Technical Design Document – Outline

# Title Page

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** |
| 0.01 | 2024-03-08 | Diego Portella |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

Contents

[1 Title Page 1](#_Toc153485133)

[2 Document History 1](#_Toc153485134)

[3 Table of Contents 1](#_Toc153485135)

[4 Game Summary 2](#_Toc153485136)

[5 Development Environment 2](#_Toc153485137)

[5.1 Development Hardware 2](#_Toc153485138)

[5.2 Programming Languages 2](#_Toc153485139)

[5.3 Development Tools 2](#_Toc153485140)

[5.4 External Code 2](#_Toc153485141)

[5.5 Game Engine 2](#_Toc153485142)

[6 Architectural Analysis 2](#_Toc153485143)

[6.1 Entities 2](#_Toc153485144)

[6.2 Components 3](#_Toc153485145)

[6.3 Systems 3](#_Toc153485146)

[6.4 Behavioral Analysis 3](#_Toc153485147)

[7 Technical Risks 3](#_Toc153485148)

# Game Summary

In "BomberMania," players engage in intense multiplayer battles in dynamic arenas reminiscent of classic Bomberman gameplay. Each player controls a character tasked with strategically navigating through a grid-based arena filled with destructible blocks, power-ups, and hazards.

# Development Environment

## Development Hardware

PC windows 10 or 11

## Programming Languages

This game engine and game will primarily be written in C++23 with SFML 2.6.1

## Development Tools

Visual Studio

## External Code

SFML (https://www.sfml-dev.org)

## Game Engine

GEX game engine developed in prog (1241-3213). This is an Entity-Component-Systems based game engine.

# Architectural Analysis

## Entities

|  |  |
| --- | --- |
| Entities | Responsabilities |
| spawnPlayer | It’s used to create the player |
| spawnPlayer2 | It’s used to create the player 2 |
| spawnBomb | Spawn the bomb |
| spawnExplosion | Spawn the explosion |
| spawnFire | Spawn the Fire |
| spawnPowerUp | Spawn the Power Up |
| spawnBrick | Spawn the bricks |
| spawnRandomBricks | Spawn randomly the bricks |
|  |  |

## Components

|  |  |
| --- | --- |
| Component | Purpose |
| CTransform | Holds the position, velocity, and direction of entity |
| CBoundingBox | Used for collision detection. All entities that participate in collisions must have this component. |
| CSprite | The sprite used to draw the entity, to be drawn an entity must have a sprite component. |
| CAnimation | Represents an animation component. |
| CShape | Represents the shape creation. |
| CInput | Represents an input component for user input. |
| CBomb | This component represents a bomb entity in the game |
| CTile | Represents a tile entity in the game world, used for creating environments |
| CPowerUp | Providing bonuses or enhancements to players when collected |
| CCollision | Handles collision detection and resolution between entities |
| CState | Represents the state or status of an entity, allowing for dynamic behavior and interactions |
| CScore | This component manages the scoring system within the game |

## Systems

|  |  |
| --- | --- |
| System | Purpose |
| sMovement | Moves all entities that have a CTransform component |
| SAnimation | Manages the animation of entities in the game |
| sBombUpdate | Handles the behavior and updates of bomb entities, such as detonation and explosion effects |
| sRender | Responsible for rendering the game scene, drawing entities and background elements onto the screen. |
| sUpdate | Updates the scene's state over time. |
| onEnd | Defines actions to be taken when the scene ends. |
| resetPlayer | Resets the player's position and state. |
| killPlayer | Handles necessary actions when the player dies. |
| updateScore | Updates the player's score based on the current position. |
| init | Initializes the scene based on a configuration file specified by the path. |
| loadLevel | Loads scene elements from a configuration file. |
| playerMovement | Handles player movement based on keyboard input. |
| adjustPlayerPosition | Ensures the player's position is within the screen boundaries. |
| checkPlayerState | Checks and manages the player's state, such as whether it's alive or dead. |
| registerActions | Registers player actions associated with control keys. |

## Behavioral Analysis

1. Start Button to start the game on Menu.
2. Receive inputs to move the frog.
3. Moving the character
4. Checking collisions during the character pursuit
5. If the character collides with something, it should block his patch or die.
6. While the character destroys the bricks, the map will be cleaned and must count points per brick destroyed.
7. If the character has remained alive or still hasn’t run out the time, the game goes on
8. The last stand alive during in the map wins the match.
9. End

# Technical Risks

|  |  |  |
| --- | --- | --- |
| Risk | Severity | Mitigation (what is to be done to eliminate or minimize this risk) |
| Uncertainty in Implementing spawnPlayer3 spawnPlayer4 and Feature | Moderate | Conduct thorough research and analysis: Investigate similar features implemented in other games or frameworks to gather insights and best practices for implementing spawnPlayer3, spawnPlayer4, and the new feature.  Prototyping and testing: Develop prototypes or proofs of concept to experiment with different approaches and identify potential challenges or limitations early in the development process.  Break down the feature: Divide the feature into smaller, more manageable tasks and prioritize them based on dependencies and complexity to facilitate incremental implementation and testing.  Collaboration and communication: Maintain open communication channels within the development team to share knowledge, exchange ideas, and address any uncertainties or challenges encountered during implementation.  Continuous evaluation and iteration: Regularly review the progress of implementation, gather feedback from stakeholders, and be prepared to make adjustments or refinements to the design and implementation based on emerging insights and requirements. |
|  |  |  |
|  |  |  |
|  |  |  |