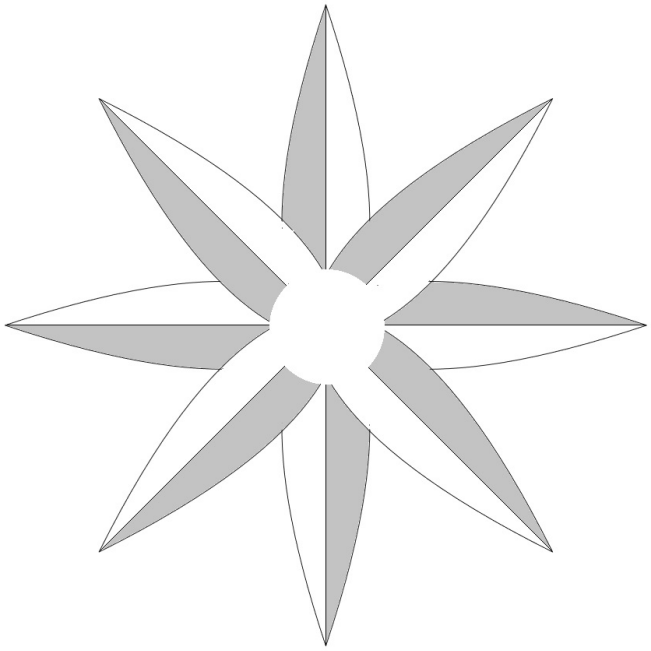
Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Summer 2023



Pastafarian Coders
VR Palliative Care

Aman Hogan-Bailey
Andrew Mondejar
Charles Pham
Mark Holcomb
Yusuf Cavus

Revision History

Revision	Date	Author(s)	Description
0.1	07.24.2023	AHB , AM , CP , MH , YC	document creation
1.1	08.05.2023	AHB , MH	Official SD1 version created

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

The VR Palliative Care system is intended to be an immersive nursing simulator that runs on the Unity using virtual reality technology. Users of the VR Palliative Care system will be able to engage with simulated patients based on four case studies provided by the University of Texas at Arlington Nursing department.

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 students 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 consists of three sections (layers): "Managers," "Game Objects," and "Data Objects." The managers layer controls how the game operates. The Game Objects layer displays the types of game objects that will be used in the game. The Data Objects layer shows how the data will be collected and stored. Each layer interacts with each other by sending messages/using each other to run the game.

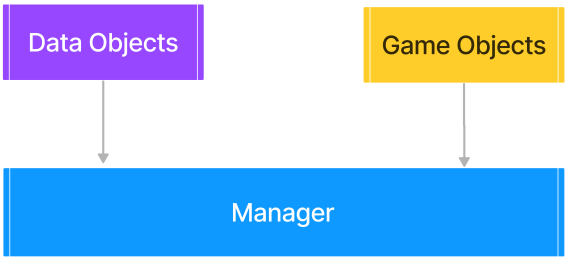


Figure 1: A simple architectural layer diagram

2.1 Manager Layer

The Manager Layer serves a pivotal role in overseeing the systems that initiate and execute the game's real-time logic. Its primary function is to segregate various functionalities into their distinct roles. The Manager Layer holds control over several domains, such as scene, patient, event, audio, player, and scenario data management. This layer interacts seamlessly with all other layers to ensure continuous visual and functional user feedback during the application's usage. It is the orchestrator that harmonizes various components of the game, ensuring smooth, uninterrupted interaction for the user throughout their time within the application.

2.2 Game Object Layer

The Game Object Layer defines the interactions of game objects within the virtual world, both with other objects and various systems. Mainly, its inputs are sourced from the Manager Layer or other game objects. Depending on the requirements of the Manager Layer, the Game Object Layer may provide structures from the Data Object Layer. It's vital that this layer can interface with global constructs, such as the haptic feedback mechanisms on VR controllers.

