Bilkent University
Department of Computer Engineering

OBJECT-ORIENTED SOFTWARE ENGINEERING PROJECT DESIGN REPORT

CS 319 Project: Dribble & Score

25/03/2017

Group name:
2C

Group Members:

Batıkan HAYTA 21301382
Berke SOYSAL 21400908
Boran YILDIRIM 21401947
Sencer Umut BALKAN 21401911

Instructor:
Prof. Dr. Uğur DOĞRUSÖZ

1.Introduction

1.1 Purpose of System

Dribble & Score is a PC game which is designed to keep the players entertained as much as possible while playing the game. The user interface will be developed in such a way that even a first-time computer user can easily understand it. This game adapts the user to the game in a very simple way on the first few levels, and when it comes to more difficult levels, there will be nothing that the user does not understand about the game.

1.2 Design Goals

The main goal for a computer game is to entertain the player. To achieve this task, it is needed to focus on little details which are not directly noticeable at a first glance.

This section details the design goals of the system such as end user criteria, maintenance criteria, performance criteria and the trade-offs that come with our chosen way of implementation.

1.2.1 End User Criteria

Target User Base: When we observe other successful games available today, we see that they have one common attribute. It is the wide range of the user base. As an example, Candy Crush is very successful because a kid and an old

person can both enjoy it equally. Our aim is to have a wide age range, to love this game.

Ease of learning: When starting the game for the first time, the user will not know how the controls work. So, it is important to provide the user with a smooth learning curve. First couple levels will be designed in such a way that it teach the controls and the mechanics of the game. After these simple levels, the player will be pushed towards greater challenges to increase the fun factor.

Ease of use: Most computer games use common controls these days to increase the ease of use. The navigation menu will be easy for anyone, as well as the gameplay. Mouse will be used for menu, and the arrow buttons for the gameplay.

1.2.2 Maintenance Criteria

Extensibility: In the lifecycle of a software program, it is important to maintain the ability to add or remove features. As object oriented software engineering principles imply, the first goal in development is to create a highly maintainable software.

Portability: In today's software and hardware world, everything is changing rapidly. However, it has a solution, cross-platform application/game development which runs platform independent. Java is a platform independent programming language such that no matter what operating system or processor architecture is system used, if the system has JRE then there will be no problem. "Write once, run everywhere"

1.2.3 Performance Criteria

Smooth Graphics: Game will contain open source 8-bit pictures. Main character, bonuses, goalkeepers and the whole level environment is going to be visualised in a retrospective way. Therefore, at this point, we will not experience a noticeable loss of speed in order to process the graphics and move with the game engine.

Input Response Time: Inputs from the keyboard while playing the game needs to be very precise because the players will be very frustrated if the character crashes due to input lag.

1.2.4 Trade-offs

Portability - Performance & Memory: In this project, we will be using the Java Programming Language, which is known for it's ability to produce binaries which can run on any processor architecture. However, this results in all the applications being run in an interpreter called the Java Virtual Machine. When compared to native languages like C or C++, programs written with Java use more resources while doing less. On the flip side, coding our game with Java will decrease the development time due to the broad range of libraries available and the game will be running on any Java compatible desktop device.

