Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

CS319 Project Group 1G: Super Mario Bros.

Analysis Report

Ahmet Çandıroğlu

Course Instructor: Uğur Doğrusöz
Course Assistant: Hasan Balcı

Progress/ Iteration 2
Oct 7 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Object Oriented Programming course CS319/1

Contents

1 Introduction

Super Mario, first released in 1985, is a game developed by Nintendo. This project will aim to create a game that is very similar to the original Super Mario game, but with some differences.

The game can be categorized as a level based adventure game, with Mario being the main character. The main objective of the game would be to overcome obstacles and reach the end point.

This game will be a desktop application and will be developed using the Java programming language.

2 Game Overview

The objective of the game is to clear as many levels as one can with a limited number of lives. Each level will consist of obstacles, monsters and traps. There will also be bonuses like coins and power-ups. The game will also have a net score of the player. The score will be dependent upon the number of coins collected, the number of monsters killed and the number of lives remaining.

The game will be controlled by just the keyboard. Additionally playing the game is very simple as the user only needs to use the arrow keys and the spacebar to send fireballs when Mario is in fire form.

2.1 List of Prizes

- **Coin:** Player collects coin to gain points. Coins can be located in *Surprise Brick*s and some of the *Ordinary Brick*s.

Figure 1: Coin

- **Super Mushroom:** This boost item turns Mario into Super form. Mario can break breakable bricks in this form. Super Mushrooms can be located in *Surprise Brick*s.

Figure 2: Super Mushroom

- **1-up Mushroom:** This boost item gives 1 extra life to Mario. It can be located in *Surprise Brick*s.

Figure 3: 1-up Mushroom

- **Fire Flower:** This item gives ability to throw fireballs which can kill enemies. It can be located in *Surprise Brick*s and *Invisible Brick*s which are basically invisible *Surprise Brick*s.

Figure 4: Fire Flower

- **Super Star:** This boost item makes Mario invincible for a short period of time. This item can be located in *Surprise Brick*s and some of the *Ordinary Brick*s.

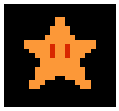


Figure 5: Super Star

2.2 List of Brick Types

- **Surprise Brick:** This bricks always contain a prize inside. Turns into empty unbreakable brick when hit.

Figure 6: Surprise Brick

- **Ordinary Brick:** This bricks may contain prize but it is uncommon. They are breakable according to Mario's state. It cannot be broken when Mario is in small state.

Figure 7: Ordinary Brick

- **Empty Unbreakable Brick:** This bricks do not contain price and cannot be broken.

Figure 8: Empty Unbreakable Brick

2.3 List of Mario's Forms

- **Small Mario:** Mario's initial state. Dies when hits enemy and cannot break bricks.

Figure 9: Small Mario

- **Super Mario:** Mario can break in this form. It reverts to *Small Mario* when he hits to enemy.

Figure 10: Super Mario

- **Fire Mario:** Mario can throw fireballs in this form. It reverts to *Small Mario* when he hits to enemy.

Figure 11: Fire Mario

- **Invincible Mario:** Mario is invincible in this form and he can kill enemies just by touching to them. Only dies when time is over or he falls into pit.

Figure 12: Invincible Mario

2.4 List of Enemies

- **Goompa:** They can move right and left. Dies when stomped on.

Figure 13: Goompa

- **Koopa Troopa:** They can move right and left. Dies when stomped on and leaves its shell which can be thrown to other enemies to kill by Mario.

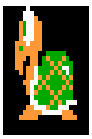


Figure 14: Koopa Troopa

3 Requirements

3.1 Functional Requirements

- User will be able to control Mario:
    - By speeding him up with a constant acceleration to a constant velocity.
    - By making him jump.
    - By making him duck.
- User will be able to break some bricks on the maps by making Mario jump to them.
- User will be able to proceed to right on the map by going right when Mario is at the one thirds of the game screen.
- User will be able to kill monsters.
- User will be able to collect coins:
    - By going over them.
    - By jumping under coin boxes.
- User will have three lives in the beginning.
- Game will be over when user spends all lives
- User will get one life when he/she collects 100 coins.
- User will start a new level when he/she reaches to end of a level.
- User will be able to pause/resume the game.

3.2 Additional Functional Requirements

- User will be able to choose a map to play. There will be an achievement system for this, user must pass and unlock previous maps in order to play next.
- Game will support sound for various actions.

3.3 Non-Functional Requirements

- Game will be implemented in Java.
- Game will support over 60 FPS.
- Game will support only key inputs due to nostalgia of the game.

3.4 Pseudo Functional Requirements

- Games code will be written in java.

