Post Arcana / Inventory

**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:** 04/11/22

**Changes:** Document created

**Version:** 0.2

**Modifier:** Danylo Cardoso

**Date:** 04/12/22

**Changes:** Detailed class design added

**Version:** 0.3

**Modifier:** Danylo Cardoso

**Date:** 04/13/22

**Changes:** Use case 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 the inventory 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 inventory system are:

• The design should minimize complexity and development effort.

• The design should give easy access to the game’s items.

# **3 System Behavior**

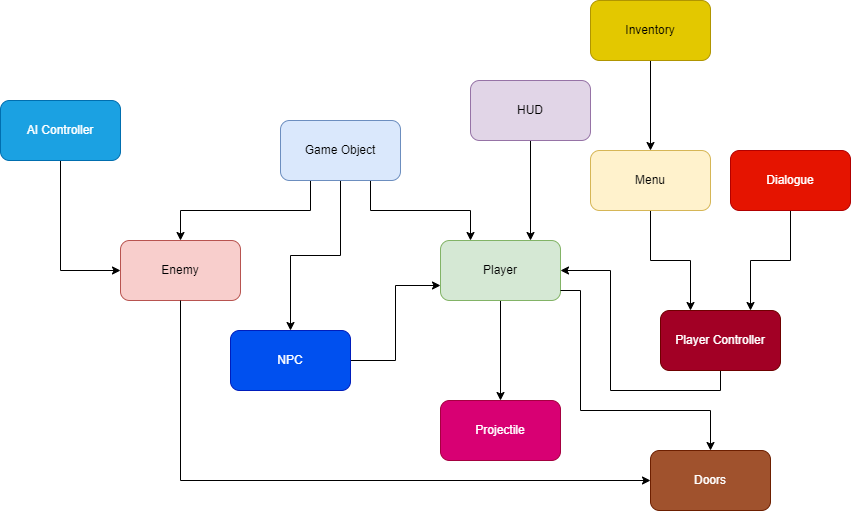
The inventory is compromised of multiple item slots. These item slots can each hold one item and create a drag and drop operation when dragged on while holding an item, as well as creating a dragged item visual. If this visual is dropped on an item slot, it will override that slot’s texture and give it a pointer to the item that was being dragged.

# **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**

****

* 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
* Inventory: Displays the items that the player owns

# **4.2 Mid-Level Design**

**Diagram

Description automatically generated**

**UPostArcanaInventorySlot:** A slot that can carry 1 item and displays that item’s icon. It can detect drags and drops.

**UInventoryDragDropOperation:** A drag and drop operation dedicated to the inventory drag and drop.

**AItem:** An item that can occupy an inventory slot.

**UInventoryDraggedItemVisual:** A widget that is generated alongside the drag and drop operation, it represents the visual that is carried by the mouse.

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

# 

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

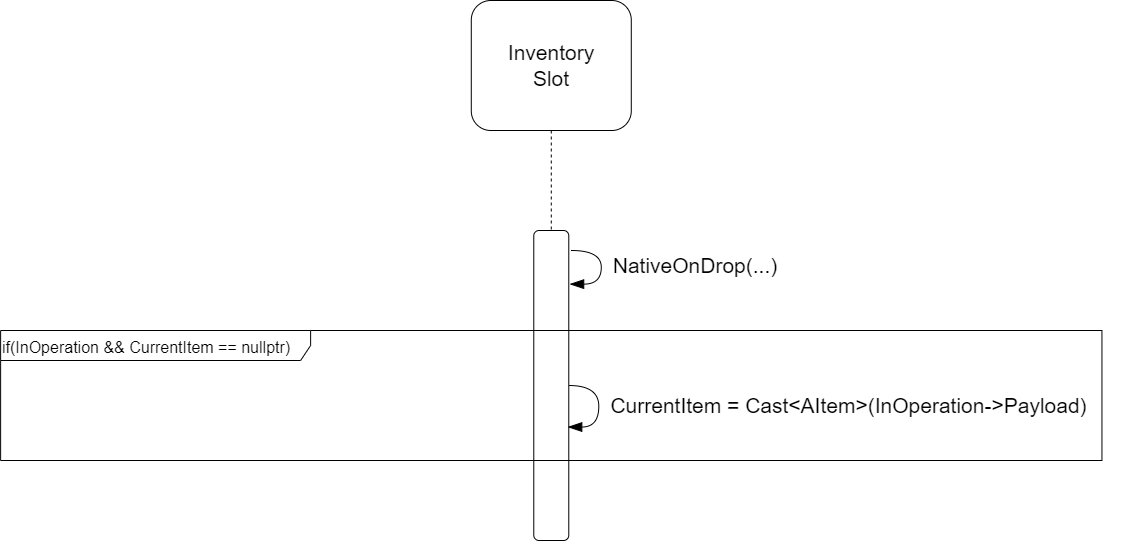
Mouse drag detection

Diagram

Description automatically generated

When the mouse is dragged over the inventory slot widget, it is checked if the slot currently holds an item and if the dragged visual blueprint property has been set. If both cases return true, the slot loses its item, and a dragged visual is created.

Mouse drop detection



# When a dragged visual is dropped on an inventory slot, it’s checked if the slot is not occupied. If not occupied, the slot’s item is set to the dragged visual’s item, and the slot’s item texture is set to the dragged visual’s texture.

# **6 Use Case View**

Creating a new inventory slot

Start by opening the inventory panel blueprint (“Content/FirstPersonCpp/Blueprints/WBP\_InventoryPanel”).

Graphical user interface, website

Description automatically generated

Search the palette for “wbp inventory slot”. Graphical user interface

Description automatically generated

Drag it on the center screen.

Graphical user interface

Description automatically generated

To have a slot start with an item in it, set Current Item BP to BP\_Item.

Graphical user interface, text

Description automatically generated with medium confidence

The inventory slot should now be displayed alongside the menu.

