PROJECT GROUP 1G

Super Mario Bros.

ANALYSIS REPORT

Table of Contents

2. Game Overview

The player takes on the role of the main protagonist of the series, Mario.

The player moves from the left side of the screen to the right side in order to reach the flag pole at the end of each level.

The objective is to race through the Mushroom Kingdom, survive the main antagonist Bowser's forces, and save Princess Toadstool.

The game world has coins scattered around it for Mario to collect, and special bricks marked with a question mark (*?*) and "secret", often invisible, bricks

Mario's primary attack is jumping on top of enemies.
Another attack, for enemies standing overhead, is to jump up and hit beneath the brick that the enemy is standing on.
Another is the Fire Flower; when picked up, this item changes the color of Super Mario's outfit and allows him to throw fireballs

The game consists of eight worlds with four sub-levels called "stages" in each world.

In addition, there are bonuses and secret areas in the game. Most secret areas contain more coins for Mario to collect, but some contain "warp pipes" that allow Mario to advance to later worlds in the game, skipping over earlier ones.

2.1. List of Bonuses

Clock
Duration of the gameplay. Determines the score

2.1.1. Collectibles

2.1.1.A Items

Coin
when picked up, game score increases

2.1.1.B Special Items

Starman

often appears when Mario hits certain concealed or otherwise invisible blocks. This item makes Mario temporarily invincible to most hazards and capable of defeating enemies on contact.

Fire Flower
when picked up, this item changes the color of Super Mario's outfit and allows him to throw fireballs or only upgrades Mario to Super Mario if he has not already.

Magic Mushroom
when picked up, Mario grows to double his size and can take one extra hit from most enemies and obstacles, in addition, it causes Mario to boost jump level being able to break bricks above him. If he is already big, collects 100 coins.

1-Up Mushroom
when picked up, Mario changes form small to big. If he is already big, collects 100 coins.

2.1.2 Secrets

Treasure Room
Most secret areas contain more coins for Mario to collect

Warp Pipes
allows Mario to advance to later worlds in the game, skipping over earlier ones.

2.1.3 Bonus Actions

Bouncing on Koopa shell
Mario may use it as a projectile, collects 100 coins, defeating several enemies in a row with a

Bouncing on Goomba
Mario collects collects 100 coins, if he bounces successively without touching the ground.

2.2 Player

Mario

move along X & Y axis, may *throw* fireballs when Fire Flower collected. Has-a *health bar* an form

move:

W: jump, along positive Y-dir
D: run, along positive X-dir
A: run, along negative X-dir
S: may enter the pipe, if Mario is on top of the pipe

throw:

P: throws fireball, along X and Y

health bar: number of lives, fixed.

form: small, big, invincible

2.3 List of Collidables

2.3.1 Enemies

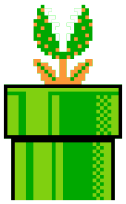
Goomba

initial: moving back and forth along the X direction
collision: Mario may loose a life or gets smaller.
defeated: will flatten in response to Mario's attack.
noticing: AI will chase after Mario.

Koopa Troopa

initial: moving back and forth along the X direction
collision: Mario may loose a life or gets smaller.
defeated:
noticing: AI will chase after Mario.

Piranha Plant



initial: rest in pipes
collision: Mario may loose a life or gets smaller.
defeated: will go back in to the pipe,
noticing: AI will bite anything near along the Y direction

2.3.2 Immovables

? Block

initial: rest in the tileset
collision: a hit from below by Mario reveals more coins or a special item.
noticing: non-exits
defeated : non-exists

Hidden Block

initial: rendered into the Scene but not visible
collision: Mario may reveal hidden block by a hit from below
noticing: non-exits
defeated : non-exists

Bottomless pit

initial: rendered into to the Scene

collision: Mario loses a life if he 'falls' in the pit

noticing: non-exists

defeated: non-exists

2.3.1.3 Bosses

Bowser

initial: rests on the the bridge

collision: Mario loses a life if collides with Bowser or his projectiles.

noticing: AI spits fireballs and tossing hammers, move on X and leaps on Y direction

defeated: grabbing the axe on the other side of the bridge causes him to fall into the lava

Fake Bowsers

initial: same with Bowser

collision: same with Bowser

noticing: same with Bowser

defeated: same with Bowser or five fireball hits

3. Requirements

3.1. Functional Requirements

- User can control Mario through action keys Such as:
  - W: jumps
  - A: run left
  - S: enters pipe
  - D: run right
- User can change the settings of the game. Such as:
  - key-bindings.
  - music on/off
- User can access the help menu. Contains:
  - information about controls & key-bindings.
- User can check the scoreboard.
- User can save. Saving stores the current level and the progress in the current level
- User can throw fireballs, If Mario collects Fire Flower
- User will be able to collect coins

