****

**COLD NITES**

Alpha-1

**Menu Interface Design Document**

Designed & Implemented by

Vrund Soni

**Change Log:**

**Version –** 0.1

**Name –** Vrund Soni

**Date –** 15th March 2021

**Description –** Added Introduction, Design Goals and System Overview

**Version –** 0.4

**Name –** Vrund Soni

**Date –** 16th March 2021

**Description –** Added Logical View

**Version –** 0.7

**Name –** Vrund Soni

**Date –** 16th March 2021

**Description –** Added Process View and Use Case View

**Version –** 1.2

**Name –** Vrund Soni

**Date –** 16th March 2021

**Description –** Modified Process view B.

**Table of Contents**

1. **Introduction 4**
2. **Design Goals 5**
3. **System Overview and Behaviour 6**
4. **Logical View 7**

**A. High-Level Design Architecture of the Entire System**

**B. Mid-Level Design of Menu Interface**

**C. Detailed Design of Menu Interface**

1. **Process View 10**

**A. Process view of cycling thorough 3D MenuButtons**

**B.** **Process view of toggling PauseMenu**

**C.** **Process view of toggling ResOptionsMenu**

1. **Use Case View 14**
2. **Getting all the 3D MenuButtons**
3. **Setting up all the 3D MenuButtons**

**1. Introduction**

Cold Nites is a Grid-style turn-based game. The player must strategically navigate through the level to survive the cold night, protecting the boy from all the mischievous elements of the city. And, there are always multiple ways to solve the puzzles along the way.

This design module will focus on the construction(architecture) and implementation of the Menu Interface (Main menu and Pause menu) in the game.

This document will describe the architecture and design choices that make Menu Interface’s implementation easy to use, understand and reusable for all the fellow programmers, artists and level designers, and major stakeholders.

Below are interest points for the mentioned parties:

**Programmers** – Menu Interface is designed with the intention of easy to work on by any new programmer. The programmer will not have to worry if the number of 3D buttons in the 3dMenu level is changed by any designers.

**Artist/Designers** – Menu Interface allows the artist to place the 3D Menu buttons in the level without worrying about their proper placement in the level. Further, the Interface also allows the designers to use various functions inside the blueprints to get the working of both menu’s easily.

**Project Manager (and the Team)** - All the tasks during the group meetings were assigned with everyone's and the Project Manager's agreement. The Menu Interface design module and the code implementation will address all the concerns and will fulfill all the requirements in the game's and team's best interest.

**2. Design Goals**

The design priorities for the Menu Interface are mentioned below:

* The design should minimize the complexity of the main menu and pause menu in the system.
* The design should allow the level designer/artists to use any number of 3D buttons for the main menu without any difficulty.

**3. System Overview and Behaviour**

The Menu Interface should allow the player to access various options available throughout the game using MainMenu and PauseMenu.

The MainMenu will allow the player to access different options such as Starting a new game, changing the screen's resolution, or quitting the game using 3D interactable widgets/buttons.

The Pause Menu will allow the player various options such as save game and load game through 2D buttons while the game is in the paused state.

The working of both the menus will be handled by two different GameMode and PlayerController classes.

