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

Architectural Design Specification
CSE 4316: Senior Design I
Spring 2023

VR Nursing Team
VR Palliative Care

Cesar Cantu-Perez
Braddock Bresnahan
Carlos Cruz
Nelson Lam

Revision History

Revision	Date	Author(s)	Description
0.1	04.03.2023	CCP	document creation
0.2	04.10.2023	CCP, BB, CC, NL	complete draft
2.0	05.10.2023	CCP, BB, CC, NL	Update for version 2

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Jennifer Roye and Shawn Gieser have commissioned a VR simulation to help nurses with hospice care. The simulation is currently being called "VR Palliative Care" but the name can be subject to change. VR Palliative Care is designed to give nursing students at UTA an educational experience inside an interactive world. Currently the nursing program at the University of Texas at Arlington does not have a way to prepare their students for the realities of hospice/end-of-life care. This simulation will be addressing that problem by providing a way for nursing students to practice hospice care protocols before having to perform the tasks in the real world.

The VR simulation is comprised of four scenarios. In scenario 1 the patient is in a hospital room where the nursing student will be introduced to the patient and his condition. By using the text-based dialogue options, the nurse will assess the situation and make decisions on how to proceed caring for the patient. In scenario 2, the nurse will visit the patient's home and meet his wife. The objective of scenario 2 is for the nurse to identify potential hazards for the patient before he is released into at home hospice care. The nursing student will be scored on how many hazards they found during their visit. In scenario 3, the patient is now at home and it is the nurse's job to follow up and perform some tests. The Glasgow-Coma Scale and the focused assessment of the patient will be required. It will also be important for the nursing student to understand and implement the correct order in which the tests should be administered. In scenario 4, the patient has died and the nurse is required to perform any final tests and comfort the family. The nurse will also be responsible for moving the body out of the house.

The VR Palliative Care system is designed to work with the XR SDK which allows compatibility with multiple headsets and controllers but will be exclusively tested on the Meta Quest 2 headsets. The performance of VR Palliative Care will be designed to be compatible with the Meta Quest 2 headsets while they are un-tethered. VR Palliative Care is intended to be a supplemental tool for nursing student at the College of Nursing in the University of Texas at Arlington. The system will not be made public to anyone besides members of the College of Nursing unless the sponsors decide to make the simulation public.

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: A simple architectural layer diagram

2.1 Manager Layer

The Manager Layer is responsible for the systems which will make up the starting and moment to moment game logic once inside the application. The purpose of the Management layer is keep functionality divided into their own respective roles for the sake of simplicity during development. This layer will control areas such as scene, patient, event, audio, player, and scenario data. This layer will work in tandem with every other layer to provide visual and functional feedback to the user throughout the time inside the application.

2.2 Game Object Layer

The Game Object Layer details how the game objects in the world interact with other game objects and other systems. This layer will primarily get its inputs from the Management layer or other game objects. The layer may supply structures from the Enum Layer or the Data Object layer to the Management layer depending on what the Management layer needs. This layer needs to be able to interface with global constructs, like the vibration controls on the VR controllers, and other constructs such as the dialog controller (not pictured).

2.3 Enum Layer

The Enum Layer is responsible for holding the values that quantify player performance and patient stats. It has 2 subsections one for Patient Health and one for Player Score. The purpose of the Enum Layer is to communicate with the other layers to deliver the needed values to the layers who need it. When another layer requests a value, the Enum layer sends the required value needed to keep the progression of the simulation going. The inputs are typically set by the developer in the case of patient health but others are also dependent on the users actions and choices throughout the simulation.

2.4 Data Object Layer

The Data Object layer is designed to communicate the application to the database in order to manage data objects and entities such as the scores, the environment, and the patient during each scenario play-

through by the user. This layer will store these objects and entities to be stored, updated, and retrieved by the application for the user in order to display the user's completion grade over a scenario. When the user interact with a specific object or entity in the scenario, the data object layer will be able to retrieve the object/entity to be updated for the scenario progression. Thus, the primary input for this layer will be the individual objects/entities and tasks for the various classes within this layer in order to validate and update the scenario statuses and data objects.

3 Subsystem Definitions & Data Flow

The following is the class diagram for the project.