4 System Models

4.1 Use Case Model

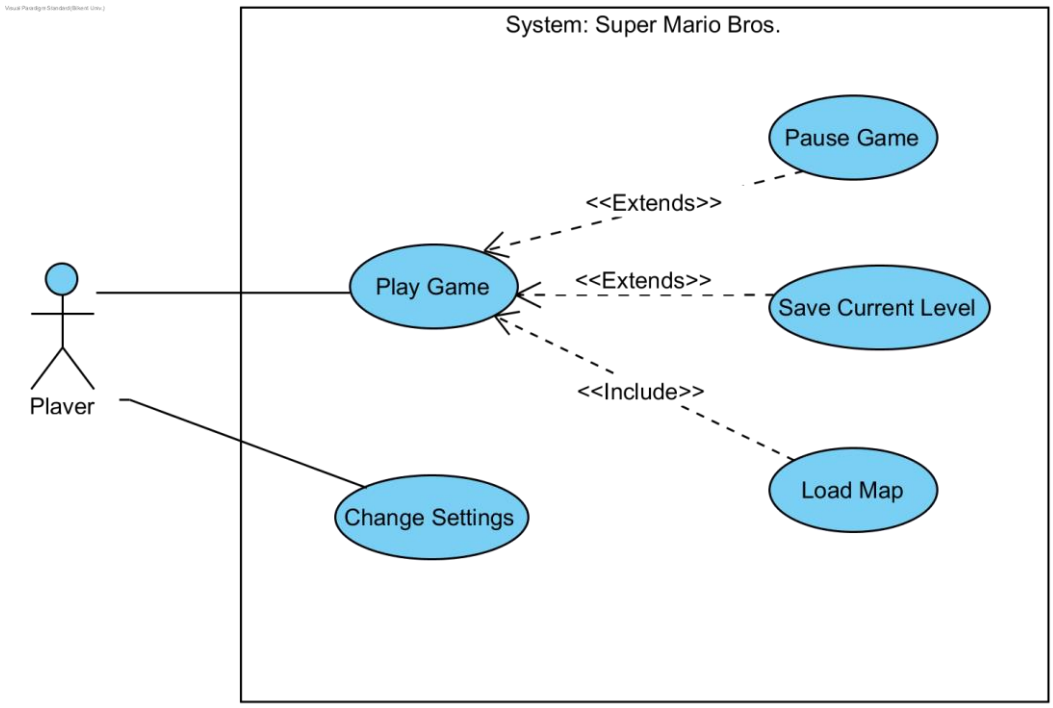


Figure 15: Use case diagram of the game

4.1.1 Use Case Descriptions

Use Case #1

Use Case Name: Play Game

Participating Actors: Player

Entry Condition: Player has opened the game and clicked “Play Game”

Exit Condition:

- Player closed the game
- Player lost all his/her lives
- Player completed all the levels of game

Main Flow of Events:

1. Start game
2. Engine constructs level
3. Player beats the level
4. If all levels are not beaten go to (1.) else display “game over”
5. Display score of player
6. Go to Main Menu

Alternative Flows of Events:

- If player loses all his lives go to (4.)

4.2 Dynamic Models

Scenario Name: Start Game and Load Map

Scenario: Ahmet clicks on game icon and see main menu in the game. Then he chooses *Start Game* from options by pressing enter key (default selection is start game). After then he chooses a map from possible options (he cannot choose a map without passing every map before it). Finally map is loaded and Ahmet plays the game.