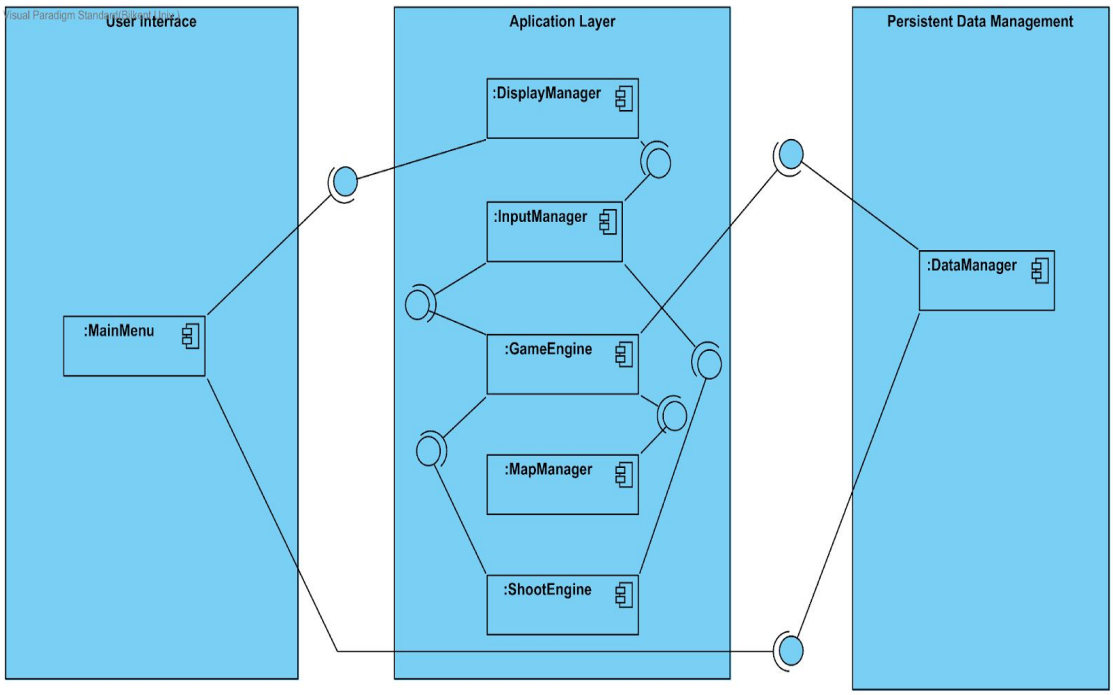
Simplification - User Base: In our end user criteria, we detailed the target user base to be very broad. When a game targets a broad range, it requires over simplification of some game mechanics. If we could have used very detailed and intensive graphics which

would have required a powerful desktop computer, which many users don't have. So, we decided to keep the game simple and have a broader user base.

2. Software Architecture

2.1. Subsystem Decomposition

In this project, we chose the three-tier architecture to design our system because three-tier architecture system is most suitable design for our system structure. In below, there is a tree-tier component diagram of our design.



In user interface layer, there is MainMenu which users interact with the game. MainMenu interact with “DisplayManager” to display necessary things into MainMenu. Also, it interacts with DataManager to get settings of the games.

In application layer we have 4 components which are “DisplayManager”, “GameEngine”, “MapManager”, and “ShootEngine”. “DisplayManager” responsible to give display information to the MainMenu and it sends those information according to the “InputManager”. For “GameEngine” part, this part responsible to create map by the help of the “MapManager” and “DataManager”. “Map Manager” creates random map according to the specific level and it send map objects to the “GameEngine” to create them. When the dribble part end and shoot part take place, “ShootEngine” takes information from “Input Manager” and uses those information to dedicates where the balls should go. “Input Manager” which is listener for our projects. It reads the data which users entered in the keyboard and send them to the responsible components.

In persistent data management, We have “DataManager” which reads and write information to the users' hard disk drive. “DataManager” is one of the important things to our project. It keeps the important data such as score, default settings, levels of the game etc. and send those information to responsible components.

2.2. Hardware/software mapping

Our game will developed by java so it will require a Java Runtime Environment to be executed. For the hardware requirements, keyboard is necessity to play this game to interact with the game.

About the graphics of the game, we decided to use 8-bit images so that required space of the game will be decrease. As a result of this, low system computers can handle. We uses keyboard the takes input from the users in real-time.

2.3. Persistent Data Management

Since our game does not need a complex database system, game data will be stored in the user's hard disk drive. “Data Manager” stores necessary information to the users hard drive disk. Also, there will be images and sounds which are used for games and those things will be stored in user's hard drive disk as well.

2.4. Access Control and Security

There will be no user authentication system for our games and games does not required any internet connection to play. So that there will be no critical information security leak. Some of variables will declare as a private and constant so that outsiders could not change it. “GameEngine” has a access to important files and datas such as map difficulty, level system etc. so that outsiders could not change.

2.5. Boundary Conditions

If there is a corrupted data in the game, program will give an error. While playing a game, if player's life are gone, game will display score of this game and gives two option. One is “Go Main Menu” and other one is “Retry”. Depending on the player respond it will either display main menu or starts the level again. Game will consist of 10 different level and only way to unlock other level is complete the level which comes before. After unlocking all the levels, player can choose any level he/she desires.