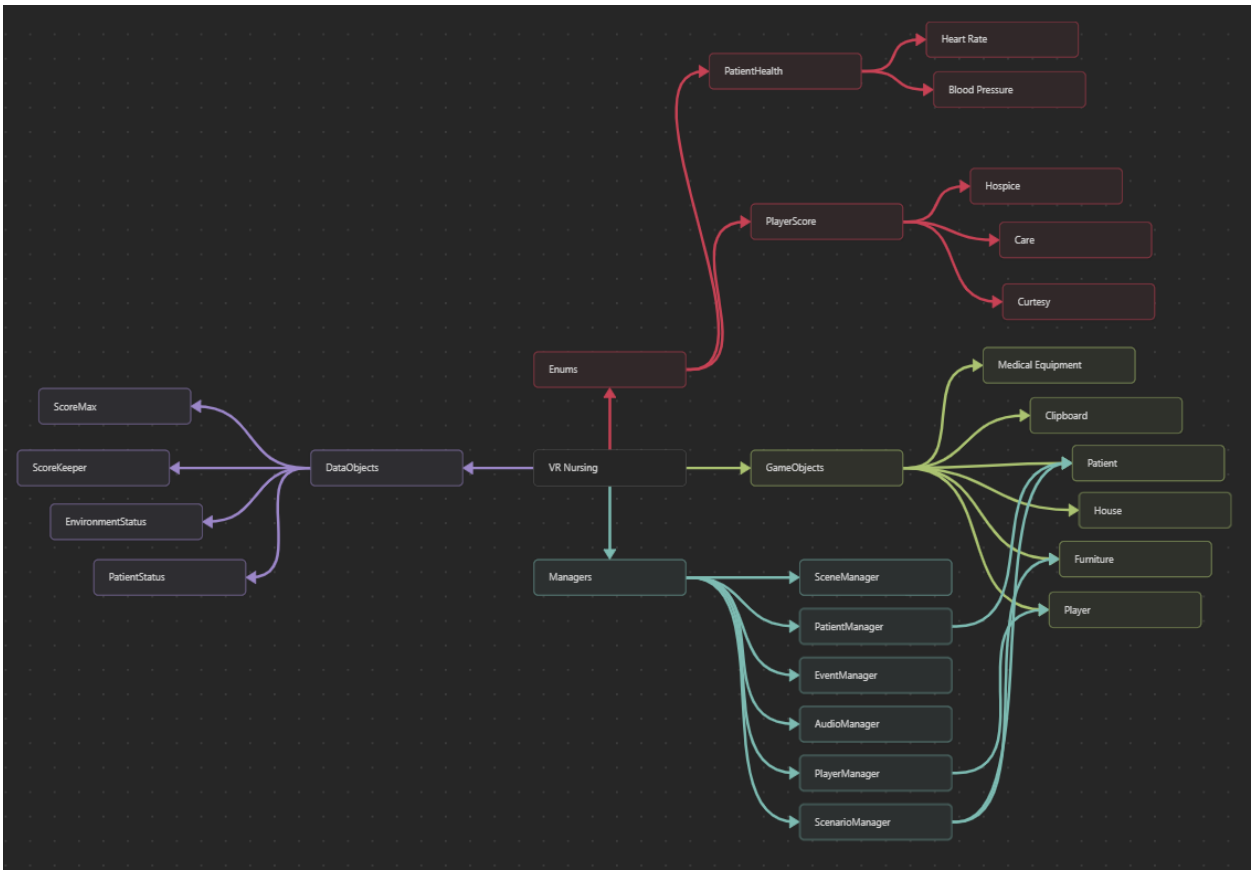

Figure 2: Class diagram

4 Manager Layer

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.

4.1 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).

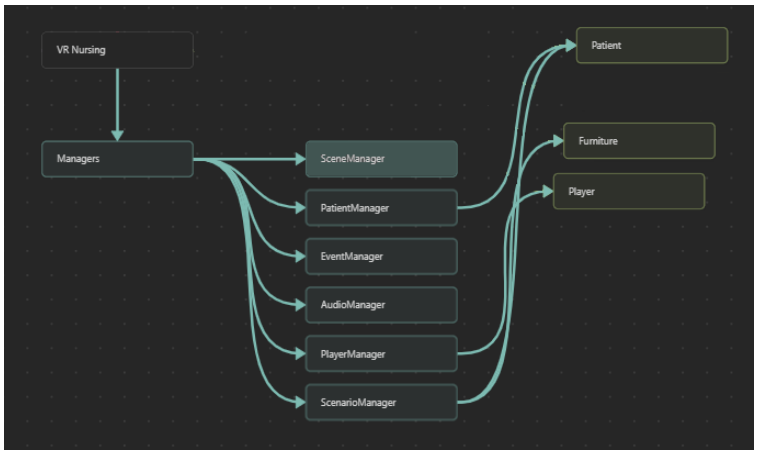

Figure 3: Scene manager diagram

4.1.1 Assumptions

We assume that the scenes being handled are the desired versions of the scene as given by the developer.

4.1.2 Responsibilities

The responsibility of the scene manager is to load a scene, transition between scenes, and exit out of a given scene either to the main menu scene or to desktop.

4.1.3 Subsystem Interfaces

Table 2: Scene manager subsystem interfaces

ID	Description	Inputs	Outputs
#1	Load scene	scene number	Opens scene
#2	Next scene	scene number	Opens next scene
#3	Exit scene	N/A	Quits to main menu
#4	Quit game	N/A	Exits application

4.2 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.

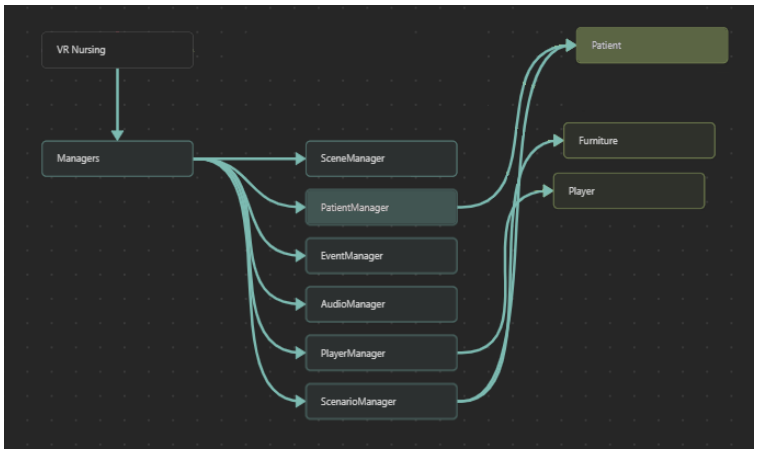

Figure 4: Patient manager diagram

4.2.1 Assumptions

We assume the patient model and animations will separate from the scope of this manager.

4.2.2 Responsibilities

The Patient manager must keep track of the health of the patient by recording the state of different measurements (i.e heart rate, blood pressure, etc.). This will also output the measurements to appropriate UI elements when measurements are performed by the player.

4.2.3 Subsystem Interfaces

Table 3: Patient manager subsystem interfaces

ID	Description	Inputs	Outputs
#1	Patient Object	Scenario data Start state data	health/condition data

4.3 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.

4.3.1 Assumptions

We assume that other game objects will be subscribed to the event triggers in this handler.

4.3.2 Responsibilities

This handler will trigger events for an objective is complete, and when a player progresses in a task, and when the scenario calls for changes in the environment.

