Requirements and Analysis Document for Smurf Survivors

Isac Snecker, Isak Söderlind, Emil Svensjö,
Jakob Schült

2023-12-15

Version 1.0

1. Introduction

The purpose of this application is to create an entertaining experience for gamers. The game consists of a map in which different types of enemies will be summoned in an attempt to kill the player. The player's objective is to kill and avoid the enemies in order to survive as long as possible meanwhile the difficulty of the game increases over time and the player earns weapons and upgrades for killing enemies.

1.1. Definitions, acronyms, and abbreviations

- **Enemy**: A entity with objective to inflict damage on the player
- **Player**: Entity in the game controlled by the user
- **Map**: The main stage of the game. Withstands of a tile map in which entities roam upon.
- **XP**: Experience points, given to the player after killing a enemy, used to level up and require new weapons
- **HP**: Health points, related to the player. When HP reaches zero the game is over.
- **Weapon**: Object used to inflict damage on the enemies

2. Requirements

2.1. User stories

2.1.1. Diagram and definitions

- **Epics**: A user story that can be broken down into smaller user stories
- **Priority** 1: Required for the application to function
- **Priority** 2: Not required for the application to function, but required for the application to be enjoyable for the user
- **Priority** 3: Not required but nice to implement to make the application more user friendly.

2.1.2. Epics

- "As a casual gamer, I want a game that is easy to understand so that I can play it without spending time to understand it."

- "As a gamer, I want to play the game to have fun"

- "As a hardcore gamer, I want to be able to move around freely and make use of different weapon combinations to MAXIMIZE my chances of succeeding within the game."

- "As a hardcore gamer, I want the game to follow the same standard game design as for other games I have played so that I can easily learn, and eventually, master the game."

- "As a hardcore gamer, I want there to be many different threats within the game that I have to use my skills and experience that I as a hardcore gamer have in order to neutralize."

- "As a casual gamer, I want a game where you can gain items/abilities so that the playstyle changes. So that gameplay changes over time."

2.1.3. Priority 1 user stories

- **"As a player I want to move my player around in order to progress the game"**
  Implemented: Yes
  Confirmation:
  - Get input from the keyboard to move the player
  - Render player on screen
  - Create player class
  - Implement methods for moving player

- **"As a player I want to kill enemies in order to progress"**
  Implemented: Yes
  Confirmation:
  - Create model for enemy
  - Render enemies
  - Make projectiles follow enemies
  - Make enemies take damage
  - Implement model for weapon
  - Implement method for shooting projectiles

- **"As a player, I want a timer rendered on the screen to track my progress"**
  Implemented: Yes
  Confirmation:
  - Implement model for Clock
  - Render clock on screen

2.1.4. Priority 2 user stories

- **"As a player, I want to see my health so I know if i'm about to die"**
  Implemented: Yes
  Confirmation:
  - Render healthbar
  - Make player take damage when colliding with enemy
  - Implement health in player model

- **"As a player, I want to be able to see my level, so that I can see my progress."**
  Implemented: Yes
  Confirmation:
  - Render XP bar
  - Create model for XP bar

2.1.5. Priority 3 user stories

- **"As a player, I want to be able to pause the game so that I can take a break."**
  Implemented: Yes
  Confirmation:
  - Implement pause function
  - Create pause screen

- **As a player, I want to be able to change the settings so that I can change the volume/resolution/difficulty.**
  Implemented: Yes
  Confirmation:
  - Create settings screen
  - Create settings model
  - Implement settings in controller

- **"As a player, I want there to be a quit button so that I am able to quit the game."**
  Implemented: Yes
  Confirmation:

  - Create quit button in controller
  - Implement quit method

- **"As a player, I want a menu screen so I can choose setup to play the game"**
  Implemented: Yes
  Confirmation:
  - Create stage for main menu
  - Render graphics for main menu
  - Add menu buttons for menu in controller