**4. Logical View**

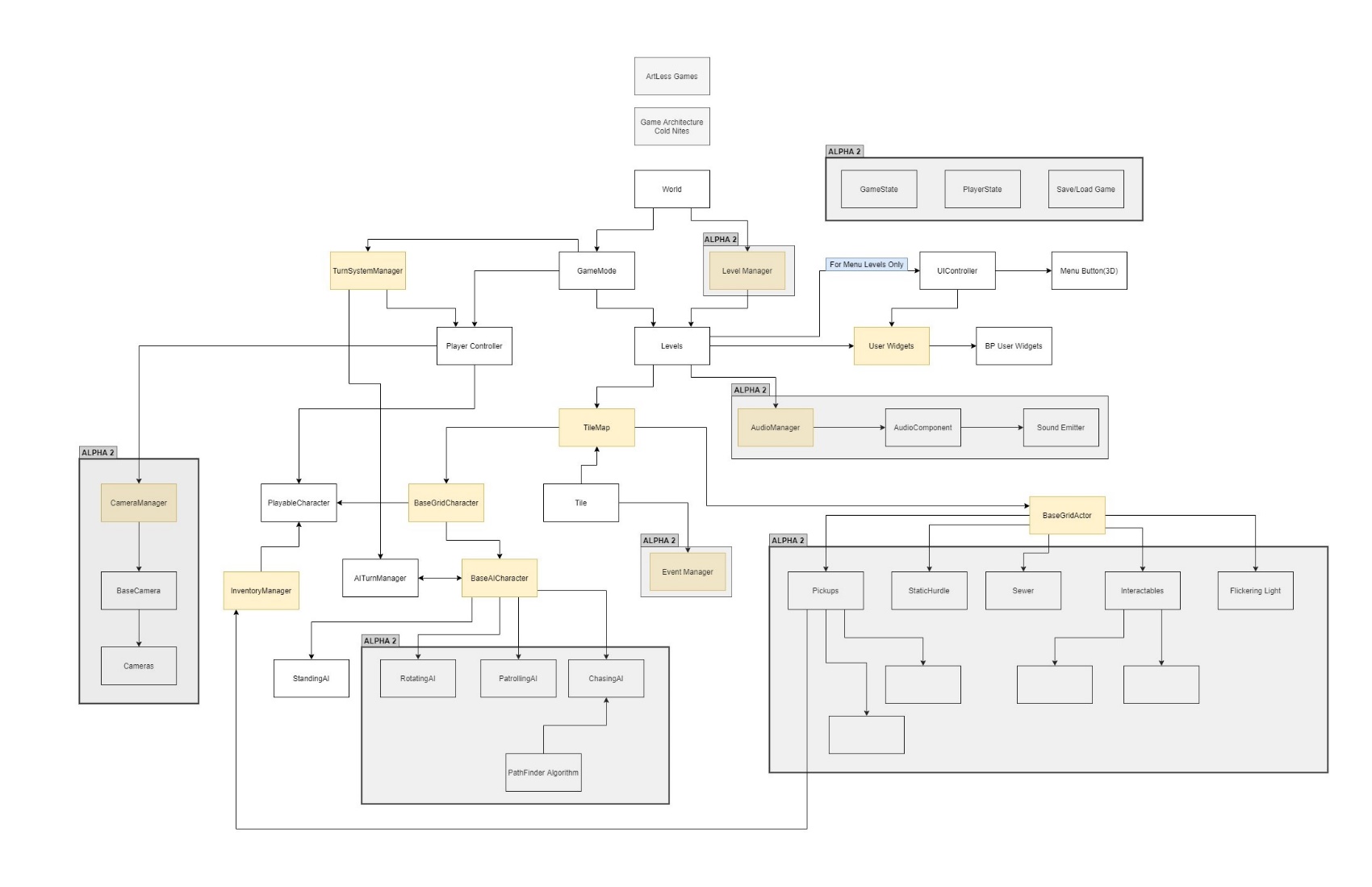
The logical view describes the high-level architecture for the entire game from all the core classes to high-level relations and interactions between them with a flow chart making it easy to read and understand.

Later, it will dive deep into the high-level and detailed design for the Menu Interface Module, using a UML Class Diagram.

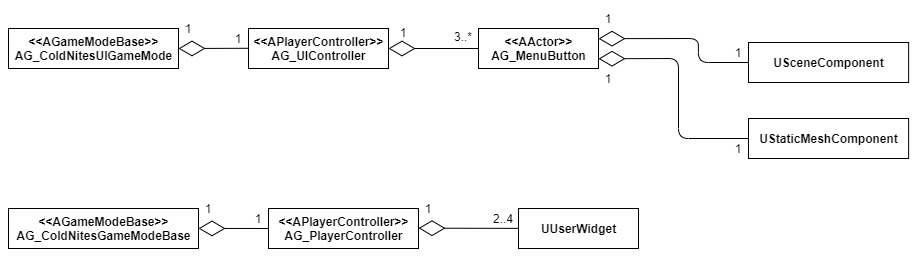
**A. High-Level Design Architecture of the Entire System**

The primary features for the Alpha 1 release:

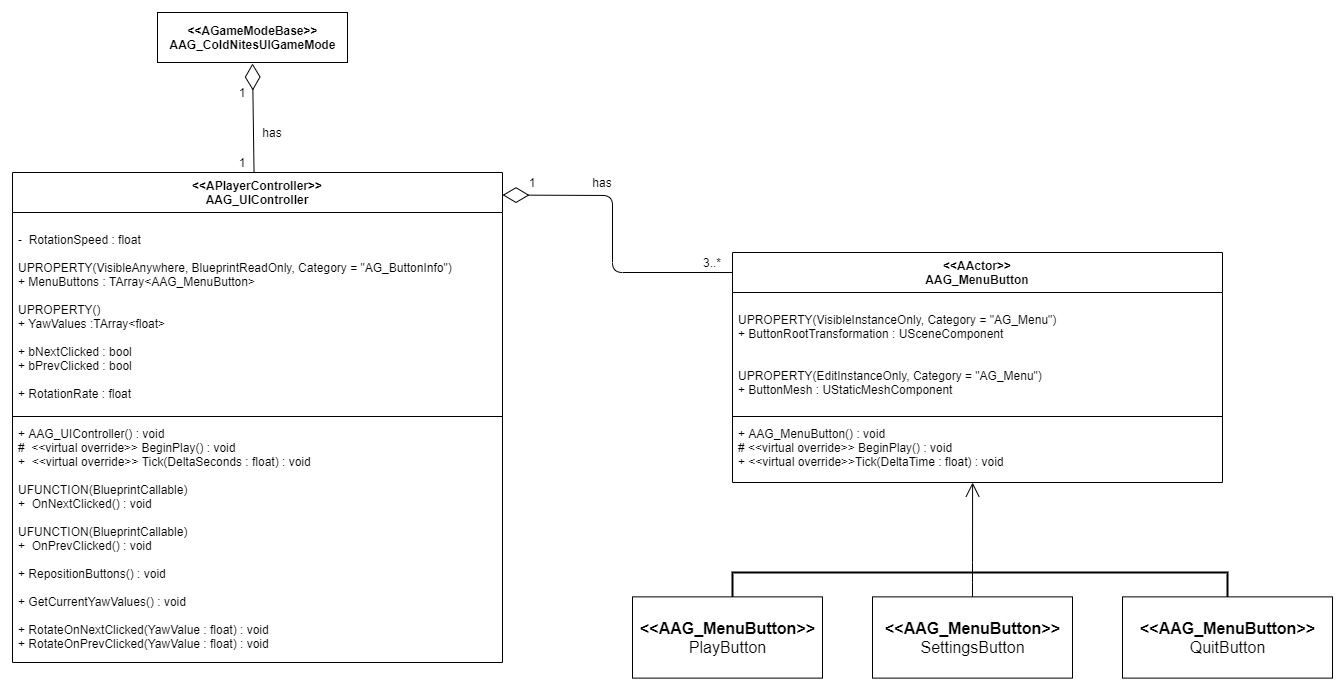
1. **TileMap** - TileMap provides the grid-based behaviour for the game and will facilitate the event system, based on the actor present on the Tiles.
2. **Turn-Based System** - This provides the turn-based aspect for the game. It is responsible for maintaining the turn order for all the world elements(actors) and the player.
3. **Base Grid Classes** - These classes work as a foundation class for all the actors/characters spawned in the game. These classes are closely integrated with handling the TileMap(Grid-Base) Behaviour of the game.
4. **Player Character** - Player is a controllable character that inherits from BaseGridCharacter, which takes user inputs to perform appropriate moves.
5. **Inventory System** - The pickup function helps the player grab the items on the map. Inventory stores the items for the corresponding actor and will allow the player easy access to any collectible throughout the game, and it also assists in equipping the stored items.
6. **Menu Interface** - The Menu Interface will be responsible for Main Menu and Pause Menu with which the player can interact.