4.4 Audio Manager

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

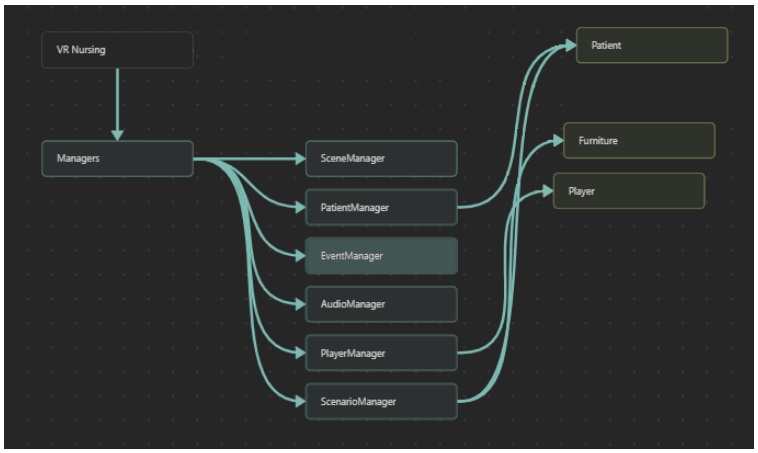

Figure 5: Event manager diagram

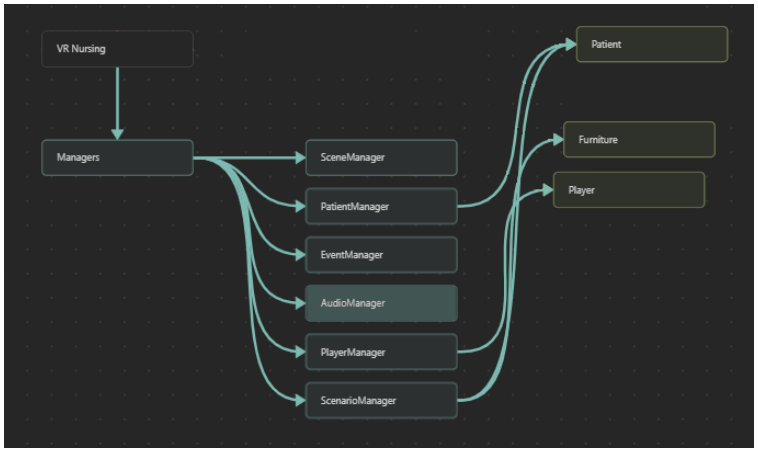

Figure 6: Audio manager diagram

4.4.1 Assumptions

Assuming the audio files will be provided by the developers as the project develops. Also that the necessary game objects using audio files will be connected appropriated in the Unity editor.

4.4.2 Responsibilities

The Audio Manager is the central point for holding all audio files and will distribute them to the necessary game objects. A "getter" function for each audio file will be held within the script so that other scripts can access the files.

4.4.3 Subsystem Interfaces

Table 4: Audio manager subsystem interfaces

ID	Description	Inputs	Outputs
#1	Sound interface	Audio data	Audio source location

4.5 Player Manager

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

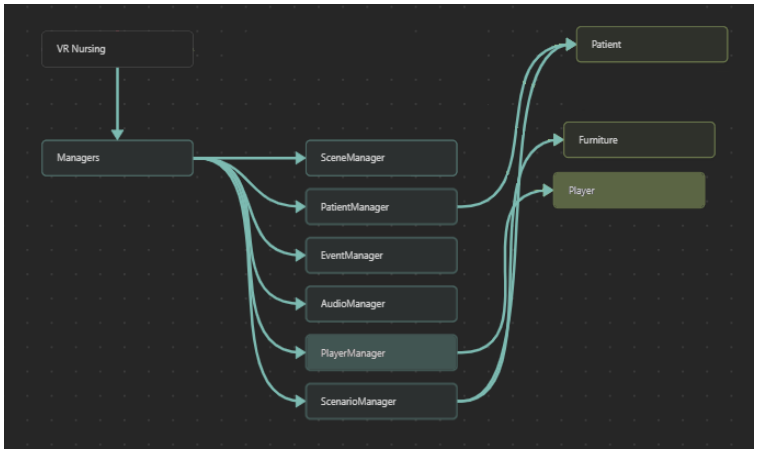

Figure 7: Player manager diagram

4.5.1 Assumptions

Assuming the XR input manager provided by Unity functions as given and interfaces with the VR headset. The only headset to test this functionality on will be the Meta Quest 2.

4.5.2 Responsibilities

The responsibilities of the player manager include handling player input, player movement within space, and handling XR headset movement/input.

4.5.3 Subsystem Interfaces

Table 5: Player manager subsystem interfaces

ID	Description	Inputs	Outputs
#1	Headset interface	Tracking data Controller input	3D movement

4.6 Scenario Manager

The scenario manager holds data specific data that pertains to the spawn locations of different objects / npcs and quest objectives.

4.6.1 Assumptions

We will assume data contained in this manager will remain static during run-time.

4.6.2 Responsibilities

The scenario manager must contain start data pertinent to the locations of objective specific objects and NPC locations. The scenario manager will also hold dialog data for the NPC's and handle the progression of conversations.

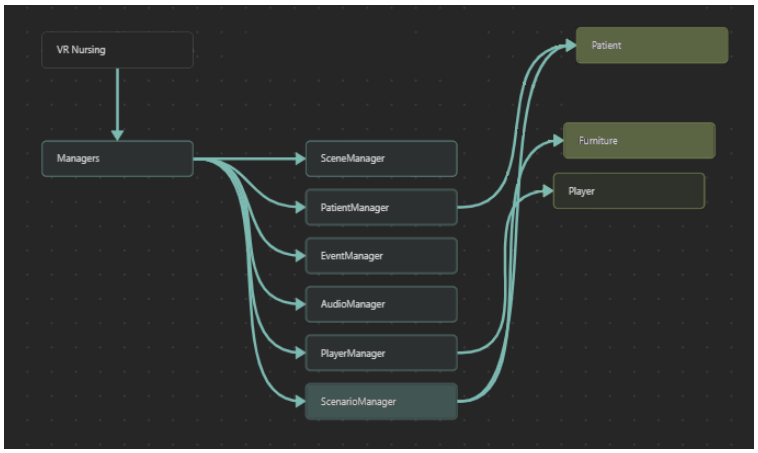

Figure 8: Scenario manager diagram

4.6.3 Subsystem Interfaces

Table 6: Scenario manager subsystem interfaces

ID	Description	Inputs	Outputs
#1	Dialog interface	Dialog raw text	visual dialog

5 Game Object Layer

In this section, the layer is described in some detail in terms of its specific subsystems. Describe each of the layers and its subsystems in a separate chapter/major subsection of this document. The content of each subsystem description should be similar. Include in this section any special considerations and/or trade-offs considered for the approach you have chosen.

5.1 Medical Equipment

For scenes 1 and 3, there will be medical equipment the player needs to interact with to complete the scene. For scene 1 there is the phone, the computer, the stethoscope, the drug dispenser, and the IV that the user can interact with. For scene 3 there is the suction pump, the stethoscope, etc.

