Bilkent University

Department of Computer Engineering

CS319 – Object-Oriented Software Engineering Project

Project short-name: Object-Oriented Implementation of Donkey Kong Game

Analysis Report

Group 3E
Fuad Ahmadov
Çağatay Küpeli
Sine Mete
Arkın Yılmaz

Instructor: Bora Güngören

Analysis Report Draft
October 6, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object Oriented Software Engineering CS319/3.

Contents

Analysis Report

Project short-name: Object-Oriented Implementation of Donkey Kong Game

1 Introduction

We decided to make Donkey Kong for our project. Donkey Kong is a well-known old arcade game which has been played for many years. It is a basic platform-action game which saving the Pauline from Donkey Kong by climbing over the platform and dodging from enemies aimed of. This report contains an overview of the game, system requirements, use-case's models including scenarios, use-case diagrams, class and dynamic models.

2 Current System

Donkey Kong was created by Shigeru Miyamo in 1981, he came up with some new characters, Jumpman (soon to be Mario) and Donkey Kong. The idea for the name Kong came from the movie King Kong and the 'Donkey' part apparently meant stupid according to a Japanese/English dictionary [1]. Additionally, Donkey Kong is the debut game of Mario and also is notable for being one of the first complete narratives in video game form, told through simplistic cut scenes that advance the story [2], which affected our choice.

3 Proposed System

In addition to original game, we are planning to add some new features and cancel some features from original Donkey Kong.

Our game will have different levels. Aim of the game is to reach Princess by avoiding enemies. Main enemy in the game is monkey who kidnapped the princess. The game will be desktop application and will be controlled by a keyboard.

3.1 Overview

3.1.1 Gameplay

The gameplay focuses on controlling the Jumpman across platforms while dodging and jumping over obstacles. Player in each level had to save Pauline, pink dressed character, from the giant ape Donkey Kong.

To complete the level, Jumpman must reach Pauline without hitting any obstacles. To do so, the player should utilize timely jumps, ladder climbing skills, navigating the correct path, collecting extra lives and using hammer power-ups to destroy barrels while collecting coins to get additional points.

Donkey Kong require great amount of skill and little bit of luck, especially dealing with barrels. During the game, Donkey Kong will throw some barrels which might just fall without rolling, or roll and fall from a random ladder. As well as barrel might spawn fire elementals which is another type of enemy can climb ladders. For example, barrel might fall the ladder you are using at that moment.

3.1.2 Enemy Types

Fire Elemental: Fire Elemental spawns when a barrel hits oil box. Its movement pattern is similar to Jumpman. It can go up and down from ladders. As well as, it can go right and left. It cannot be destroyed by hammer.

Falling Barrel (Blue): This particular barrel only fall from the platforms, it does not need any ladder to fall; however, it cannot roll.

Rolling Barrel (Yellow): This particular barrel can both roll and fall; however, it needs ladder or empty space at the end of the platform to fall.

3.1.3 Enhancements/Power-Ups

Hammer: Hammer allows Jumpman to destroy barrels in front of him. To use hammer, player must reach hammer icon in game and it will last only small amount of time.

Extra Life: Player start with 3 life and he will lose one every time they hit by any obstacles. This allow player to have another chance if they can reach extra life icon.

- **Remark:** If player fall from at the end of the platform, they die and lose life.

Coin: Allow player to earn more score.

3.1.4 Other Objects

Jumpman: Jumpman is an user controlled unit. To complete the level, you must reach Pauline without hitting any obstacles. It can jump, climb and strike.

Pauline: Pink dressed character. Player must reach

Oil: Oil spawns fire elemental when it is hit by a barrel.

Monkey: There is no point reaching monkey. If you get so close, you die.

Ladder: Player uses ladders to climb platforms.

- **Remark:** Player cannot climb from all ladder. Some of them are broken; however, fire elementals can climb from broken ones. Therefore barrels can fall from them.

3.2 Functional Requirements

- Starting a game; this is a mandatory option for games.
- Picking a level; user can pick any level s/he wants before starting the game.