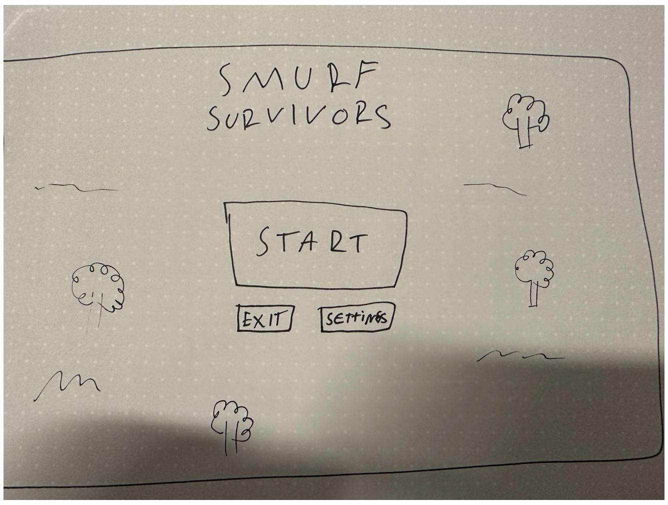
- **"As a player I want sound effects to play when performing actions in order to increase immersion"**
  Implemented: Partly
  Confirmation:
  Implement audio manager
  Implement audio observer pattern
  Play audio from collision handler

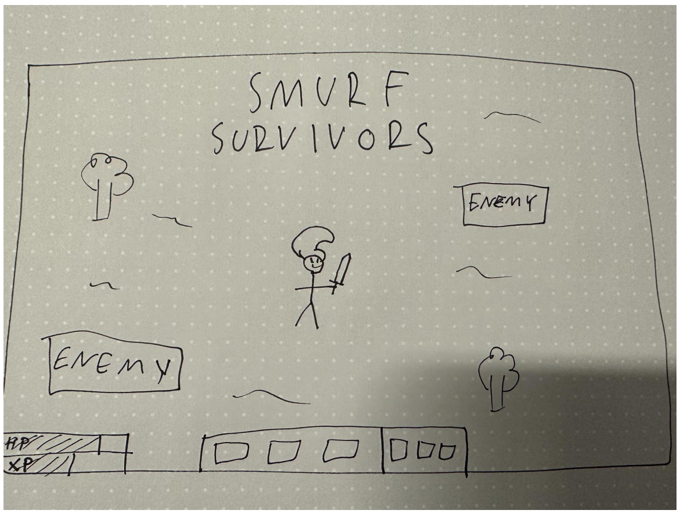
- **"As a player I want a toolbar so I can see which weapons I have and their levels."**
  Implemented: Partially
  Confirmation:
  - Implement Toolbar class
  - Add Toolbar to HUD
  - Render Toolbar
  - Make Toolbar able to store items
  - Render weapon level on toolbar item