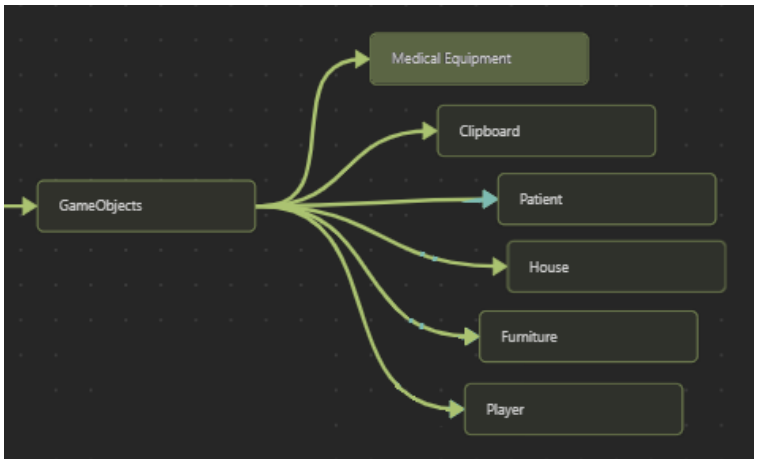

Figure 9: Medical equipment diagram

5.1.1 Assumptions

This subsystem assumes that it has global access to the Dialog system as well as access to VR haptic controls. It also assumes all object in the category are non-static.

5.1.2 Responsibilities

This group of objects can be interacted with by the user. They will be able to be picked up and used by the player on the patient. Some objects, like the phone, will interact with the dialog system (not pictured); while others, like the computer, will display an enlarged view of its screen. This group is also responsible for giving feedback to the user when the object is used. For example, then morphine is administered through the IV, the object will give visual or haptic feedback that the task has been done.

5.1.3 Subsystem Interfaces

The objects may interface with the dialog system (not pictured) when the user interacts with an object that calls for it; and the haptic components of the controllers (not pictured) activating a short vibration when an action has successfully completed.

Table 7: Medical equipment subsystem interfaces

ID	Description	Inputs	Outputs
#1	Dialog System	Dialog option selection	Display dialog box
#2	Haptic controller feedback	N/A	Vibration command

5.2 Clipboard

There will be a central object that the user can interact with for meta operations (like quit, scenario select, options, etc.) The object will take the form of some kind of clipboard, smart phone, or tablet. It will need to interact with the scene manager for scene selection.

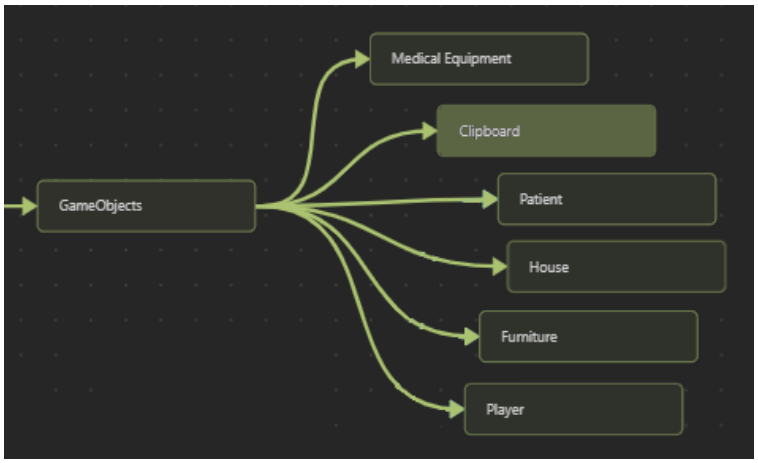

Figure 10: Clipboard diagram

5.2.1 Assumptions

The clipboard will be able to interface with the manager to perform its meta functions.

5.2.2 Responsibilities

The user will be able to interact with the clipboard by grabbing it from their imaginary "belt". It will be hanging off to their side. Then the user will be able to interact with the clipboard by selecting the options with their other hand.

5.2.3 Subsystem Interfaces

The clipboard will interact with the Manager subsection to allow the changing of scenes as well as other meta operations.

Table 8: Clipboard subsystem interfaces

ID	Description	Inputs	Outputs
#1	Scene Manager	N/A	Change scene, quit, options

5.3 PATIENT

The patient is an object that the player and medical objects can interact with. For scenes 1, 3, and 4, the patient is the center of attention. The user will interact with the patient through dialog and by using medical object when required.

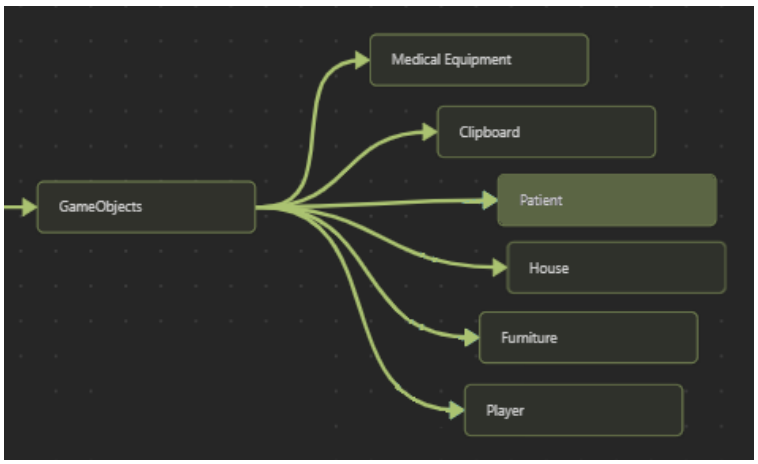

Figure 11: Patient diagram

5.3.1 ASSUMPTIONS

The patient will be a static object and does not need to move or perform animations.

5.3.2 RESPONSIBILITIES

The patient is the pace maker for scenes 1, 3, and 4. The user will interact with the patient to progress in those scenarios.

5.3.3 SUBSYSTEM INTERFACES

The patient will interact with the patient manager to keep track of what state the patient is in. The scenario manager will also interact with the scenario manager to keep track of the progress in the scenario.

Table 9: Patient subsystem interfaces

ID	Description	Inputs	Outputs
#1	Patient Manager	State the patient should be	Any interaction from the player
#2	Scenario Manager	What the user is expected to do	If the user performed the task

5.4 HOUSE

The house is where scenes 2, 3, and 4 take place. The house will house all the furniture, other NPCs, the patient, the player, and medical equipment.

5.4.1 ASSUMPTIONS

The user teleportation works on the house floors, including second floor.