- User will be able to control the Jumpman with keyboard by using WASD or direction keys. Player also use space key to jump and strike with hammer after taking it.
- User can change the settings of the game. Available settings are; sound:on/off and choosing a level.
- User will be able to access the help menu. Help menu contains information about game and controls.
- The game will offer various power-ups.
- The game will contain a credits panel.

3.2.1 Possible Future Functional Requirements

- There can be pausing the game option.
- There can be high scores table.
- We can add more levels.
- We can add save game option which is not necessary for this type arcade games.
- We can add a setting for personalizing the game like changing the colors etc.

3.3 Non-functional Requirements

- The response time will be low and effective so that enables users to play with minimal delay. The original game is not good at response time.
- The design of the game will be simple and understandable.

3.4 Pseudo Requirements

- The game will be implemented in Java.
- The game will be a desktop application.
- Adobe Photoshop will be used for design some elements.

3.5 System Models

3.5.1 Use-Case Model

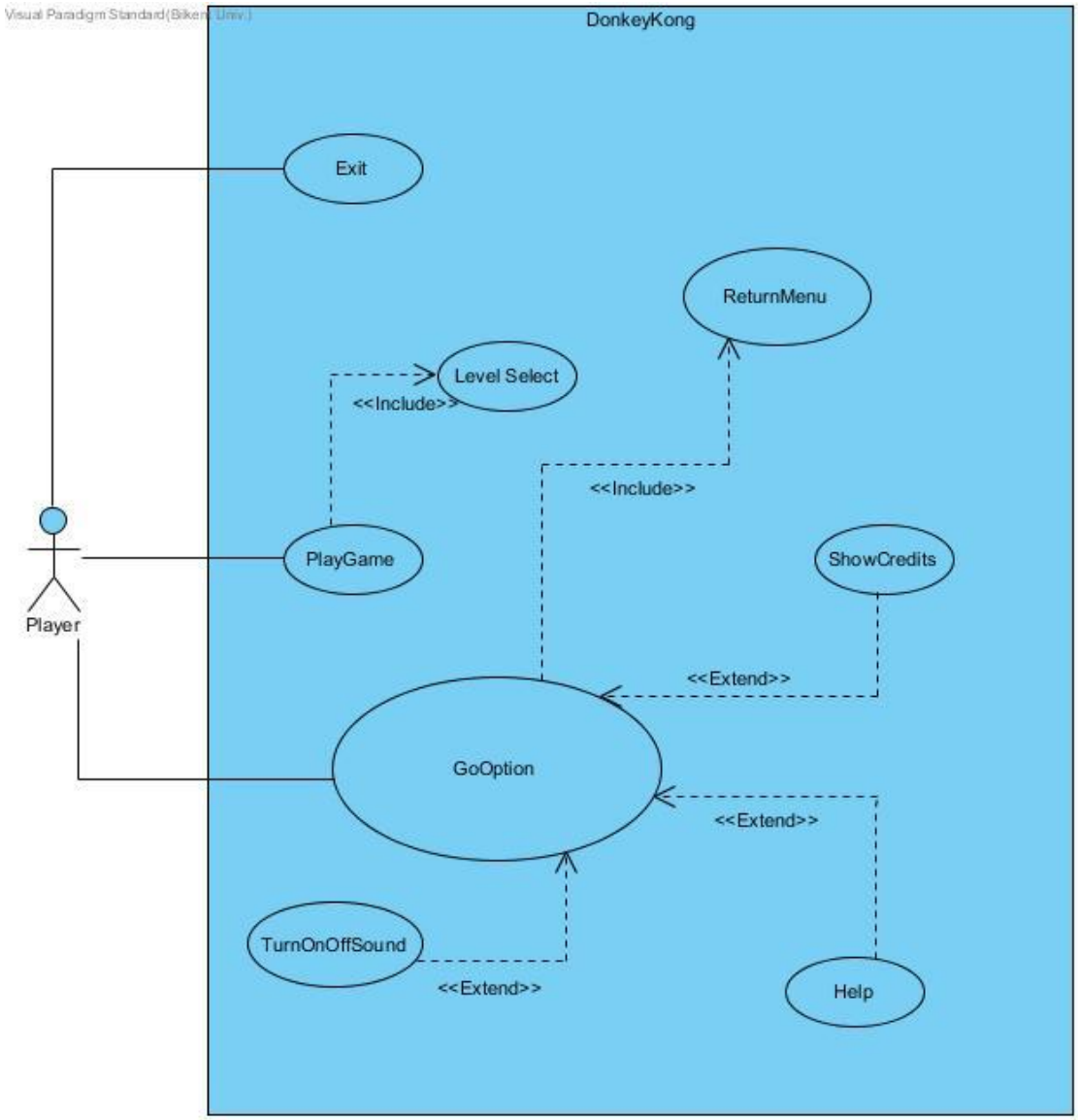

Figure 3.5.1.1: Use Case Diagram

3.5.1.1 Use-Case Descriptions

Use Case #1

Use case name: PlayGame

Participating actors: Player

Entry condition: Player has opened main menu and selected the PLAY GAME button.

Exit condition:

- Player has to complete all of the levels, OR
- Player has to lose all of the lives in levels, OR
- Player has to exit the game by clicking the exit button.

Main Flow of Events:

1. Player clicks the "Play Game" Button
2. System comes up with level choosing page.
3. Player chooses particular level where he/she wants to start from.
4. Player completes all levels.
5. Player returns to the main menu of the game.

Alternative Flow of Event:

- Player loses all of his/her 3 lives and game returns to Main Menu page.(System comes up with "GAME OVER" pop-up message)
- Player can exit the game any time.

Use Case #2

Use case name: Help

Participating actors: Player