3. Subsystem Services

In this section, we will explain the detailed information about the interfaces of our subsystem.

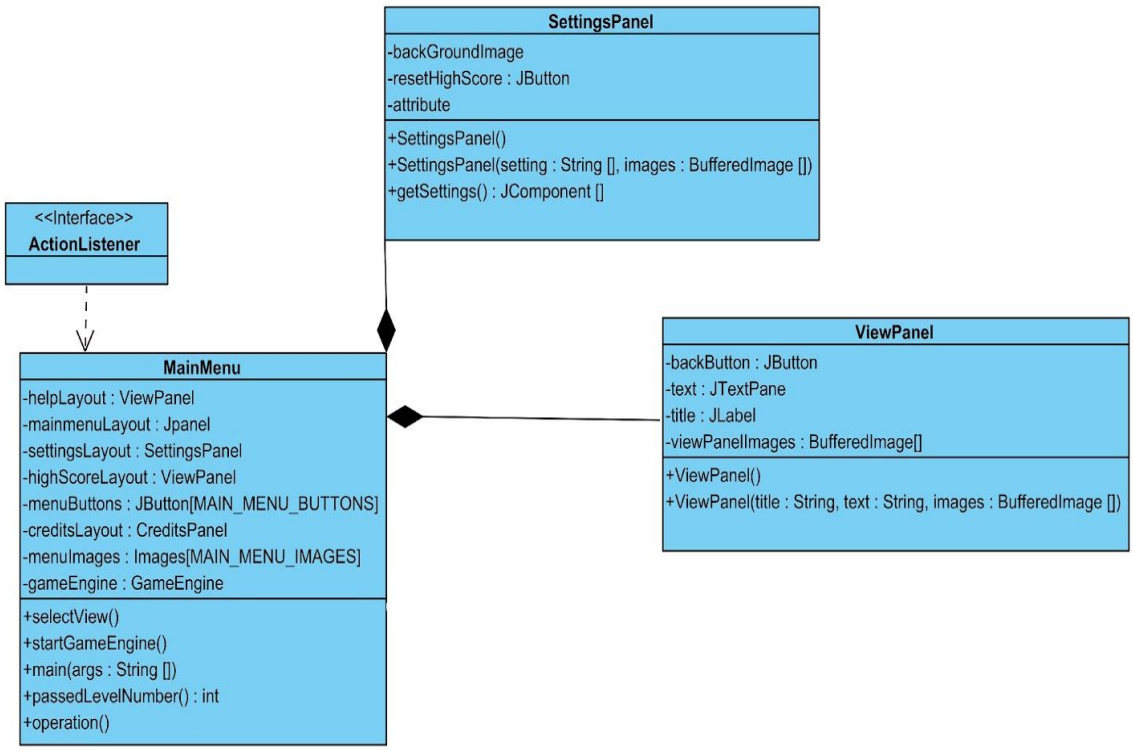
Façade Design

We choose façade design because this pattern provides maintainability, reusability and extendibility. Any change in the components of this subsystem can be reflected by making changes in the façade class so that its easier for us to change and add little things.

In our project, “GameEngine” will be our façade class. We choose “GameEngine” to be façade class because its orginize the game and game objects and has more interaction between other classes.

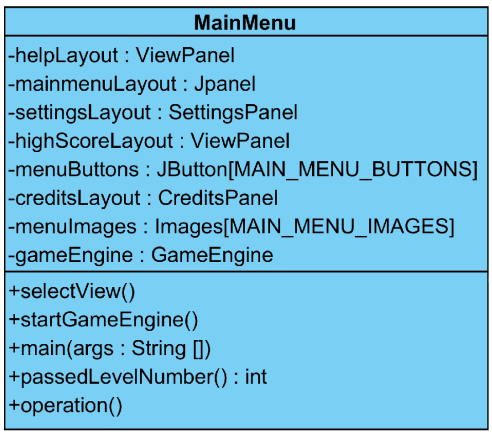
3.1 User Interface Subsystem

This subsystem responsible to provide an interface between user and the system so that users can interact with the system easily.



