent and validate the statuses of environment objects and entities within each scenario.

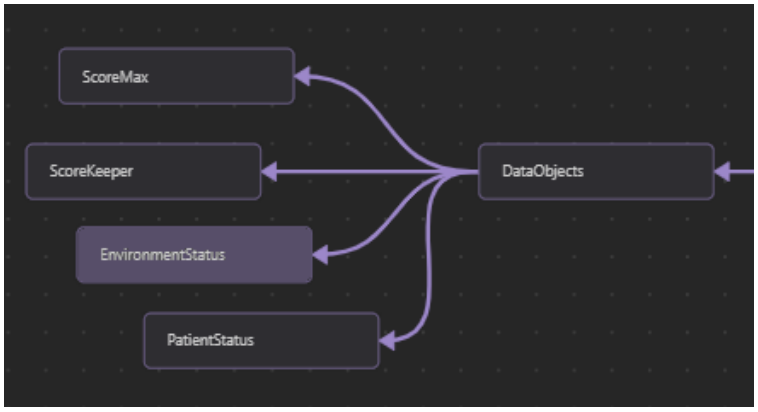

Figure 22: Subsystem of Data Object Layer: Environment Status

6.5.1 Subsystem Operating System

It is compatible with many operating systems for Unity development: Windows, MacOS, and Linux. However, for optimal performance, it will require the Windows OS.

6.5.2 Subsystem Software Dependencies

The primary dependency for developing the subsystem in Unity is the Unity game engine itself. Unity provides a comprehensive set of tools and features for development, including support for scripting, rendering, and user interface.

6.5.3 Subsystem Programming Languages

Unity uses C# as its language.

6.5.4 Subsystem Data Structures

It will be encapsulated in a class to hold the classes of objects and entities–such as task objects and environmental objects, within the environment of a scenario to be validated and allow the user to play through the scenario.

6.5.5 Subsystem Data Processing

With the classes of objects and entities that needs to be validated for the user, they'll be stored within a list that can be linearly validated and allow the scenario to be executed.

6.6 Subsystem 4: Patient Status

The purpose of the Patient Status subsystem is to represent and validate the status of the patient within each scenario.

Figure 23: Subsystem of Data Object Layer: Patient Status

6.6.1 Subsystem Operating System

It is compatible with many operating systems for Unity development: Windows, MacOS, and Linux. However, for optimal performance, it will require the Windows OS.

6.6.2 Subsystem Software Dependencies

The primary dependency for developing the subsystem in Unity is the Unity game engine itself. Unity provides a comprehensive set of tools and features for development, including support for scripting, rendering, and user interface.

6.6.3 Subsystem Programming Languages

Unity uses C# as its language.

6.6.4 Subsystem Data Structures

It will be encapsulated in a class to hold the classes of objects and entities–such as task objects and the patient's status, with the patient's status of a scenario to be validated and allow the continual update of the patient's status for simulation progression.

6.6.5 Subsystem Data Processing

With the class of the patient's status and the task objects, Patient Status will iterate through a list upon each user interaction with the patient in order to validate their task and proceed the Patient's status corresponding to the user's action within the user.

7 APPENDIX A

Include any additional documents (CAD design, circuit schematics, etc) as an appendix as necessary.

References