3.2. Non-Functional Requirements

- Game graphics is represented as 8-bit sprite-sheets, to solve the significant efficiency problem created by rendering individual image files on CPU
- Some game objects which has AI is handled by AIEngine, such as
  - Enemies
  - Bosses
- User inputs are handled by InputEngine
- Game physics is handled by PhysicsEngine
- World data such as Collidables, Collectibles and, Map is generated by WorldEngine
- Game sounds is handled by SoundEngine
- World data is handled by GraphicsEngine/SceneManager/MapHandler
- User Interface is handled by GraphicsEngine/UIManager
- Camera, which follows Mario, is handled by GraphicsEngine/SceneManager/CameraHandler

3.3. Pseudo Functional Requirements

- Game implemented with Java
- Graphic objects will be designed using Adobe® Photoshop CS6.

4. System Models

4.1. Use Case Model

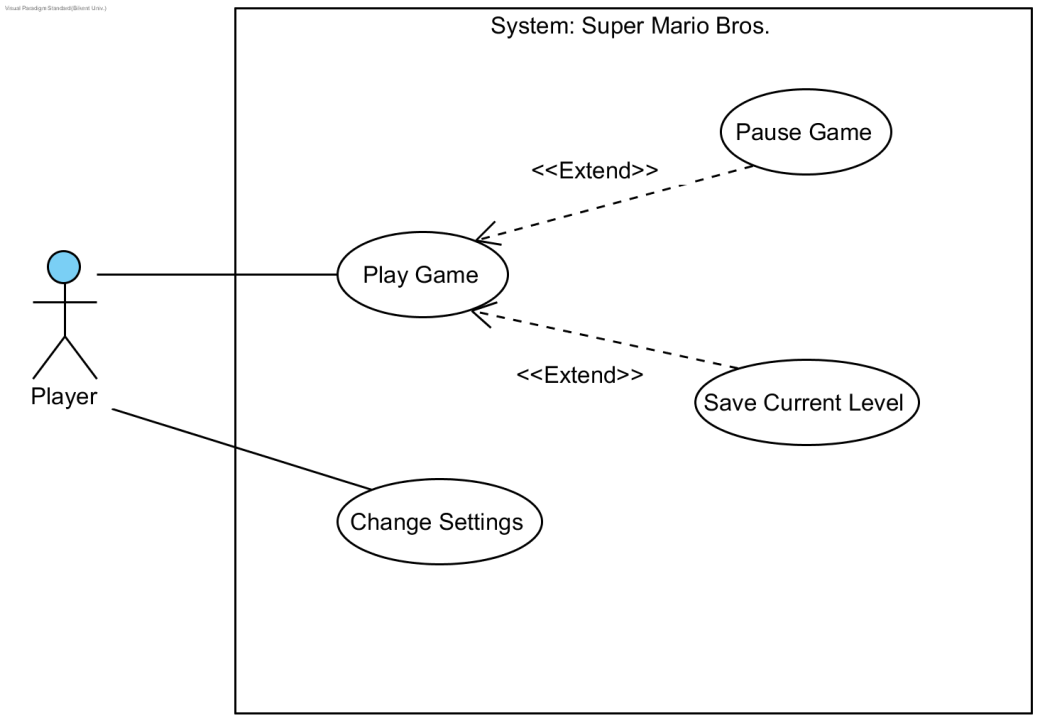


Figure 15: Use case diagram of the game

4.2 Dynamic Model

Scenario Name: Start Game

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.)