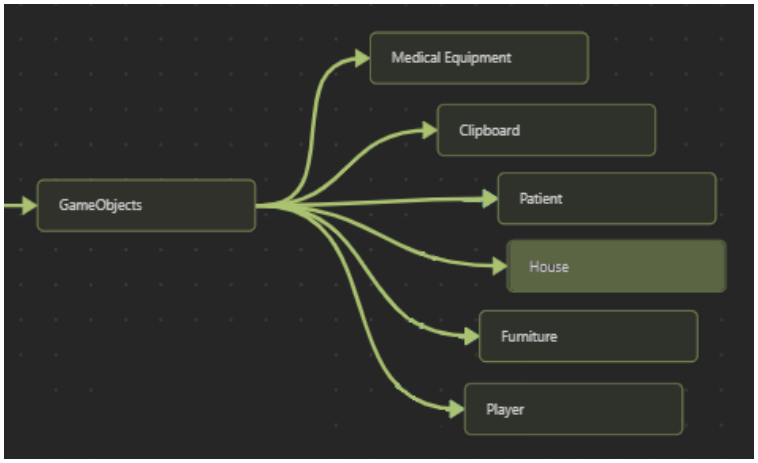

Figure 12: House diagram

5.4.2 Responsibilities

The house will facilitate scenarios 2, 3, and 4. Only the rooms needed to perform the scenario will be available for the player to walk around in. The front and back door can be interacted with, but no other door to blocked off areas should be operable. There will be an option at the bottom and top of the stairs that allow the user to teleport up and down.

5.4.3 Subsystem Interfaces

The only object the house needs to interact with is the player for teleportation and stair traversal.

Table 10: House subsystem interfaces

ID	Description	Inputs	Outputs
#1	Player object	Player wants to teleport upstairs	Move player position

5.5 Furniture

Furniture will be dotted throughout the office and the house to make the scenery seem lived in. Some furniture will interact with the player in some scenarios.

5.5.1 Assumptions

All furniture is static and immovable.

5.5.2 Responsibilities

The House and Office will be populated with believable furniture. The furniture cannot be moved but some can be interacted with, like the sink in the bathroom to wash hands.

5.5.3 Subsystem Interfaces

The furniture will need to interact with the Manager layer for scene specific goals (like washing hands).

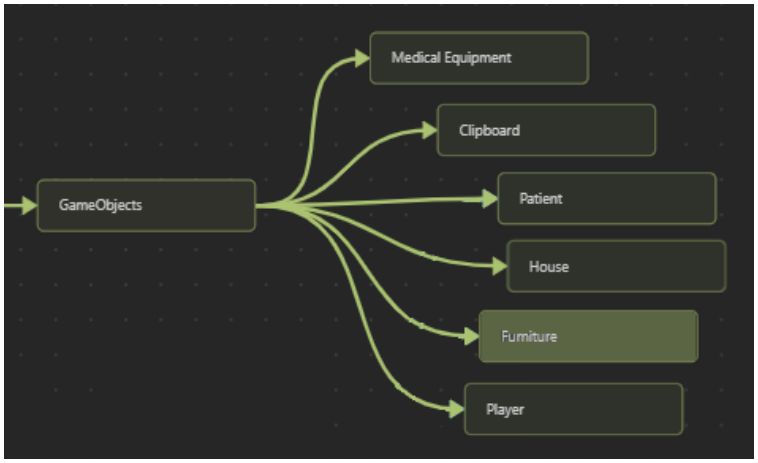

Figure 13: Furniture diagram

Table 11: Furniture subsystem interfaces

ID	Description	Inputs	Outputs
#1	Scenario Manager	N/A	Action performed by user

5.6 Player

The player object represents the user in space. It will need to interact with VR systems to simulate the user in VR.

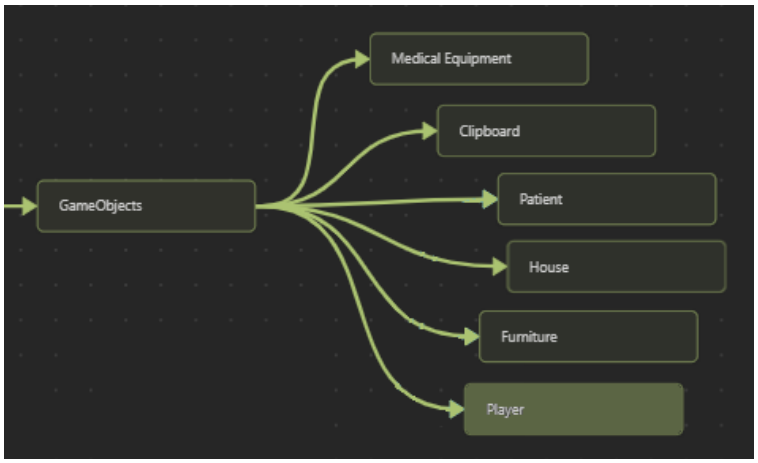

Figure 14: Player diagram

5.6.1 Assumptions

Furniture objects are static. The user will interact with the player object through the player manager.

5.6.2 RESPONSIBILITIES

The player object will disallow movement into invalid places, like going through walls, and disallow collisions with the furniture in a scene. This is just an object that represents the player.

5.6.3 SUBSYSTEM INTERFACES

The player manager controls the player object in all functions (practically).

Table 12: Player subsystem interfaces

ID	Description	Inputs	Outputs
#1	Player Manager	user position	N/A

6 Enum Layer

This section will describe the enum layer which is where the constant variables for the simulation will be held. These variables will be used in every sub section to determine logic and keep the progression of the simulation at a reasonable pace.

6.1 Patient Health

The Patient Health Enum keeps track of the patients heart rate and blood pressure used to check the dialogue options for correct responses.

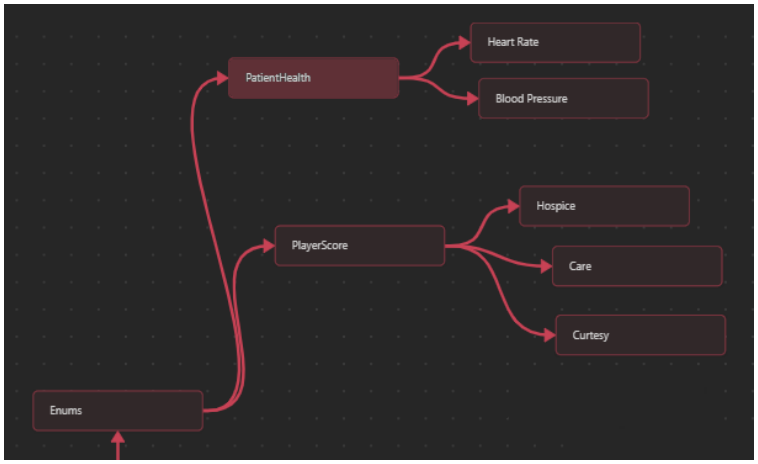

Figure 15: Patient health diagram

6.1.1 Assumptions

The heart rate and blood pressure should change from scene to scene as the patient condition worsens.

