Project Abyss / Example Module

Architecture/Design Document

**Table of Contents**

Change Log.. 2

1 Introduction.. 3

2 Design Goals.. 3

3 System Behavior.. 3

4 Logical View... 4

4.1 High-Level Design (Architecture) 4

4.2 Mid-Level Design. 5

4.3 Detailed Class Design. 6

5 Process View... 7

6 Use Case View... 9

Change History

**Version:** 0.1

**Modifier:** Jarod Beach

**Date:** 11/2/23

**Description of Change:** Started

**Version:** 0.2

**Modifier:** Jarod Beach

**Date:** 11/5/23

**Description of Change:** Completed first draft

**Version:** 0.3

**Modifier:** Jarod Beach

**Date:** 11/5/23

**Description of Change:** Doing second draft

**Version:** 0.4

**Modifier:** Jarod Beach

**Date:** 11/9/23

**Description of Change:** Final draft for release 1

**Version:** 1.0

**Modifier:** Jarod Beach

**Date:** 12/7/23

**Description of Change:** Final draft for release 2

# **1 Introduction**

This document describes the architecture and design for “Project: Abyss”, a game being developed by Astralwater Interactive. Embark on a groundbreaking cooperative underwater adventure with "Project Abyss," where the uncharted depths of Hollow Earth reveal a hidden world teeming with enigmatic flora, fauna, and ancient mysteries. As intrepid researchers contracted by a major tech conglomerate, you and your partner dive into the abyssal unknown armed with cutting-edge technology. However, beneath the waves, the shadows of corporate indifference threaten your every move.

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

● Developers – they want an architecture that will minimize complexity and development effort.

● Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all game logic.

● Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain on into the future.

# **2 Design Goals**

The design priorities for the Ai Fish system are:

* The design should minimize complexity and development effort.
* The design should allow Level Designers to easily change variables around without having to dive deep into the code itself.

**3 System Behavior**

The Fish Class which will inherit from the creature class will be what controls the behavior of our fish, this will use the UfishmovementComponent to move around the world and move along splines and randomly generated points.

# **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 (Architecture of the Entire system)**

A diagram of a computer program

Description automatically generated with medium confidence

* **Player System** is the central component that manages the player's actions, interactions, and progression throughout the game. It includes the player character, input handling. Key responsibilities involve controlling the player's movements, handling inputs, and triggering movement states such as swimming.
* **Photography System** is responsible for implementing the mechanics related to underwater photography. It includes the Photography Camera and associated components. This system allows players to take pictures during dives, manage a limited film capacity, store pictures in a gallery, and upload relevant ones to the Collection journal, contributing to the player's knowledge about underwater creatures.
* **Creature System** manages the behavior, interactions, and characteristics of underwater creatures. This system includes various types of fish, their AI, and the logic governing their movement, appearance, and responses to the player's actions. The system ensures a dynamic and realistic underwater ecosystem, providing challenges and opportunities for the player.
* **AI System** governs the artificial intelligence of both hostile and non-hostile entities in the game. It includes the behaviors and decision-making processes of underwater creatures, as well as any AI-driven challenges or enemies the player may encounter during dives.
* **Missions System** oversees the organization of missions into distinct categories, monitors the progression of objectives, and governs the allocation of rewards and unlocks. Accessible through the PC Terminal, players can seamlessly navigate through available, active, and completed missions. The PC Terminal serves as the central hub for mission management in the form of an email system, allowing players to accept new missions and submit completed ones, providing a comprehensive interface for tracking and advancing in the game's narrative.
* **Health Component System** manages the player's health. It includes components that handle damage, healing, and any other health-related mechanics. This system ensures that the player's survival is a key consideration during dives and encounters with underwater creatures.
* **Stress Component System** adds a psychological horror element to the game. It manages the player's stress levels based on in-game events, environmental factors, and encounters. High stress may affect performance or trigger adverse effects, adding an immersive layer to the overall experience.
* **Boat System** encompasses the functionality related to the player's boat. This includes navigation between dive locations, boat inventory, and serving as a safe zone between dives. The boat system also facilitates equipment upgrades, PC Terminal interaction, and overall progression.
* **DiveCage System** is used as a later game upgrade that can be used to speed up travel time to get to deeper locations.
* **Equipment System** is the main item system in the game. Players will be able to buy and use equipment from the shop, including a camera, deployable light sources, and more. Some equipment is consumable, and some is permanent.
* **Divebot System** encompasses the functionality to the controllable Divebot available in the game. The Divebot will be possessed on death as a spectator mechanic, and can be possessed on command to allow remote control for scouting purposes. This will include a custom movement component to assist in executing some of these tasks.
* **Upgrades System** enables the enhancement and customization of the player's equipment, boat, and dive-related tools. Players can earn or purchase upgrades to improve their camera, dive cage, and other components, providing a sense of progression and empowerment.
* **Shop System** allows players to buy in-game equipment and upgrades. It provides a marketplace where players can spend earned resources to acquire new equipment, upgrade existing gear, or replenish consumables.
* **PC System** refers to the personal computer within the game. It serves as the hub for managing various aspects, such as reviewing your research journal, viewing photographs, accessing the in-game shop, and planning future dives. The PC system contributes to the player's overall agency and strategic decision-making.
* **Inventory System** refers to the inventory component that is attached to the player and storage chest. The player will be able to swap equipment to and from the storage chest.