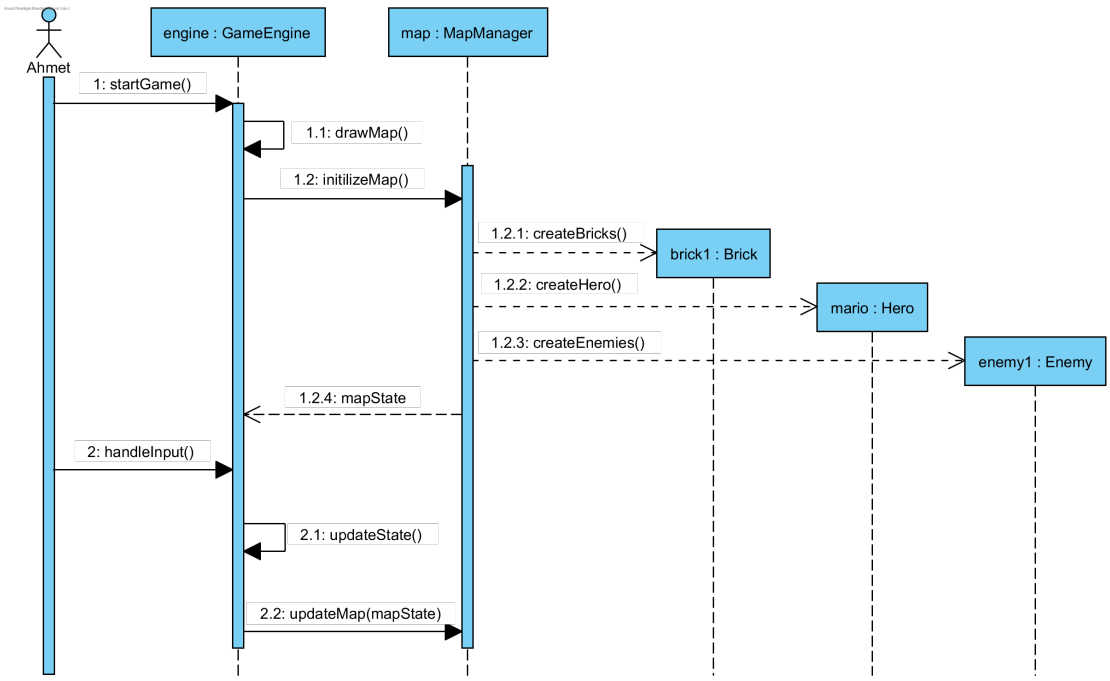


Figure 16: Sequence diagram for initial start of game

Scenario: Ahmet clicks on game icon and see main menu in the game. Then he decides to play without any configuration. He clicks on play game and starts to play

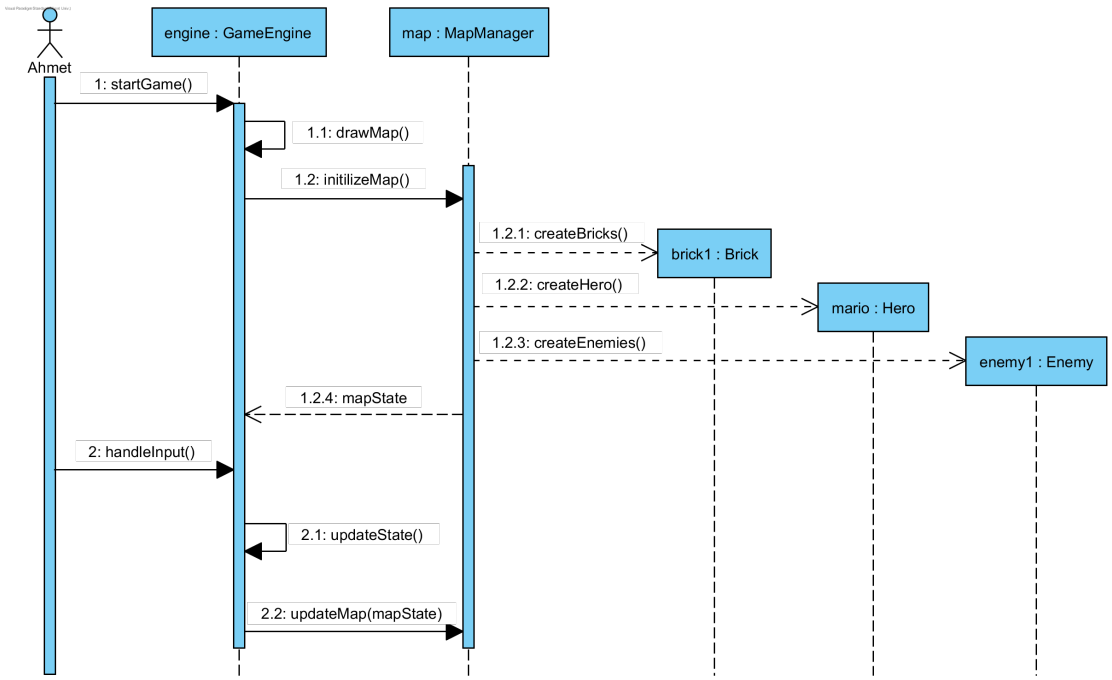


Figure 16: Sequence diagram for initial start of game

Description: Ahmet wants to play the game so he starts the game. This process initialized by *startGame()* method of *GameEngine*. Then, *GameEngine*, draws the map, not visually, and calls *initilizeMap(mapState)* method of MapManager to create required classes. *mapState* is the information of bricks, hero, enemies etc. which is gathered from *drawMap()* method. After that, MapManager creates asked components then return *mapState* again to *GameEngine.* From now on Ahmet gives input to *GameEngine*, *GameEngine* updates the state with *updateState()* method and then calls *updateMap(mapState)* method which changes view.