“ViewPanel” and “SettingsPanel” is responsible to display other submenus such as “Settings”, “Credits”, “Scores”, “Levels” etc. “Main Menu” controls these clases so that when user wants to reach those submenus.

3.1.1 MainMenu Class

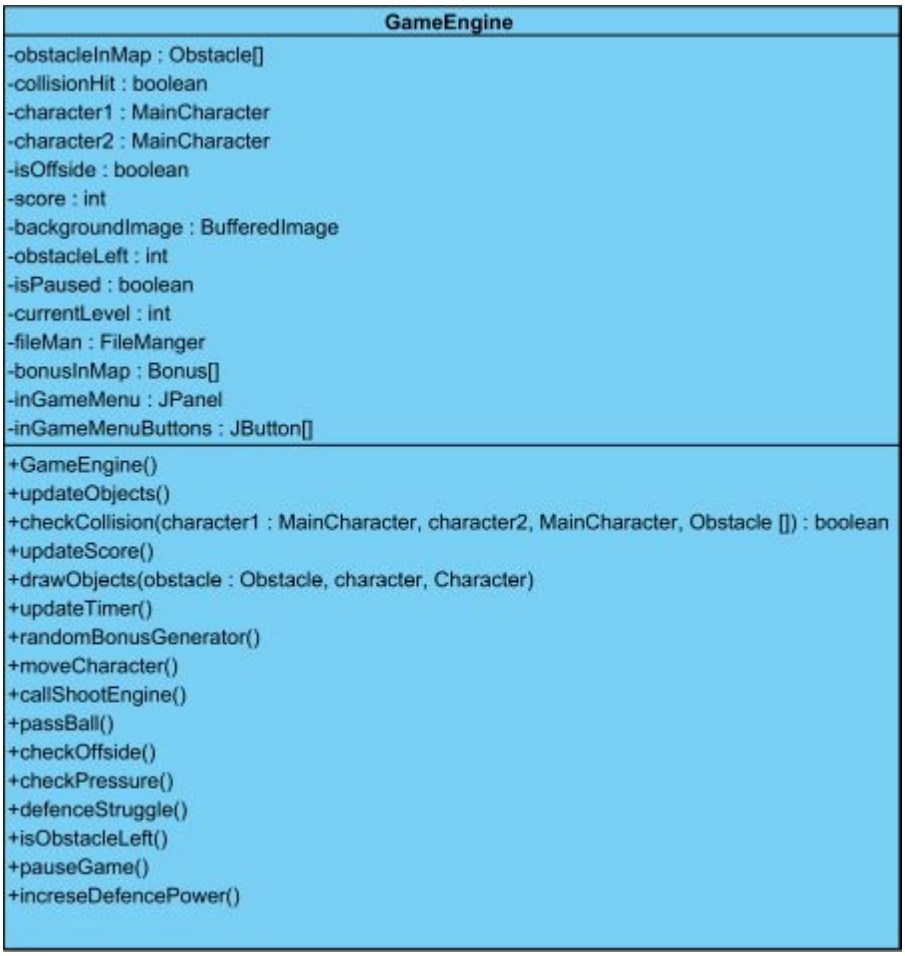


“Main Menu” class is the first class when game is first executed and its first interface that users can see. In this menu, users can see six buttons which are “Levels”, “Settings”, “Scores”, “Help”, “Credits”, and “Exit”. “passedLevelNumber” methods has levels that users unlocked succesfully through completing them. “View Panel” and “Setting Panel” helps Main menu class to view those interfaces to the users.

3.2 Application Subsystem

Application layer is responsible to handle game mechanics such as collision check, create map and display. Our façade class “GameEngine” is responsible to create random map and other methods that helps gameplay mechanics.