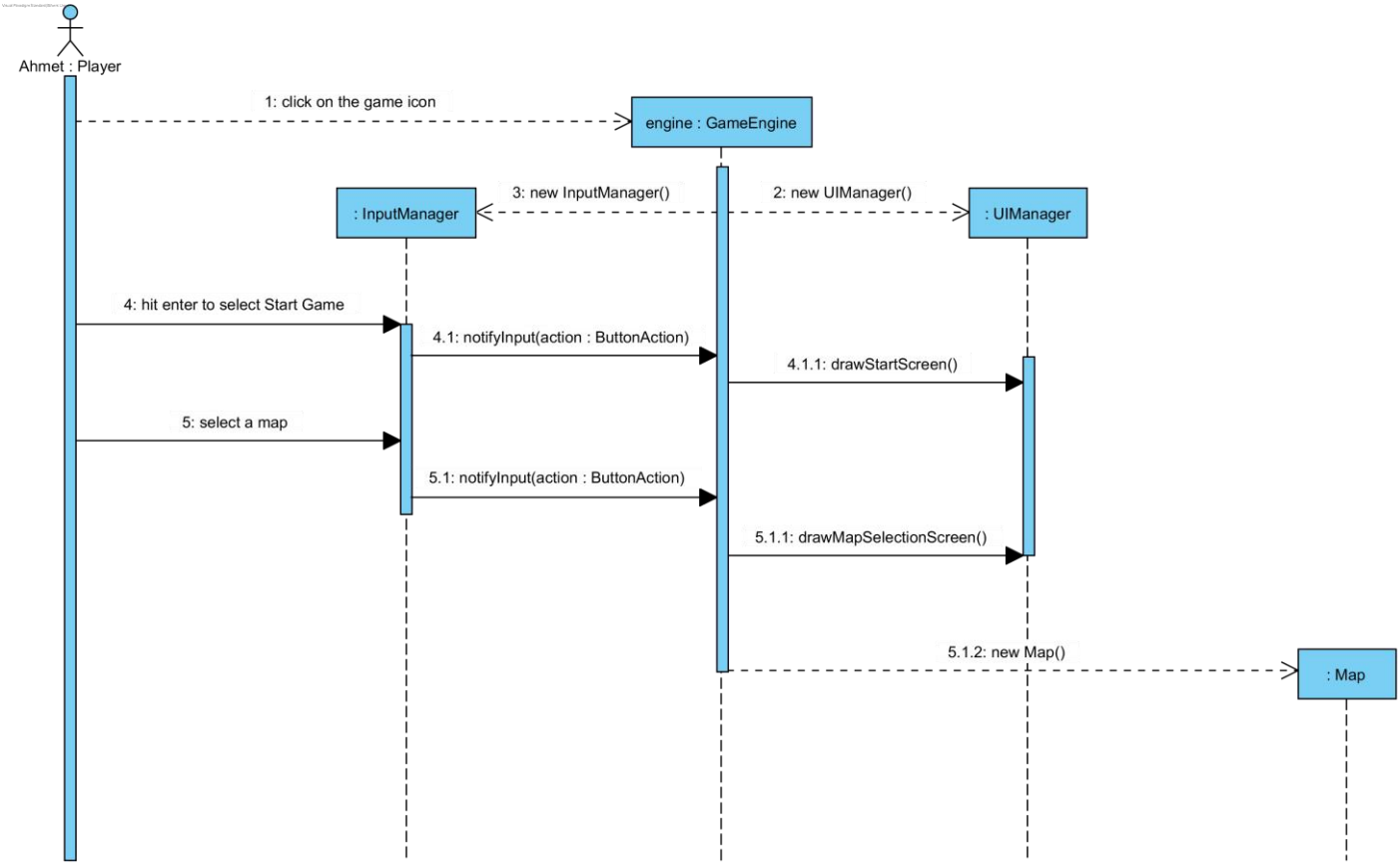


Figure 16: Sequence diagram for initial start of game

Description: Ahmet wants to play the game so he starts the game. *GameEngine* class is the main class of the game so it is created. *GameEngine* creates *UIManager* and *InputManager* instances and calls *drawStartScreen*() method of the *UIManager* which shows an option panel to player. Player chooses *Start Game* option so *GameEngine* calls *drawMapSelectionScreen*() method of *UIManager*. Ahmet selects a possible map and *GameEngine* creates a map accordingly.

Scenario Name: Revealing Super Mushroom

Scenario: Ahmet has been playing Super Mario Bros. with great joy for a while. He is trying to get coins and mushrooms by hitting to *SurpriseBrick*s from bottom. Turns out that the brick contains a SuperMushroom which is revealed. That SuperMushroom instance is added to Map.

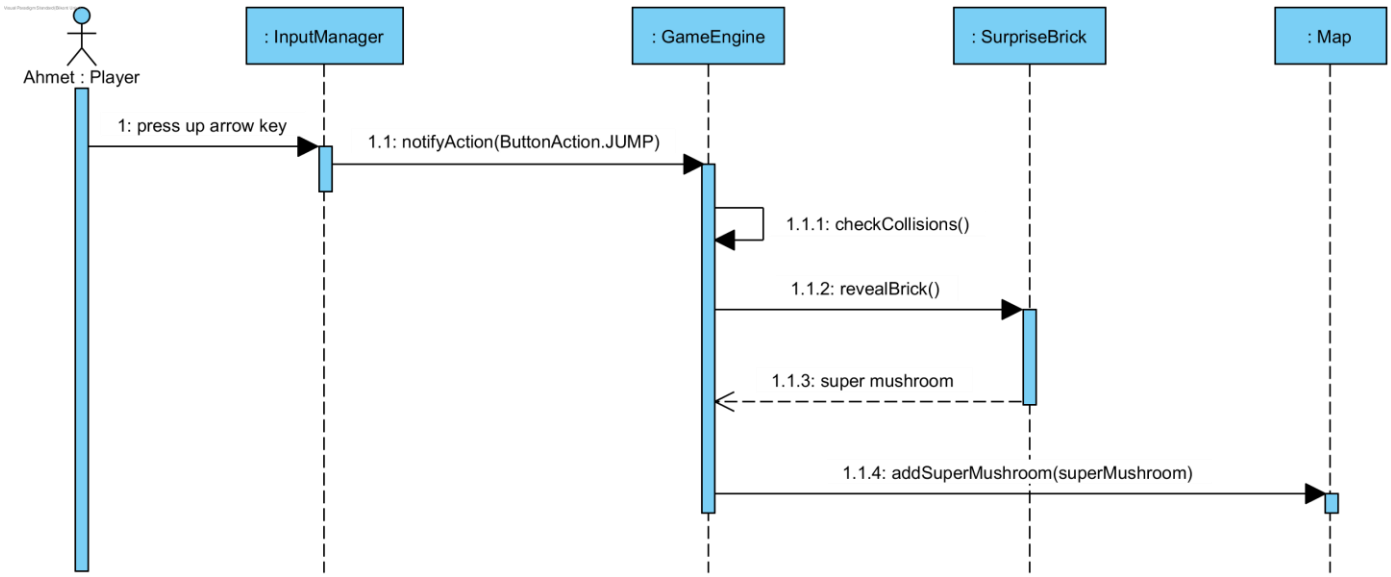


Figure 17: Sequence diagram of revealing a super mushroom

Description: Ahmet jumps under a *SurpriseBrick* which contains a *SuperMushroom*. First *GameEngine* is notified of the jump action. Engine updates locations of all of the game objects and checks for collisions. At this moment *Mario* and the *SurpriseBrick* intersects so the *Brick* will be revealed and the *Prize* it contains will be returned. This *SuperMushroom* instance is put into *Map* instance in *GameEngine* and so, it will move and can be eaten.

Scenario Name: Eating Super Mushroom

Scenario: After great success of revealing mistery of *SuperMushroom*, Ahmet seeks more and wants to eat it. So, he moves to the right where the

SuperMushroom lies and touches it. Then, *Mario* gains great power via this magical mushroom and turns into super form.