Scenario Name: Eating Super Mushroom

Scenario: Ahmet plays Super Mario Bros. with great joy. He is trying to get coins and mushrooms by hitting to *SurpriseBrick*s. He opens a brick which contains *SuperMushroom*. He eats it and his Mario becomes super.

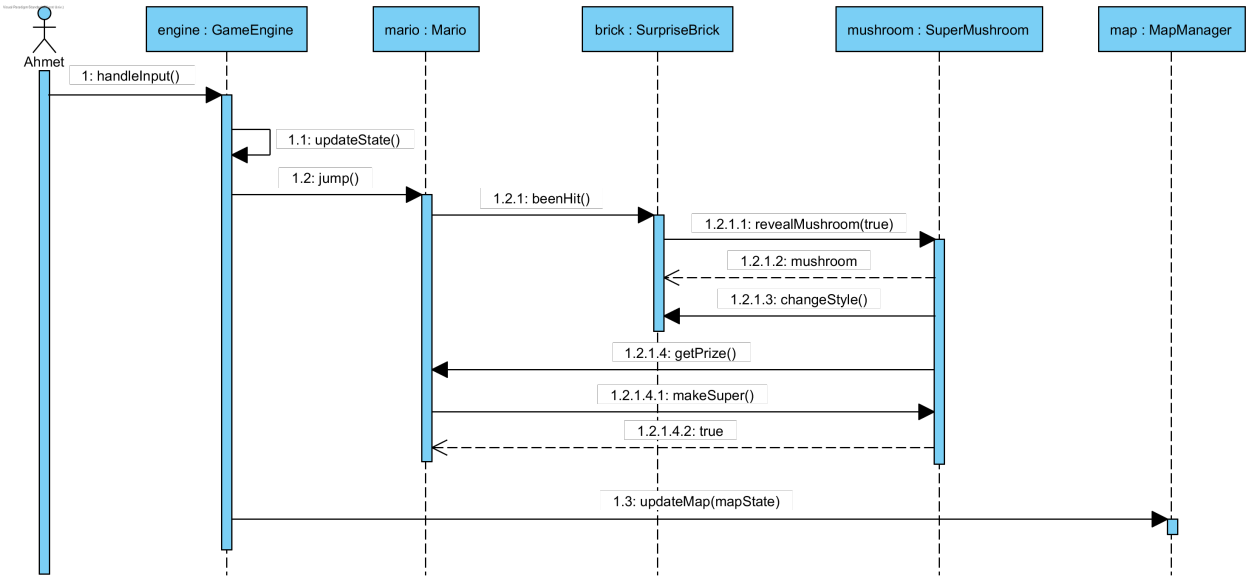


Figure 17: Sequence diagram of eating super mushroom

Description: Ahmet jumps under a *SurpriseBrick* which contains a *SuperMushroom*. *GameEngine* updates the state after getting input and then calls *jump()* method of Mario. Mario calls *beenHit()* function of *SurpriseBrick* which leads to mushroom to be revealed. Then, Mario calls *makeSuper()* function of mushroom by eating it. After all happened finally MapManager is updated and so view.

