Software Requirements Specification
For 2D Game: BroncoRun

Prepared by Pratik
Saurabh,
Radhika,
Poojan

Version 1.0

Santa Clara University

March 15, 2018

1. Introduction

1.1 Purpose

To design and develop a 2D game in JAVA which will implement most of the object oriented concepts along with multi- threading. The project will demonstrate use of JAVA swing elements.

1.2 Document Conventions

Every requirement statement is to have its own priority in this SRS.

1.3 Intended Audience and Reading Suggestions

The document is intended for developers, project managers, marketing staff, users, testers, and documentation writers. The rest of this SRS contains requirements of our project in detailed sections generalized by the titles. We suggest reading the document starting with the overview sections and proceeding through the sections that are most pertinent to each reader type.

1.4 Product Scope

The scope of the project is limited to single level in game. We will be using a single map and it will be a single player game. The game will have just one type of enemy, which on collide will reduce the health of the player. Here we are basically trying to showcase our understanding of

OOADP concepts by structuring classes and using proper data structures and design patterns where ever needed.

This application is developed using JSwing components and is a stand-alone application which is not hosted anywhere, and multiple users cannot collaborate to this application through web. Hence, to sum it up this is a standalone application which is scoped to a single user.

1.5 Glossary of terms:

Term	Description
Game	The 2D game that this project implements
Player / Bronco	The main character of the game
Enemy	The other character which when collides with player reduces players health
Map	Collection of image blocks on which the player will be moving
Game state	Whether it is in menu or game mode
Collide	Intersection of player with enemy

2. Overall Description

2.1 Project Perspective

The game will have a menu which will allow user to select from following three categories: Play, Help or Exit. On selecting Play, a map will be loaded on a JPanel with a background.

The origin of the project is based on a single user game where the character 'Bronco'- derived from sprites is expected to move forward, backwards, jump and glides. The map is designed from blocks of images where some blocks are opaque and some are transparent. Bronco should jump over the blocks of maps which are opaque, whereas Bronco should be able to pass through the transparent ones.

The game also has enemies which moves backward and forward. If Bronco collides with the enemies, his health will be reduced. Although, Bronco can attack the enemy through scratching in which case the enemy dies(disappears), as we have not maintained the health of the enemy.

2.2 Product Functions

The primary functions are:

- Allow user to select menu
- Load background image and map as per screen size
- Allow forward and backward movement
- Allow jump on up key pressed
- Allow drop on key release
- Allow player to glide when there is long distance to cover
- If the player is not able to glide through to ass valley, it fells and the game gets over.
- Maintain health of the player
- Player can attack on enemy by scratching
- After the player scratches the enemy, it disappears(dies)
- Player's health decreases on each collision with enemy
- After the game gets over, user is directed to the instructions page.

2.3 User Classes and Characteristics

User has the ability to choose from the menu and once selected to play he will be the controller of bronco. The user will be moving the main character in the game forward or backward. That's the only thing that will be under user control. Rest maps and background and enemies will be loaded by the game.

2.3.1 Provides menu:

As soon as the user loads the game, he will be shown a menu with three options: play, help or exit. The user should be able to navigate through the menu by up and down arrow key and the navigation should be in circular form.

2.3.2 Selects menu:

The user should be able to select from the menu. If he chooses to select play, the game should be started. If he chooses help, a help dialogue box should be displayed with instructions to play the game. and if he clicks on exit, the window should be closed.

2.3.3 Game loads background:

If user selects to play game, it should load the background image according to the size of window. When the player moves back and forth, the background image should also load synchronously giving the user a feel of continuous image.

2.3.4 Game loads map:

If user selects to play, game should load the map for user. The map should consist of opaque and transparent tiles. Transparent tiles should let player pass through them and opaque tiles should black players passage compelling him to jump/glide through it.

2.3.5 Game loads map objects:

If user selects to play, along with background and map, the game should also load the main player bronco, its appropriate sprite (image) and dimensions. It should also load other moving objects like enemies with their image and proper dimensions.

