Requirements and Analysis Document for BattleRace

October 1, 2020

Filip Nordqvist, Simon Andersson, Viktor Berggren,
Rasmus Otterlind, Gustav Wadström

1 Introduction

Our project idea is to create a two-dimensional racing-game that you can play online with your friends. Our idea that the map is supposed to be randomly generated in the shape of a sine curve. The game is also supposed to have power ups that will be put out on the map and that the player should be able to pick up to get some sort of aid in winning. There's also going to be physics in the game such as gravitation so if you come at high speed on a hill your car will jump. The game is supposed to be played multiplayer/online but can also be played alone if you want to practice or test things. The game can be played by everyone but is targeted towards younger adults and kids.

2 Requirements

2.1 User Stories

The stories are ordered after what priority they have.

2.1.1 Epic

I want to be able to play a game where I compete against my friends in a race.

1. As a user I want to be able to navigate in a menu where I can change settings and start the game.

- The user can change settings
- The user can choose to player singleplayer or multiplayer
- The user can start his own multiplayer game that other people then can join and connect to

1. As a user I want to be able to roam free in the game.

- The use can move backwards, forwards, up and down
- The user can reach a finish line

2. As a user I want there to be obstacles to make it harder.

- The user should be able to avoid obstacles
- The user should get "hurt" is he or she hits an obstacle
- When the user hits an obstacle he or she should get some type of defect from it

2. As a user I want to make it harder for my opponents with the help of power ups.

- Power ups are laid out along the map
- Some help you, others make it harder for your opponent
- Should be generated randomly

2. As a user I want to be able to see my friends drive around by being connected to a server.

3. As a user I want to compare myself to my friends in the form of points or time.

- Add points
- Able to see others' points

3. As a user I want to have sound in the game to improve my experience.

- The user should hear sound effects from the game
- The user should her music in the game

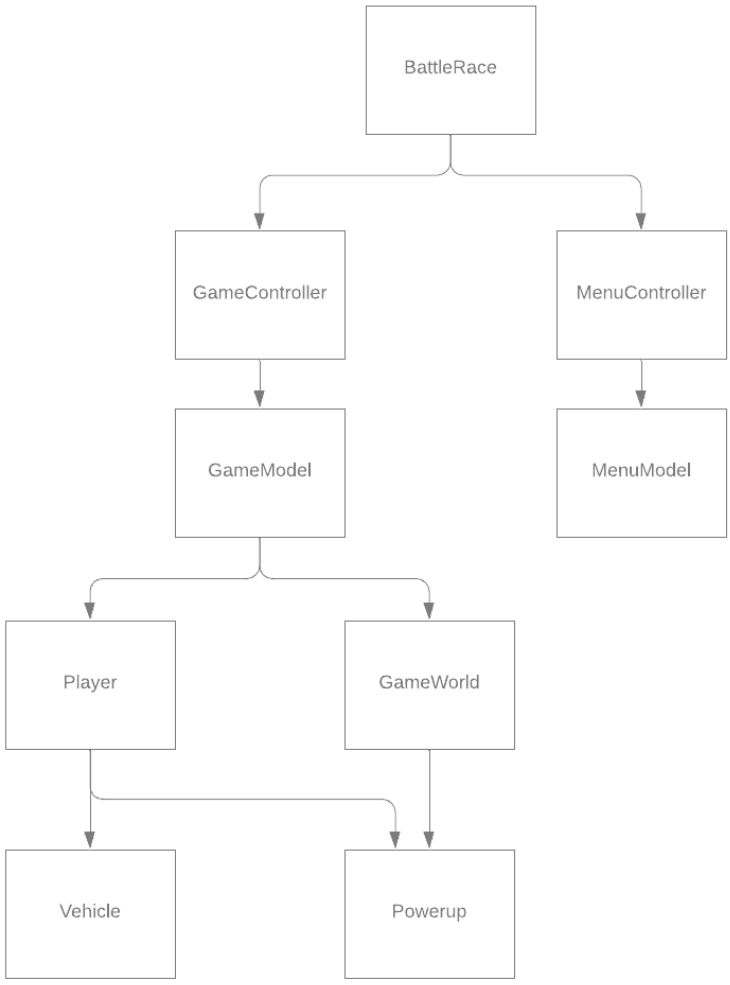
4. As a user I want to be able to chat with my friends..

- The user should be able to send messages
- Others in the session should be able to read the messages

2.2 Definition of done

For a user story to be "done" all the acceptance criteria must have been met. It must have gone through testing and pushed to the master branch. The code must also work for all members who are a part of the creation of the project. The code must also fulfill requirements described in SDD.

3 Domain-model



3.1 Class responsibilities

BattleRace is technically the main-class here, it is not the class that contains the main-method but it can be interpreted as that. *MenuController* is the controller for our menus, it handles switching between different screens in the menu. *GameController* is the class that handles the inputs from the user. It is set as the projects' input processor. *GameController* also creates the *GameModel*.

The *GameModel* holds all the data and also the logic for controlling the data. The *GameModel* also creates the *Player* and the *GameWorld*. *GameWorld*s responsibility is creating the world that the player(s) will race in. It also has access to information about the world such as position of blocks. *GameWorld* also has a responsibility of updating the world when a certain amount of time has passed (currently 60 frames per second) and the objects in it. To clarify it doesn't render anything but it needs to step the world to check for things such as collision detection and constraint solutions. *GameWorld* also generates and creates *PowerUp*s. Now all this is not done by *GameWorld* alone but it creates the classes and calls the methods in those classes that do all this.

The *Player* class creates the *Vehicle* that the player of the game controls. *Player* also has a reference to *Powerup* as the player will be able to pick them up and use them when they want. It does not create any *Powerup*. *Player* also tells the vehicle when the gas or brake is pressed and when the car is turned. It does not handle inputs but it has the methods that call the gas and brake etc. methods in *Vehicle*. The *Vehicle* package contains all the data and logic for the vehicles in the game. This can be what to do when gas is pressed or how much friction the wheels have.