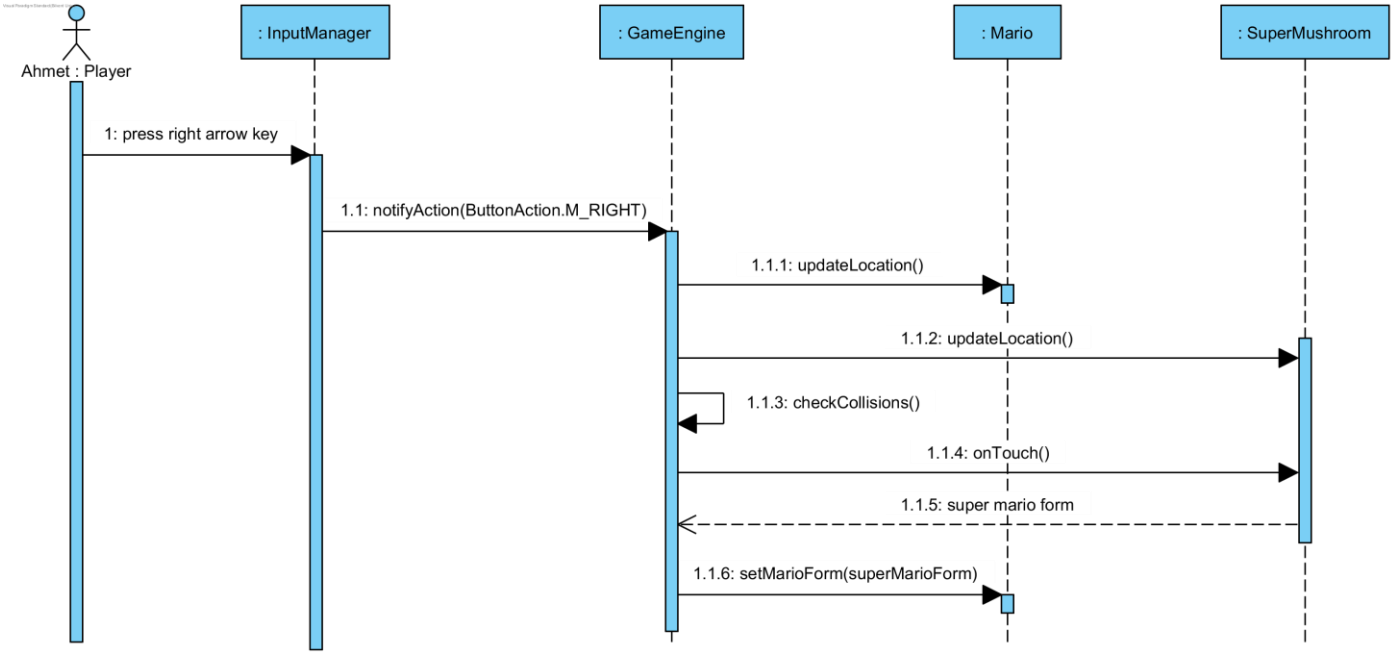


Figure 18: Sequence diagram of eating a super mushroom

Description: Ahmet pressed right arrow button to move *Mario* to the right. So, *InputManager* calls *notifyInput*() method of *GameEngine*. Then, *GameEngine* updates locations in *Map* class in this case, *Mario* and *SuperMushroom*. Engine calculates collisions and finds that *Mario* touches to *SuperMushroom* so invokes *onTouch*() method of the *SuperMushroom* which returns super mario form, an instance of *MarioForm* class. Finally, *GameEngine* calls *setMarioForm*() method of *Mario* with super mario form as an argument.

4.3 Object and Class Model

GameEngine is the main class which controls the game according to state of the actors in map and inputs received from player. *Map* class contains everything related to map and draws to the screen. It is only a visual container, it does not control anything. Hero class represents the possible characters player can control during the game. *BoostItem* changes state of the Hero, it makes Hero stronger. *Prize*, *Brick*, *Hero* classes are model classes where as *GameEngine* is controller and *Map* is view class.