## **4.2 Mid-Level Design of the Fish Module**

A black and white screen with white text

Description automatically generated

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

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

# 

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

**NewPointLocation()**

A diagram of a fish

Description automatically generated

**1.**Get the fishes Location

**2.** Based on the fish’s location. Create a point around a certain radius of the fish

**CreatePoint()**

A diagram of a fishing game

Description automatically generated

**1.** Get the fishes Location

**2.** return fish position

**3.** Based on the fish’s location. Create a point around a certain radius of the fish

**4.**If the fishPoint is null

**5.** Create a point and give the point the random point location.

**bRaycastToPoint()**

A screenshot of a computer

Description automatically generated

**1.** The fish raycasts to the point and see if it can see it

**2.** The point raycasts to the fish and see if it can see it

**3.** If both raycasts hit successfully

**4.** return that the fish can see the point

**5.** else return false

**Idle(float DeltaTime)**

A black screen with white text

Description automatically generated

**1.** Set the fish to idle for 3 seconds

**2.** After the timer is up, switch fish back to patrol

**Patrol(float DeltaTime)**

A screenshot of a computer

Description automatically generated

**1.** If the fish is swimming

**2.** If the fish detects the player

**3.** switch it into attack state if not continue

**4.** if the FishPointActor doesn’t exist

**5.** create a Fishpoint actor

**6.** Call movement component on the fish with the FishPointActors location

**7.** the movement component will move the fish towards the point

**8.** If the fish is close enough to the FishPoint Actor

**9.** Move the point to a new location

**10.** Raycast to the point

**11.** if the point is not valid

**12.** move the point until its valid

**Attack(float DeltaTime)**

A screenshot of a computer screen

Description automatically generated

**1.** Find all actors around the fish

**2.** if it finds a actor, search if its found a player

**3.** if it has move towards the player

**4.** If the fish gets close to the player

**5.** Attack the player animation

**6.** Player receives the damage of the attack

**7.** Return the fish to the idle state

**8.** if not player was found from step 2

**9.** Return fish to the idle state

# **6 Use Case View**

Grab one of these blueprints and put them into the world or duplicate them to make your own!

A screenshot of a game

Description automatically generated

If you want to change the titles and description of fish, you will change them in the statistics and Designer tabs

A screenshot of a computer

Description automatically generated

Change the Collision type of the fish to Fish-Water in the collision box

A screenshot of a computer

Description automatically generated

Change Static mesh and Materials in the mesh tab

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Next go to the fish movement component, you can change the movement type as well as the values of speed and turning. Be sure to always change the controller to BaseFishAIController under the pawn tab.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated