Post Arcana / Menu

**Table of contents**:

1. – Introduction................................................2
2. – Design Goals..............................................2
3. – System Behavior.........................................2
4. – Logical View...............................................3
   1. – High-Level Design (Architecture)........3
   2. – Mid-Level Design.................................4
   3. – Detailed Class Design...........................5
5. – Process View................................................6
6. – Use Case View.............................................7

**Change History**

**Version:** 0.1

**Modifier:** Danylo Cardoso

**Date:** 03/13/2022

**Changes:** Document created

**Version:** 0.2

**Modifier:** Danylo Cardoso

**Date:** 03/15/2022

**Changes:** Mid-level design and process view added

**Version:** 0.3

**Modifier:** Danylo Cardoso

**Date:** 03/16/2022

**Changes:** High-Level design added

**Version:** 0.4

**Modifier:** Danylo Cardoso

**Date:** 03/17/2022

**Changes:** System architecture added

# **1 Introduction**

This document describes the design and architecture of Post Arcana by the Hex Decimals. Post Arcana is a single player open world RPG, set in a post apocalyptic small Canadian town after the introduction of magic to the real world.

The purpose of this document is to define the architecture and design of menu systemin a manner that assists the interests of all major stakeholders. The major stakeholders and their interests are as follows:

* Developers: A design that is easy to implement that minimizes complexity
* Project Manager: A design that can be easily divided among the skills of the programmers
* Maintenance Programmers: A design that can be improved upon easily

# 

# **2 Design Goals**

The design priorities for the Menu system are:

• The design should minimize complexity and development effort.

• The design should give developers access to player stats and

experience level, being able to view and modify them.

# **3 System Behavior**

The Menu module inherits from UUserWidget. It displays a window that shows the player’s stats, current xp value, level and eventually inventory. The player’s stats can be modified through the menu. It can be toggled on and off with the press of a key, and when active it overrides the player’s inputs and pauses the game.

# **4 Logical View**

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section, the modules of the system are first expressed in terms of high-level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

# **4.1 High-Level Design**

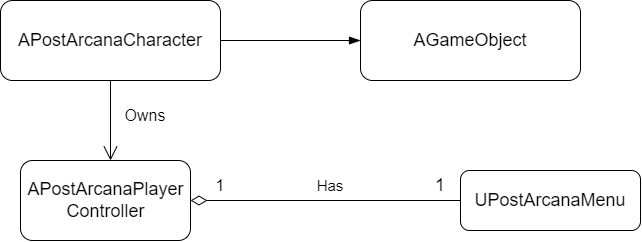
**Diagram

Description automatically generated**

# 

* GameObject: Base class for the game entities (Players and enemies). Stores abstract and concrete stats.
* Enemy: The class used for enemy-only stats and components
* AI Controller: Moves and controls the enemies
* Player: The class used for player-only stats and components
* Player Controller: Receives player input and controls the player
* Projectile: Is launched by the player, and deals damage to Game Objects
* Menu: Places a UI element on the screen given a specific input
* HUD: Shows the player important stats