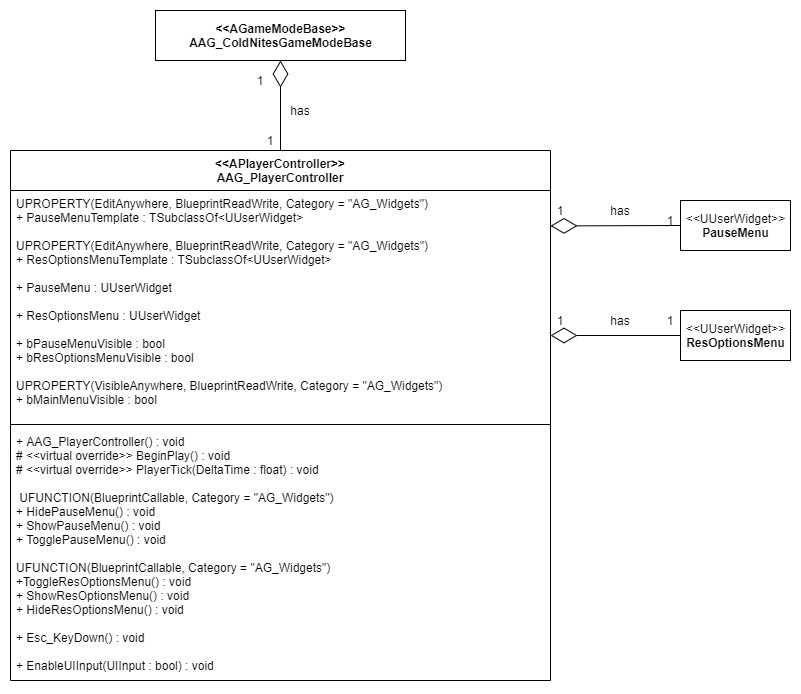


**B. Mid-Level Design of Menu Interface**



**C. Detailed Design of Menu Interface**

****

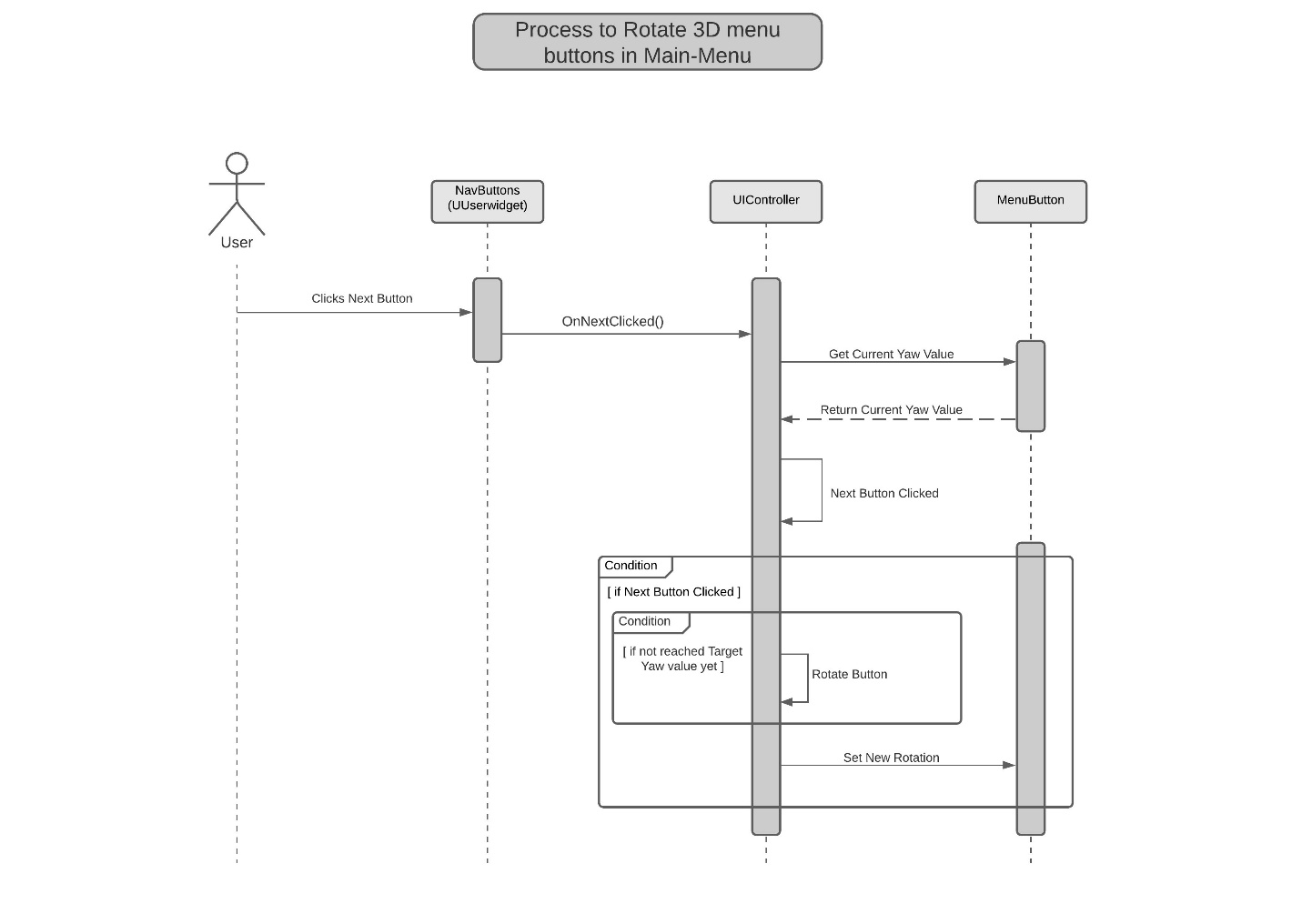
****

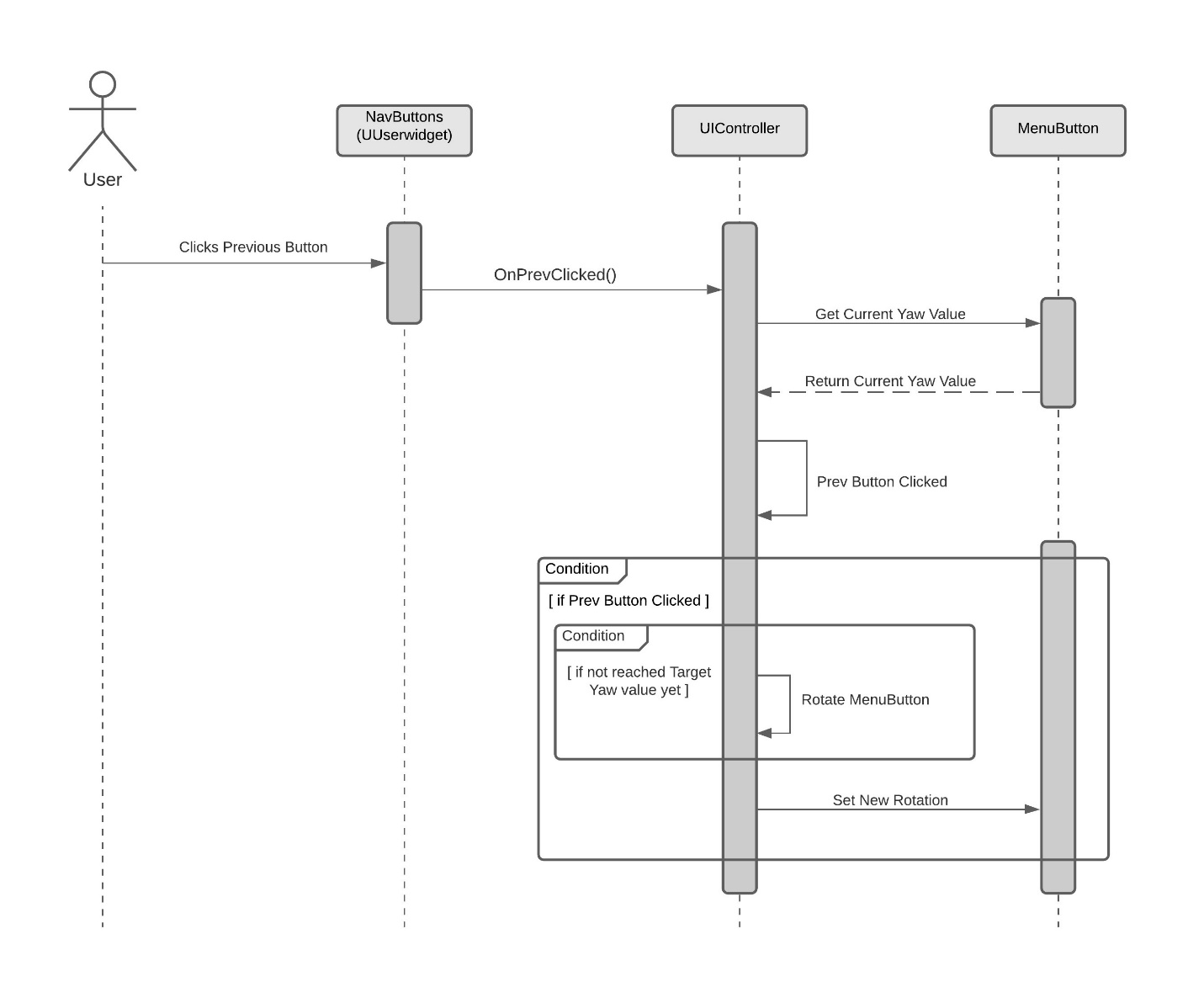
**5. Process View**

The process view will explain the relation and interaction between various cases using Sequence and Collaboration Diagrams.

