Technical Design Document – Outline

# Title Page

Wonder Boy – Implementation of the classic wonder boy arcade game

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Changes |
| 0.2 | January 16, 2013 | Emile Cormier |  |
| 0.3 | January 05, 2016 | David Burchill |  |
| 0.4 | March 07, 2024 | Jung Dae Kwon |  |

# Table of Contents

# Game Summary

The adventurous young boy goes out on a long journey to chase the Evil King, who had kidnapped his girlfriend. Players should guide "Wonder Boy" through an island and rescue her.

In this game, players encounter a dynamic and visually captivating arcade experience. The game features a diverse range of areas, each presenting unique challenges and enemies for Wonder Boy to overcome.

# Development Environment

## Development Hardware

HP laptop with Windows

## Programming Languages

Visual C++ 2022  
SFML

## Development Tools

Visual Studio 2022

Texture Packer

AESPRITE

Git hub

## External Code

SFML

<https://www.sfml-dev.org/>

## Game Engine

Professor David Burchill provided a base game engine to build this game.

I modified the play scene and map. Implementing a tile system with uphill was challenging, it is solved with double for loops. From that, I could learn deeply about centering origin position and bounding boxes. Collision with tiles with gravity system also very hard to implement properly, because it should work properly and stably spontaneously.

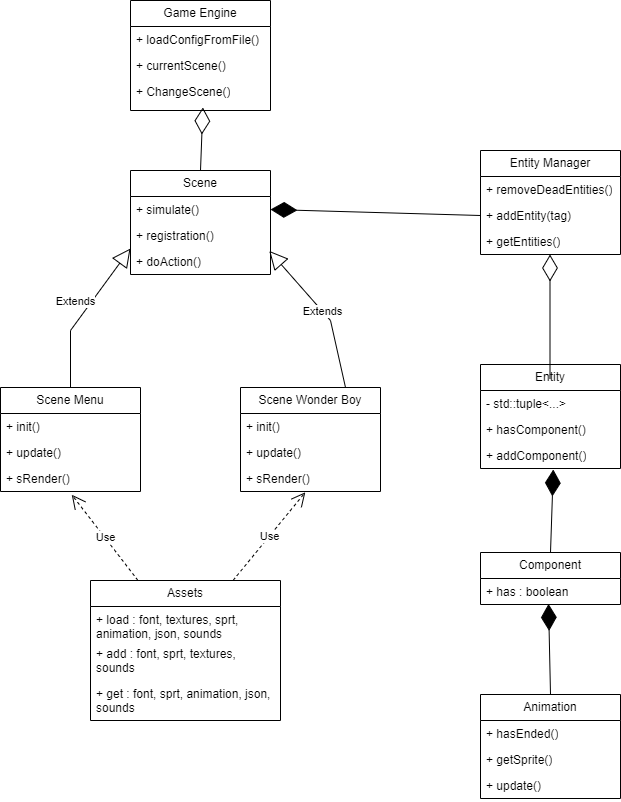
# Architectural Analysis

## Classes

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

|  |  |  |
| --- | --- | --- |
| Class | Responsibilities | Collaborations |
| Animation | Managing sprites to render | It is a part of Component class, works with render system. |
| Assets | Load files to store | It loads sources for font, sound, images by config.txt |
| Entity | Manage Components | Set component and use them. |
| Entity Manager | Manage Entities | Add entity, return entities, remove entities from the list to render.  Work with Game Engine and Scene class. |
| Game Engine | Operating basic mechanism of the game | It has a map of scenes. |
| Scene | Implementing each scene | Scene uses Game Engine. |

Present class diagrams that show the relationships between classes. Show only the most essential attributes and methods for each class.



## Behavioral Analysis

Present statecharts, flow charts (activity diagrams), sequence diagrams, etc. that model complicated behavior. If your game has actors that implement a state machine, this would be the section where you’d present the statechart.

enum playerState {

isAlive = 1, // 1 set: alive, unset: dead

isFacingLeft = 1 << 1, // 2 set: facing left, unset: facing right

isRunning = 1 << 2, // 4 set: running, unset: not running, stand

onSkate = 1 << 3, // 8 set: on skate, unset: not on skate

isGrounded = 1 << 4, // 16 set: grounded, unset: not grounded

isThrowing = 1 << 5, // 32 set: throwing, unset: not throwing

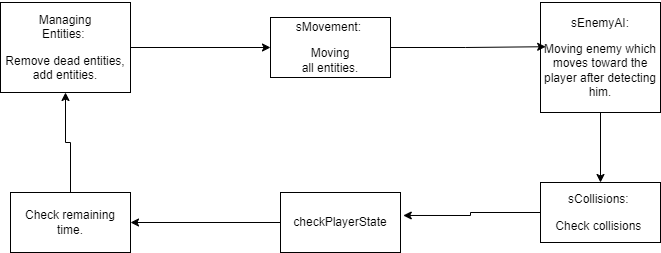
isBurned = 1 << 6, // 64 set: burned, unset: not burned

isTripping = 1 << 7, // 128 set: tripping, unset: not tripping

};

## 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.



# Technical Risks

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

* Uncertainty on how to implement a certain feature
* Uncertainty on if a certain feature can be executed fast enough in real time
* First time using a certain library

|  |  |  |
| --- | --- | --- |
| Risk | Severity | Mitigation (what is to be done to eliminate or minimize this risk) |
| Gravity and Collision with tile bounding box | High | If I assign gravity to enemies, after 4~5 seconds, they fall down to the bottom through tile’s collision system.  Only using gravity component to the player during figuring out why it happens. |
| Setting proper speed to animations | Medium | I analyzed the game engine and identified where to adjust the speed. |
| Implementing uphill tiles | Low | Using double for loop with bounding boxes. |
|  |  |  |