Requirements and Analysis Document for Generic Platformer

Oscar Arvidson, Sam Salek, Anwarr Shiervani, Erik Wetter, Malte Åkvist
Group 5: Hmmm

October 23, 2020
version 2.0

Contents

1 Introduction

The aim of this project is to create a 2D platformer game. It's a single-player game where the player can use a multitude of weapons to fight off different kinds of enemies. The game progress is built up using multiple levels and wave logic. This means when the current wave of enemies is defeated a new wave will appear.

The purpose of this project is to entertain our users and create stress relief in their everyday life. Generic platformer is supposed to create a fun and relaxing atmosphere, unlike competitive games which can form a very stressful environment at times. To achieve these goals and still build an enjoyable game to play the project used a cartoon-like design rather than a more realistic design.

1.1 Definitions, acronyms and abbreviations

Wordlist:

- User story: Description of a software feature from an-end user perspective. A user story describes the type of user, what they want and why. User stories help to create a simplified description of a requirement.
- Enemies: An in game character that follows the player and tries to deal damage to the player.
- Wave: The enemies are placed in the game in groups. Each group has a set amount of enemies in it. An individual group of enemies is referred to as a wave.
- Levels: Different worlds/environments the player can choose between to play on.
- Platformer: A platformer is a video game in which the game-play revolves heavily around players controlling a character who runs and jumps onto platforms, floors, ledges, stairs or other objects depicted on a single or scrolling (horizontal or vertical) game screen.
- Player: The character you play as.
- Block: An object the player can place down on the level.
- Weapon: An object used by the player to kill enemies.
- Entity: An object in the game world with physics properties.
- Hitbox: An invisible box that surrounds entities. When this box touches other entities it detects a collision with the entity.
- User: A person using the application.

2 Requirements

2.1 User Stories

Story Identifier: GPG001

Implemented: Yes

Story Name: Basic World

Description: As a user I need to see a world in the game so that I have something to explore and interact with. A game world would let me see how I can interact when I'm in the game for example seeing where I can go.

Confirmation:
Functional:

- The player needs to be able to stand on some kind of ground.
- The game world should have 2-3 platforms.
- Platforms should have a collidable hitbox.

Non-functional:

- The game world's terrain should be built up using a grid.
- The game world will have three different texture tiles.

Story Identifier: GPG002

Implemented: Yes

Story Name: Player and Movement

Description: As a user I want to be able to see my character and to control my character by moving and jumping in order to make progress in the game.

Confirmation:
Functional:

- Have a movable player.
- The player moves right when the user presses the D key.
- The player moves left when the user presses the A key.
- The player jumps when the user presses the W key.

Non-functional:

Story Identifier: GPG003

Implemented: Yes

Story Name: Enemies

Description: As a user I want different types of enemies to appear in the game world and follow the player. Three different types will create a bigger variation for my gameplay and with enemies following me I will get a reason to move.

Confirmation:
Functional:

- Have three different types of enemies.
- Enemies will run towards players.
- If an enemy reaches the player, it will stop.
- The enemy will primarily follow the player in the x-direction.

Non-functional:

- Enemies should look different in order to tell them apart.

Story Identifier: GPG004

Implemented: Yes

Story Name: Player Shooting

Description: As a user I want to be able to shoot possible threats e.g enemies, in order to defend myself.

Confirmation:
Functional:

- The user aims with the mouse.
- The user shoots by clicking the Mouse1 button.
- The weapon can be reloaded.
- The player can only shoot when the weapon has reloaded.

Non-functional:

Story Identifier: GPG005

Implemented: Yes

Story Name: Weapon Projectiles

Description: As a user I want the projectiles fired from my weapon to move in the direction I am aiming to prevent enemies from coming too close to me.

Confirmation:
Functional:

- The projectile travels towards the designated target selected by the user.
- The projectile travels forward in the angle that has been calculated based on user input.
- After the projectile has traveled for more than 3 seconds it is destroyed.
- The projectile collides with enemies and pushes them away.
- The projectile travels at a constant speed.

Non-functional:

Story Identifier: GPG006

Implemented: Yes

Story Name: Jump Collision Reset

Description: As a user I want my character to regain the ability to jump when touching the ground so that I can jump more than one time during the game and not jump while I am in the air.

Confirmation:
Functional:

- Regain the ability to jump when touching the ground/platforms.
- Stop the player from being able to jump when in the air.
- Prevent the player to jump up and sit on the walls.
- Do not allow the player to climb walls when the user is spamming the jump button.

Non-functional:

Story Identifier: GPG007

Implemented: Yes

Story Name: Collision Detection

Description: As a user I want the game to react when e.g my character shoots an enemy or when an enemy hits me, so that an enemy and a player can take damage/lose health.

Confirmation:
Functional:

- Collisions between the player and enemies should lower players' health.
- Collisions between enemies and projectiles should lower the enemy's health.
- Collisions between enemies and projectiles should remove projectiles.
- Collisions between the player and platforms should reset the player's amount of jumps.

Non-functional:

Story Identifier: GPG008

Implemented: No

Story Name: Basic Wave Manager

Description: As a user I want to be able to fight waves of enemies so that there is a challenge and goal in the game.

Confirmation
Functional:

- A new wave will spawn x amount of different enemies (depending on which wave it is).
- Enemies will spawn at the edges of the map.
- The higher the wave, the more enemies will spawn.
- When the timer for each wave runs out a new wave will spawn.
- When all enemies in a wave are dead a new wave will spawn.

Non-functional:

Story Identifier: GPG009

Implemented: Yes

Story Name: Main Menu

Description: As a user I want to be able to see a main menu so that I can get an overview of the game before it starts.

Confirmation
Functional:

- When you press the play button, the game will open up the play menu.
- The player will be able to select a level in the play menu.
- When you press the settings button, the settings menu will open.
- When you press the exit button, the game will exit.

Non-functional:

Story Identifier: GPG010

Implemented: Yes

Story Name: Enemy Pathfinding

Description: As a user I want enemies to be able to traverse the map in order to reach the player and follow it if it moves from one place to another.

Confirmation
Functional:

- Enemies should walk towards the player and stop when the player is reached.
- Enemies shouldn't get stuck on obstacles.
- Enemies shouldn't walk through the map or obstacles to reach the player.
- Enemies shouldn't fly or jump uncontrollably.
- Enemies should take into account the players X and Y coordinates.

Non-functional:

Story Identifier: GPG011

Implemented: Yes

Story Name: Player Building

Description: As a user I want to be able to place blocks that can defend me from enemies in order to enhance my chances to complete the next wave.

Confirmation
Functional:

- The player can only build within a certain amount of tiles (a building range).
- You can only place a block on a tile that is connected to the ground or another building block.
- Blocks can only be placed on one of the tiles inside the map.
- The blocks can be destroyed by enemies when collided with.

Non-functional:

Story Identifier: GPG012

Implemented: Yes

Story Name: Build View

Description: As a player I want to be able to see where I can build in the form of a grid.

Confirmation
Functional:

- The grid the user can see is a transparent overlay covering certain tiles.
- The tiles the grid covers is the player's buildrange.
- This grid is constantly updated, following the player around.

Non-functional:

Story Identifier: GPG013

Implemented: Yes

Story Name: Weapon Selection

Description: As a user I want to have different kinds of weapons with more damage

Confirmation
Functional:

- Have at least 3 types of different weapons
- Different weapons have different damage.
- Different weapons have different projectiles.
- A player can only hold x amount of weapons at one time and change between them.

Non-functional:

Story Identifier: GPG014

Implemented: Yes

Story Name: UI

Description: As a user I want to be able to see the necessary information about my player as well as my progress in the game while I am playing. In order to get a better understanding of how the game is going for me.

Confirmation
Functional:

- The health bar changes when the player gains/loses HP.
- The wave indicator is incremented whenever the user completes a wave.
- A counter displays the amount of ammunition the player has left.
- A text indicates which weapon is selected by the user.

Non-functional:

Story Identifier: GPG015

Implemented: Yes

Story Name: Enemy Tile Destroy

Description: As a user I want enemies to attack and destroy tiles I have placed to make the game more challenging.

Confirmation
Functional:

- Enemies can destroy tiles placed by the user when they collide with blocks.
- The predefined tiles in the level varies depending on level.

Non-functional:

Story Identifier: GPG016

Implemented: Yes

Story Name: Various Worlds

Description: As a user I want to be able to be in various worlds (levels) in the game, with different tiles and backgrounds.

Confirmation
Functional:

- The environment of the game world changes every time the user switches between a level.
- Different enemies destroy tiles with different speeds (depending on their attack).

Non-functional:

- The game world's terrain should be built up with the use of different tiles (e.g. grass tiles).

Story Identifier: GPG017

Implemented: Yes

Story Name: Exit Menu

Description: As a user I want to be able to pause the current game by pressing the Escape button. After that, I can decide whether or not to return to the main menu.

Confirmation
Functional:

- The user can press the Escape button to pause the game. After that the user will be given an option to return to the main menu.

- The user can resume the game by pressing the Escape button again.

Non-functional:

- Game is paused when the exit menu is opened.

Story Identifier: GPG018

Implemented: Yes

Story Name: Movement Animation

Description: As a user I want to see an animation when my character and the enemies are running or jumping, in order to get the visual information of which action the player or the enemies are performing currently.

Confirmation
Functional:

- Have a sprite of the player running when it is running.
- Have a sprite of the player jumping when it's jumping.
- Have a sprite of the enemy running when it is running.
- Have a sprite of the enemy jumping when it is jumping.

Non-functional:

Story Identifier: GPG019

Implemented: Yes

Story Name: Game Settings

Description: As a user I want to be able to change the characteristics of the game through a settings menu. As a user this will make it easier to see all my options and alternatives I have in the game and make me have more control over my experience.

Confirmation
Functional:

- The user can rebind a key when the user presses a control button in the settings menu.

Non-functional:

Story Identifier: GPG020

Implemented: No

Story Name: Audio

Description: As a user I want to be able to hear sounds from various sources in the game while playing it.

Confirmation
Functional:

- Enemies make sounds.
- Jumping makes a sound.
- Placing blocks makes a sound.
- Firing a projectile makes a sound.
- Taking damage makes a sound.

Non-functional:

Story Identifier: GPG021

Implemented: No

Story Name: Player Pickup

Description: As a user I want to be able to pick up weapons and building blocks from the ground.

Confirmation
Functional:

- The player can pick up weapons and building blocks by walking over them (when they are on the ground as a dropped entity).
- The player can not pick up more weapons than the maximum carry amount.
- The player can not pick up building blocks if the stack is full.

Non-functional:

Story Identifier: GPG022

Implemented: No

Story Name: Player XP System

Description: As a user I want to be able to level up after I kill enough enemies so that I can get stronger and have an incentive to play the game.

Confirmation
Functional:

- The player gets XP by killing enemies.
- The player levels up after enough XP has been reached.

- When the player levels up, it's stats (attack, speed, hp) increases.
- On each level-up it gets progressively more difficult to level up (more and more xp is required to level up again).

Non-functional:

2.2 Definition of Done

For a user story to be defined as done it has to meet all of its own acceptance criteria. All involved code and classes related to the user stories must be documented with java docs, in order to be understood by everyone associated with the project. The user stories acceptances will be tested through concrete test classes, human code evaluation, and user acceptance testing.