Entry condition: Firstly, game and then option has been opened by Player and he/she is on Options menu.

Exit condition: Player returns to main menu.

Main Flow of Events:

1. Player chooses"Help" from Options menu.
2. Player reads instructions shown by the system.
3. Player returns to the Option menu.

Use Case #3

Use case name: ShowCredits
Participating actors: Player
Entry condition: Firstly, game and then option has been opened by Player and he/she is on Options menu.
Exit condition: Player returns to main menu.
Main Flow of Events:

1. Player chooses "Credits" from Options menu.
2. Player reads name of the developers shown by the system.
3. Player returns to the Options menu.

Use Case #4

Use case name: GoOption
Participating actors: Player
Entry condition: Player has opened main menu.
Exit condition: Player returns to main menu.
Main Flow of Events:

1. Player chooses "Options" from main menu.
2. Player can turn off/on the sound, open credits, get help.
3. Player returns to main menu.
4.

Alternative Flow of Event:

- Player does not change anything. (go to step 3)

Use Case #5

Use case name: ReturnMenu
Participating actors: Player
Entry condition: Firstly, game and then option has been opened by Player and he/she is on Options menu.
Exit condition: Player returns to main menu.

Main Flow of Events:

1. Player chooses "Return Menu" from Options menu.
2. Player returns to main menu.

Use Case #6

Use case name: LevelSelect
Participating actors: Player
Entry condition: Player is on main menu.
Exit condition: Player returns to main menu.
Main Flow of Events:

1. Player clicks "Play Game" button from main menu.
2. Player chooses the particular level.
3. Player completes all levels.
4. Player returns to the main menu of the game.

Alternative Flow of Event:

- Player returns to main menu without playing the game.

Use Case #7

Use case name: TurnOnOffSound
Participating actors: Player
Entry condition: Firstly, game and then option has been opened by Player and he/she is on Options menu.
Exit condition: Player returns to main menu.
Main Flow of Events:

1. Player chooses "Turn On/Off Sound" from Options menu.
2. Player changes the sound.
3. Player returns to the Options menu.

