[Just Survive AI Module]

Architecture/Design Document

**Table of Contents**

[**1**](#_heading=h.gjdgxs) **Introduction 3**

[**2**](#_heading=h.30j0zll) **Design Goals 4**

[**3**](#_heading=h.1fob9te) **System Behavior 4**

[**4**](#_heading=h.3znysh7) **Logical View 5**

[**4.1**](#_heading=h.2et92p0) **High-Level Design (Architecture) 5**

[**4.2**](#_heading=h.tyjcwt) **Mid-Level Design 6**

[**4.3**](#_heading=h.3dy6vkm) **Detailed Class Design 8**

[**5**](#_heading=h.1t3h5sf) **Process View 11**

[**6**](#_heading=h.3rdcrjn) **Development View 12**

[**7**](#_heading=h.2s8eyo1) **Physical View 12**

[**8**](#_heading=h.17dp8vu) **Use Case View 12**

Change History

**Version:** <1.0>

**Modifier:** <Vanessa>

**Date:** 03/03/2020

**Description of Change:** <File created and set up>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version:** <1.1>

**Modifier:** <Vanessa>

**Date:** 03/04/2020

**Description of Change:** <Added UML and sequence diagrams>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version:** <1.2>

**Modifier:** <Vanessa>

**Date:** 03/06/2020

**Description of Change:** <Finished document>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version:** <1.3>

**Modifier:** <Vanessa>

**Date:** 03/06/2020

**Description of Change:** <Updated Introduction>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Version:** <1.4>

**Modifier:** <Vanessa>

**Date:** 03/06/2020

**Description of Change:** <Added AI elements, updated UML>

# Introduction

The purpose of this document is to describe the architecture and design for our AI module, which will focus on pawn sensing components and detection used in the game Just Survive, developed by Samurai Intellectuals. Pawn sensing is used to detect, target and damage the AI in the game by our damage-dealing actors.

The purpose of this document is to describe the architecture and design of the AI module application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

* Game Programmers
  + Want to make sure the code implemented is easy for the designers to user and edit
  + keep code consistent and structured so all programmers can understand and change as needed
* Game Designers - design in a way the programmers can use and change if needed

# Design Goals

The goal of this module is to have actors, such as towers in this case, detect the AI based on “sound”. If the AI are making sounds within a certain range of the actor, the tower will be able to detect that the AI is there. We’re using this to target and spawn projectiles in the direction of the AI once sensed and stop firing after the AI leaves the tower’s range of detection. Designers can set the waypoints for the AI to follow through the map accordingly.

# System Behavior

The tower class and Enemy Base both have a pawn sensing component that is set to a controlled range. Our player and any enemies have noise emitters to set off detection. When within the targeted range, the tower senses the AI, rotates it’s position based on the AI’s position, targeting and spawning projectiles. The AI also contains a pawn sensing component that listens for noise emitted by the player so it can target and shoot at the player if in range.

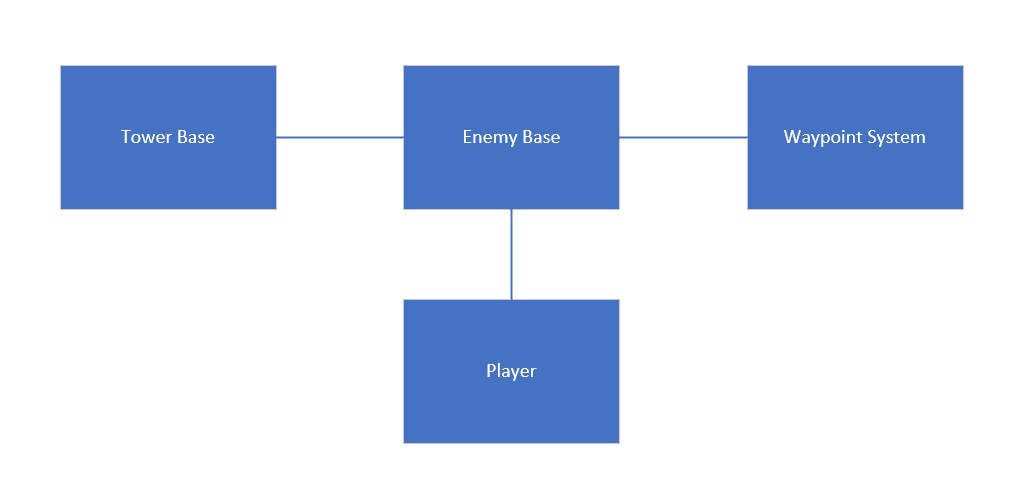
# Logical View

The pawn sensing system needs a “PawnSensingComponent”, noise emitter and an “OnNoiseHeard” function to work. Our tower class has the components and functions needed. The OnNoiseHeard function works to change towers rotation once a pawn is heard and handles logic of spawning and firing projectiles.

The AI class contains a noise emitter which emits noise whenever moving or firing.