2.2 User interface

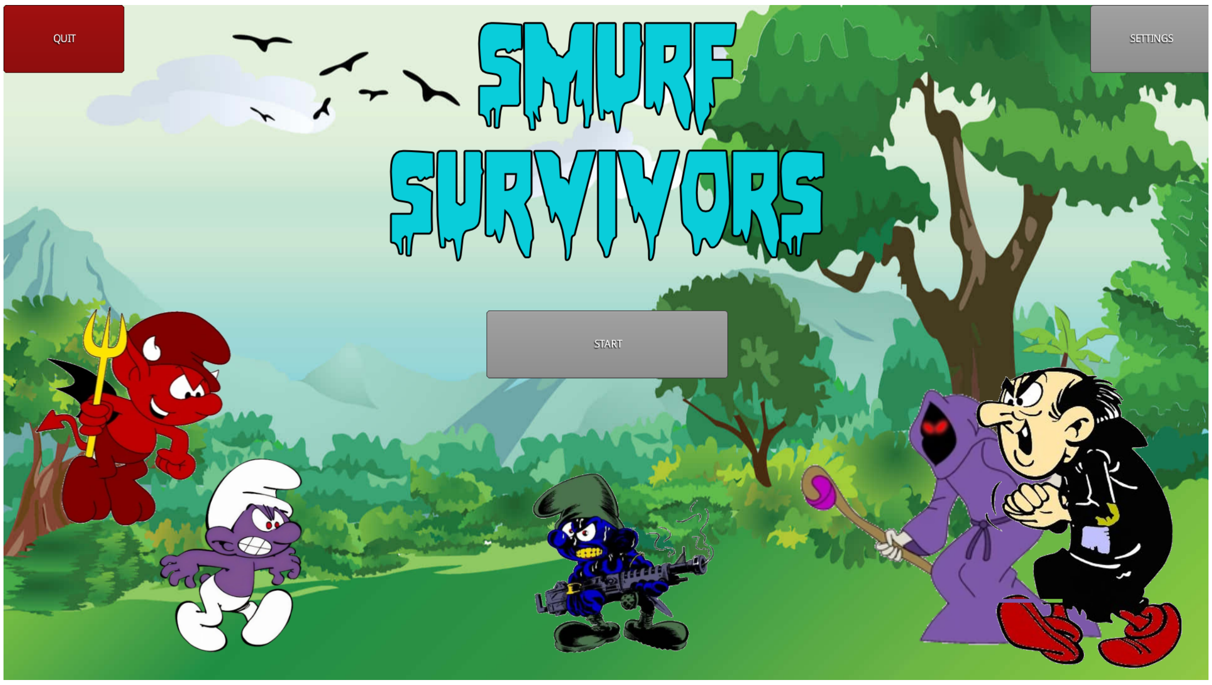
The first initial prototype of the UI is a drawing of two different views found in the game. The main menu and one markup of the view of the gameplay itself. The prototype resembles the last iteration of the UI fairly well although there are some key differences which have been changed.

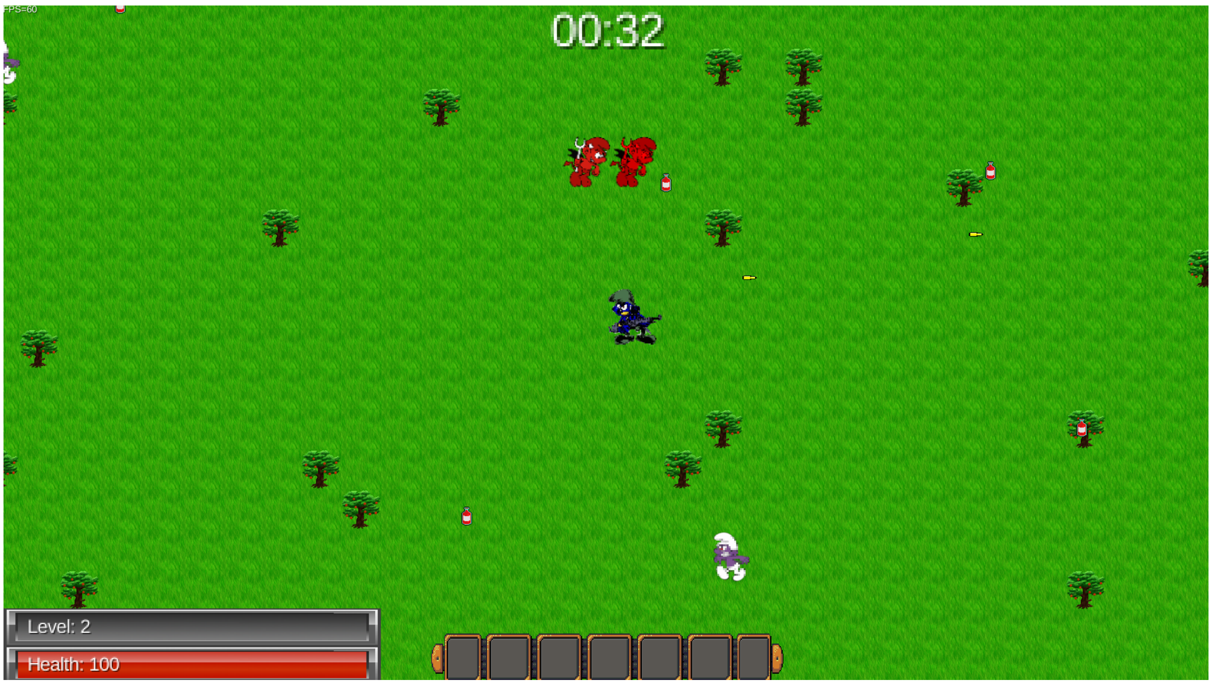
First iteration:





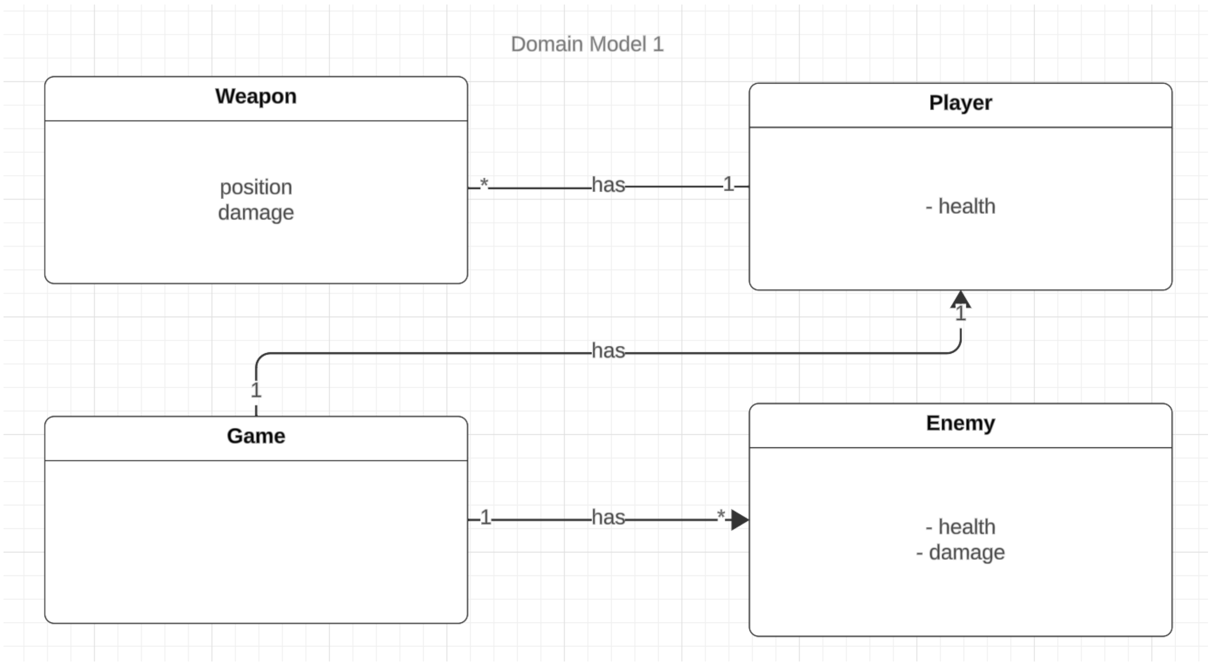
Current iteration:

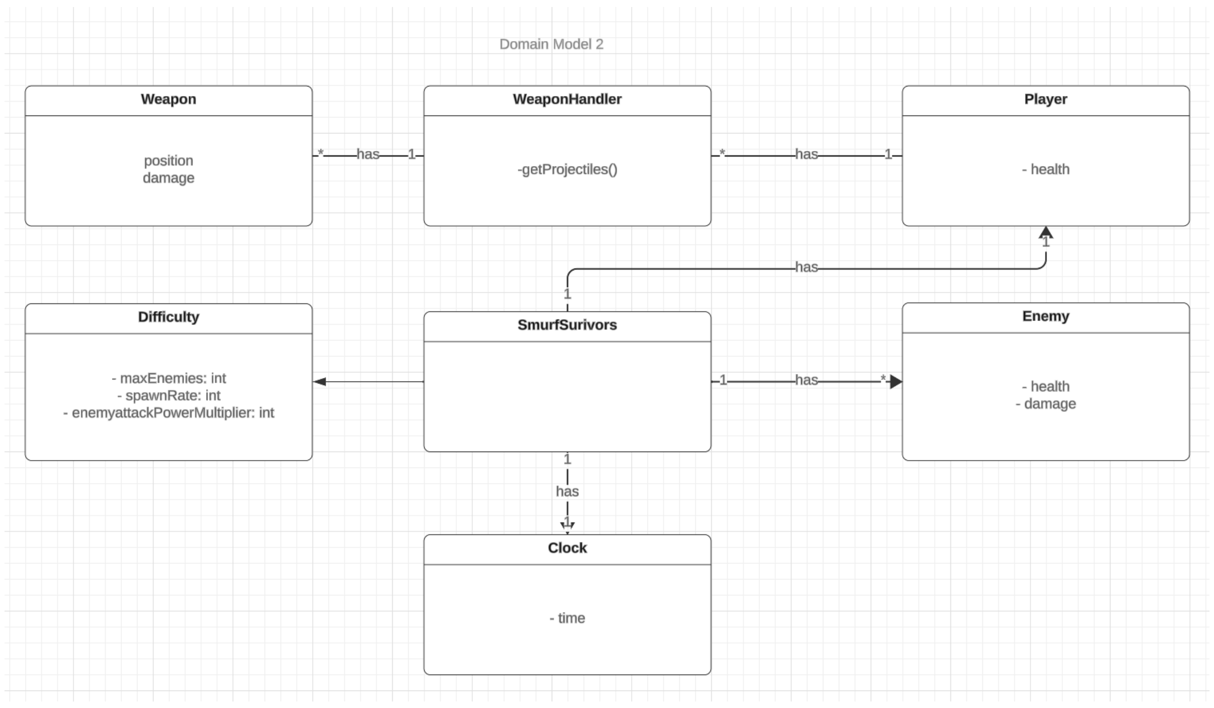


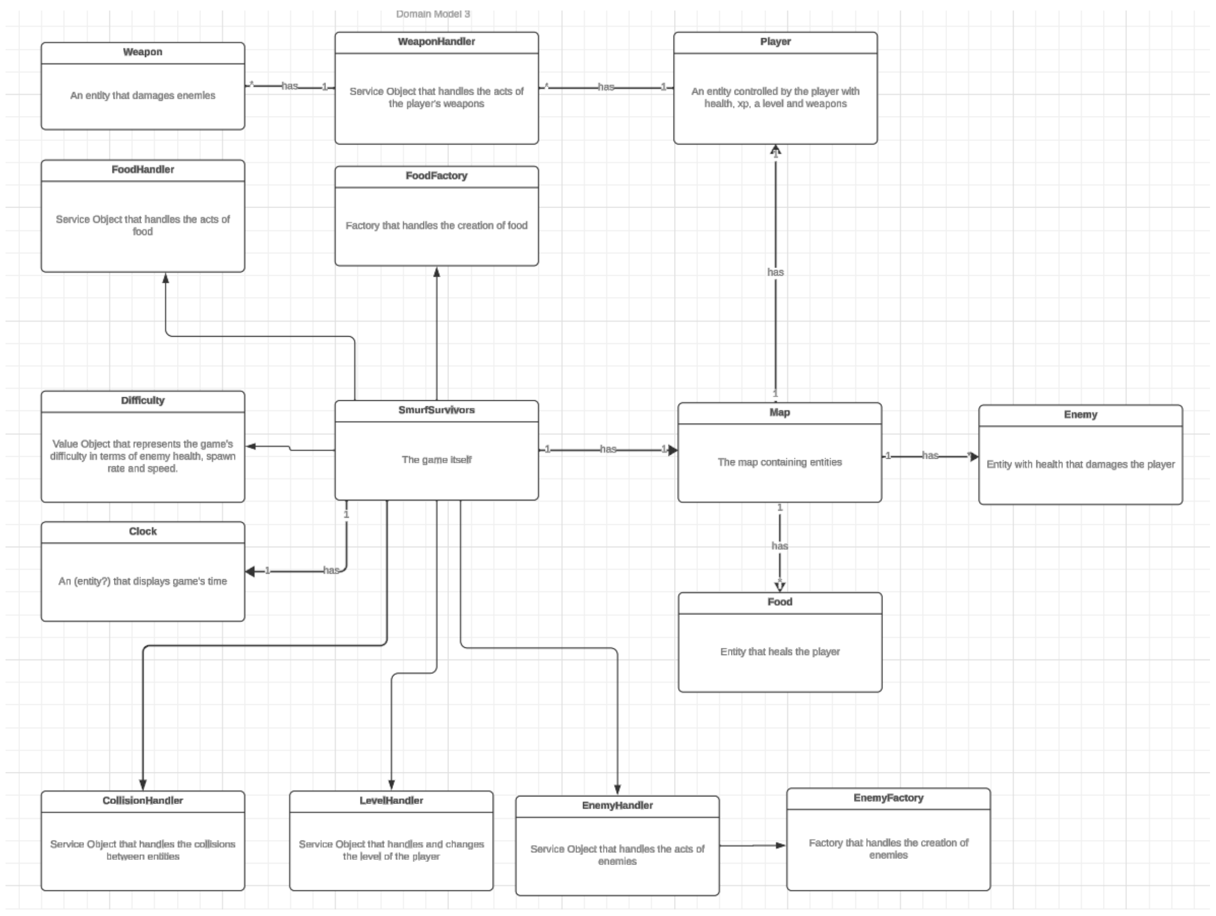


3. Domain model

The following three images represents the three main iterations of our domain model for our game giving a overhead representation of the applications classes and its relations.




Domain Model 2
Weapon
position
damage
WeaponHandler
-getProjectiles()
Player
- health
Difficulty
- maxEnemies: int
- spawnRate: int
- enemyattackPowerMultiplier: int
SmurfSurivors
Enemy
- health
- damage
Clock
- time
has
*
1


Domain Model 3
Weapon
An entity that damages enemies
WeaponHandler
Service Object that handles the acts of the player's weapons
Player
An entity controlled by the player with health, xp, a level and weapons
FoodHandler
Service Object that handles the acts of food
FoodFactory
Factory that handles the creation of food
Difficulty
Value Object that represents the game's difficulty in terms of enemy health, spawn rate and speed.
SmurfSurvivors
The game itself
Map
The map containing entities
Enemy
Entity with health that damages the player
Clock
An (entity?) that displays game's time
Food
Entity that heals the player
CollisionHandler
Service Object that handles the collisions between entities
LevelHandler
Service Object that handles and changes the level of the player
EnemyHandler
Service Object that handles the acts of enemies
EnemyFactory
Factory that handles the creation of enemies
has
*
1

3.1 Smurf Survivors

Represents the game “itself”, acting as the center point of the application. This is what the user will be focused on.

3.2 Map