Use Case #8

Use case name: Exit
Participating actors: Player
Entry condition: Player is on main menu.
Exit condition: Program is closed.
Main Flow of Events:

1. Player clicks "Exit" button from main menu.
2. Program is closed.

3.5.2 Object and Class Model

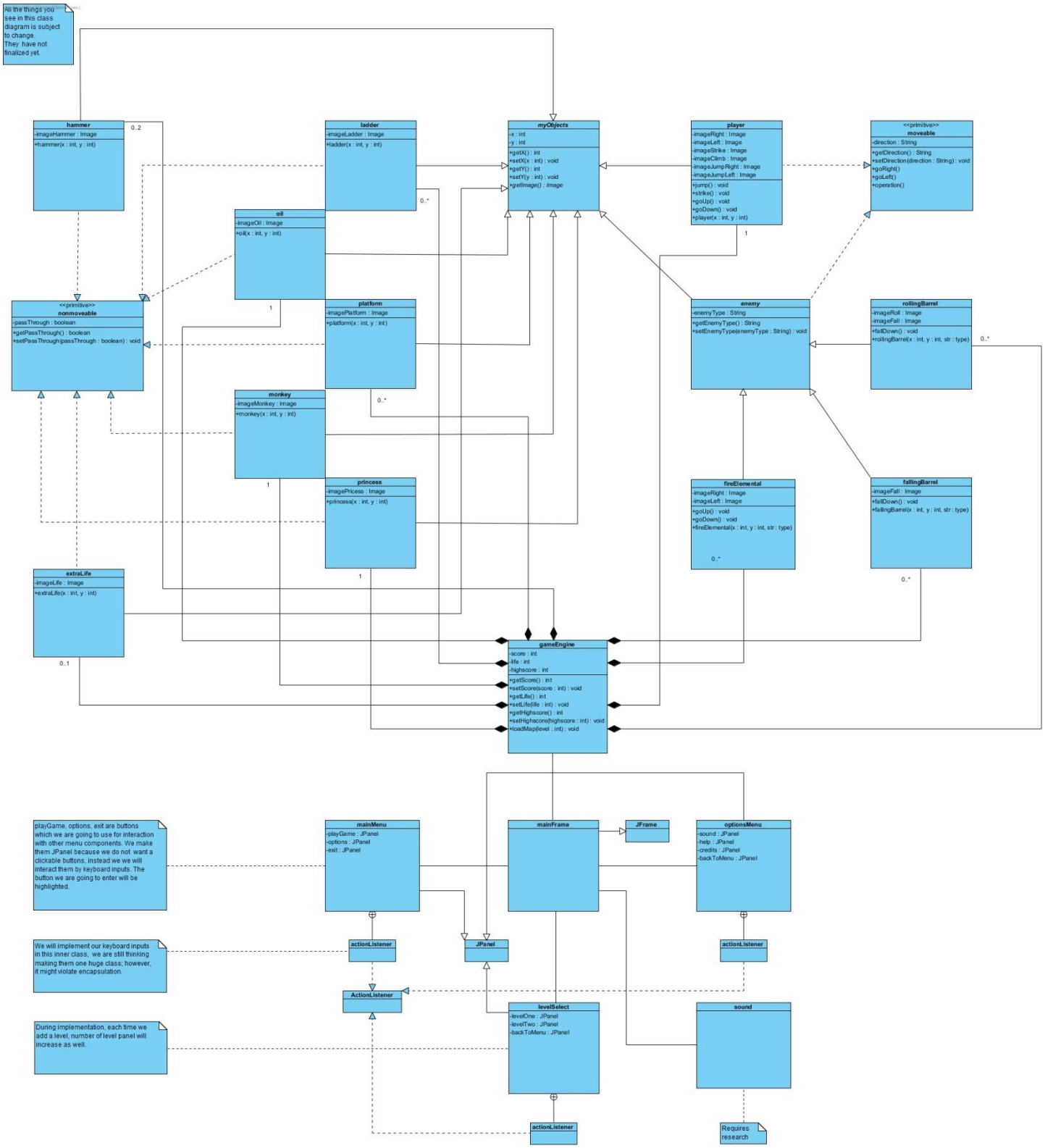

Figure 3.5.2.1: Class Diagram

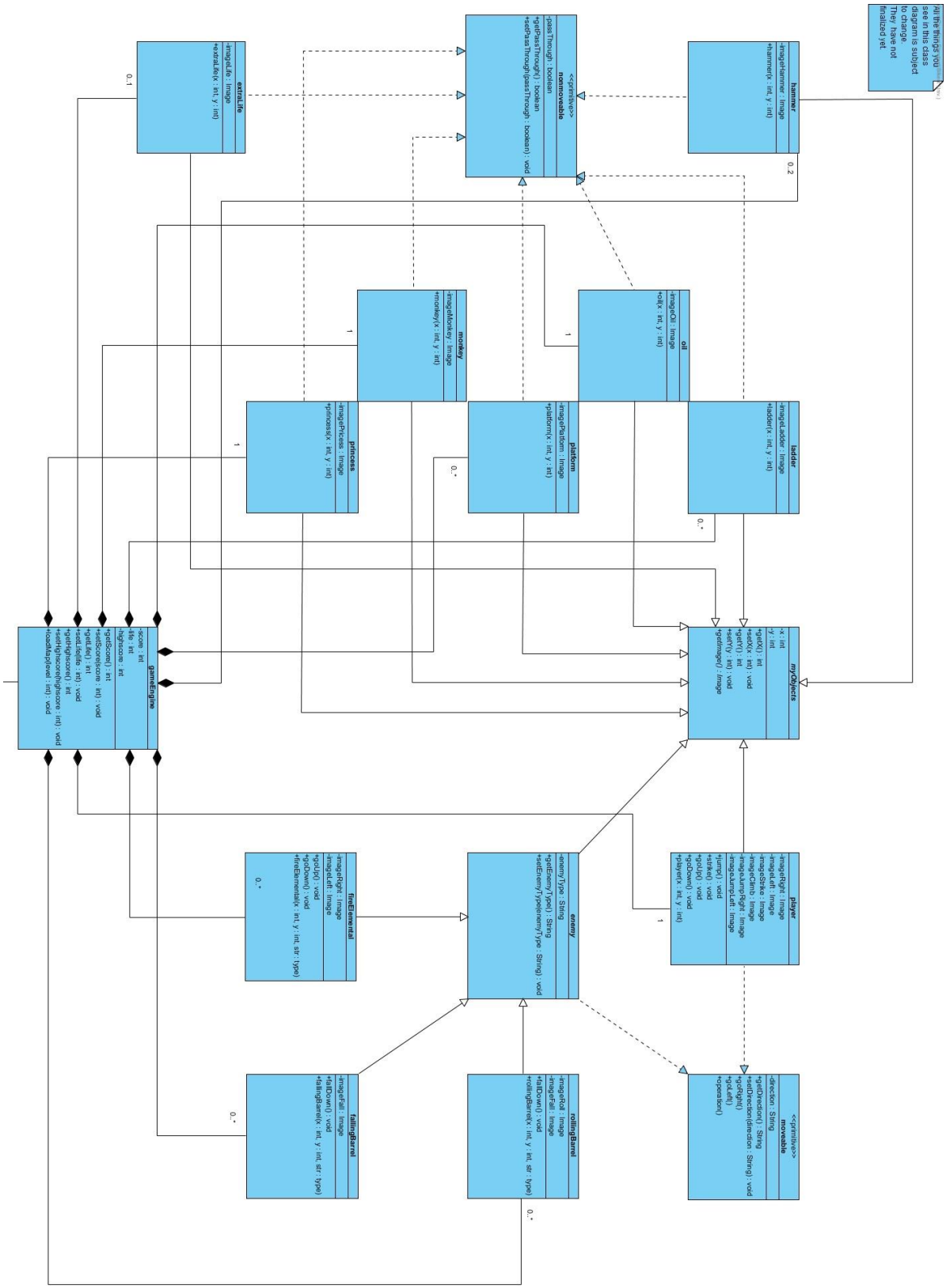

Figure 3.5.2.2: Class Diagram Object Part (For Readability)