2.3 Data Object Layer

The Data Object Layer functions as a mediator, facilitating the communication between the application and the database to manage various data objects and entities. These may include scores, the environment, and patient details for each user's scenario play-through. When a user interacts with a specific object or entity in a given scenario, the Data Object Layer is primed to retrieve and update that object/entity, thereby influencing the scenario's progression. As such, the primary inputs for this layer are the individual objects/entities and tasks tied to various classes within this layer.

3 Subsystem Definitions & Data Flow

The following is the class diagram for the project.

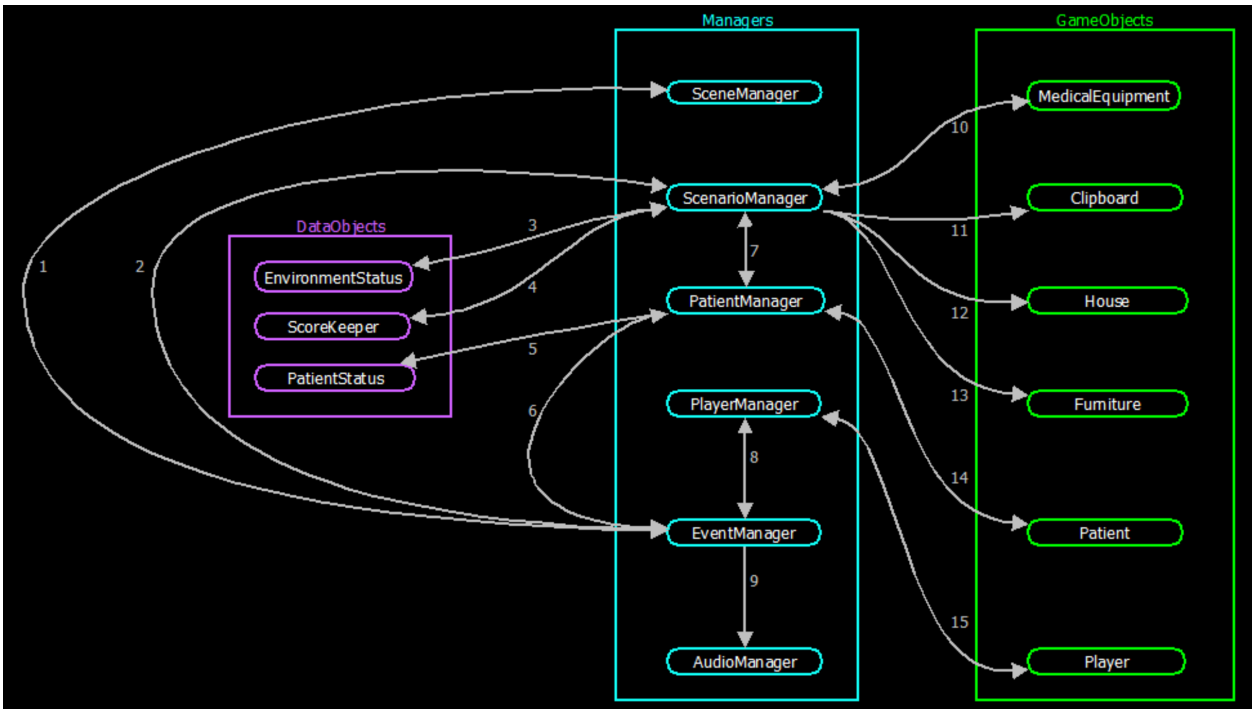


Figure 2: Class Diagram

4 Manager Layer

This section describes the Managers layer of the project. In this section, managers which handle the flow of logic in the game project are illustrated and described.

4.1 Scene Manager

The scene manager's purpose within the game project to handle changing and saving 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).