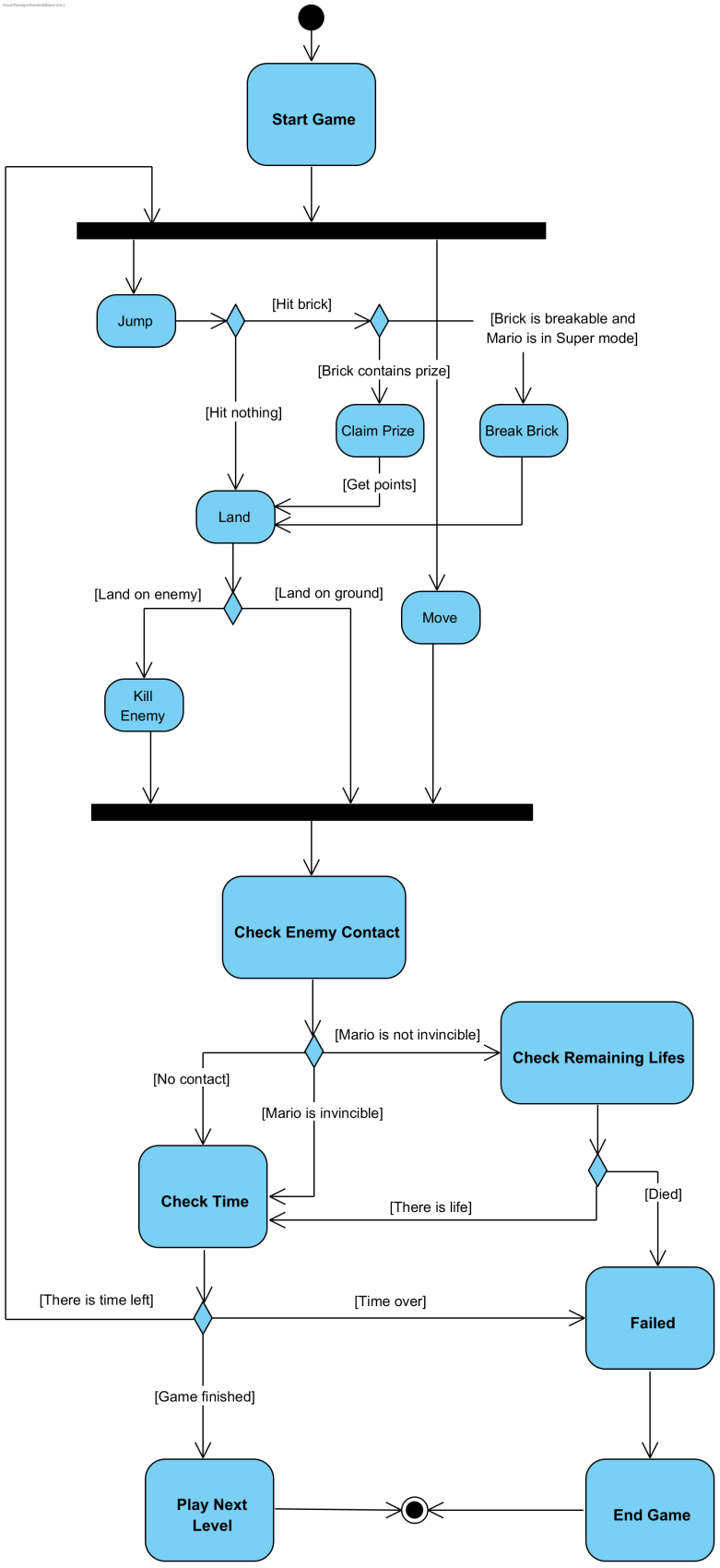
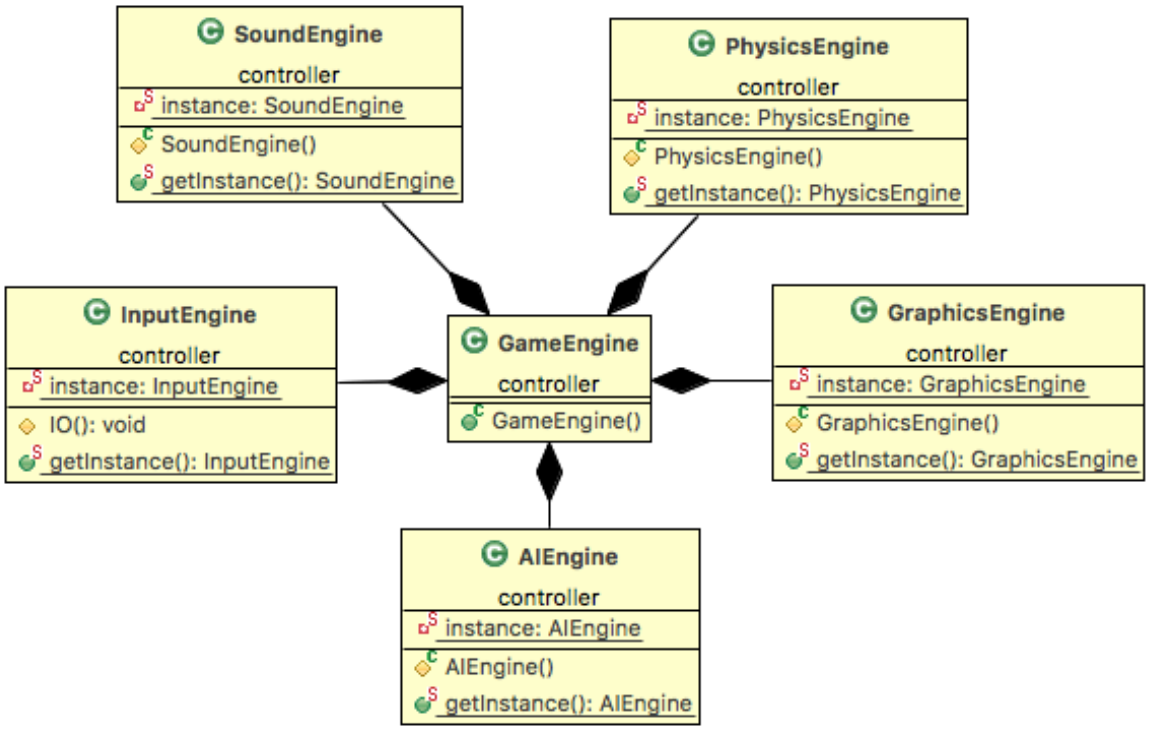


