Post Arcana / Stat System

Architecture/Design Document

**Table of Contents**

1. Introduction

2. Design Goals

3. System Architecture

4. Mid Level Design

5. Detailed Class Design

6. Process View

7. Use Case View

**Change History**

**Version:** 0.1

**Modifier:** Colin Brandow

**Date:** 03/12/2022

**Changes:** Design Document Created

**Version:** 0.2

**Modifier:** Colin Brandow

**Date:** 03/17/2022

**Changes:** System Architecture and Detailed Class Design

**Version:** 0.3

**Modifier:** Colin Brandow

**Date:** 03/18/2022

**Changes:** Document completed

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 stat systemin a manner that assists the interests of all major stakeholders. The major stakeholders and their interests are as follows:

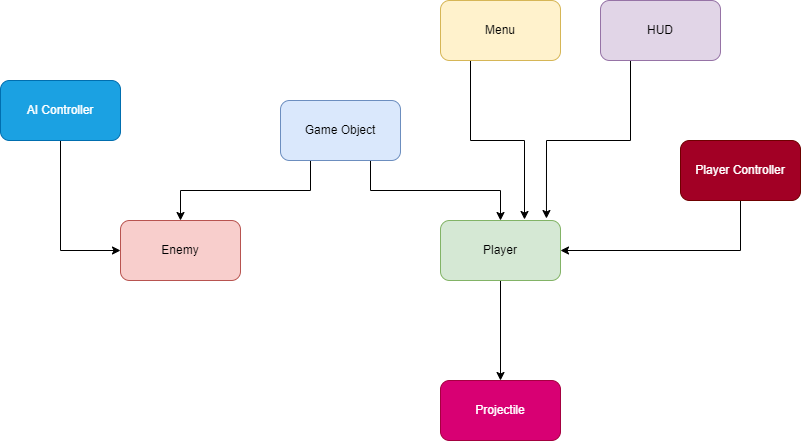
* Developers: A design that is easy to implement that minimizes complexity
* Project Manager: A design that can be easily divided among the skills of the programmers
* Maintenance Programmers: A design that can be improved upon easily

1. Design Goals

The design goals for the stat system are as follows:

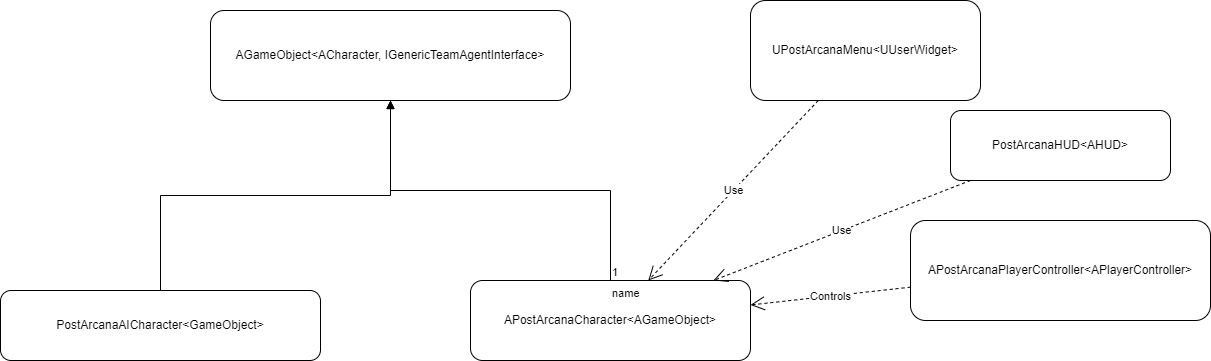
* The design must minimize complexity
* The design must allow for easy access to abstract and concrete stats
* The design should be easy to amend and balance