2.3 User interface

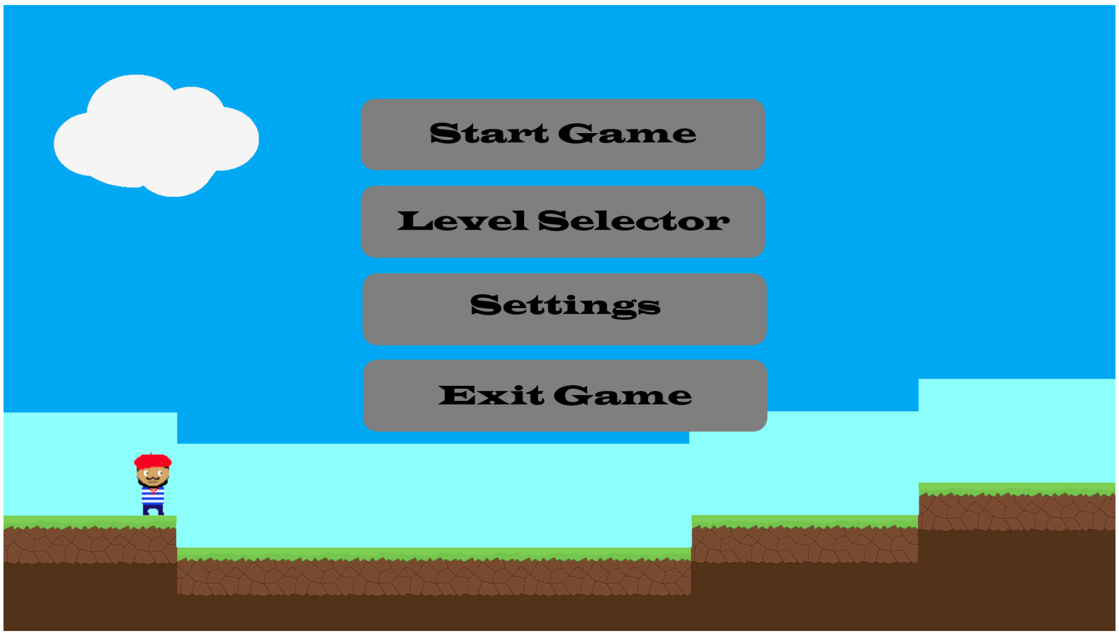

Figure 1: Sketch of main menu.

When the application first starts the user is directed to the main menu of the game. In the main menu there are buttons for selecting a level, changing settings and exiting the game.

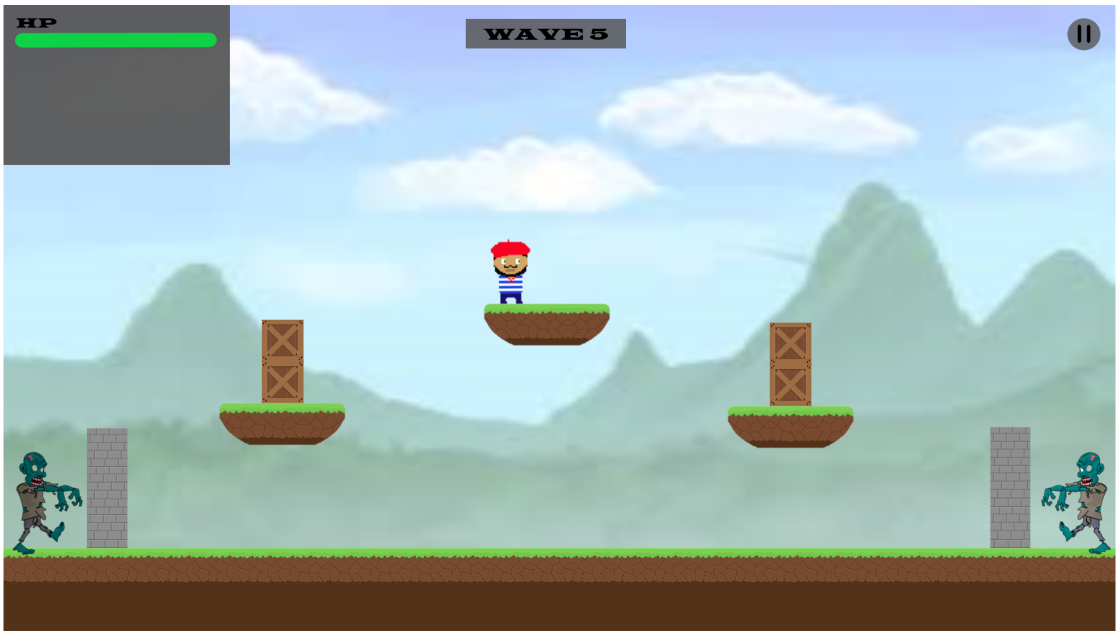

Figure 2: Sketch of in game view.

The gameplay scene is shown after the user selects a level to play. When playing a level the user is for instance able to see the wave the player is on, the player, player's hp and enemies in the game.