# **4.2 Mid-Level Design**



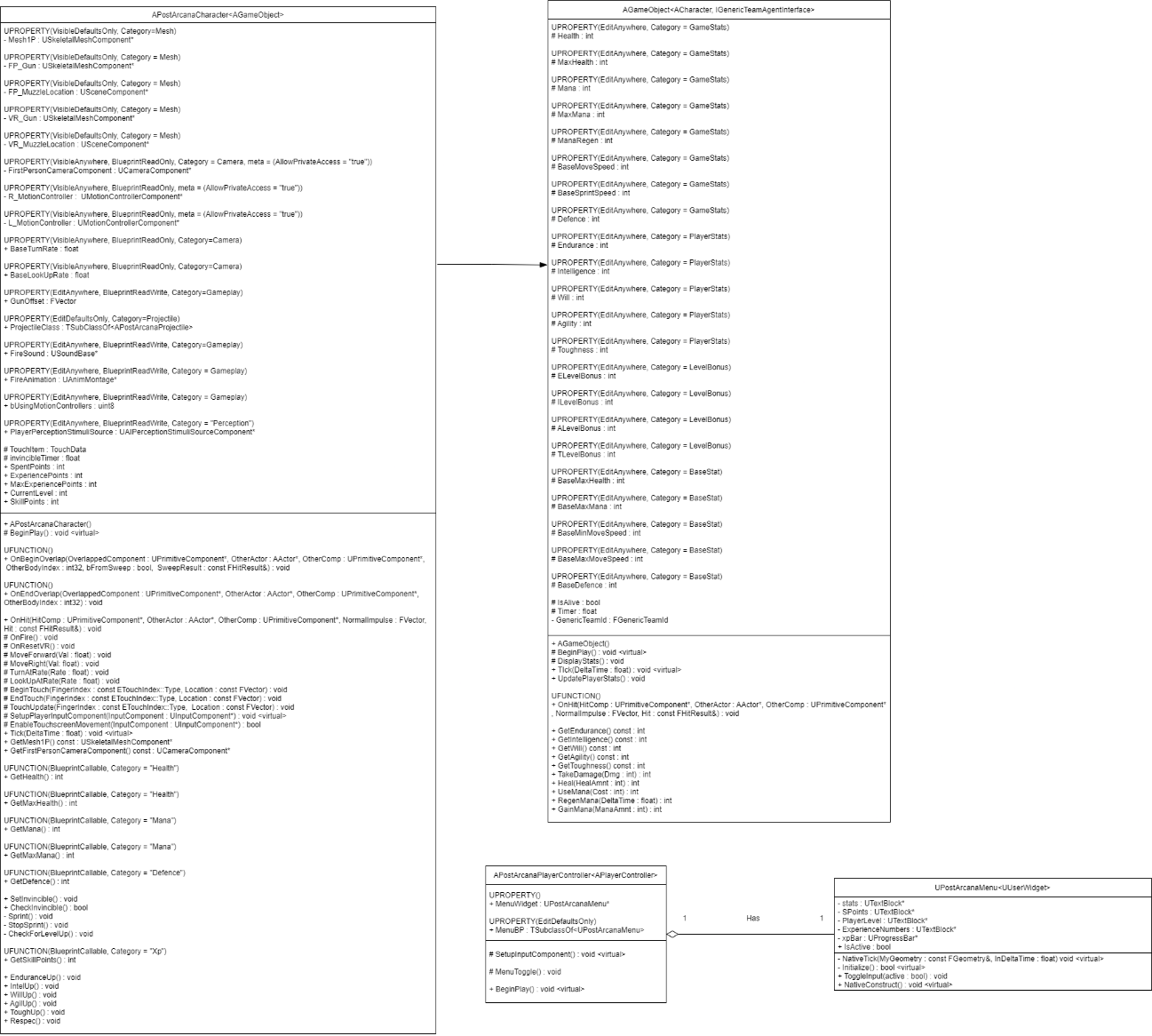
**UPostArcanaMenu:** User widget that displays the character’s stats in a window and allows them to be modified.

**APostArcanaCharacter:** Game object controlled by the player through inputs. While the menu widget is being displayed the player’s input is disabled except for mouse inputs.

**APostArcanaPlayerController:** Contains a menu variable and is owned by the Player. Handles the input to open and close the menu window.

**AGameObject:** A character with health, stats, and a level. APostArcanaCharacter inherits from it.

# **4.3 Detailed Class Design of the Menu Module**



# 

# **5 Process View of the Menu Module**

Toggling the menu window

# 

# When the menu key is pressed the player controller will invert the menu widget’s current state, and then will call the appropriate methods based on said state. When enabled the menu will disable the player’s input, and when disabled the input will be returned to the player.