Figure 18: Activity diagram of the game

As it is shown in activity diagram there are lots of checking and decision making during gameplay. After each move or jump of the player, *GameEngine* needs to check several conditions. Player needs to reach end of the map within given time so after each move *GameEngine* checks time. Also, there are enemies on the ground so *GameEngine* also check for possible enemy contact and changes state accordingly. If player reaches final checkpoint without wasting every life within given time, then he/she can go to next level which is a different map or exit the game. Points will be gained by killing enemies and obtaining prizes.

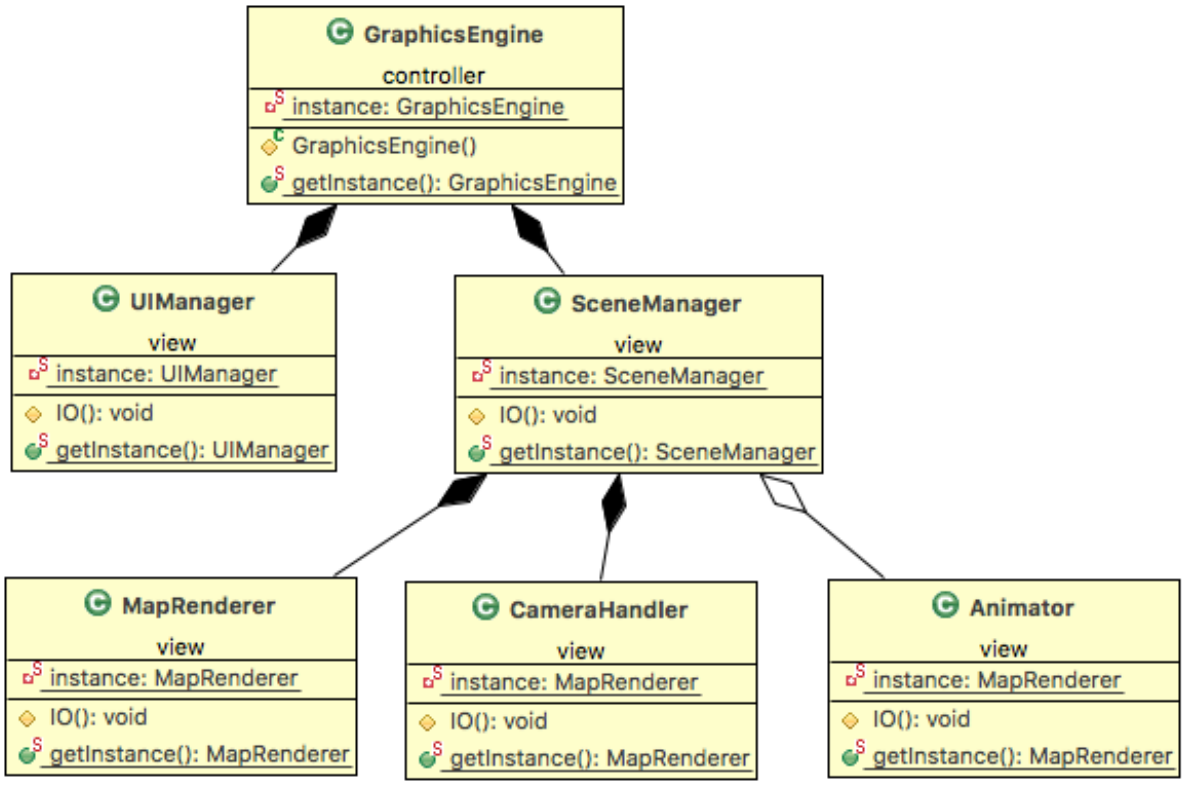
4.3 Object and Class Model

4.3.1 Controller Package



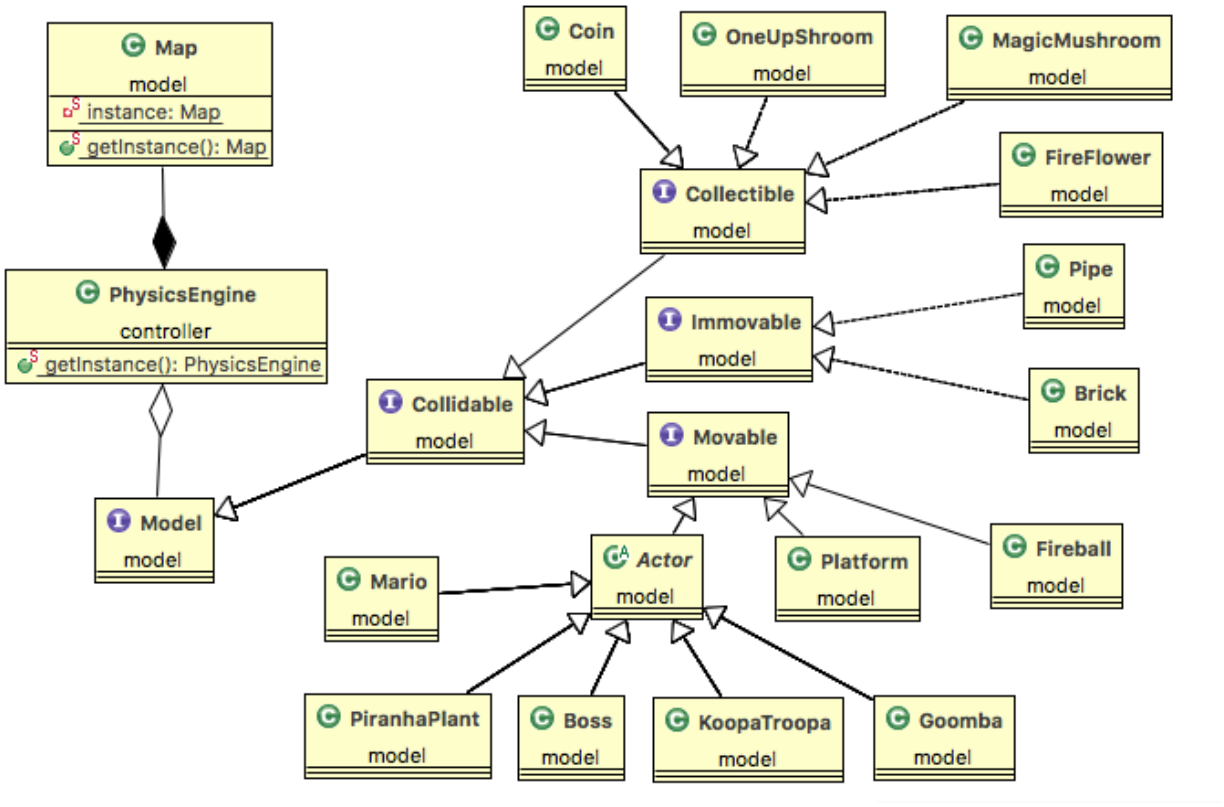
- *GameEngine* is at the center of all Engine interactions.
- *GraphicsEngine* renders the animation of game objects, UI and Camera.
- *AIEngine* handles the Enemy & Boss objects' behavior upon noticing
- *InputEngine* handles the user input
- *SoundEngine* handles game sounds
- *PhysicsEngine* handles position of objects and their interaction. (collision)

4.3.2 View Package



UIManager renders user interface. Such as: Buttons, Fonts
MapRenderer renders the Map from tileset
CameraHandler renders the Camera, to follow Mario during gameplay
Animator renders the animation of objects from sprite-sheet

4.3.3 Model Package



PhysicsEngine is responsible for creating objects such as *Model* and *Map*.
Map is in Singleton Pattern to prevent multiple copies.
Other game objects classified according to their physicality. Such as *Moveable*, *Collectible* etc.