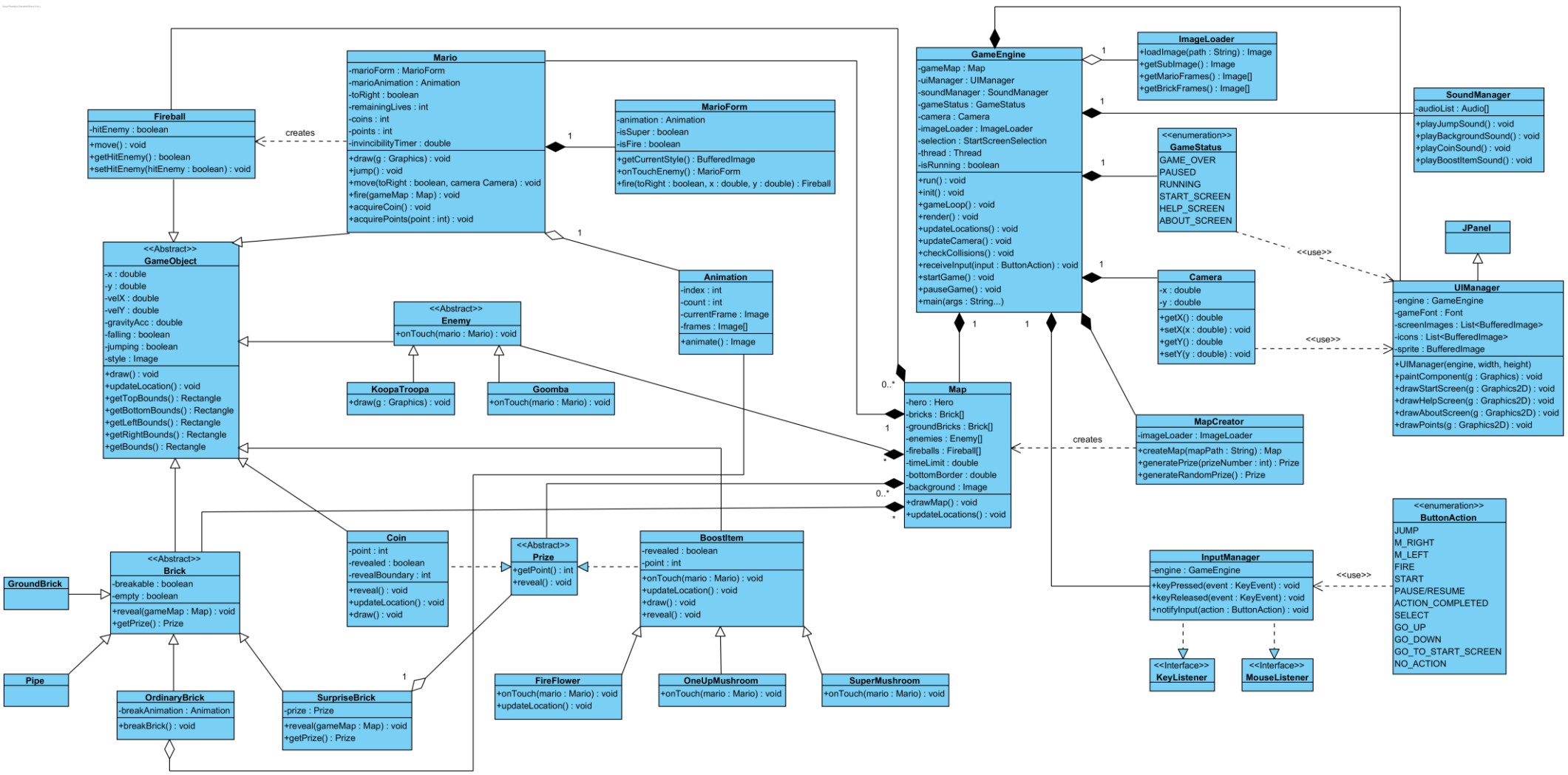


Figure 19: UML class diagram of the game

4.4 User Interface and Screen Mock-ups

