Requirements

Project Runner [wip]

Group 5

TI2206 Software Engineering Methods
of the Computer Science curriculum
at the Delft University of Technology.

L. van Aanholt levi.vaan@gmail.com 4084101
M. Wolting matthijswolting@gmail.com 4356063
M.E. Kuipers marcelkuipers1996@gmail.com 4389042
N.E. Hullegien nilshullegien@gmail.com 4389069
H.M. Yeh H.M.Yeh@student.tudelft.nl 4386027

Contents

1	**Functional requirements**	**2**
	1.1 Must haves	2
	1.2 Should haves	2
	1.3 Could haves	3
	1.4 Would/ Won’t haves	3
2	**Non-functional requirements**	**4**

1 Functional Requirements

For the Project Runner game the MoSCoW model was used to define and prioritize the requirements regarding the system functions and services of the game:

1.1 Must haves

-The game shall show the main menu from which the game can be started.
-The player shall be able to start the game from the main menu
-The player shall be able to quit the game from the main menu
-The player shall be able to perform the jump action by pressing the jump key.
-The game shall show the environment scrolling to the left while the player keeps the same horizontal position.
-The game shall have a 2D gravity logic that pulls the player down if he isn't on the gameplane or an obstacle
-The game shall show the level containing various obstacles and voids.
-The player shall be able to jump over an obstacle to avoid it
-The game shall place obstacles at intervals in front of the player.
-The player loses the game by falling into a void or colliding with an obstacle
-The game shall go into a game over state when the player loses the game.
-From the game over state the player can return to the main menu.

1.2 Should haves

-The game should keep track of the score of the player.
-The player should earn points for the amount of distance traveled
-The player should earn a predefined number of points for avoiding an obstacle.
-The game should endlessly loop the level over and over again
-The game should increase the speed of the player over time
-The game should show the obstacles not at set intervals, but at random intervals.
-The game should show obstacles that is able to shoot projectiles at the player.
-The game should kill the player when the player collides with a projectile.
-The player should be able to jump for longer distances by holding the jump action key for a longer time as well as jumping shorter distances by holding the jump action key for a shorter time.
-The player should be able to end his game and as a result will lose the score of the ended game.
-The game should show the score of the player when the player loses the game.
-The game should show a player animation for running and jumping
-The game should show the transition of running to jumping and the other way around through an animation.
-The player should be able to pause the game by pressing a predefined key.
-The player should be able to resume the game by pressing a predefined key.

1.3 Could haves

-The game could have obstacles that can be avoided by performing the slide action.
-The player could be able to perform the slide action by pressing the slide key
-The game could save the five highest scores that has been achieved by the player
-The player could be able to see the high scores from the main menu
-The game could show the high scores in descending order.
-The game could replace one of the high scores with a new score of the player at the right ordered place if the new score is higher than any of the previous high scores.
-The game could have pickups, like the jetpack and weapons
-The player could pick up a pickup by colliding with it
-The player could be able to acquire a jetpack as a pickup while playing the game
-The player could be able to use the jetpack by pressing and holding the jump key.
-The game could show the amount of fuel that the jetpack contains.
-The game could show the player having an upward motion, while the jetpack is used.
-The game could remove the jetpack from the player when the jetpack fuel is depleted.
-The player could acquire a weapon as a pickup while playing the game
-The player could shoot the weapon by pressing the corresponding shoot weapon key.
- The player could be able to destroy obstacles by hitting the obstacle with the weapon.
-The game could remove destroyed obstacles.
-The player could earn a predefined number of points for destroying an obstacle.
-The player could be able to disable certain obstacles by hitting these obstacles with the weapon.
-The game could stop a disabled robot from shooting projectiles
-The game could procedurally generate itself at run-time.
-The player could be able to choose between different playable characters
-The game could show a visual representation of the theme of the game in the main menu.
-The game could show a revelatory message when the player has achieved a new high score.
-The player could be able to go back to the main menu from the revelatory screen.
-The game could use sound effects.
-The player could be able to mute and unmute the sound.

1.4 Would/Won’t have:

-The game won’t have multiple start setups at the start of the gane with benefits and negative properties
-The game won’t have RPG elements

2 Non-functional requirements

Next to the functional requirements, the game also has to comply to certain constraints regarding the development process as well as the system which will have to be met:

- A basic working prototype is finished on September 11 2015 23:55 as the master of the repository
- The final product will be finished on October 30 2015 23:55 as the master of the repository
- For this project the Scrum paradigm of design will be used with weekly sprints (starting on monday, ending on friday) where at the end of a sprint there are working results
- Will have the code of the project on a public github repository
- The game will be coded in java
- All the resources used in the end product must have a free to use license
- The visual theme of the game is escaping robot spaceships
- There will at least be a meaningful test coverage of 70%