## High-Level Design (Architecture of the Entire system)

<Tower Base, Enemy Base, Player and Waypoint System>

****

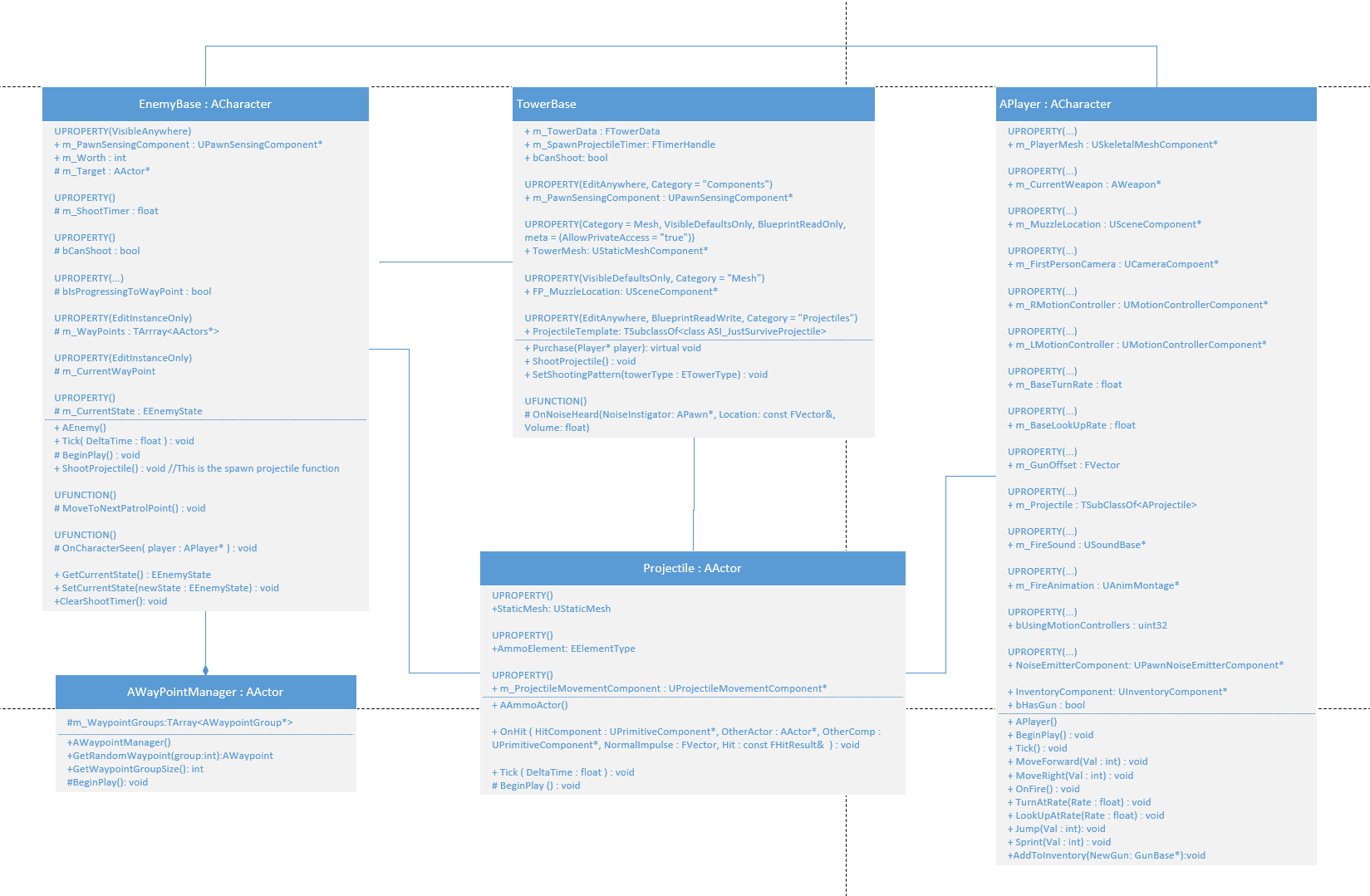
**System Architecture**

* The **Tower Base** has a pawn sensing component, which has a set radius of detection for anything that emits noise. It also has a function that will change the rotation of the tower to face the direction of the noise instigator and start firing projectiles in their direction. If the noise is no longer heard, the tower can no longer shoot until a new noise is detected.
* The **Enemy Base** has a main goal of following waypoints throughout the map. The enemy has a noise emitter and will make noise any time they walk or fire. If they enter the detection range of a tower base, they’ll be targeted. They currently don’t interact with the tower and will continue on their way to the next way point, eventually leaving the detection radius of the tower.
* The **Waypoint system** contains waypoint for the AI to follow, setting a path throughout the map
* the **Player** has a goal of taking down enemies and protecting assets like the generator

## Mid-Level Design of Module <AI Module>

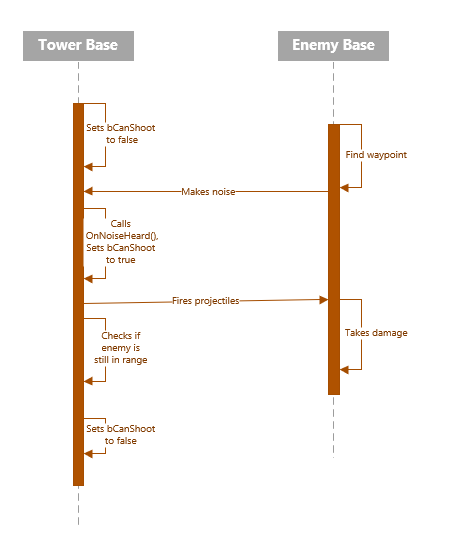
## 

## Detailed Class Design of Module <AI Module>



# Process View of Module <AI Module>

*Sequence diagram of interaction between TowerBase and EnemyBase:*



# Physical View (Applies to Multiplayer)

Since it’s a component and part of several classes, nothing will physically appear except debug spheres and messages.

# Use Case View

* Tower and enemy have a set radius of detection
* Tower and enemy check if anything has entered the PawnSensingComponent’s range of detection
* If “noise” is heard, tower and/or enemy sets a bool that it can fire projectiles to true, adjusts rotation based on noise instigators position, and starts shooting projectiles
* Once enemy or player is out of range again, bool is set back to false and tower continues to listen for next noise heard