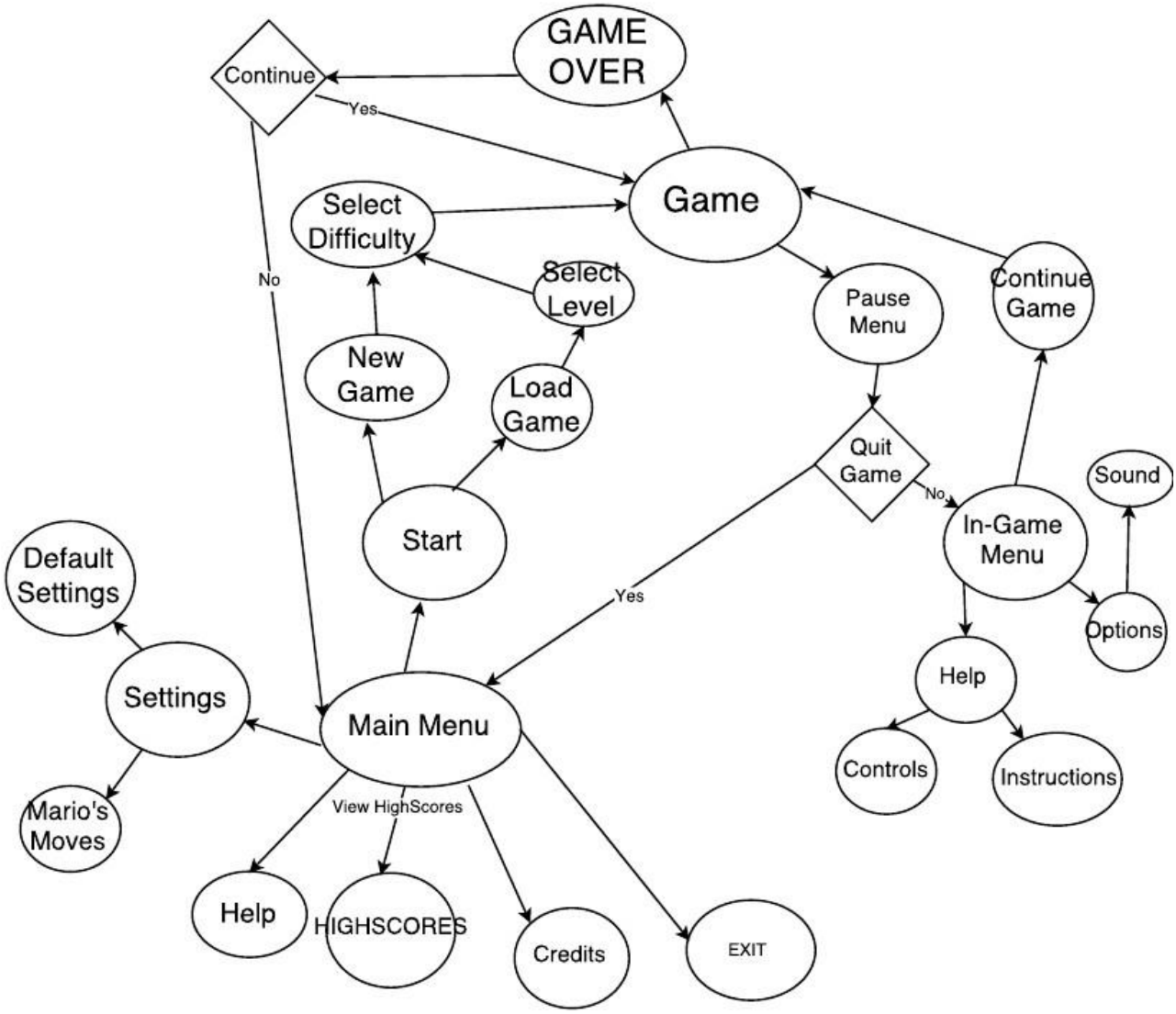
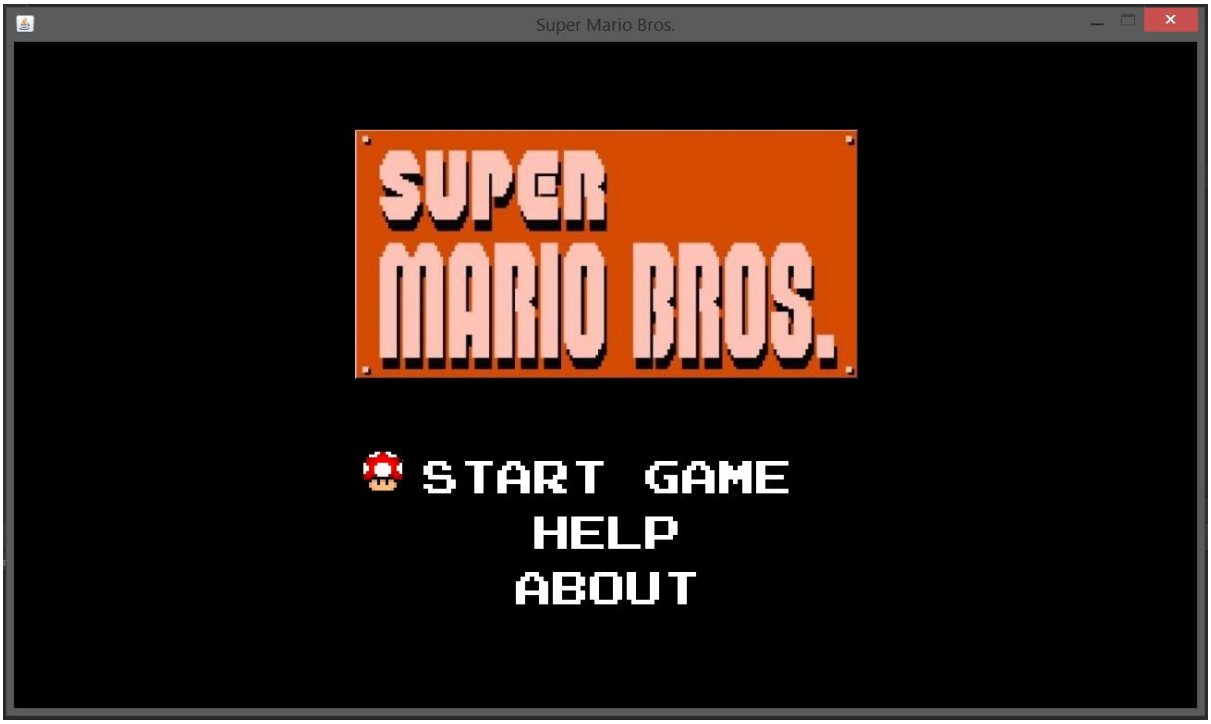
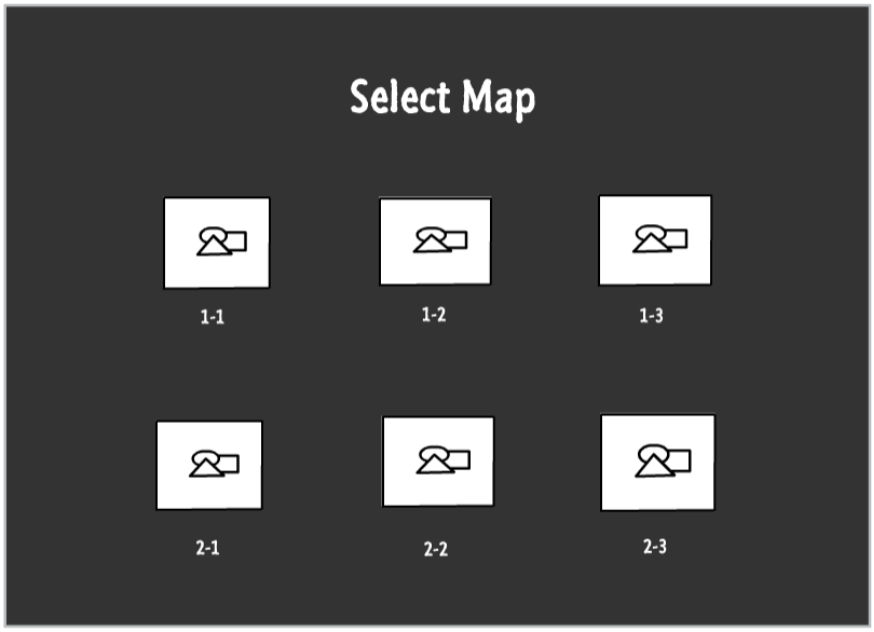