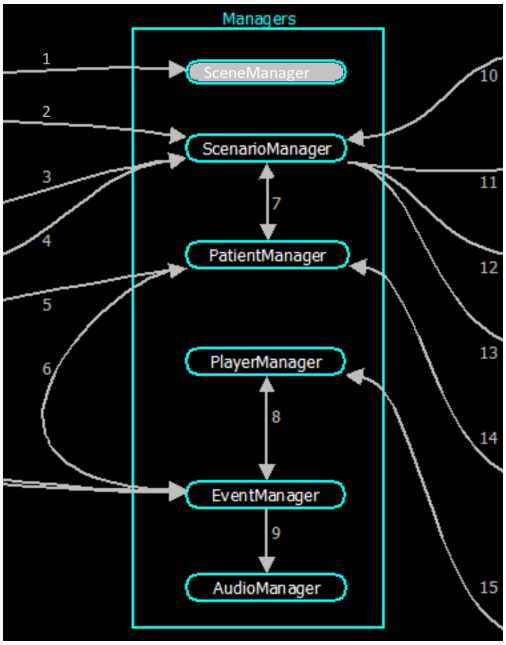


Figure 3: Scene manager

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 interfaces

ID	Description	Inputs	Outputs
#1	Load scene	scene number	Opens scene

4.2 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.

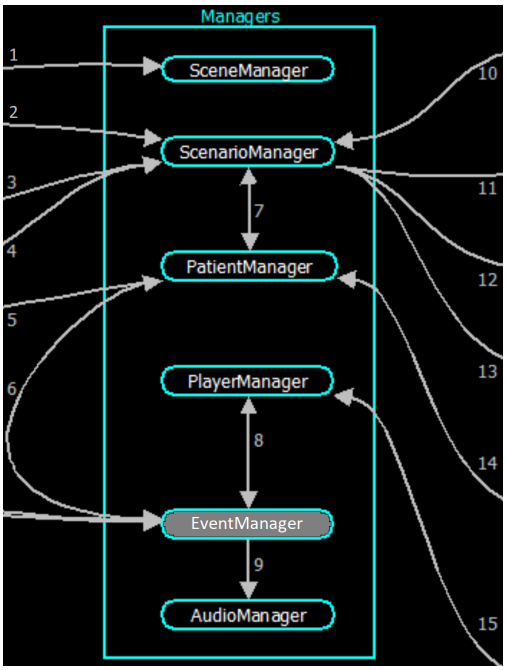


Figure 4: Event Manager

4.2.1 Assumptions

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

4.2.2 Responsibilities

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

4.2.3 Subsystem Interfaces

Table 3: Event Manager interfaces

ID	Description	Inputs	Outputs
#1	Scene Manager Event trigger	Event ID Scene ID	Messages to subscribed game objects
#2	Scenario Manager Event trigger	Event ID	Messages to subscribed game objects
#6	Patient Manager Event trigger	Event ID	Messages to subscribed game objects
#8	Player manager Event trigger	Event ID	Messages to subscribed game objects

4.3 Audio Manager

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

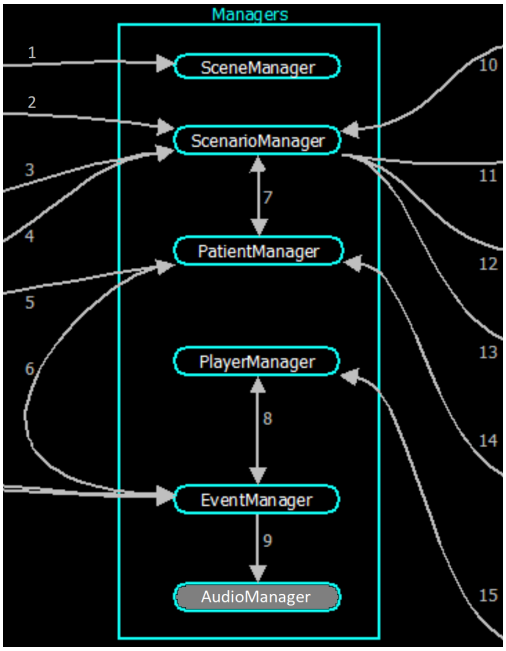


Figure 5: Audio Manager

4.3.1 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.