2.3.6 Game maintains game state:

Game state refers to whether the game is in level (play) state or menu state. And depending on the state, the game should load the window size and background.

2.3.7 Game maintains player's health:

The main player will have some amount of health when a new game is loaded and this health will be decreased by certain percent when he hits the enemy. Game should keep a track of players health and should end the game when the health becomes zero.

2.3.8 Player moves (backward/forward)

When player selects to play and the character and map is loaded on the window, the player should be able to move back or front on the press of front and back arrow key. As soon as the arrow key is released, the movement of the player should stop.

2.3.9 Player jumps or plummets

In the play mode, the user should be able to jump on the up-key press. And he should fall down on the key release.

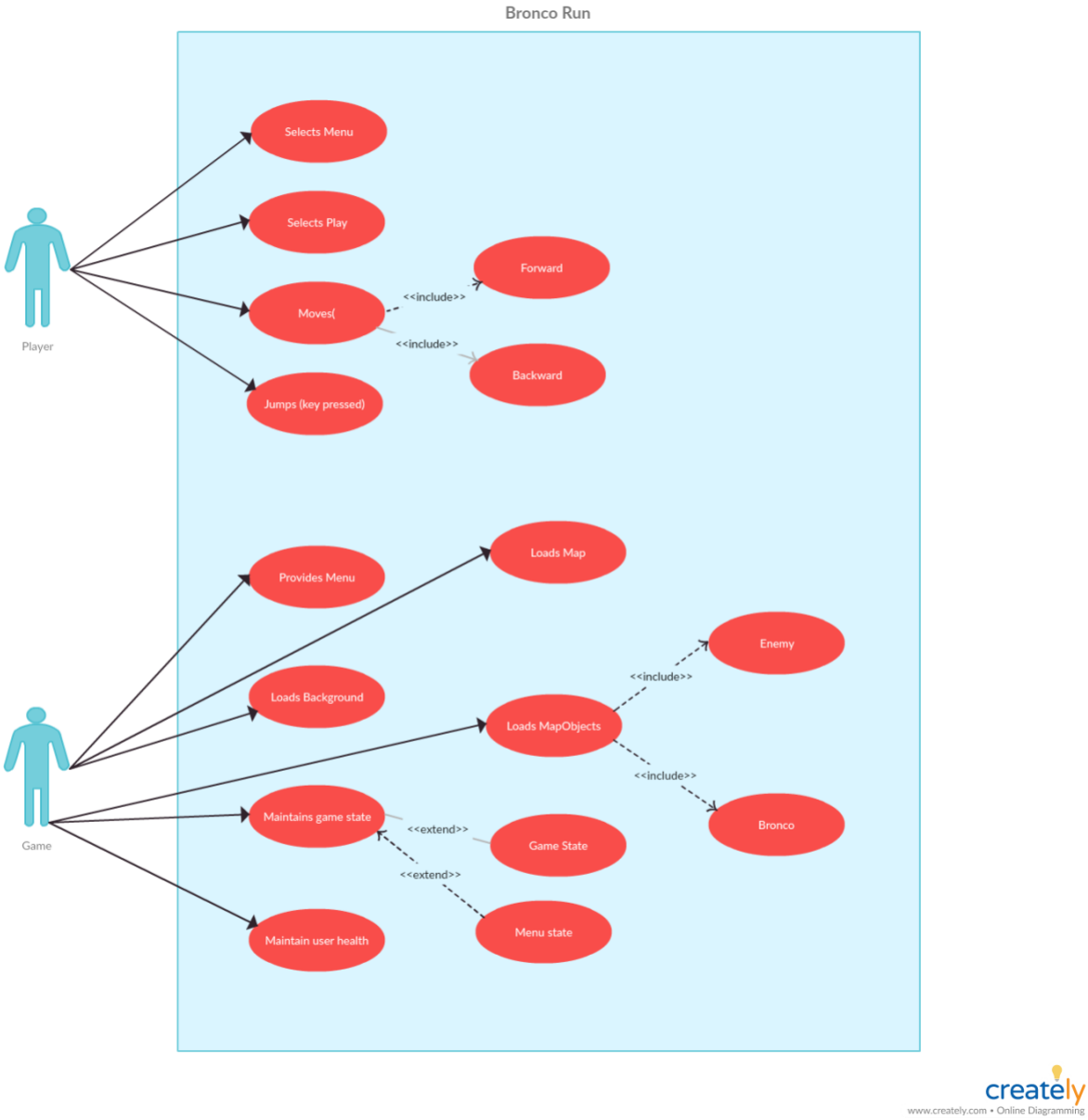
2.3.10 Player scratches the enemy

If the player comes in contact with the enemy, he should be able to scratch the enemy, doing so, the enemy should die.

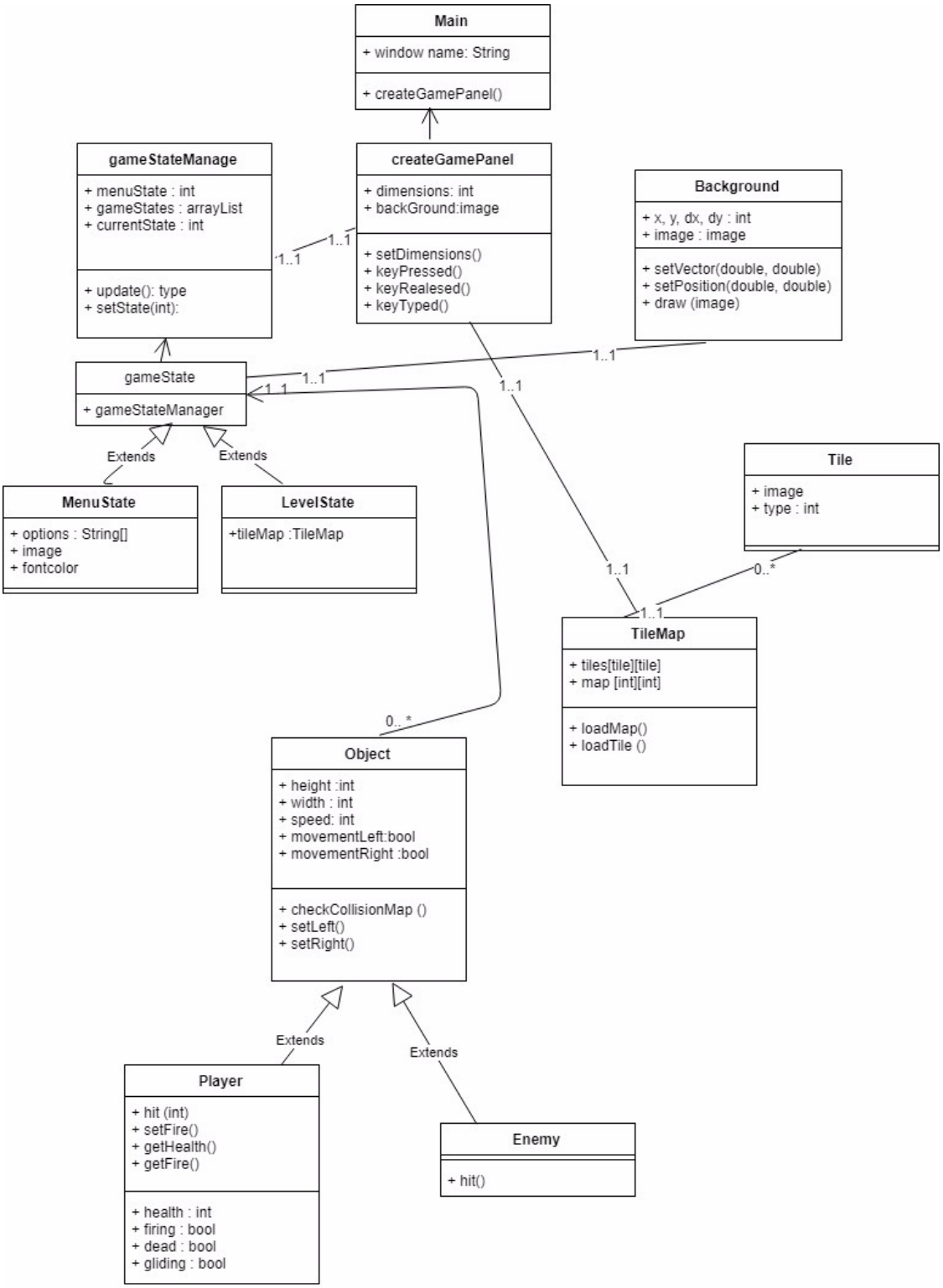
2.3.11 Player's health should reduce on colliding with the enemy