**A. Process view of cycling thorough 3D MenuButtons:**

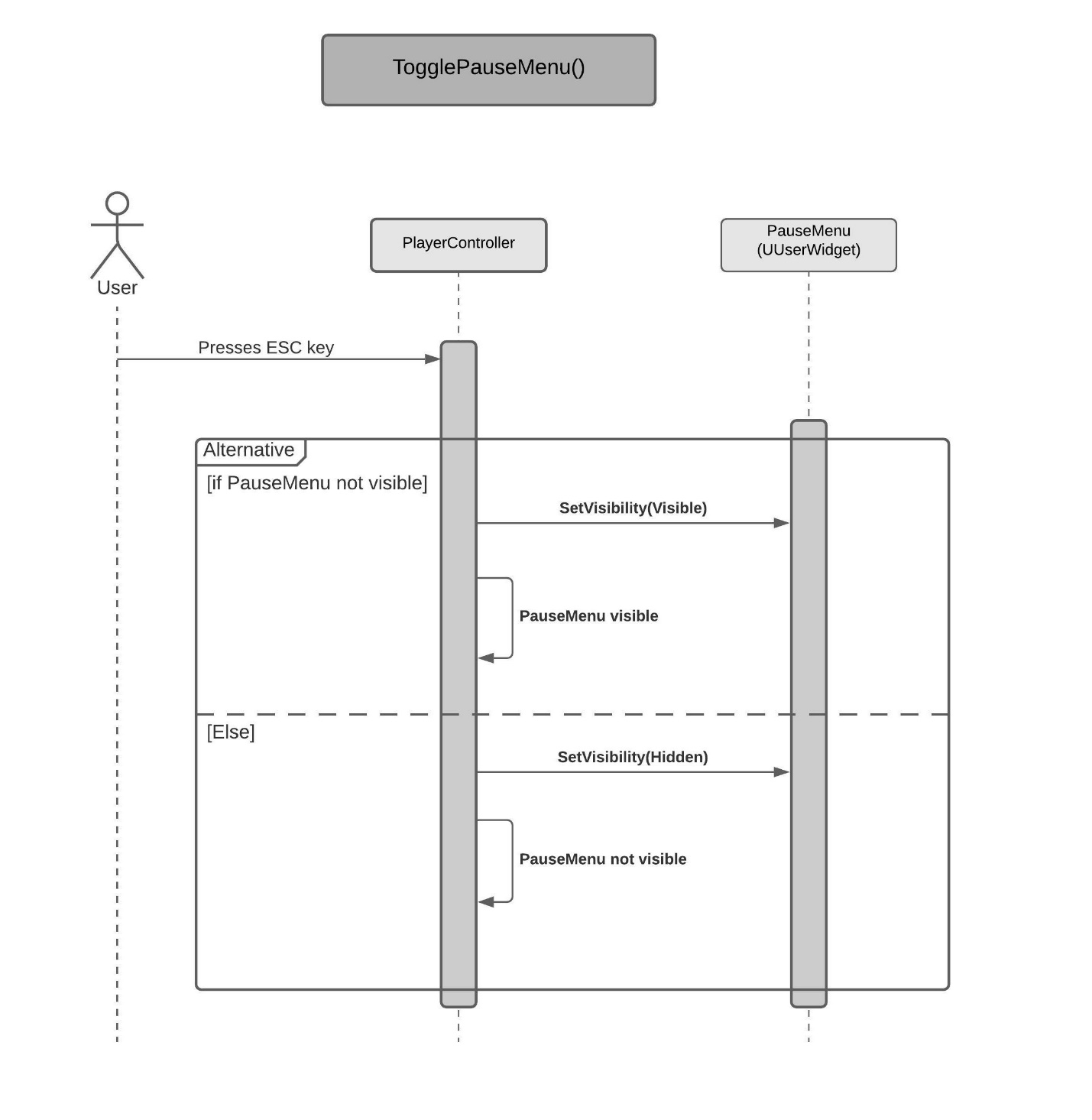
When the game starts, a UserWidget consisting of two image buttons is created, allowing the user to cycle through the 3D menu button. When the user presses any of those two buttons, the respective method in the UIController is called. The method subsequently gets the Current Yaw Values of all the buttons to set a new Target Yaw Value to rotate them. After getting the Current yaw values, the UIController sets a variable indicating a button is pressed. The UIController subsequently starts rotating the buttons until they reach the Target Yaw Value that was set in the past.