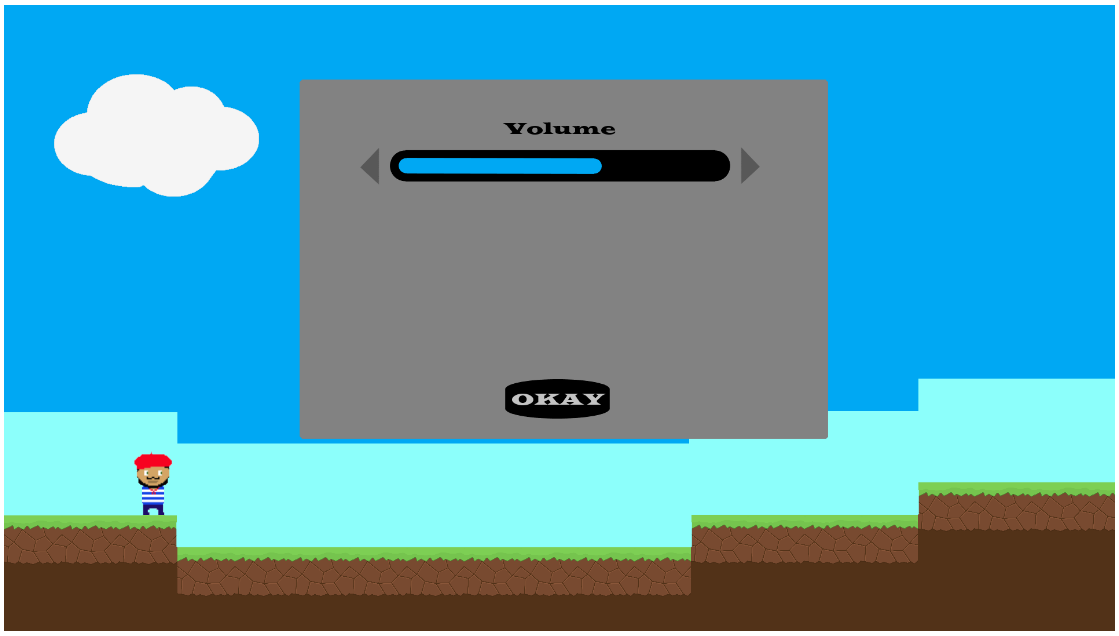

Figure 3: Sketch of settings menu.

Settings scene is shown when the user clicks on the settings button in the main menu. In the settings scene the user is able to change keybinds.

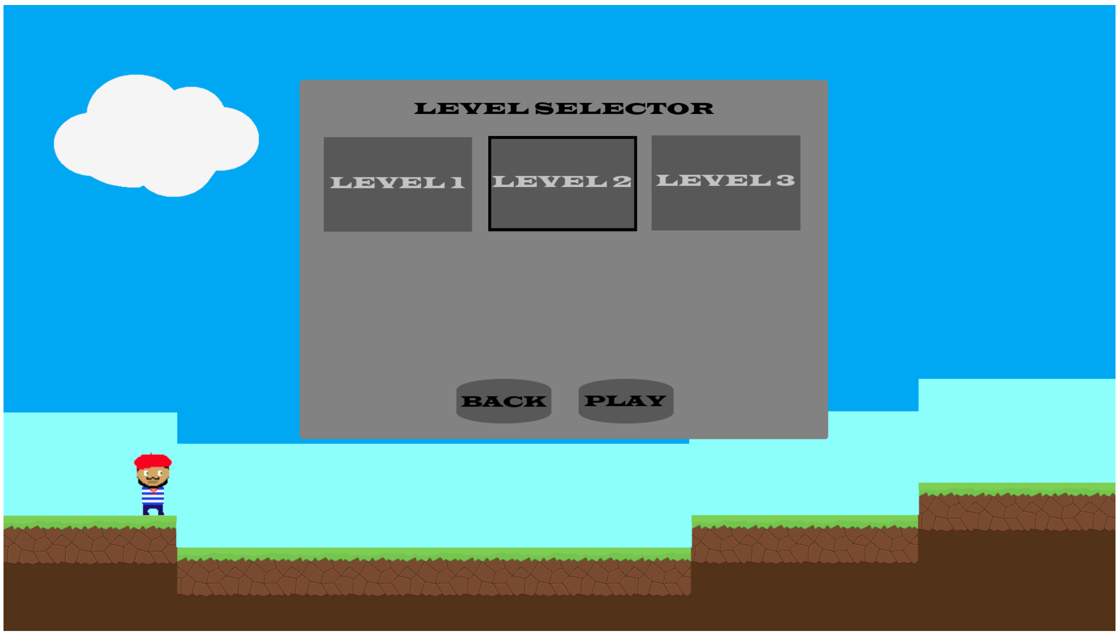

Figure 4: Sketch of level selector view.