If the player collides with the enemy, his health should reduce and the players image should flicker for a few seconds allowing user to understand that the collision has occurred. Also the reduced health should be reflected on a health bar on the screen.

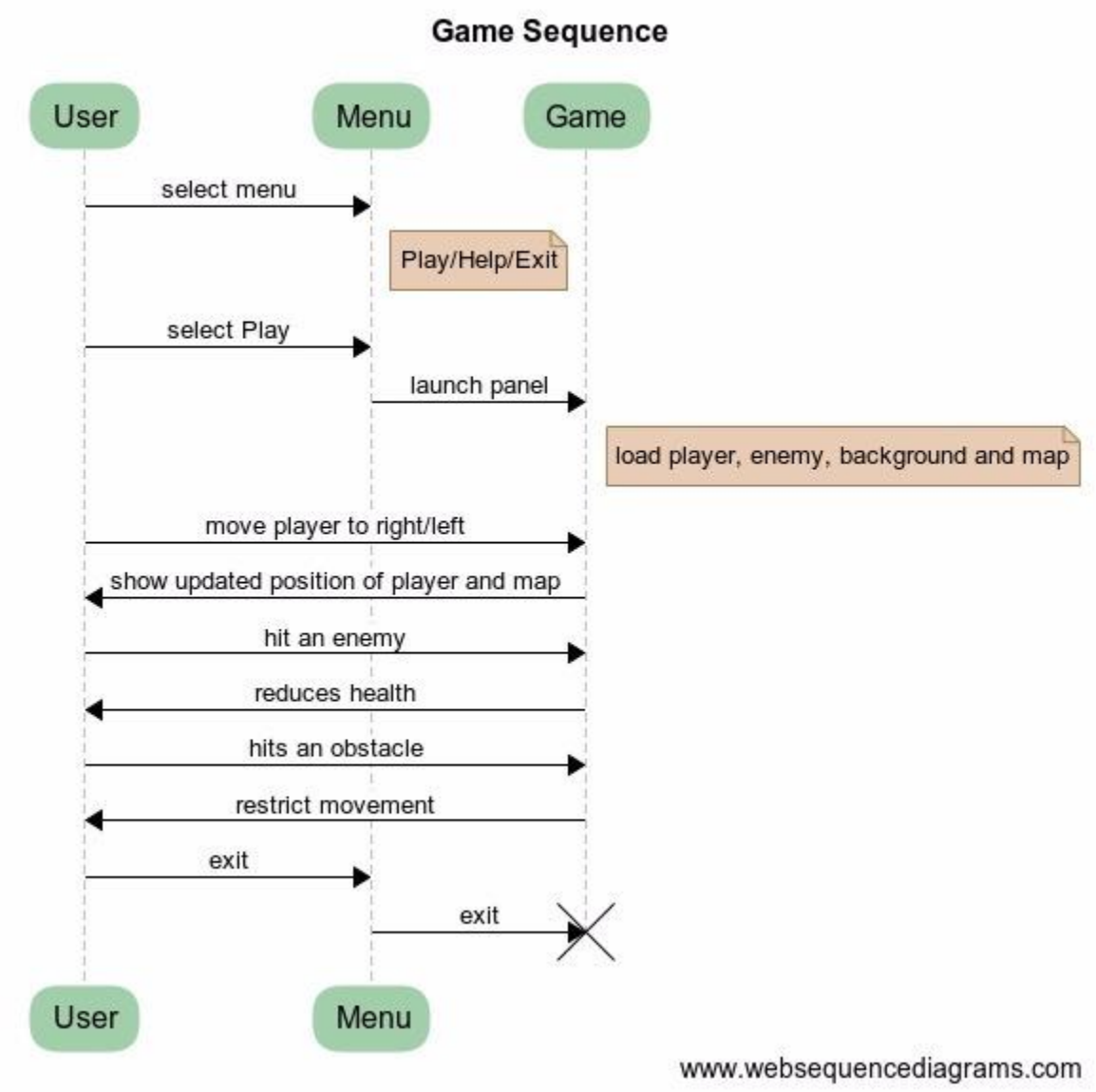
2.4 Use case diagram



2.5 Class Diagram



2.6 Sequence diagram



The sequence is as follows:

1. The game will display menu to user.
2. The menu consists of :
    a. Play
    b. Help
    c. Exit
3. If the user selects play, the game will launch a new window.
4. Game will load the background, map and all the map-objects including the main character and enemies
5. The game will also populate the player health.

6. The user can now move the player front/ back on key press.
7. The background and map should load asper users movement.
8. The map consists of opaque and transparent tiles
9. On colliding with opaque tiles, user should jump to cross them using up arrow key
10. In case of empty tile, user should jump to cross it, failure to do so will lead user to fall in the valley and the game will get over.
11. The enemies should be populated on game load.
12. If the player collides with the enemy, its health should reduce.
13. The reduction of health should be reflected by to events:
    a. Flickering of player image
    b. Reduction of health from health bar
14. If the user health reduces to zero, the game should get over.
15. The user should also be able to scratch the enemies
16. If the user scratches the enemies, the enemies should die and disappear from the screen
17. If the user selects help from menu, help window should open showing player instructions to play the game
18. If user selects exit from menu, the window should be closed.

3. Nonfunctional Requirements

As this is a game where in the user is moving forward and or backward at the speed of button press, it is important to load the game objects as equal or greater pace. We will be using images with lower resolution to make sure that the loading time is not more and the pace can be matched. Also, as developing this game involves a lot of implementation, we have made sure that the hierarchy(IS-A and HAS-A) relationship is maintained among the classes, because it gets easy to navigate through the classes as they are defined structurally. As there are a lot of classes involved, to make sure that the