6.1.2 Responsibilities

The Patient Health Enum group is responsible for holding the values for heart rate and blood pressure. This is the enum that will be called by the other systems when the heart rate and blood pressure are needed for checking dialogue options or scoring the nurse.

6.1.3 Subsystem Interfaces

Table 13: Patient health subsystem interfaces

ID	Description	Inputs	Outputs
#1	Heart Rate	Developer made for each scenario	Current heart rate of the patient
#2	Blood Pressure	Developer made for each scenario	Current blood pressure of the patient

6.2 Heart Rate

The heart rate enum holds the current heart rate and passes it into the Patient Health enum

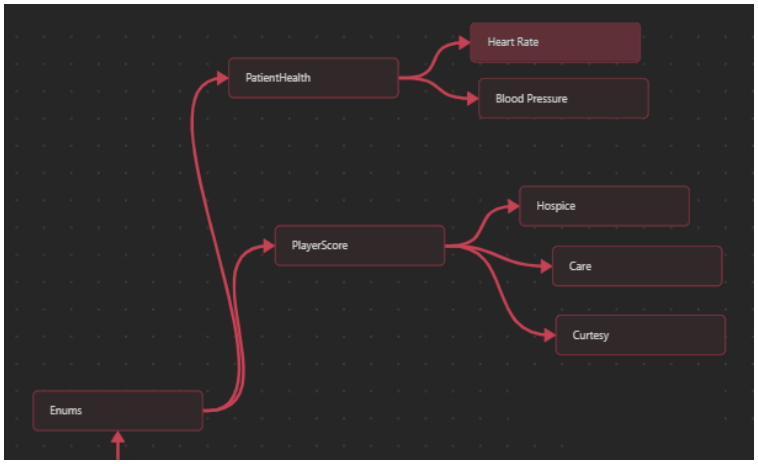

Figure 16: Heart rate diagram

6.2.1 Assumptions

The heart rate will differ from scene to scene as the patients condition worsens.

6.2.2 Responsibilities

The heart rate enum is responsible for holding the heart rate value assigned by the developer.

6.2.3 Subsystem Interfaces

Table 14: Heart rate subsystem interfaces

ID	Description	Inputs	Outputs
#1	Heart rate value	Inserted by developer	Heart rate value to send to Patient Health

6.3 Blood Pressure

The blood pressure enum holds the current blood pressure and passes it into the Patient Health enum

6.3.1 Assumptions

The blood pressure will differ from scene to scene as the patients condition worsens.

6.3.2 Responsibilities

The blood pressure enum is responsible for holding the blood pressure value assigned by the developer.

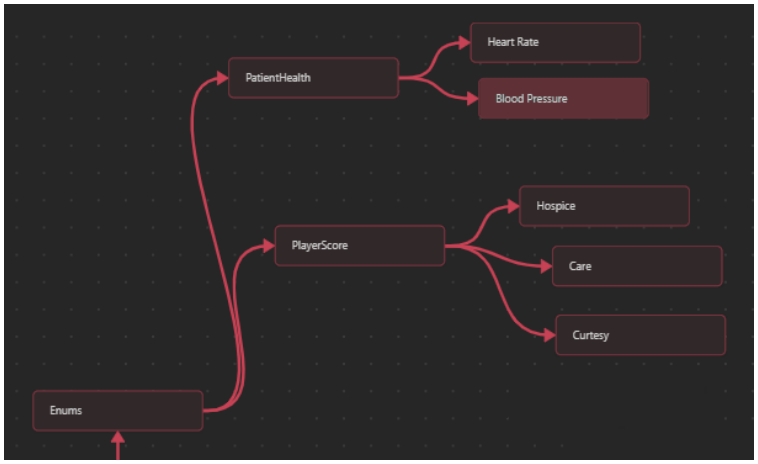

Figure 17: Blood Pressure diagram

6.3.3 Subsystem Interfaces

Table 15: Blood pressure subsystem interfaces

ID	Description	Inputs	Outputs
#1	Blood pressure value	Inserted by developer	Blood Pressure value to send to Patient Health

6.4 Player Score

Player Score is used to measure the performance of the nurse in training. By determining the percentage of the nurses correct answers out of possible correct answers a player score can be determined and will be viewing from the results screen at the end of each scenario.

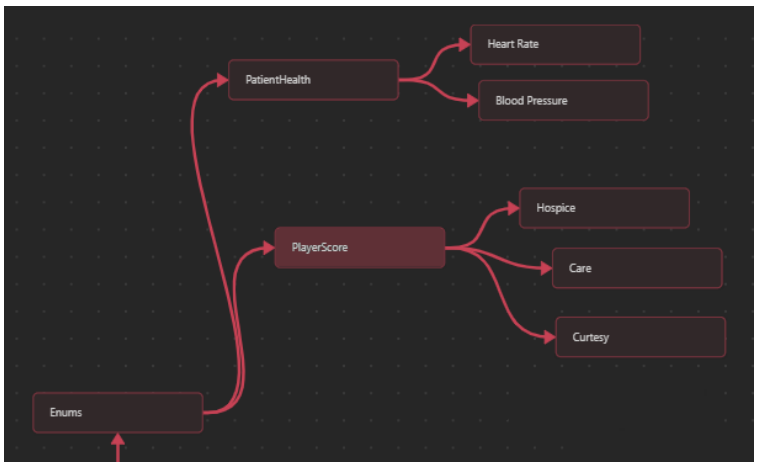

Figure 18: Player score diagram

6.4.1 Assumptions

The maximum score will be a 100% and the Player score can never be negative.

6.4.2 Responsibilities

The player score enum is responsible for taking the Hospice, Care, and Courtesy variables and using them to calculate score. It is also responsible for sending the value to the results screen at the end of every scenario.