****

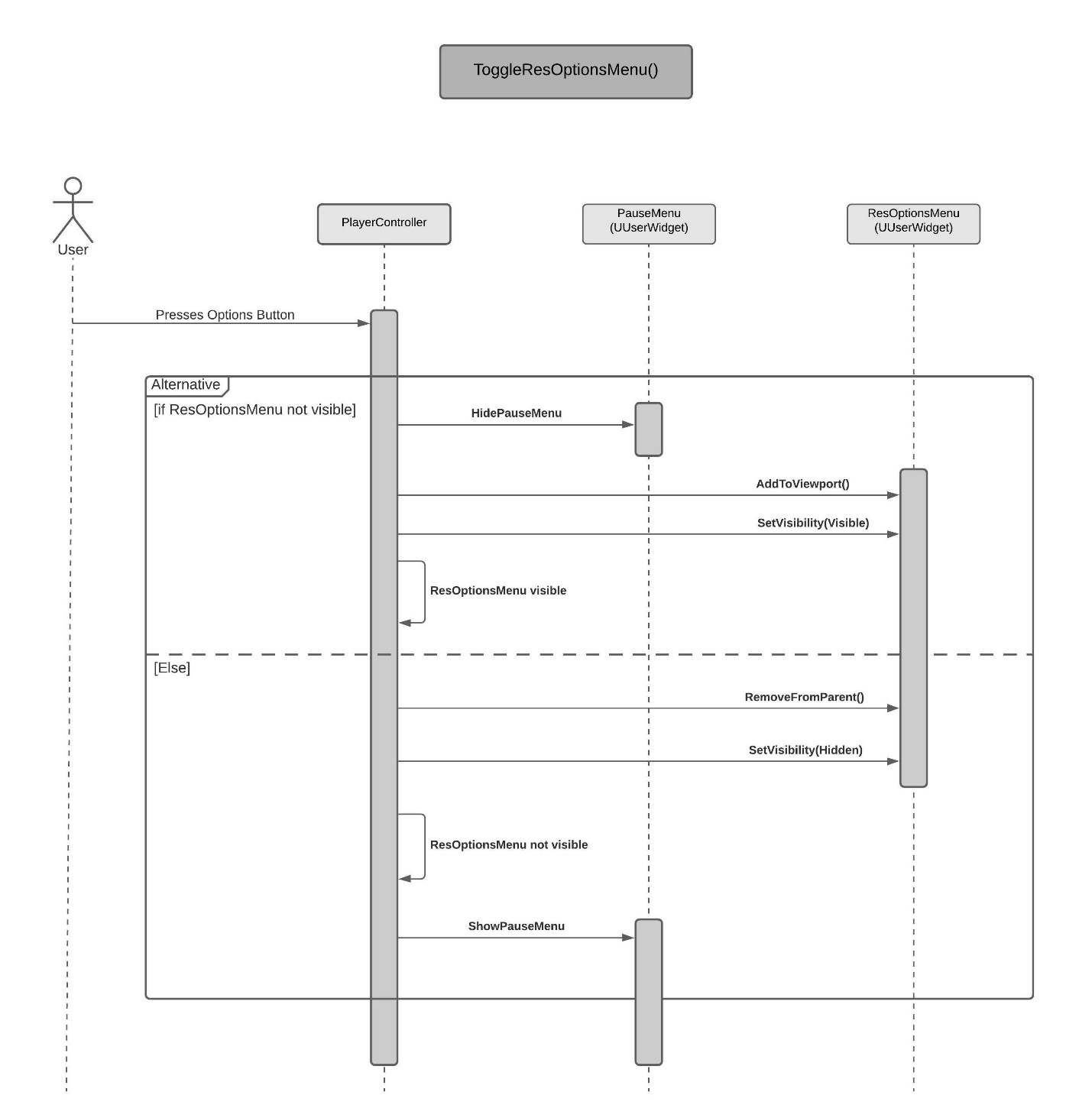
**B. Process view of toggling PauseMenu:**