4.3.2 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.

4.3.3 Subsystem Interfaces

Table 4: Audio Manager interfaces

ID	Description	Inputs	Outputs
#9	Sound interface	Event ID	Audio source location

4.4 Scenario Manager

The scenario manager holds data that pertains to the spawn locations of various objects and characters, as well as quest objectives.

4.4.1 Assumptions

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

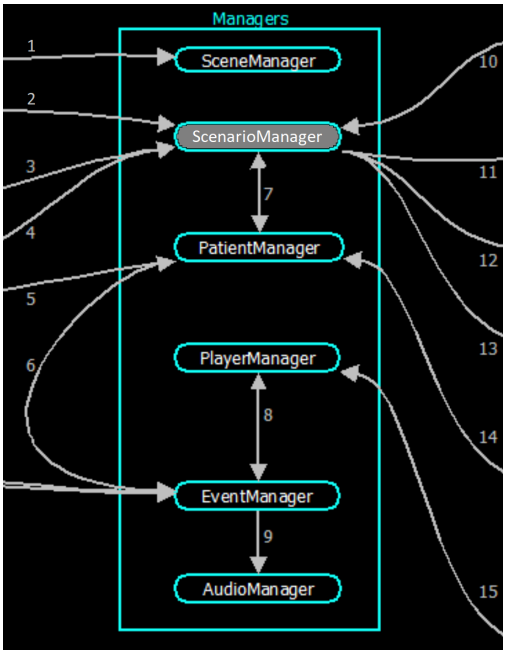


Figure 6: Scenario Manager

4.4.2 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.

4.4.3 Subsystem Interfaces

Table 5: Scenario Manager interfaces

ID	Description	Inputs	Outputs
#2	Event manager notification	Event ID	Scenario state change
#3	Environment status updates	Environment	status value
#4	Score updates and reporting	Score results	Task completion ID
#7	Patient	Patient status	Event notification
#10	Medical Equipment status	Position/ orientation/ state	Action event
#11	Clipboard interaction	N/A	Text
#12	House	N/A	State
#13	Furniture	N/A	State

4.5 Patient Manager

The patient 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.

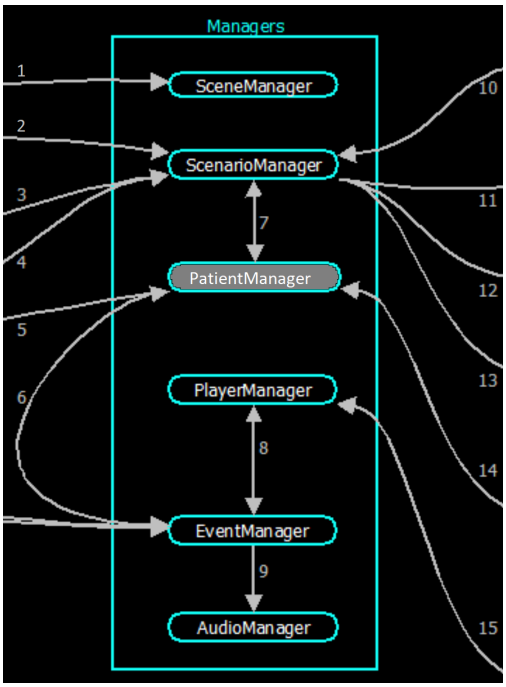


Figure 7: Patient Manager

4.5.1 Assumptions

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

4.5.2 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.