4.4 In-Game & End-Game Screens

4.5 User Interface and Screen Mock-ups

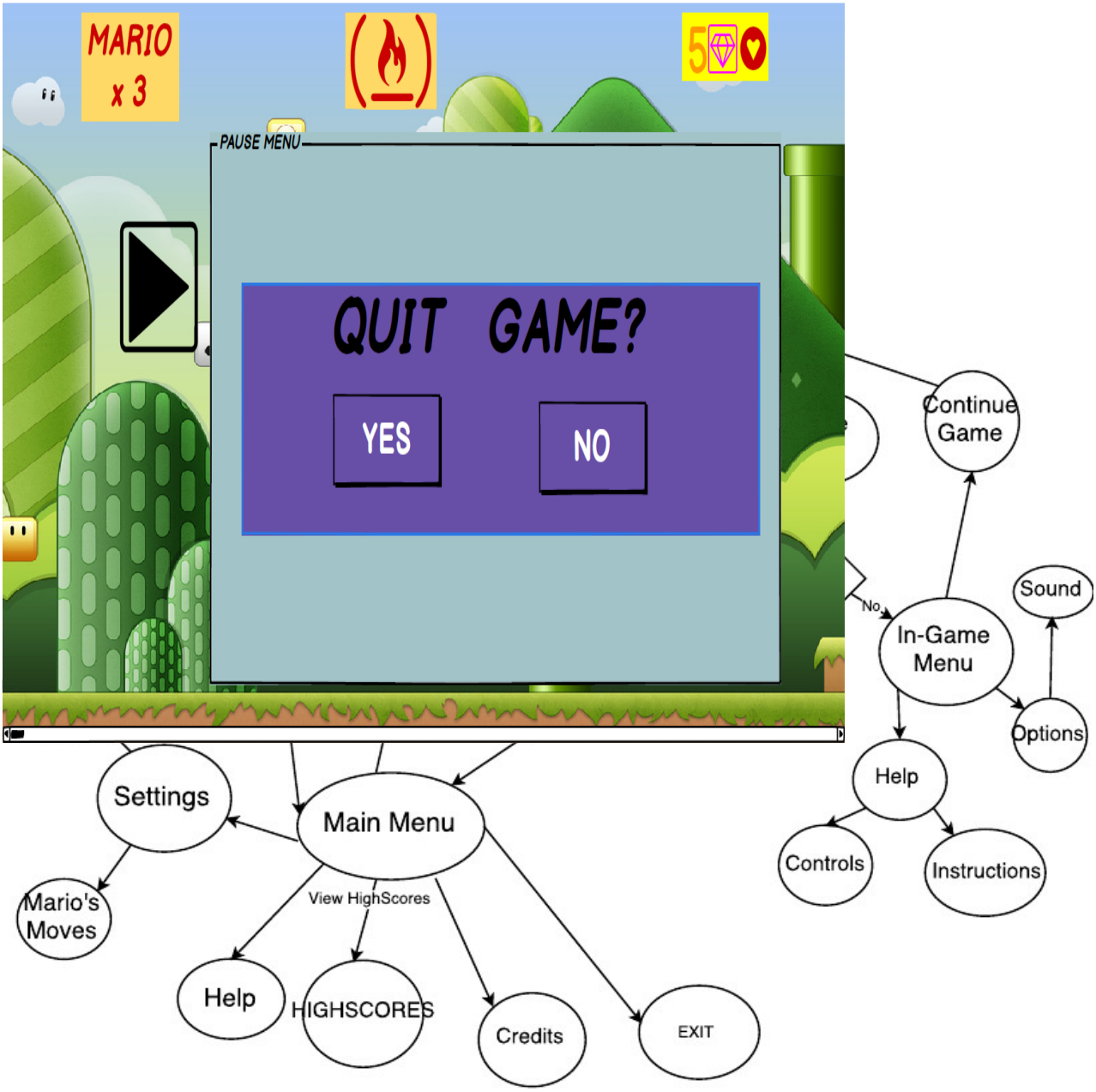


Figure 20: Navigational path diagram

Load Game: After player selects “Start Game” from the main menu, if player does not select “New Game” and selects “Load Game”, player can select any game from the previous levels; which are shown in red buttons. Player will not be able to play a game from the levels; which are shown in black buttons; because these are the levels; which the player hasn’t accomplished to pass before.

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 Menu: If the player press the “NO” button in the pause menu; this player encounters with 3 options.

Figure 24: Help menu interface

Control Menu: If the player selects “Controls” button in the help menu, he/she can see which keyboard keys should be used to move, run, jump.

5. References

NES - Super Mario Bros. Tileset https://www.mariowiki.com/Super_Mario_Bros.

Figure 23: 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
`