Technical Design Document – Outline

# Title Page

6th Sense: Eternal Abyss

*“Survive the Shadows, Seize the Shade”*

William Herbert C. Balingit

April 14, 2024

Copyright © 2024 William Balingit. All rights reserved.

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Changes |
| 0.1 |  | William Balingit | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[1 Title Page 1](#_Toc164124172)

[2 Document History 2](#_Toc164124173)

[3 Table of Contents 2](#_Toc164124174)

[4 Game Summary 3](#_Toc164124175)

[5 Development Environment 3](#_Toc164124176)

[5.1 Development Hardware 3](#_Toc164124177)

[5.2 Programming Languages 3](#_Toc164124178)

[5.3 Development Tools 3](#_Toc164124179)

[5.4 Game Engine 3](#_Toc164124180)

[6 Architectural Analysis 3](#_Toc164124181)

[6.1 Classes 3](#_Toc164124182)

[6.2 Behavioral Analysis 4](#_Toc164124183)

[6.3 Game Loop 5](#_Toc164124184)

[7 Technical Risks 6](#_Toc164124185)

# Game Summary

"Eternal Abyss" immerses players in a chilling dark ambient survival shooter set within the depths of the Underworld, cloaked in perpetual darkness. Armed with a diverse array of weapons and abilities, players must endure through an eerie environment, facing relentless enemies. The goal: survive the perpetual onslaught or uncover the mysteries hidden within the eternal abyss. To aid in the struggle, players can utilize power-ups like increased light for visibility, slowed-down enemies, quick reloads, and brief periods of unlimited ammo. With atmospheric gameplay, strategic decision-making, and an enticing blend of survival and mystery, "Eternal Abyss" delivers an intense and haunting gaming experience.

# Development Environment

## Development Hardware

* Laptop Computer: 16GB RAM, Nvidia 3050Ti Graphics, Ryzen 7 5th Gen Processor
* Keyboard and Mouse
* Headphones
* Windows 10 Home

## Programming Languages

* C++

## Development Tools

* Visual Studio IDE
* Macromedia Flash Pro 8 – Image editor
* Audacity – Audio editor
* Texture Packer – Sprites
* Github – version control software

## Game Engine

SFML (Simple and Fast Multimedia Library) version 2.6.1

# Architectural Analysis

## Classes

Describe the classes that will have to be implemented. For each class, provide:

* Its responsibilities
* How it collaborates with other classes

|  |  |  |
| --- | --- | --- |
| Class | Responsibilities | Collaborations |
| Scene | Manages current scene, loads/unloads scene-specific assets | Game, Assets, Entity Manager |
| Game | Controls game flow, manages game states | Scene, Menu, Settings |
| Menu | Displays and manages in-game menus | Game, Command |
| Settings | Manages game settings and configurations | Game |
| Assets | Stores and manages game assets | Scene, Entity |
| Command | Represents user commands/actions | Menu, Game |
| Entity | Represents interactive objects/characters | Entity Manager, Physics |
| Entity Manager | Manages lifecycle of entities in game world | Scene, Entity |
| Music Player | Controls background music playback | Game |
| SFX Player | Handles playback of sound effects | Entity, Utilities |
| Physics | Simulates physical interactions between game objects | Entity, Utilities |
| Utilities | Provides general-purpose utility functions/classes | Used by various other classes for common tasks |

## Behavioral Analysis

* Player
  + States:
    - Idle: The default state when the player is not moving.
    - Moving: The state when the player is actively moving in a specific direction.
    - Shooting: The state when the player is shooting bullets.
  + Transitions:
    - Idle to Moving: Triggered when the player presses WASD keys.
    - Moving to Idle: Triggered when the player releases WASD keys.
    - Idle/Moving to Shooting: Triggered when the player presses the shoot key.
* Enemies
  + Symbiote (Follows the Player):
    - The enemy continuously moves towards the player's current position.
  + Wraith (Periodically Hides Texture):
    - The enemy periodically changes its appearance by hiding its texture.
    - This behavior can be represented as a timed event within the enemy's behavior loop.
  + Revenant Enemy (Occasionally Disables Hitbox):
    - The enemy occasionally disables its hitbox, making it temporarily immune to player bullets.
    - This behavior can also be represented as a timed event within the enemy's behavior loop.
  + Abyssal Serpent (Hides Texture and Disables Hitbox):
    - The enemy occasionally hides its texture and disables its hitbox simultaneously.
    - This behavior combines the characteristics of Type 2 and Type 3 enemies.
* Bullet
  + Pistol Bullets:
    - Bullets are shot in a straight line from the player's position towards the direction they are facing.
    - Upon hitting an enemy, bullets disappear.
  + Shotgun Bullets:
    - Bullets spread in an arch pattern, with each shot diverging slightly from the others.
    - Each bullet has its trajectory but follows the general direction indicated by the player's aim.
    - Bullets disappear upon hitting an enemy.
  + Sniper Bullets:
    - Bullets shoot in a straight line and pierce through all enemies in their path.
    - Bullets continue their trajectory until they reach the end of the game area or hit an obstacle.

## Game Loop

Describe, in order, the sequence of activities that happen during each game loop. You must document this even if you’ll be using the “Clown Cannon” game engine.

1. Sound Management -> remove stopped sounds.
2. Entity Management -> update all entities within the scene.
3. Pause Check -> returns early and skips executing the rest of the game loop if game is “paused”.
4. Animation Update -> updates animations for various entities in the scene based on “dt”.
5. Player Movement -> handle player movement based on input and time.
6. Adjust Position -> ensure the player is within the play/game boundaries/window.
7. Manage mobs -> manage enemy spawning, de-spawning, movement and behavior.
8. Collisions -> handle collision detection and resolution.
9. Render -> draw backgrounds, entities, and UI

# Technical Risks

List all technical risks that could make it difficult or impossible to complete the game. Examples:

|  |  |  |
| --- | --- | --- |
| Risk | Severity | Mitigation (what is to be done to eliminate or minimize this risk) |
| Uncertainty in implementing complex features | High | Conduct thorough research and prototyping to explore different approaches. Break down features into smaller, manageable tasks. Seek advice from experienced developers or mentors. |
| Performance concerns | High | Profile code and identify performance bottlenecks early in development. |
| Scalability concerns | Medium | Design a modular and scalable architecture from the outset. |
| Resource limitations | Medium | Optimize assets, code, and content to minimize resource consumption. |
| Lack of documentation or support | Medium | Document code, architecture, and development processes comprehensively throughout the project. |