1. System Architecture

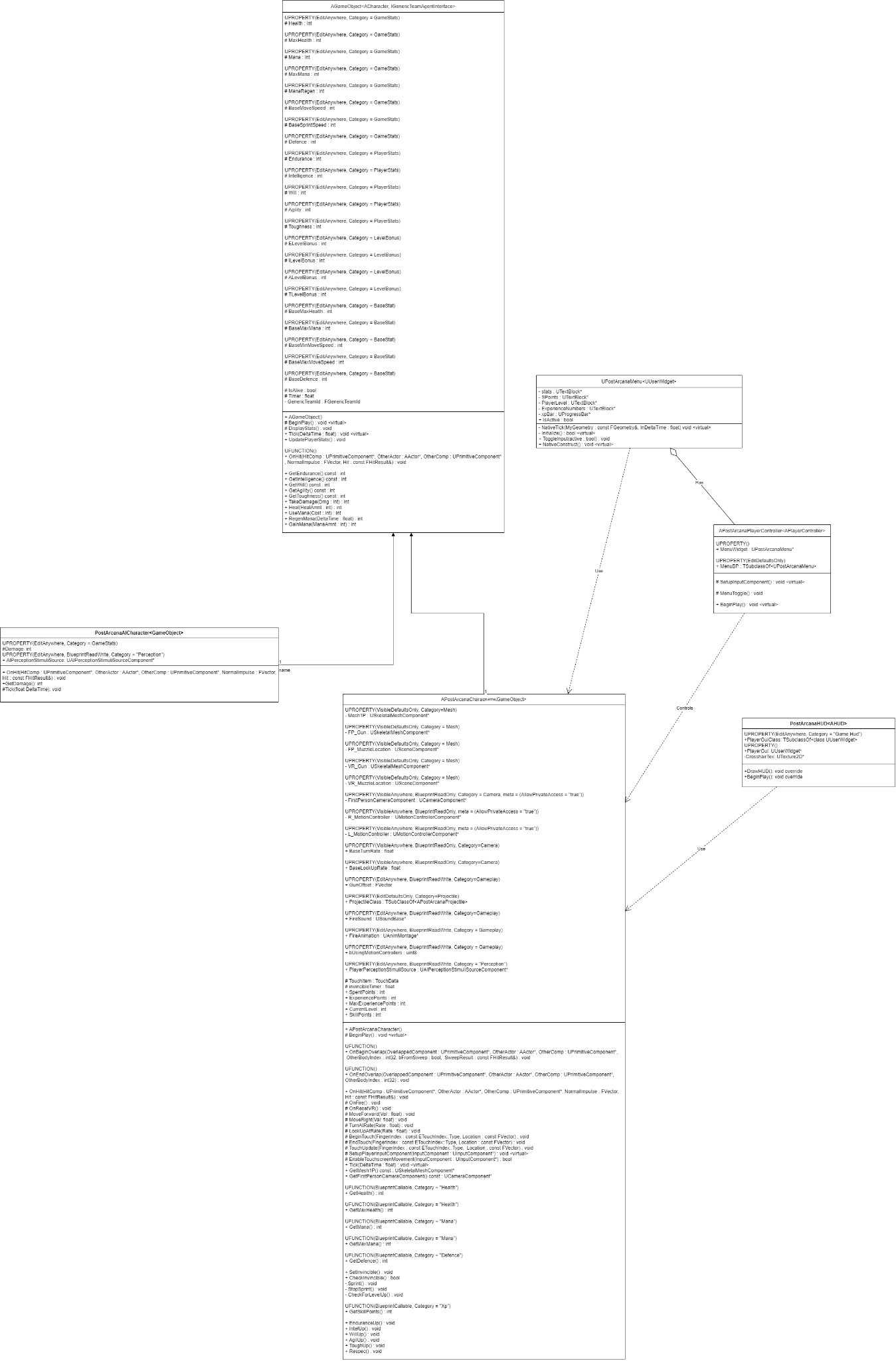
****

* 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 launchd 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

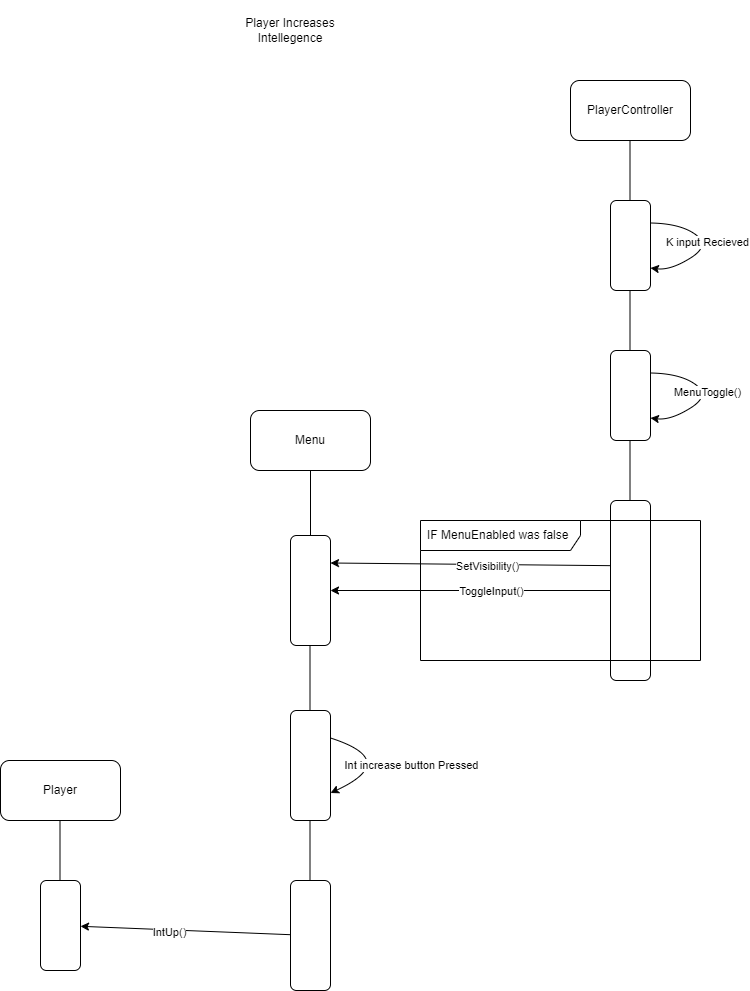
1. Mid Level Design

****

1. Detailed Class Design

****

1. Process View

****

The sequence for stat increases for the game is fairly simple. First, the player controller senses the menu button being pressed, and checks to see whether it needs to open or close the menu. If the menu was already closed, it will open. Then, when the player presses one of the stat increase buttons (This example uses intelligence but they are all very similar). When the menu senses this button is pressed, it will call the IntUp() function from the player, increasing the player’s intelligence and decreasing their available stat points.

1. Use Case View

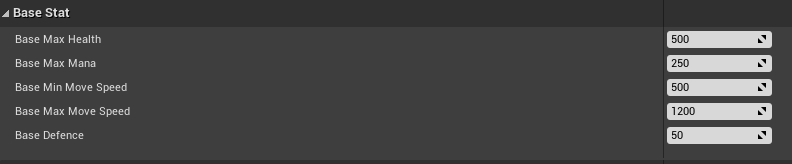
Using the stat system in practice is fairly easy. The system applies to all game objects, which currently includes players and enemies. There are also 2 sets of stats: concrete stats and abstract stats. The concrete stats are stats that directly affect gameplay and they are as follows:

* Health: The amount damage an entity can take before they are defeated
* Mana: A resource used for attacks and abilities
* Maximum Mana: The maximum amount of mana an entity can store
* Mana Regeneration: The rate at which mana is restored (measured in points per second)
* Defense: How much incoming damage is reduced (1 defense reduces incoming damage by 1)
* Damage: How much damage this entity deals

There are also 5 abstract stats which modify the concrete classes. They are as follows:

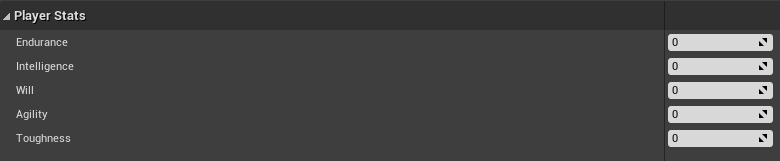
* Endurance: 1 point of endurance increases entities’ health by 25
* Intelligence: 1 point increases the entities’ maximum mana by 25. It also increases the projectile’s damage by 50 per point
* Will: 1 point increases mana regeneration by 1
* Agility: 1 point will increase the entities’ movement speed by 15 (both walking and running)
* Toughness: 1 point will increase the entities’ defense by 5

If a developer wishes to change the base values of any concrete stats for a specific entity they can increase it using the Base Stat heading in it’s blueprint as shown below:



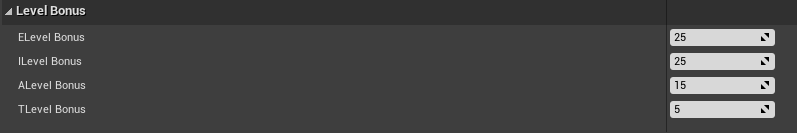
This is best used for increasing the concrete stats of an enemy like a boss, as it will allow a developer to increase its concrete stats without increasing its abstract stats should a developer wish to do so.

Similarly the abstract stats can be modified under the player stats heading:



This is best used for standard enemies, as increasing these stats will keep them comparable to a player of a similar level.

If a developer wishes to change the amount each abstract class modifies their concrete stat it can be changes under the Level Bonus heading:



This is best used for rebalancing, however it can be used to increase concrete stats at a different rate than the player which could make more interesting enemy designs