Representation of the “game board”, the visual and logical representation of the game world in which the player and enemies wander upon.

3.3 Enemy

Represents the entities responsible for moving around and damaging the player

3.4 Food

Responsible for increasing the health of the player. Is randomly stored on the map for the player to pick up.

3.5 Player

Responsible for keeping track of the players data such as health, xp and position. Also contains some functionality for manipulating this data.

3.6 Weapon Handler

Responsible for keeping track of weapons linked to a specific entity. Such as the player.

3.7 Weapon

Responsible for keeping track of data related to a weapon, such as damage and fire rate

3.8 Food Handler

Responsible for spawning Food on the map if conditions allow it to.

3.9 Difficulty

Contains data related to the difficulty of the game such as spawn rate, damage and speed multiplier for enemies.

3.10 Clock

Responsible for keeping track of the time in which the game has progressed. Also contains functionality for manipulating this data.

3.11 Collision Handler

Responsible for looking for collision between entities in the game and call relevant methods when detected

3.12 Level Handler

Responsible for keeping track of the players level, and giving the player upgrades and new weapons.

3.13 Enemy Handler

Responsible for spawning and deleting enemies from the map

3.14 Enemy Factory

Responsible for creating different types of Enemy objects

4. References

LibGDX: Library used for rendering the game.
JUnit: Library used for testing methods in codebase.