Requirements Document

Game: Temple Run

by

Group 9

Course: TI2206

Software Engineering Methods

Maarten Sijm
Robin van Heukelum
Mathias Meuleman
Mitchell Dingjan
Maikel Kerkhof

TA: Jurgen van Schagen

Teacher: Alberto Bacchelli

1 FUNCTIONAL REQUIREMENTS

For the game Temple Run, the requirements regarding functionality and service are grouped under the Functional Requirements. Within these functional requirements, four categories can be identified using the MoSCoW model[1] for prioritizing requirements: Must haves, Should haves, Could haves and Won't haves.

1.1. Must haves

- The game will start in the main menu when it is launched.
    - A new game run will be started when clicking the 'New Game' button in the main menu.
    - The player will be able to quit the game by clicking on the 'Quit' button both in the main menu.
- The game run will consist of a character running over a randomly generated track that is infinitely long.
    - The player character will be viewed from third person.
    - The track will consist of three lanes.
- The player will:
    - start in the middle lane.
    - be able to change lanes by pressing certain keys on the keyboard.
    - not be able to move outside of the outer lanes.
    - be able to collect items (e.g. coins) on the tracks.
- The lanes will have obstacles that the player has to avoid.
    - The obstacles on the tracks will be randomly generated while running.
    - Crashing into an obstacle will result in death.
    - After death, a pop-up will be shown, containing:
        - The Player's final score.
        - A 'Try again' button, which will start a new game run when being clicked.
- The game will initiate and show the player's score at 0 at each run.
    - Running further will increase score.
    - Picking up items will increase score.
- The game will initiate and show the player's amount of coins at 0.
    - Picking up coins will increase the amount of coins.

[1] http://en.wikipedia.org/wiki/MoSCoW_method

1.2. Should haves

- The game shall have a pause menu during the game which can be viewed by clicking on the “Pause” button. In the pause menu the player can:
    - resume the game.
    - quit the game.
    - enable/disable both the soundtrack and sound effects.
- Players shall be able to pause / resume the game by clicking on the pause button that’s available during the runs.
- The game shall have a soundtrack.
- The game shall have a high-score list which can be viewed by clicking on the ‘View high-score list” button in the main menu.
- The game shall have an option menu that can be viewed by clicking on the “Options” button in the main menu, which contains all options about the game (e.g. screen resolution).
- The game shall display a distance meter, keeping track of the distance ran by the player.
- The game shall slowly increase the speed of the character over time.

1.3. Could haves

- The game could have a store which can be entered by clicking on the ‘Store’ button in the main menu. In this store the player could:
    - ‘buy’ alternate appearances of the character with coins gathered while playing.
    - ‘buy’ soundtracks with coins gathered while playing.
    - (directly) alter the appearance of the character.
    - (directly) alter the soundtrack of the game
- The amount of coins and bought appearances/soundtracks could be saved to an encrypted file.
- The Player could be able to pick up power-ups.
    - Some obstacles could be able to be destroyed with a power up which will raise the score.
- The game could have sound effects when crashing and picking up items.
- The game could have a storyline that will be told while playing the game.
    - (Suggestion: A story in which the main character has made the ducks of a certain nest angry and are therefore chasing him over the track.)
- The soundtrack shall be played faster as the running speed increases.

1.4. Won’t haves

- The game won’t have a multiplayer mode.

2 NON-FUNCTIONAL REQUIREMENTS

Besides the provided functionality and services, design constraints need to be included in the requirements specification as well. These requirements do not indicate what the system should do, but instead indicate the constraints that apply to the system or the development process of the system.

- The game shall be playable on Windows (7 or higher), Mac OS X (10.8 and higher), and Linux.
- The game shall be implemented in Java.
- A first fully working version of the game shall be delivered at September 11, 2015, before 23:55.
- For the iterations after the delivery of the first fully working version, the Scrum methodology shall be applied.
- The implementation of the game shall have at least 75% of meaningful line test coverage (where meaningful means that the tests actually test the functionalities of the game and for example do not just execute the methods involved).
- The development team will use a number of tools:
    - IDE (Eclipse/IntelliJ)
    - JUnit
    - Git(Hub)
    - Apache Maven
    - Travis
    - Waffle
    - Slack
    - Checkstyle
    - Findbugs
    - PMD