4.5.3 Subsystem Interfaces

Table 6: Patient Manager interfaces

ID	Description	Inputs	Outputs
#5	Patient Status	N/A	Health/condition data
#6	Patient event	Patient event	Health/condition data
#7	Scenario patient interaction	Scenario data Start state data	Current patient status
#14	Patient Object	Scenario data Start state data	Health/condition data

4.6 Player Manager

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

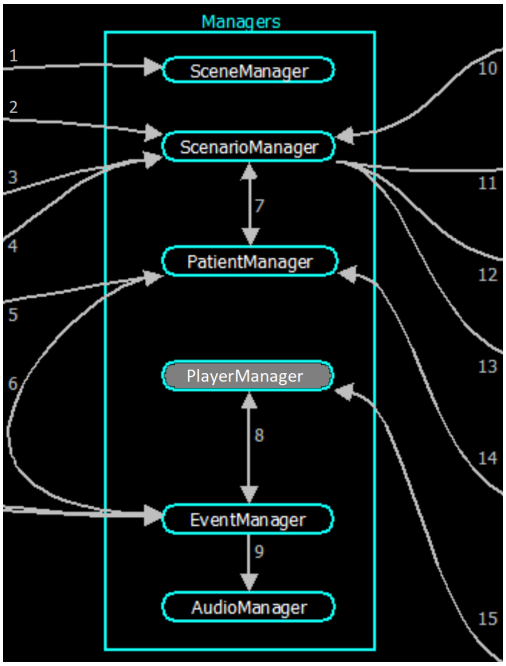


Figure 8: Player Manager

4.6.1 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.

4.6.2 Responsibilities

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

4.6.3 Subsystem Interfaces

Table 7: Player Manager interfaces

ID	Description	Inputs	Outputs
#8	Headset interface	Tracking data Controller input	3D movement
#15	Player event	Event data	Player status

5 Game Object Layer

The game object layer handles the appearance and behavior of objects throughout the game.

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.

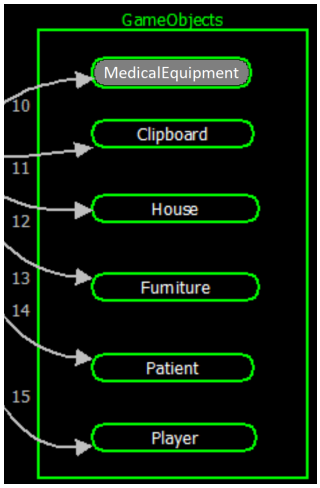


Figure 9: Medical Equipment

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 8: Medical Equipment interfaces

ID	Description	Inputs	Outputs
#10	Scenario medical equipment	scenario state data	Medical equipment status

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 scenario manager for scenario selection.

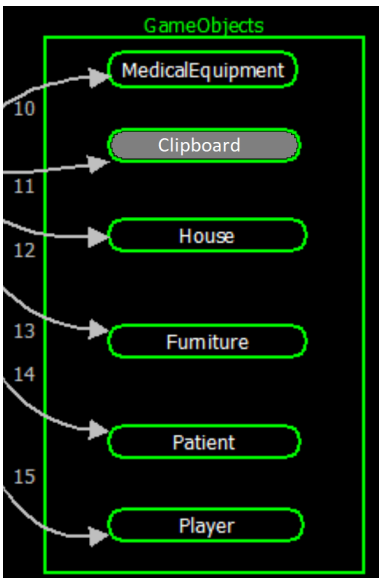


Figure 10: Clipboard

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 layer to allow the changing of scenarios as well as other meta operations

Table 9: Clipboard interfaces

ID	Description	Inputs	Outputs
#11	Metagame options	N/A	Change scenario, 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.

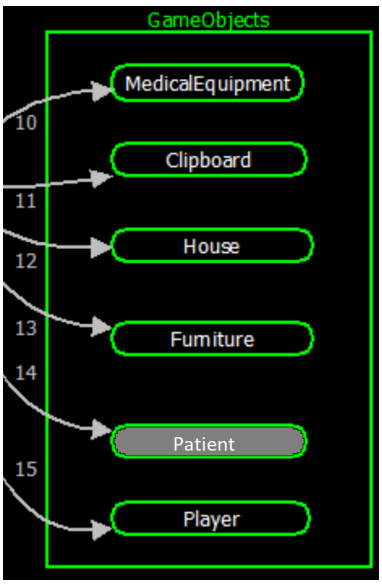


Figure 11: Patient

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 10: Patient interfaces

ID	Description	Inputs	Outputs
#14	Patient Manager	Patient events	Patient state

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.