Figure 20: Navigational path diagram

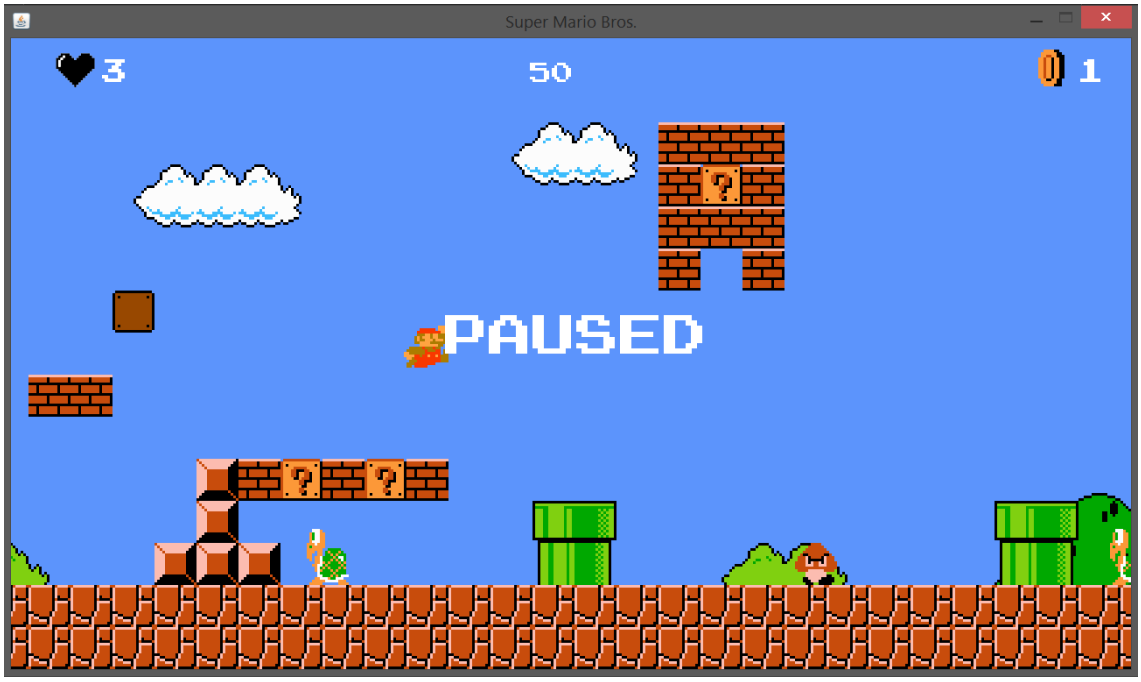
Start Screen



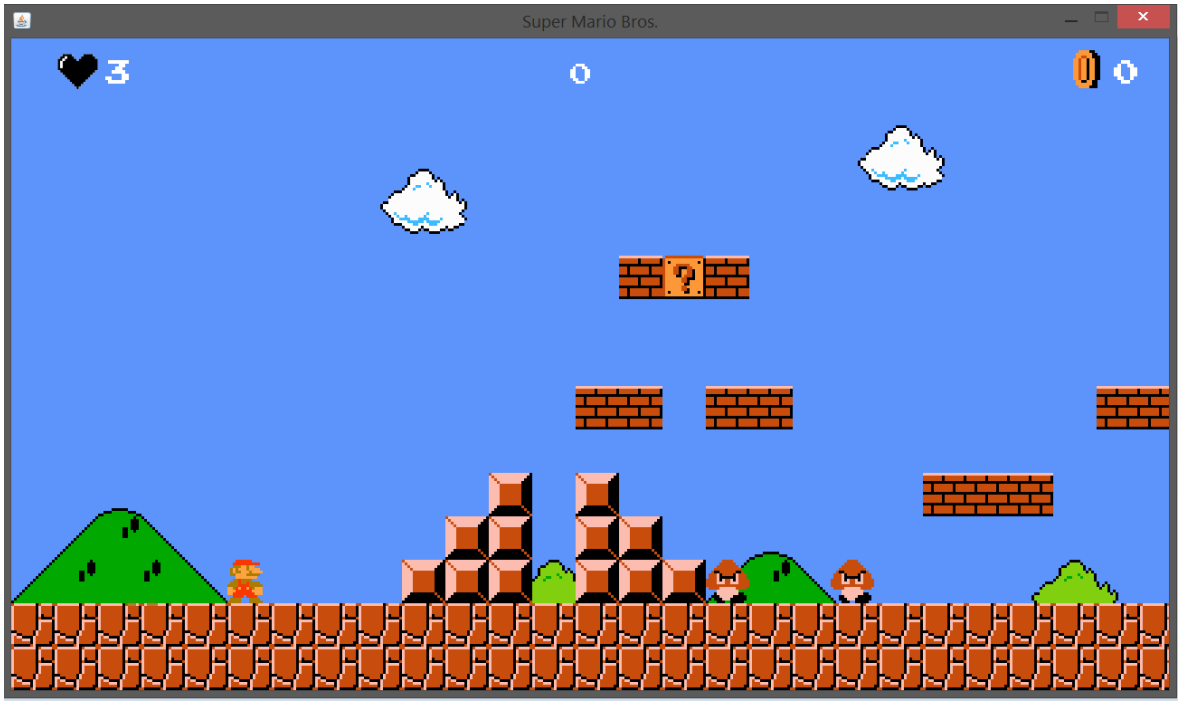
Load Game: After Player selected Start game option, he/she needs to choose a map. Maps are not unlocked one of the previous maps is unlocked and first one is always available.



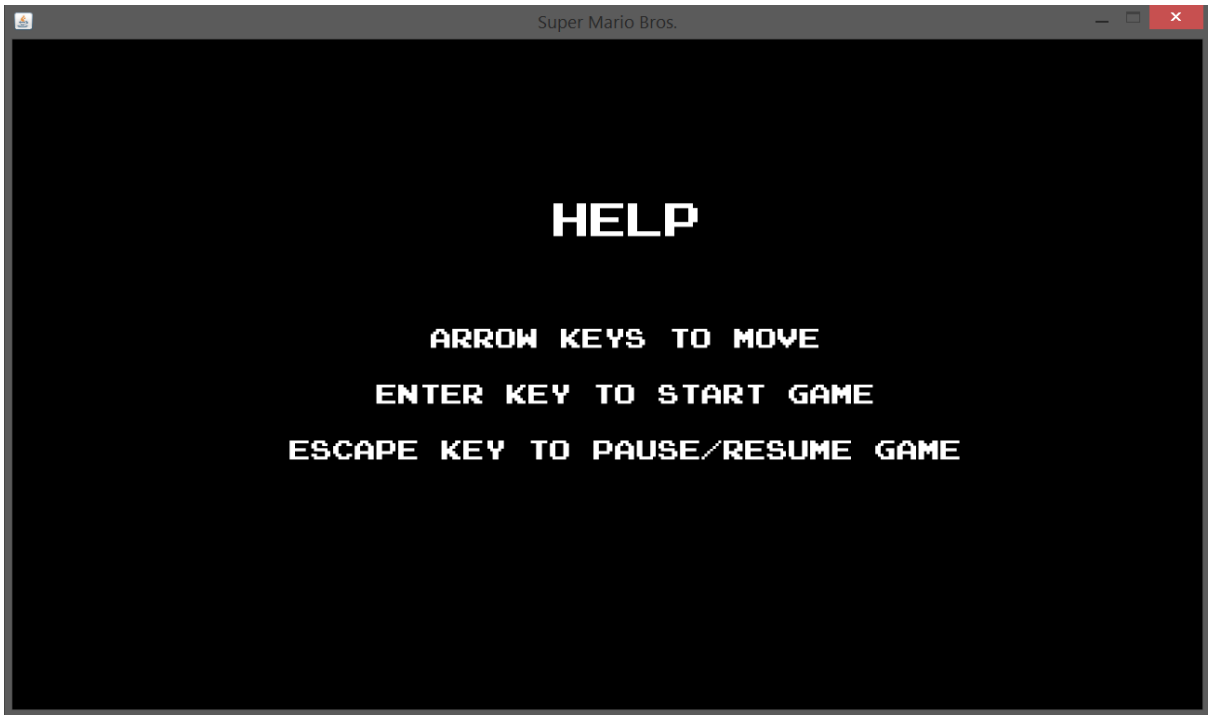
Pause Menu: If the player press the pause button; which is in the top right corner of the gameplay screen, this player encounters with the pause menu; which has 2 options.



In-Game Screen: Player can see remaining time, remaining lives, number of acquired coins and points.



Help Menu



Note: Start menu, Pause menu, In-game screen and Help menu implemented already and these images are screenshots of the game.

5 Improvement Summary

- Added – Game will support sounds now. Super Mario game cannot be played without its iconic soundtracks so *SoundManager* will be implemented which will control played audio.
- Added – New maps will be implemented which can be selected by player to play. These maps will be original as it was in the first iteration. User must unlock maps by passing each previous map successfully.
- Added – New game screens are implemented and added to User interface section of the report. These screens will summarize all the information that needs to be known without any complexity for user.
- Changed – Structure of the class design is improved.
    1. Mario subclasses (SmallMario, SuperMario and FireMario) are removed and MarioForm class is added. MarioForm class holds all information necessary to represent a Mario form. The reason behind this to implement Player-role design pattern and avoid class changes during run-time.
    2. Animation class added to show animation of a character easily.
    3. GameObject class is improved and has most of the functionality now.
    4. GroundBrick class is implemented and added to Map class to represent all the bricks on the ground which has no use other than being an obstacle.
    5. MapCreator class is implemented. This class takes a path of an image (map image) and creates a map for *GameEngine*.
- Changed – Overall design is now better and more close to MVC design pattern.

6 References

https://www.mariowiki.com/Super_Mario_Bros.

https://www.mariowiki.com/Coin

https://www.mariowiki.com/Super_Mushroom

https://www.mariowiki.com/Fire_Flower

https://www.mariowiki.com/Super_Star

https://www.mariowiki.com/1-Up_Mushroom

https://www.mariowiki.com/Small_Mario

https://www.mariowiki.com/Super_Mario

https://www.mariowiki.com/Fire_Mario

https://www.mariowiki.com/Invincible_Mario

http://www.mariouniverse.com/maps/nes/smb