
Bilkent University

Department of Computer Engineering

# CS319 Object Oriented Software Engineering Project

*Donkey Kong Game*

## Final Report

Group 3E
Fuad Ahmadov
Çağatay Küpeli
Sine Mete
Arkın Yılmaz

Supervisor: Bora Güngören

Final Report Draft
Nov 4, 2017
This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS319/3.

# Contents

# Final Report

*Donkey Kong Game*

## 1. Changes in the Implementation

### 1.1 Game Algorithm & Game Visual Subsystem

We scrapped this subsystem instead we only have Game Algorithm Subsystem right now. We already know that extending JPanel in GameEngine class was violating our design pattern; however, we could not figure out how to divede algorithm part and displaying part into two pieces. Then during implementation, we realize that we can make another class and that class can take GameEngine as an object. Therefore we end up with two classes.

First class is for Game Algorithm Subsystem which we call GameEngine currently and other class is for Game Visual Subsystem which we call GamePanel. Then we add GamePanel class to User Interface Subsystem by removing Game Visual Subsystem.

GamePanel extends JPanel and implements Runnable. Therefore it renders the game. It also has a nested class called KeyHandler which extends AbstractAction to take inputs from user. Unlike other classes in our design, this class uses KeyBindings instead of KeyListeners. The reason is the implementer (Çağatay Küpeli) has read KeyListeners might cause problems if the screen GamePanel loses its focus during runtime and implementing them takes no more than implementing KeyListeners.

In summary, GameEngine is just a regular class with no extends and implements. We are using it as the game holder. For example, when a player start a new game, we call new GameEngine class for that game in GamePanel class. Therefore we are using this new called GamePanel which extends JPanel and implements Runnable to display game state. Lastly we remove the Game Visual Subsystem part and move GamePanel class to User Interface Subsystem.

### 1.2 User Interface Subsystem

We start our implementation with MainMenu class. To improve the design, the implementer (Arkın Yılmaz) scrapped the idea of using JPanels as buttons instead he returned back to our starting point which is using JButtons. We use JLabels to display our tittles and a lot of constants to initialize the components. We implement KeyListeners to this class to move around options. Currently we did not start OptionMenu and LevelSelect. However we are expecting them to change according to how we handle with MainMenu. We also have a new brand mockup for MainMenu class.

We add a new class called GamePanel. Design goal of adding this class to this subsystem is explained under section 1.1.

### 1.3 Game Entities Subsystem

We were happy with our Game Entities Subsystem design. Therefore nothing has changed; however, we did not finish implementing our new class GamePanel class. Our plan is to reduce the number of class which implements Nonmoveable interface to one. If we can get away by only using one class and enum types to realize the type of the object and send correct Image for that object, we planning to make them one class.

### 1.4 Input Management Subsystem

Input Management Subsystem has scrapped. Instead, implement them as nested classes.

### 1.5 File Management Subsystem

GameData class has scrapped. Instead we have MapData and ScoreData classes. Our design philosophy of scraping is to add new feature to our game. Our first improvement is to show high scores as an option in main menu. In old design GameData, both interact with map files and score file. Therefore we end up with a design that MainMenu class can access map files as well. To get rid of this interaction, we decided to divide GameData class into two pieces.

## 2. Current Status of the Project

### 2.1 Game Algorithm Subsystem

We have started implementing GameEngine class with new design we come up.

- We initialized its interaction with File Management Subsystem (MapData & ScoreData).
- We start establishing connection with GamePanel class. Currently, it has not finished yet.
- Similarly we start establishing connection with Game Entities Subsystem. The implementer took commands to where and how we implement algorithm; however it is just a psedocode right now.

### 2.2 User Interface Subsystem

We are done with MainMenu class.

We have start implementing GamePanel class.

- We implement thread generating and KeyBindings successfully.
- We start establishing connection with GameEngine class. Currently, it has not finished yet.
- We add a new feature we mentioned in previouse report which is pause option. It is finished; however we need to test and debug it after we are done with GameEngine class because we are taking game state from GameEngine. Without a state, you cannot draw it to screen.

We are almost done with MainFrame class. Only thing is left is to establish movement between OptionMenu, LevelSelect and GamePanel. We did not start that part because we did not implement OptionMenu or LevelSelect.

### 2.3 Game Entities Subsystem

We are done with all classes in this subsystem. Only thing left is not code related. We need to add images to some file in order to take them as input

### 2.4 File Management Subsystem

We are done with both MapData and ScoreData. However they will change according to map size. Therefore we need to return back to our design to change some numbers.