6.4.3 Subsystem Interfaces

Table 16: Player score subsystem interfaces

ID	Description	Inputs	Outputs
#1	Player Score calculation	Hospice Care Courtesy	Player score value

6.5 Hospice

Hospice enum hold a value used to portray hospice which is the used to describe the at home care in scenarios 3 and 4. It will be used to measure the level of comfort the nurse provided for the patient.

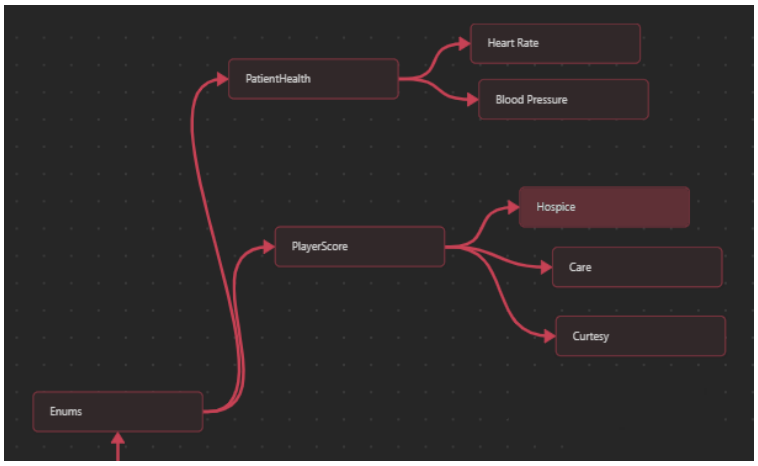

Figure 19: Hospice diagram

6.5.1 Assumptions

Hospice can only be used in scenarios 3 and 4 as those are the only scenarios where is patient is at home.

6.5.2 Responsibilities

Takes dialogue choices that were chosen and decreases when an answer choice is incorrect and belongs to a dialogue that is testing the hospice.

6.5.3 Subsystem Interfaces

Table 17: Hospice subsystem interfaces

ID	Description	Inputs	Outputs
#1	Decrease hospice value	User dialogue choices	New Hospice value
#2	Send to Player Score	User ends scenario	Value sent to Player Score

6.6 Care

Care is used to store the value that will be used to quantify the quality of care the nurse provided to the patient. Actions such as choosing the correct order of tests and properly administering the medication count towards this score.

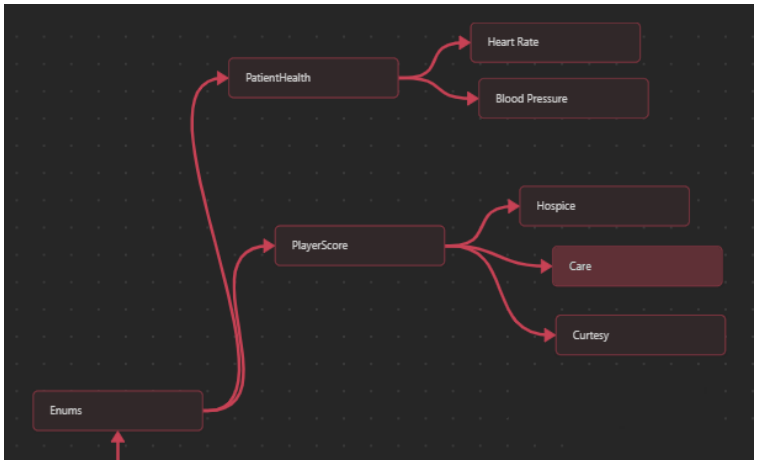

Figure 20: Care diagram

6.6.1 Assumptions

Care will be used in all 4 scenarios and will be the biggest part of all player score calculations. Will also start at max by default.

6.6.2 Responsibilities

Whenever a player chooses an incorrect action or dialogue choice, the care score decreases. The care score is sent to player score to determine the final score at the end of each scenario.

6.6.3 Subsystem Interfaces

Table 18: Care subsystem interfaces

ID	Description	Inputs	Outputs
#1	Decrease care value	User dialogue choices	New care value
#2	Send to Player Score	User ends scenario	Value sent to Player Score

6.7 Courtesy

Courtesy will be used when speaking to the patient and other people. This is used to quantify the quality of your responses to questions and how you console the family throughout the simulation.

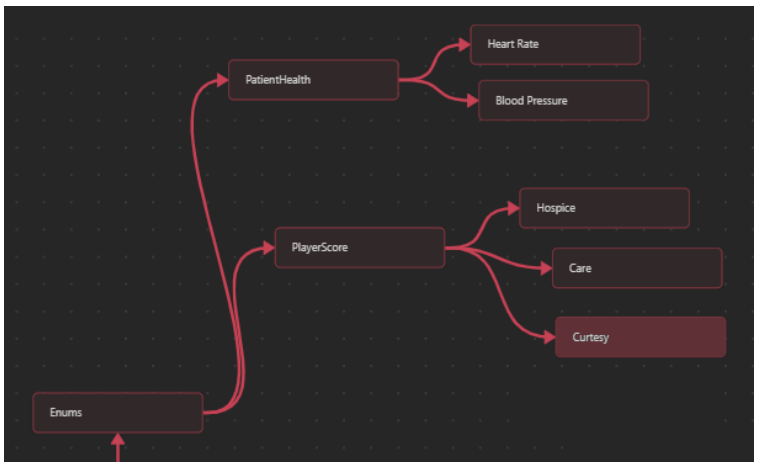

Figure 21: Courtesy diagram

6.7.1 Assumptions

As with the other scoring enums, the score will start at max and only decrease. This value will be sent to player score for the final score calculation at the end of every scenario.

6.7.2 Responsibilities

Whenever a player chooses an incorrect action or dialogue choice, the courtesy score decreases. The care score is sent to player score to determine the final score at the end of each scenario.

6.7.3 SUBSYSTEM INTERFACES

Table 19: Courtesy subsystem interfaces

ID	Description	Inputs	Outputs
#1	Decrease courtesy value	User dialogue choices	New courtesy value
#2	Send to Player Score	User ends scenario	Value sent to Player Score

7 Data Object Layer

The data object layer is a component of the virtual reality nursing application's software architecture that is responsible for managing the data objects/entities. The layer is utilized to represent an abstraction layer between the application and the data storage system to represent various data object types such as classes and objects contained within this layer.

7.1 Score Max

Score Max is a class within the data object layer to represent the maximum score the user can achieve within a specific scenario. This will allow the application to retrieve the Score Max for the user to compare the score they achieve post-simulation of a scenario in order to see if they were able to complete the scenario or they had missed specific objectives and tasks within the scenario.