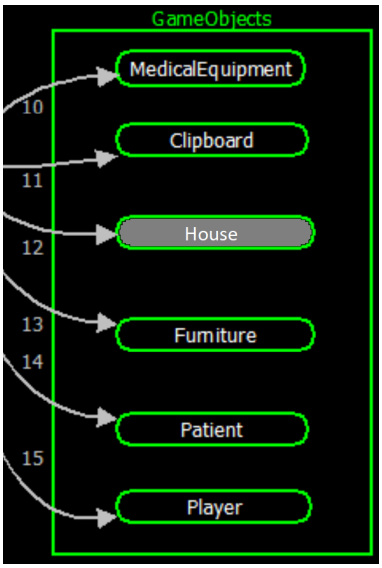


Figure 12: House

5.4.1 Assumptions

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

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 11: House interfaces

ID	Description	Inputs	Outputs
#1	Scenario house	Scenario state	House state

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.

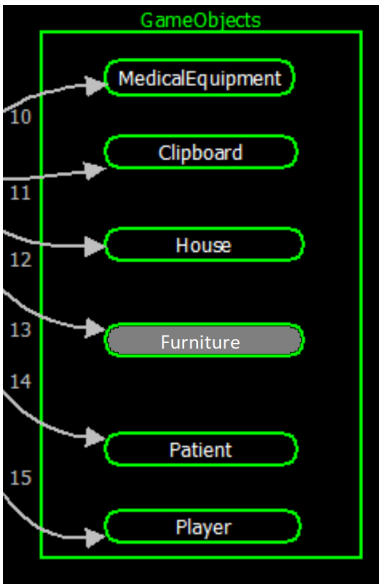


Figure 13: Furniture

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).

Table 12: Furniture interfaces

ID	Description	Inputs	Outputs
#13	Scenario furniture	Furniture event	Furniture state

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.

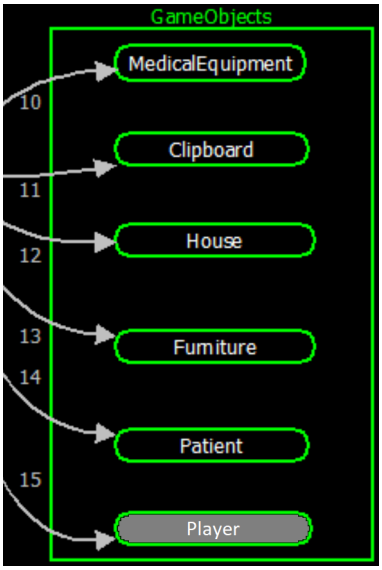


Figure 14: Player

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 13: Player interfaces

ID	Description	Inputs	Outputs
#15	Player Manager	user position	User events

6 Data Object Layer

The Data Object Layer mediates communication between the application and the database, managing data objects like scores, environment, and patient details. It updates these entities based on user interactions, thereby influencing scenario progression. Its key inputs are individual objects/entities and related tasks.

6.1 Score Keeper

Score Keeper, a class in the Data Object Layer, reflects the user's current score in a particular scenario. It enables the application to update the score as the user interacts with the scenario and accomplishes tasks or objectives.

