TODO: finish UCs, write non-functional reqs, references(?)

Requirements and Analysis Document for the Infinity Run project (RAD)

Version: 1.0
Date: 2017-04-17
Authors: Jacob Rohdin,
Anders Bäckelie,
Leo Oinonen,
Mikael Törnqvist

This version overrides all previous versions.

1 Introduction

The project aim is to make a game where the goal is to travel as far as you can to the right. When moving you will be facing different hazards such as spikes, moving platforms and even moving AI. You will have basic movements such as moving right and left, ducking and jumping. You will also be able to attack monsters and the monsters will become harder the further you have travelled.

As mentioned earlier the monster will become harder as you get further away from the start point. This is a scaled difficulty which means spikes will deal more damage if you have travelled a long distance and possibly kill you instantly depending on your max HP. The character will have a damage, HP - health points and a movement speed.

The more the more coins, which is this game's currency, you have collected the “stronger” or more advanced your character will become. When you die you will be sent to a shop where you can buy upgrades for the coins you have collected. You will be able to get a higher HP as well as a higher movement speed. You will be able to buy different weapons that will increase your damage.

The app will be a desktop, singleplayer game for the platforms Windows/Mac/Linux and will be using a simple graphical user interface.

General characteristics

- The application will be a single-player 2D platformer-style game, meaning that the player character will have to move in two dimensions (up/down; left/right) in order to traverse a world filled with platforms.

- The world will be procedurally generated using pre-defined building blocks designed to fit together in order to make an “infinitely” large world. The point of origin will always be the same.
- The application will keep track of the player’s “score”. The score is defined as the number of rooms you have traversed since the start of the game. The highest score achieved will be saved as the “high score”.
- In addition to the score, the player will be able to collect coins generated onto the world. These coins will persist through several game sessions, i.e. they are not bound to the current session.
- The game session will end when the player’s health points reach zero, referred to as “death”. This occurs by taking damage, and is caused by various hazards (such as spikes or mobile enemies) generated onto the world.
- Upon dying, the player will have access to the shop. In the shop the player will be able to upgrade their character using the coins collected during the game sessions.
- The application will be using a GUI that will include the world and all its constituents, a coin counter, and a health counter. The shop will have a number of upgrades to choose from, their price, and the coin counter.

1.1 Definitions, acronyms, and abbreviations

Technical definitions

- **GUI** - graphical user interface.
- **Java** - platform independent programming language
- **JRE** - the Java Runtime Environment. Additional software needed to run an Java application.
- **AI -** Artificial Intelligence or in other words bots that are programmed to exhibit situation-dependent behaviour.

Game Definitions

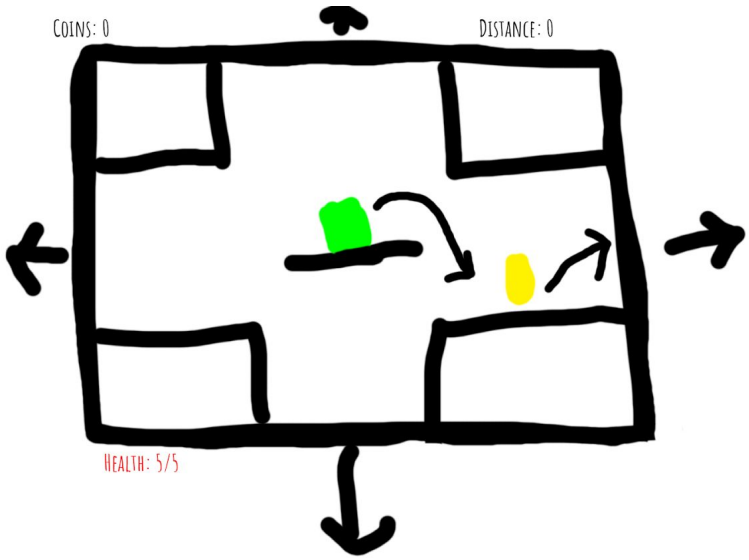
- **HP** or Health Points, the total number of damage something can take before dying.
- **Movement Speed** is the speed which things can move.
- **Coins** which is the currency of the games, used for purchasing upgrades and similar things.
- **Distance** this is how far your character have travelled on the x - axis from the starting point.
- **Room block** the created rooms that the game generates the world with.
- **Death** when your HP reaches zero, you will then be transported to the shop. This only means you have to start from the starting point again - your character will not be removed.
- **Shop** is the place where you can spend your coins for upgrades.
- **Upgrades** will be making your character better through different weapons, more HP and movement speed.
- **Top Score** is the highest distance you managed to travel with your character. The goal is to reach as high a distance as you possibly can.