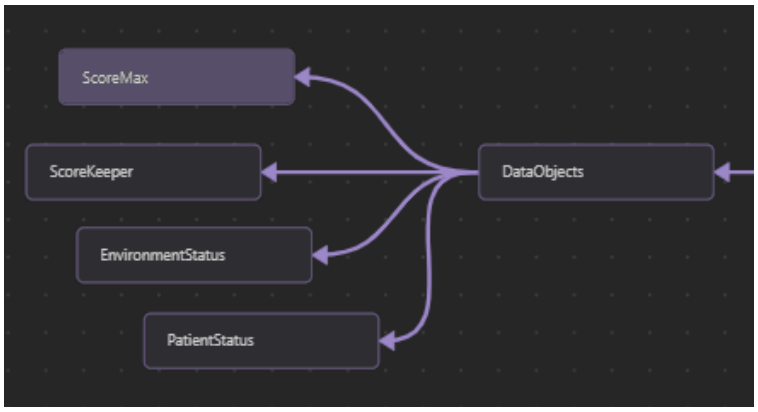

Figure 22: Score Max diagram

7.1.1 Assumptions

The application will be able to access the Score Max data object within the layer to update and/or retrieve.

7.1.2 Responsibilities

Score Max is responsible for store the total scenario point system to be displayed after the user completes each scenario to validate whether the user was able to complete the scenario or had missed any task or objective.

7.1.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem, Score Max, are defined here.

Table 20: Score max subsystem interfaces

ID	Description	Inputs	Outputs
#22	TotalMaxScore	TaskPoint ObjectivePoint	MaxScorePoint

7.2 Score Keeper

Score Keeper is a class within the data object layer to represent the current score of the user have achieved during a specific scenario. This will allow the application to update the object for the user as they interact within a scenario and complete a specific task/objective.

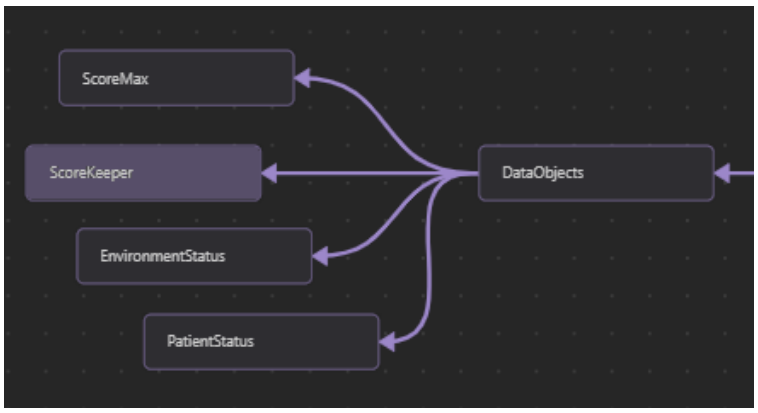

Figure 23: Score keeper diagram

7.2.1 Assumptions

The application will be able to access the Score Keeper data object within the layer to update and/or retrieve.

7.2.2 Responsibilities

Score Keeper is responsible for storing the user's total current point as they navigate through a scenario, handling and validating whether the user succeeded a task/objective to update the user's current point, and be retrieved by the system to display for post-simulation to display the user's point when they complete a scenario.

7.2.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem, Score Keeper, are defined here.

Table 21: Score keeper subsystem interfaces

ID	Description	Inputs	Outputs
#23	CurrentScenarioPoint	TaskPoint ObjectivePoint	CurrentScore

7.3 Environment Status

Environment Status is a class within the data object layer to represent and validate the statuses of the environment objects and entities within each scenario. This will allow the application to update specific objects and entities within a specific scenario that are interacted with by the user to handle tasks and objectives completion as well as validate whether the environment is open and ready to be ran through the scenario for the user.

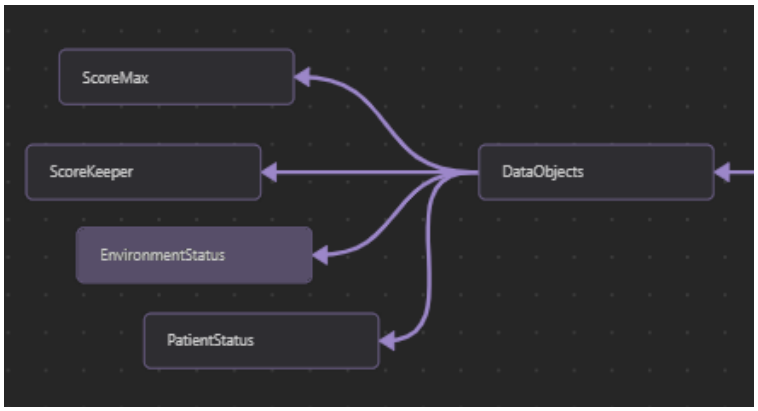

Figure 24: Environment status diagram

7.3.1 Assumptions

Objects and entities are loaded correctly in each scenario and are able to be handled by the user.

7.3.2 Responsibilities

Environment Status is responsible for storing statuses of objects and entities within a scenario, handling and validating whether the object/entity is rendered within the scenario and successfully ran by the application, and retrieve points related to objects and entities within a task or objective are completed.

7.3.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem, Environment Status, are defined here.

Table 22: Environment status subsystem interfaces

ID	Description	Inputs	Outputs
#24	EnvObject	Object Entity	Status
#25	ObjectPoint	Object Entity Task	TaskPoint

7.4 Patient Status

Patient Status is a class within the data object layer to represent and validate the status of the patient within each scenario. This will allow the application to update the patient's status when the user is able to progress in a specific scenario and change the user's sequential tasks to be complete and finish the scenario.

7.4.1 Assumptions

Objects and entities within specific tasks are accessible by patient status for user to complete the sequential task that follows.

7.4.2 Responsibilities

Patient Status is responsible to store the patient's status during a specific scenario, handle and validate patient's status for scenario progression, and retrieve the status of the patient for scenario completion.

Figure 25: Patient status diagram

7.4.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem, Patient Status, are defined here.

Table 23: Patient status subsystem interfaces

ID	Description	Inputs	Outputs
#26	Patient Status	Scenario Timeline Task	Patient Status
#27	Scenario Objects	Patient Status	Object Status

REFERENCES