communication between different classes is ensured becomes crucial. In order to ensure this, we have defined classes in such a manner that their attributes and methods does not invoke any type of complexity. In other words, we made sure that the concept of 5Cs (Clarity, Cohesion, Completeness, Convenience, Consistency) is taken care of while designing classes.

This game needs Robust Software Testing because as there are so many modules involved in making the game, every time we change something in a module, we have to test that whether or not that change causes any type of discrepancy in other modules. If it doesn't then we are fine and good to go, but if does then we have to be extra careful while doing any other type of change. Also, while changing any module we have to make sure that we test that module as soon as the change is made, because if we test the module after making too many changes then we might lose the track of which change made caused which discrepancy. That's why Regression testing is very crucial when we have to implement plentiful classes in game development. However, as there was time constraint while implementing the project, we focused upon only Manual Testing, where we designed P1 and P2 test cases.

4. Other Requirements

From future enhancement perspective of product, we can implement further levels to give more difficulty levels to the user. In game development, there is a wide scope of future enhancement We can go all day long describing the future scope. Some of which can be:

1)We can add a soundtrack as a part of background music to improve the user interactive experience.

2)We can add more enemies having special powers to increase the difficulty of each level.

3)If we add more enemies, we can always see that the complexity within the same class doesn't increase.

4) Or we can add more superpowers to the player in addition to that of scratching and make the user experience subtler.

5)We can enable saving the game and loading the saved game in order to provide more flexibility to user in case user decides to discontinue playing the game after specific period of time.

6)We can also add multiple spawning of the enemies.

5. References :

[1] http://zetcode.com/tutorials/javagamestutorial/

[2] http://www.mohamedtalaat.net/portfolio/

[3] Davison, Andrew. Killer game programming in Java. " O'Reilly Media, Inc.", 2005.