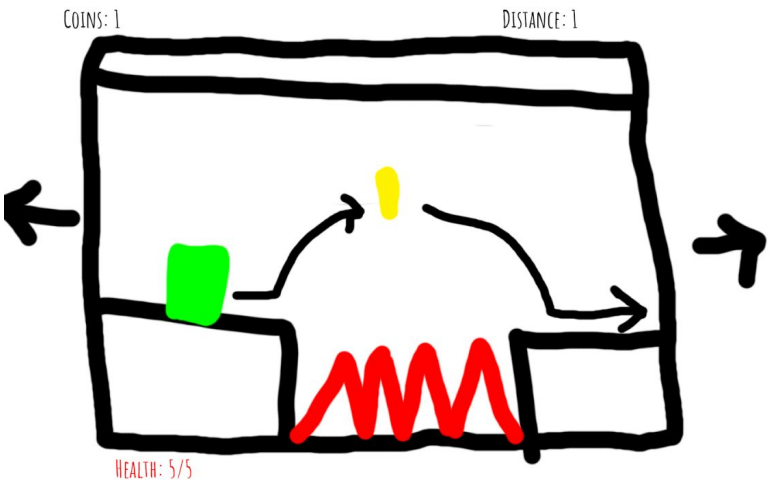
- **Damage** is the number of HP something can deal in one hit.
- **Monsters** will have some sort of AI, they can be killed if you have enough damage or outrun with enough movement speed.
- **Character** is the one you as a player control, you can make the character stronger by collecting coins and then purchasing upgrades.

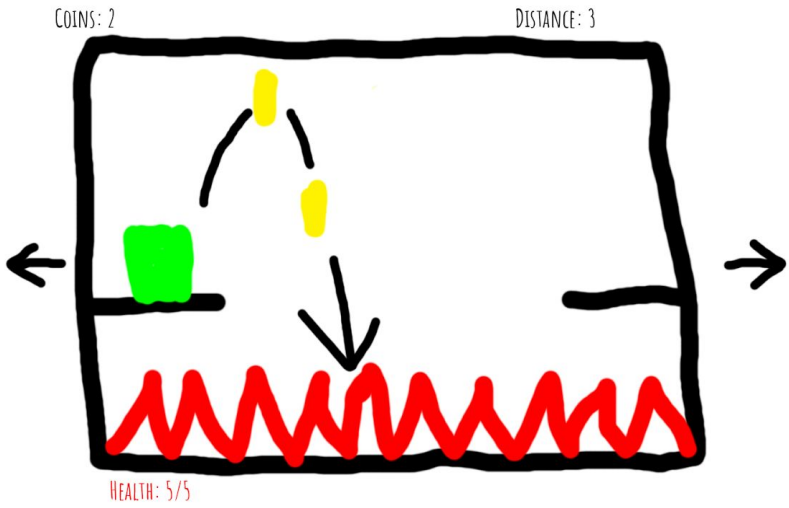
2 Requirements

2.1 Graphical User Interface

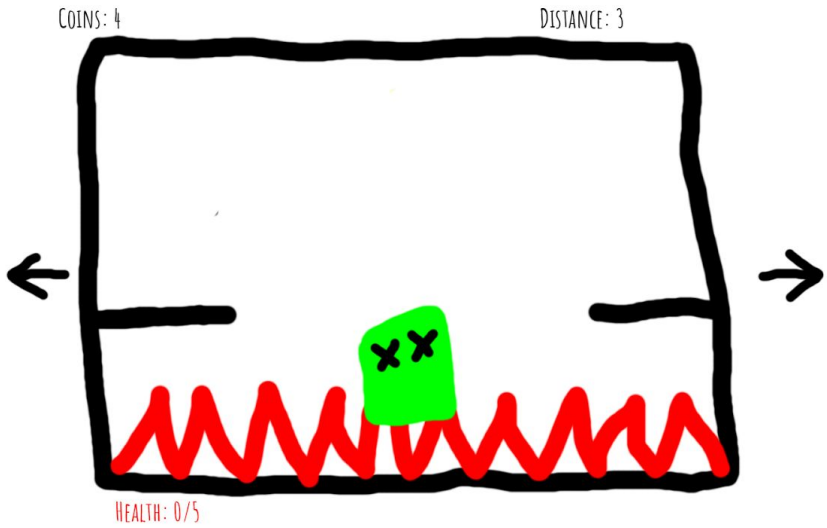
The game GUI will consist of a character, coins, hazards, coin counter, distance counter and health. In the sketches below, the player is green, hazards are red, and coins are yellow.





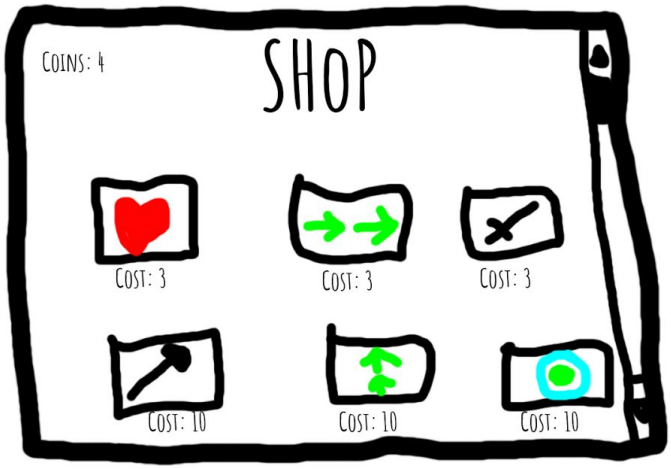


The player jumps for the coins but misses the platform on the other side, they hit the deadly hazard and lose all their HP.