The PlayerController can toggle the PauseMenuWidget by using the TogglePauseMenu() method. When the PlayerController gets an input from the user to display the PauseMenu, it firsts checks if it is already not on screen. If not, then it will set the PauseMenu widget's visibility to visible. If the widget is already on the screen, then it will hide it by simply setting its visibility to hidden.

****

**C. Process view of toggling ResolutionOptionsMenu:**

The PlayerController can display the ResOptionsMenu widget using the ToggleResOptionsMenu(). The method will check if the widget is not already on the screen. If not, that means the PauseMenu widget is on the screen. So, the method will first hide the PauseMenu widget by setting its visibility to hidden. Then it will add the ResOptionsMenu widget to the viewport and set its visibility to visible. On the flip side, if the ResOptionsMenu widget is already on the screen, then the method will remove it from the viewport and set its visibility to hidden. Finally, it will set the PauseMenuWidget's visibility to visible to get it back on the screen.



**6. Use Case View**

The use case will focus on showing the uses of Menu Interface at different stages in the game and will explain its application so that the Menu Interface module can act as a guide/reference for someone not quite familiar with the Menu Interface codebase.

1. **Getting all the 3D MenuButtons present in the level:**

* In the code, any class that need to deal with the menu Buttons in the level can easily get all the buttons using:

*TArray<AActor\*> ButtonActors;*

*UGameplayStatics::GetAllActorsOfClass(this, AAG\_MenuButton::StaticClass(), ButtonActors);*

* To work on the MenuButtons, first cast the actors in the ButtonActors array to AAG\_MenuButton and store them in a different array using the following lines.

*if(ButtonActors.Num() > 0)*

*{*

*for (int i = 0; i < ButtonActors.Num(); i++)*

*{*

*MenuButtons.Insert(Cast<AAG\_MenuButton>(ButtonActors[i]), i);*

*}*

*}*

1. **Setting up all the 3D MenuButtons in the level when the game starts:**

* The MenuButtons can be adjusted in a proper circular pattern during the start of the gameplay by calling the following function in the BeginPlay():

*void AAG\_UIController::RepositionButtons()*

*{*

*RotationRate = 360.0f / MenuButtons.Num();*

*float Angle = 0.f;*

*if(MenuButtons.Num() > 0)*

*{*

*for(int i = 0; i < MenuButtons.Num(); i++)*

*{*

*MenuButtons[i]->SetActorLocation(FVector(0.f));*

*MenuButtons[i]->SetActorRotation(FRotator(0.f, Angle, 0.f));*

*Angle += RotationRate;*

*}*

*}*

*}*

Calling this function will make sure that all the menu buttons are arranged in

proper circular manner nonetheless how they are placedinside the level in the editor.