- The reason of future changes is we are using static 2D arrays to store our map as matrix.

# 3. Changes in Implementation with Class Diagrams

## 3.1 Game Algorithm Subsystem

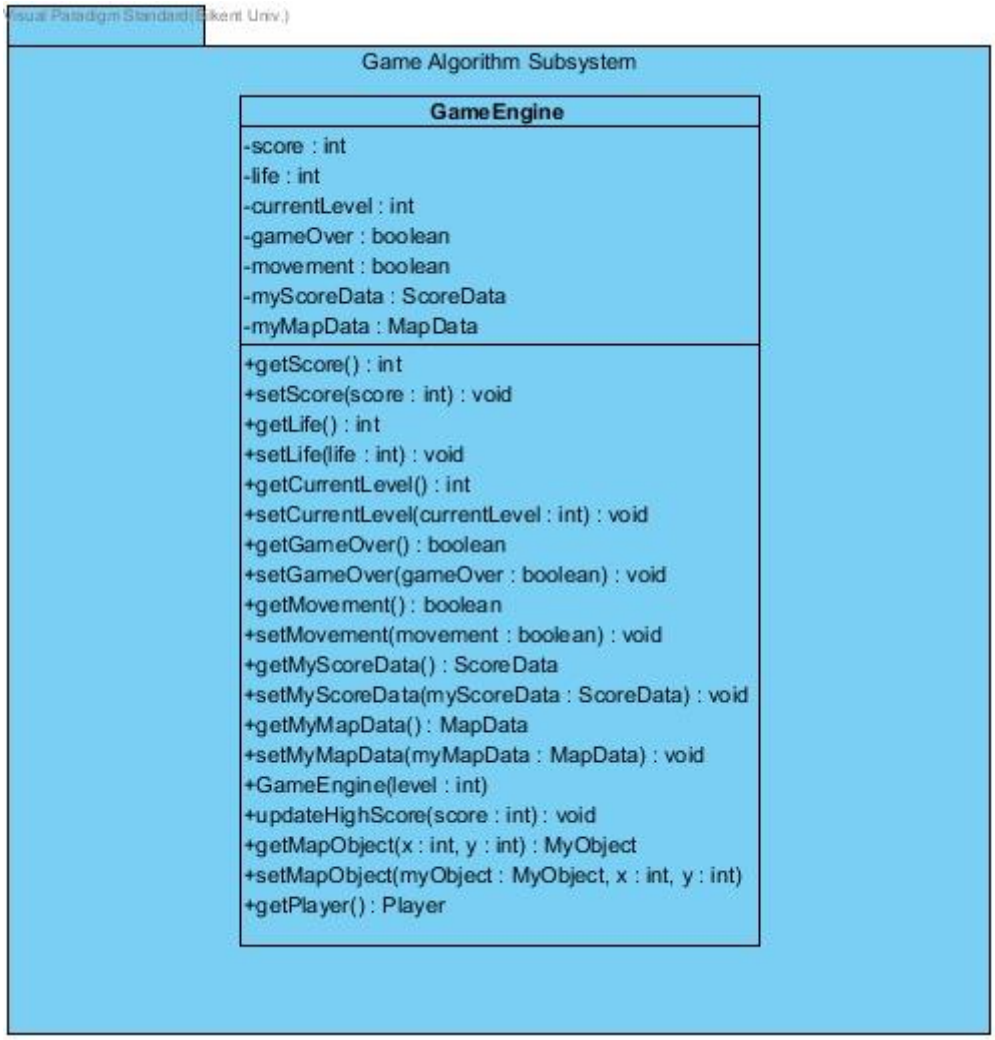


*Figure 1 Game Algorithm Subsystem*

## 3.2 User Interface Subsystem

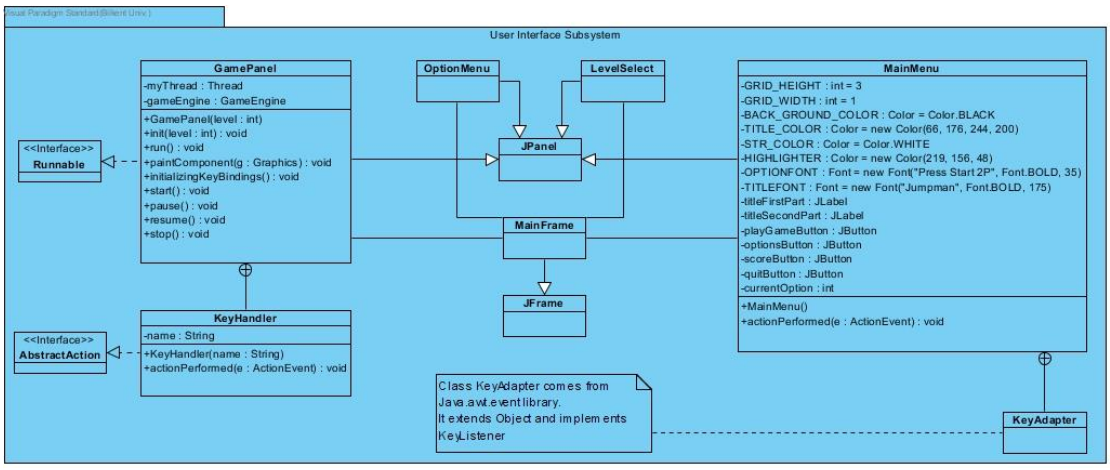

*Figure 2 User Interface Subsystem*

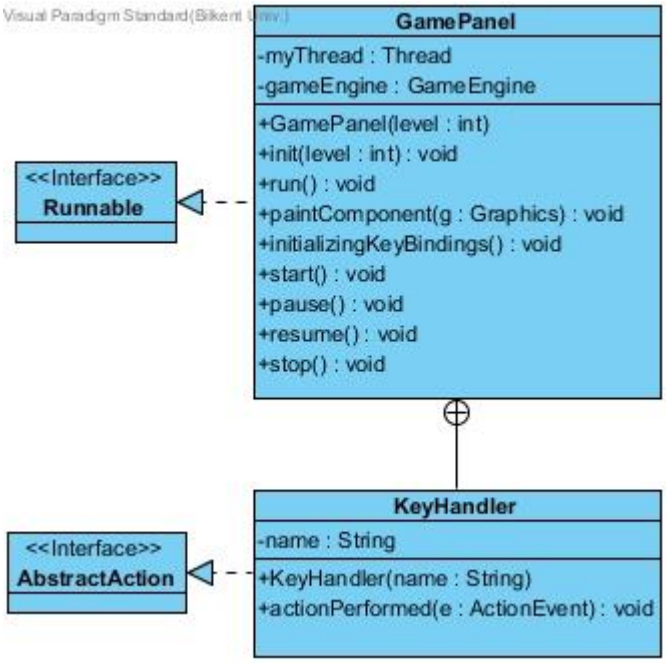


*Figure 3 GamePanel Class*

*Figure 4 MainMenu Class*

### 3.3 File Management Subsystem

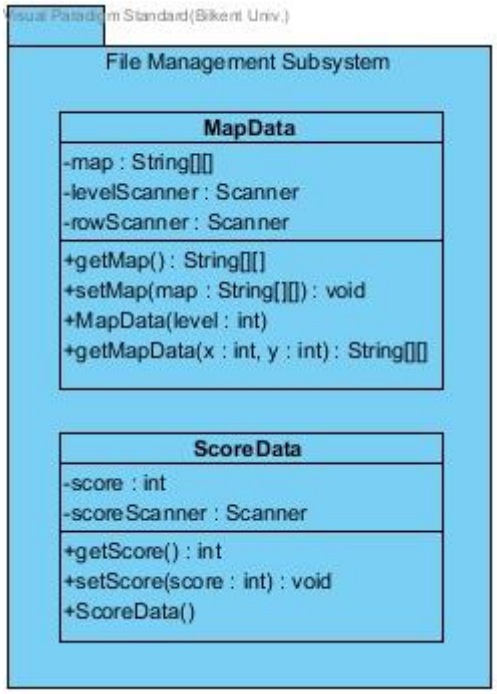


*Figure 5 File Management Subsystem*

## 4. User Guide

### 4.1 System Requirements

Donkey Kong Game premastered version of famous arcade game called Donkey Kong. For the sake of CS 319 (Object Oriented Software) class, we decided to re-implement Donkey Kong by using Java. For this reason, Java Run Environment (JRE) must be installed before playing the game. You can download it in following link;

http://www.oracle.com/technetwork/java/javase/downloads/

Minimum System Requirements

- Screen Resolution: 750x750

### 4.2 Installation

Currently game is not finished yet; however our plan is to make it a runnable jar file. Therefore we will provide the code as well and you will be able to run it with Java IDE.

### 4.3 Changes in Mockup Design