# 

# **6 Use Case View**

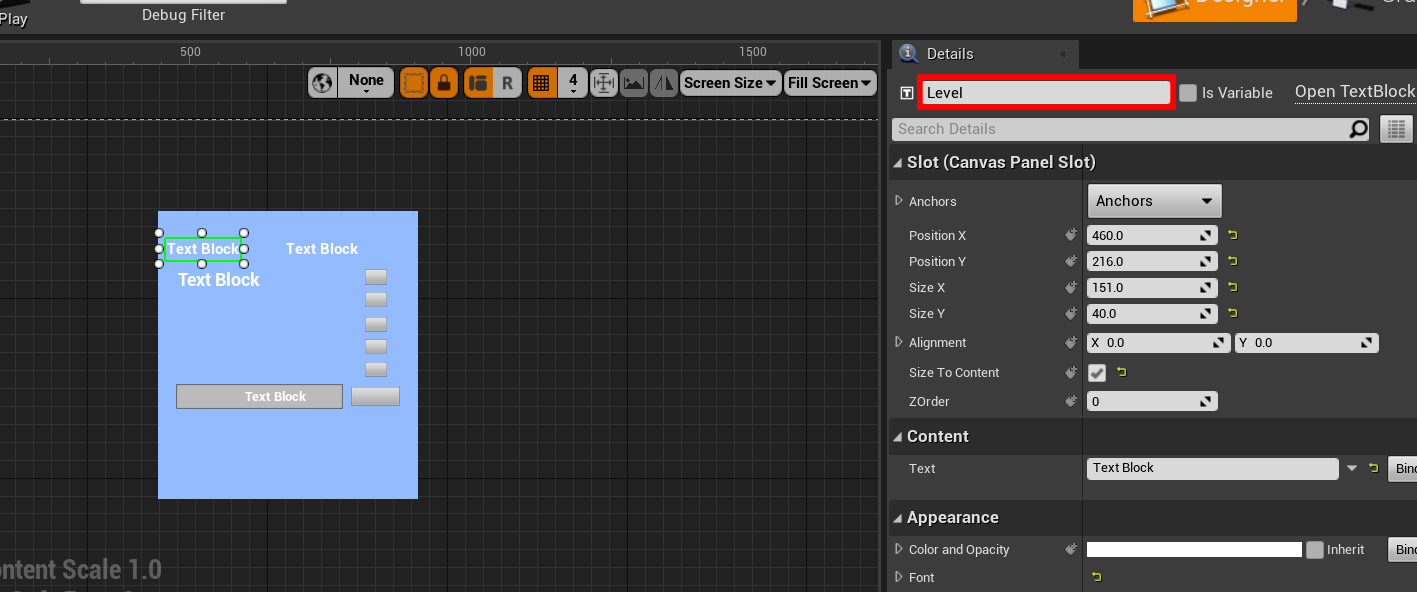
**Adding to and modifying the menu window**

First open the menu widget blueprint (“Content/FirstPersonCpp/Blueprints/WBP\_PostArcanaMenu”).

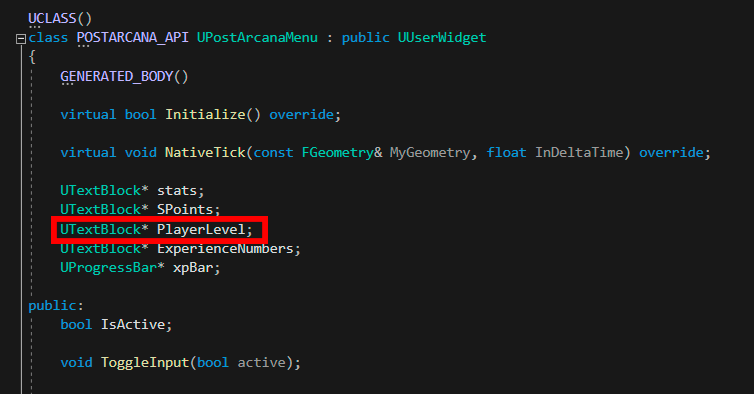
Graphical user interface, application, website

Description automatically generated

There you can edit and add whatever component you want. Make sure to name each component appropriately.



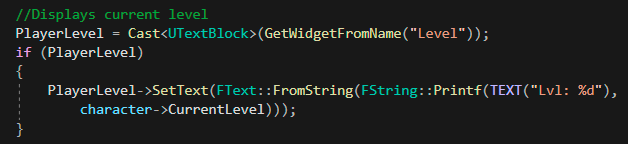
Next go on to UPostArcanaMenu (“Content/C++ Classes/PostArcana/UI/PostArcanaMenu”) and add a field pointer of the same data type as the component you want to edit.



On UPostArcanaMenu::NativeTick assign your field to the return value of GetWidgetFromName(), passing in the name you used on the blueprint as an argument, and casting the return value to the correct data type of the field.



Now whatever edits you do on your field will be reflected on the widget in the menu window. For example, you can use SetText() to change the output of a UTextBlock.



Graphical user interface

Description automatically generated