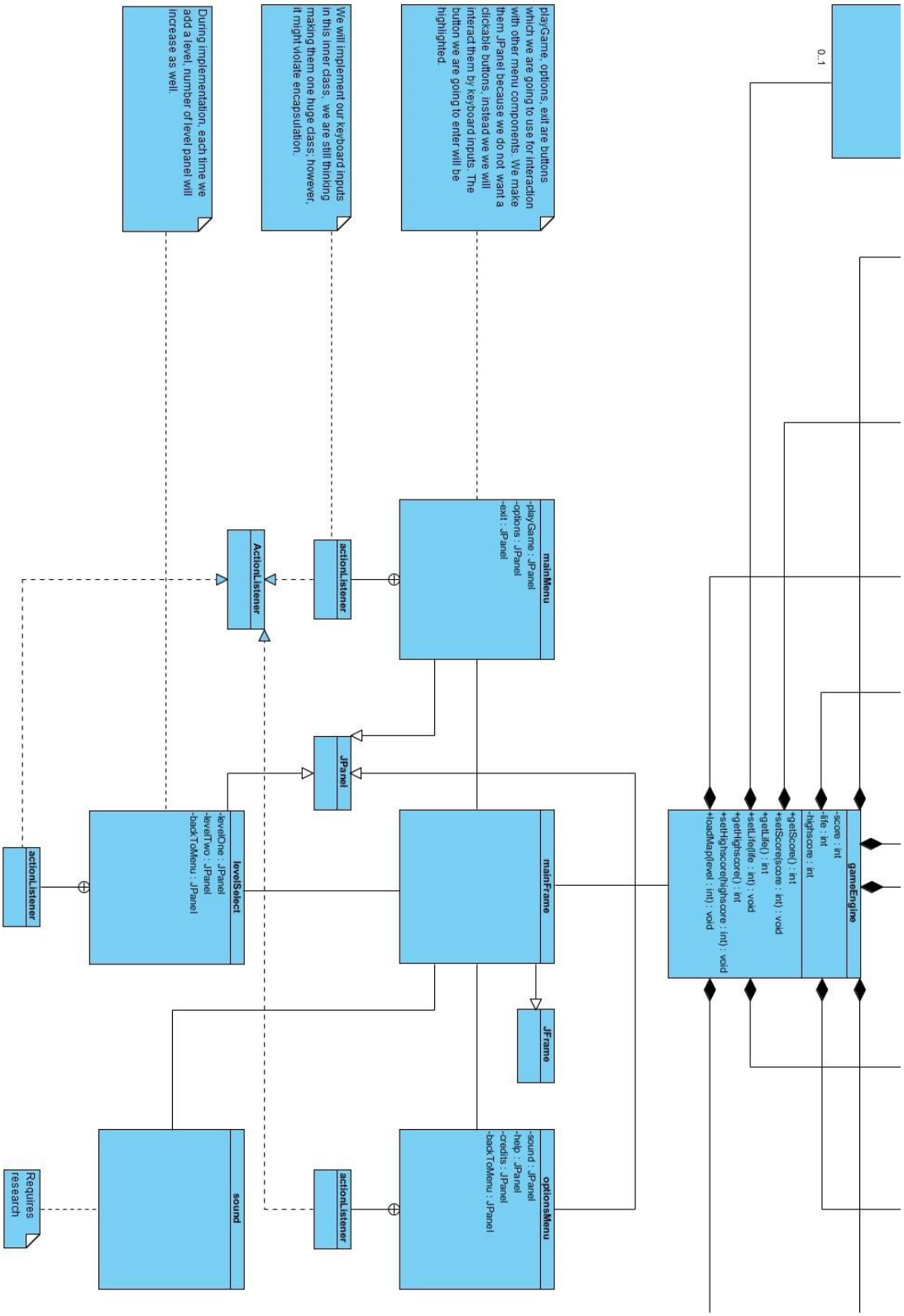

Figure 3.5.2.3: Class Diagram GUI Part (For Readability)