The level selector scene is shown when the user clicks on the select level button in the main menu. In the selector scene the user is able to select the level to play.

3 Domain model

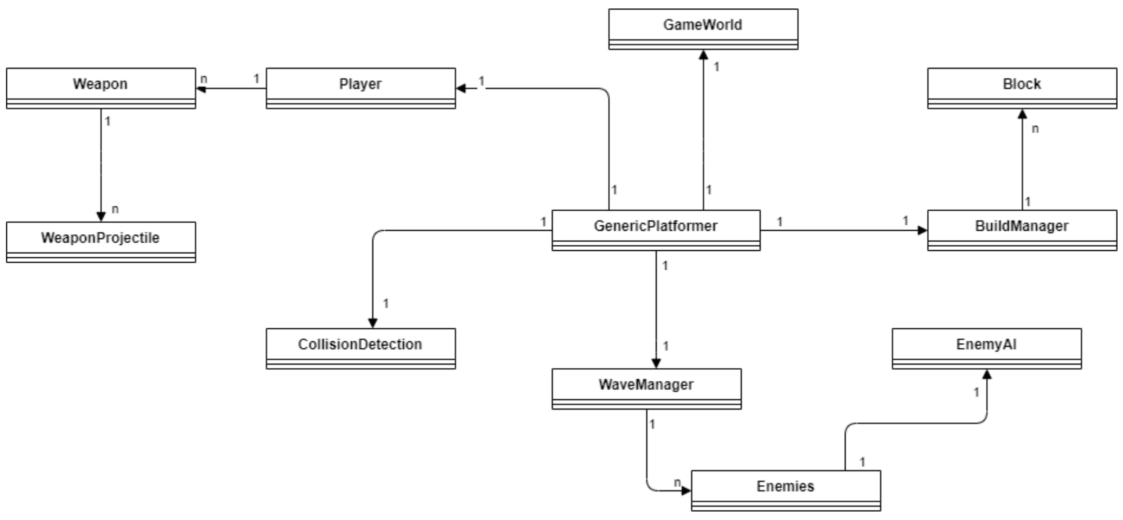

Figure 5: UML of the domain model.

3.1 Class responsibilities

GenericPlatform:
Is used as a root class for the game and the model package. Every data and information passing from the model to either the view or controller should pass through GenericPlatformer in order to achieve low coupling.

BuildManager:
Handles player being able to spawn and place blocks. A block is able to spawn if the player can reach it, the tile to spawn it on is empty, the tile to spawn it on is connected to another block and the block to spawn isn't on the player.

Block:
This class creates a collidable block entity which can take damage and be destroyed.

CollisionDetection:
Manages all collisions between different entities in the game and is responsible for calling all necessary methods used to resolve/react to said collision.

WaveManager:
The class handles waves in the game. It calculates which types of enemies should spawn and creates new waves of enemies. The amount of enemies that spawns depends on which wave it is (the higher the wave the more enemies). It will generate a new wave when all enemies are dead and/or when the timer for the wave runs out.

Enemies:
The class responsible for creating a component with the ability to give an entity all common

attributes and behaviours of an enemy.

EnemyAI:
Manages and contains all functionality for the Enemy AI. Functionality includes moving to the player while he is moving or standing still, jumping over obstacles when path to player is interrupted, traversing floating platforms to reach player, etc.

GameWorld:
This class contains all methods for spawning the game world entities, such as the player and platforms.

Player:
The class responsible for creating a component the ability to give an entity all attributes and behaviours of a player to a specific entity.

Weapon:
The class that is responsible for creating instances of WeaponProjectile as well as managing a "weapon's" ammunition counter and the attributes of different types of "weapons".

WeaponProjectile:
The class that is responsible for creating a projectile entity and calculate the movement direction of this entity.

References

[1] A. Baimagambetov. (2020). Fxgl 11 wiki, [Online]. Available: `https://github.com/AlmasB/FXGL/wiki/FXGL-11` (visited on 09/30/2020).

[2] ——, (2020). Fxgl library, [Online]. Available: `https://github.com/AlmasB/FXGL` (visited on 10/21/2020).