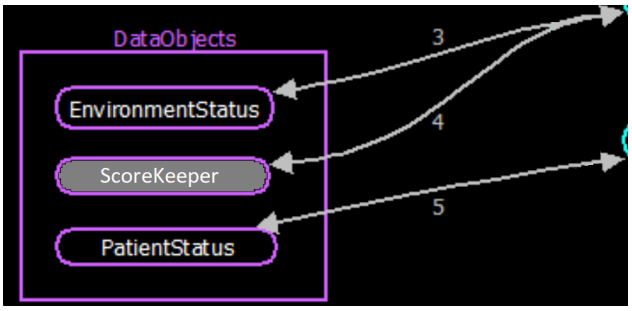


Figure 15: Score Keeper

6.1.1 Assumptions

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

6.1.2 Responsibilities

Score Keeper's role is to track and store the user's ongoing point total as they progress through a scenario. It handles the validation and updating of points when the user achieves a task or objective. Post-simulation, the system retrieves this data to display the user's final score upon scenario completion.

6.1.3 Subsystem Interfaces

Table 14: Score Keeper interfaces

ID	Description	Inputs	Outputs
#4	CurrentScenarioPoint	TaskPoint ObjectivePoint	CurrentScore

6.2 Environment Status

Environment Status, a class within the Data Object Layer, signifies and validates the states of environmental objects and entities within each scenario. It facilitates the application to update particular objects and entities within a scenario based on user interaction. It handles task and objective completion, and confirms whether the environment is prepared and available for the user to navigate the scenario.

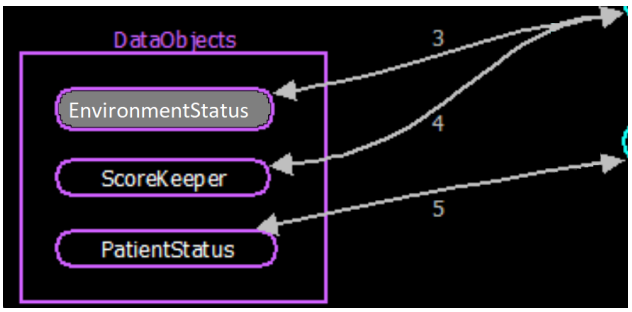


Figure 16: Environment Status

6.2.1 Assumptions

In every scenario, objects and entities are properly loaded and are able to be handled by the user.

6.2.2 Responsibilities

Environment Status is responsible for storing the states of objects and entities within a scenario. It manages and verifies whether these components are accurately rendered within the scenario and successfully operated by the application. Additionally, it retrieves points pertinent to objects and entities once a task or objective is completed.

6.2.3 Subsystem Interfaces

Table 15: Environment Status interfaces

ID	Description	Inputs	Outputs
#3	EnvObject	Object Entity Task	Status TaskPoint

6.3 Patient Status

Patient Status, a class within the Data Object Layer, signifies and verifies the condition of the patient within each scenario. It enables the application to update the patient's status in response to user progression within a specific scenario, subsequently altering the user's subsequent tasks to mark scenario completion.

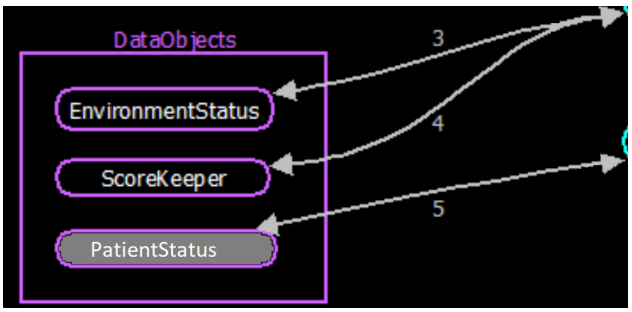


Figure 17: Patient Status

6.3.1 Assumptions

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

6.3.2 Responsibilities

Patient Status holds the responsibility of storing the patient's status during a specific scenario, managing and validating the patient's status to facilitate scenario progression, and retrieving the patient's status to denote the scenario's completion.

6.3.3 Subsystem Interfaces

Table 16: Patient Status interfaces

ID	Description	Inputs	Outputs
#5	Patient Status	Scenario Timeline Task	Patient Status

REFERENCES