Currently our design uses Java Swing library; however, it is subject to change. When you run the game, first thing it does is creating a JFrame called mainFrame. We will use this frame as our starting point. Everything will render on this frame and all other menus are run under this frame. Every menu has its own JPanel's and Action Listener's.

myObjects is an abtract class which is extended by almost every game object in the diagram. Even though it looks complicated, they can be divided into 2 different tittle. First one is moveable which will help us to design object which moves in the game such as player and enemies. Other one is nonmoveable which will help us to differentiate interactive and uniteractive objects such as hammer and monkey. In gameEngine, by creating instance of these object, we will launch the game.

gameEngine class is the center of all classes. It will interact both game mechanics, user inputs and data transaction. When user choose a level from levelSelect menu, gameEngine construct the chosen level and take care of everything which is part of the gameplay. The rules of the game, such as player cannot go through platform etc., will be implemented under this segment. gameEngine class will also deal with user input and rendering. Furthermore during the gameplay, score will be calculated under gameEngine class. For design purposes if this class becomes hard to deal with it, it might end up divided into pieces.

3.5.3 Dynamic Models

3.5.4 User Interface

Figure 3.5.4.1: Main menu of the game

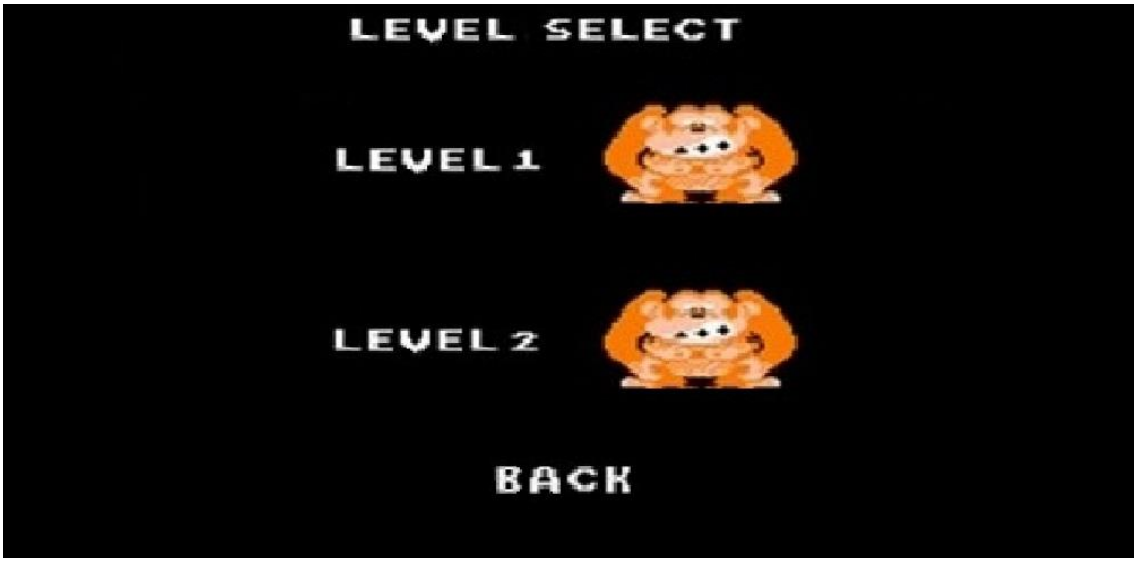

Figure 3.5.4.2: Level Selection

Figure 3.5.4.3: Options Menu

4 References

[1] "The History of Donkey Kong." *Classic Gaming,* <http://www.classicgaming.cc/classics/donkey-kong/history>

[2] "Donkey Kong." *MobyGames,* <www.mobygames.com/game/donkey-kong>.