As the player died, they are taken to the shop. Below is a simple sketch of what the shop

might look like:



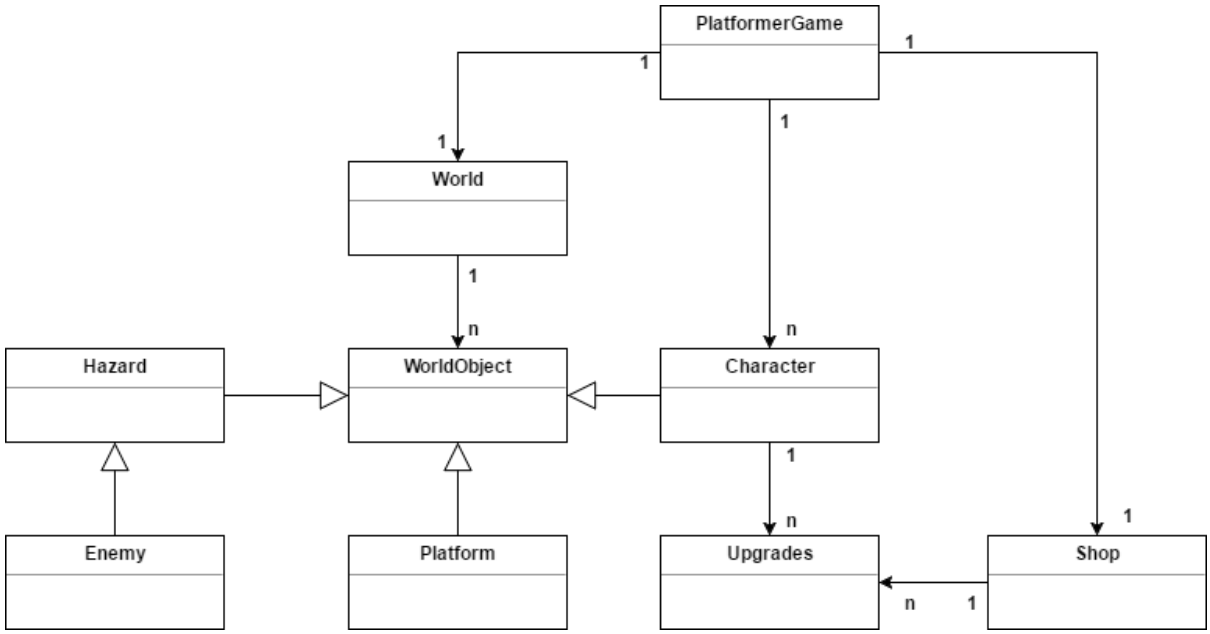
2.2 Functional Requirements

1. Purchase and browse upgrades with accumulated coins
2. Start the game at the beginning area
3. Traverse the world:
    - The player can move
    - The player can collect coins
    - The player can attack monsters
    - The player will be able to get hurt by monsters
    - Generate following rooms when a room is entered
    - Score points based on distance.
4. Death upon losing all HP
    - This returns you to the upgrade screen.

2.3 Non-functional requirements

3 Use cases

4 Domain Model



4.1 Class Responsibilities

PlatformerGame is the main class we have when we open the game. The player will choose which character he/she wants to play on and then the main class will call the world or shop depending on the player's choices and if the world is picked then a new game will start.

The World class is kind of a object container. It will have a list of all the objects in the world, so we can display them properly for the players and some objects may just be used for collisions.

WorldObject is the "base" class in the visible objects hierarchy. It will have some basic functions like Position and a method Render.

Hazard extends WorldObject and acts similar to it but a Hazard is supposed to represent spikes or similar, something that can damage the character. Objects like this is better treated separately because we will make different things if the player intersect with one of these objects.

Enemy extends Hazard. The main reason that enemy is not represented as a hazard is because the enemy will have some kind of AI that controls it and may require some kind of special framing, etc a 15FPS clock to control the AI with.

Platform extends WorldObject, Platform is just a simple object in the world but it has the ability to have intersections in different directions. For example one platform could have a

intersection with the character if the player is falling but not if he is jumping, meaning that the character may pass through it from the underside.

Character extends WorldObject. The character class will represent the actual player in the world. The Characters movement will be controlled by the keyboard and/or mouse. The character class will also have a List with upgrades that this Character has previously bought. The first character is picked by the player at the game startup and is then sent as a reference to the World by the PlatformerGame class.

Shop is a class where the “user” can buy upgrades to its current Character. When one upgrade is bought the Shop will return one Upgrade class instance and the PlatformerGame will add this Upgrade instance to the player.

Upgrade is a class that contains information about an “upgrade” that the player can have. A upgrade could be a sword that deals 3 damage instead of 2 for example.

5 References