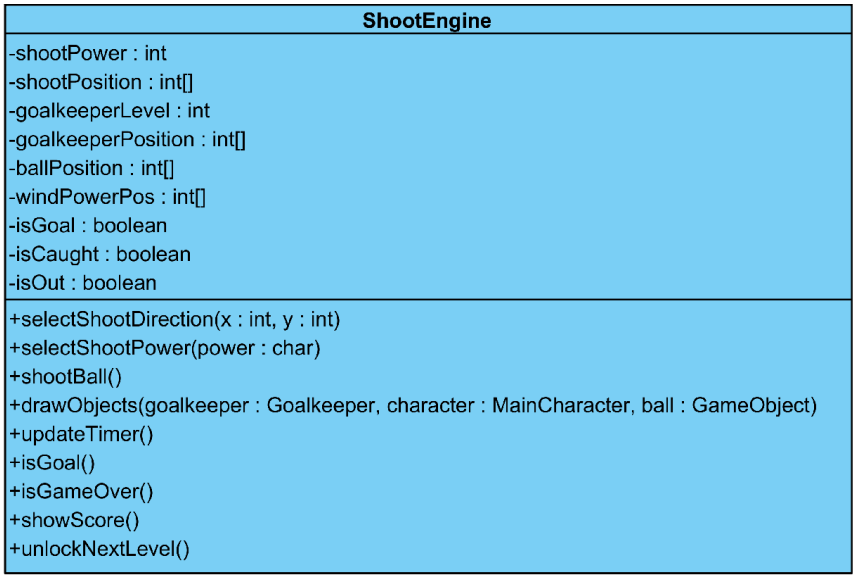
3.2.1 GameEngine Class



After player starts the level, “GameEngine” takes place to create map according to the selected level and obstacles. In map creation phase, it takes information from “MapManager” to create“ the map and DataManager” gives the settings and levels information to the “GameEngine” class. After creating objects of the game and map,

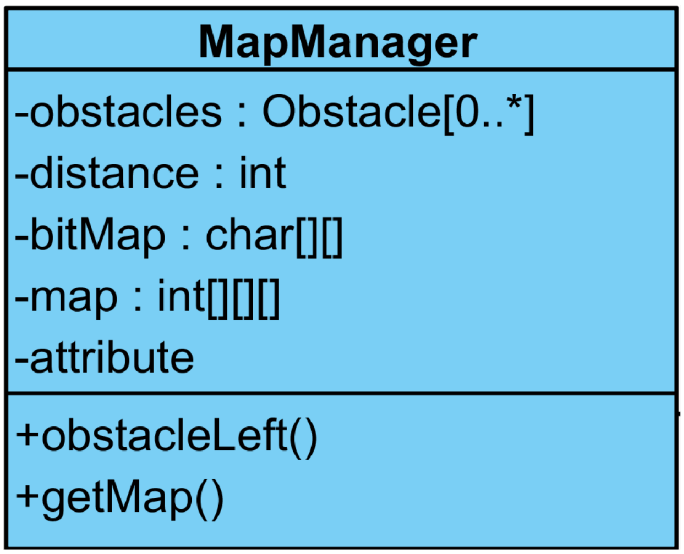
updateObjects() methods helps to adjust speed and positions of the objects. CheckCollision methods checks whether mainCharacter1 or if it is multiplayer game, mainCharacter2, collide with obstacles or not. Finally, isObstacleLeft() check whether obstacle left or not. If it return true, dribble parts will finish and shoots part takes place and shoots part, “ShootEngine” is responsible.

3.2.2 ShootEngine Class



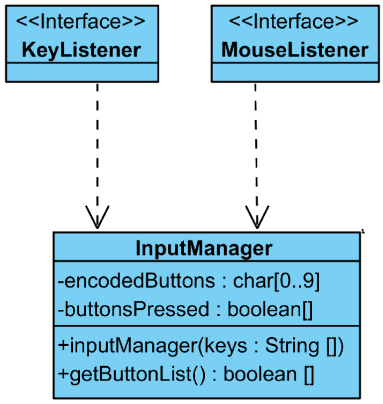
“ShootEngine” class determines the users input from “InputManager” class. According the this information, it adjust ball speed, direction, positions. Also, It adjust goalkeeper level and position according to the respective levels. If isGoal() method return true, it display score of the level and unlockNextLevel() methods unlocked the next level.

3.2.3 Map Manager



“MapManager” is responsible for the creating the map and sending this map to the “GameEngine” to creat them. In map, there are random obstacles and random length of the map.

3.2.4 InputManager Class

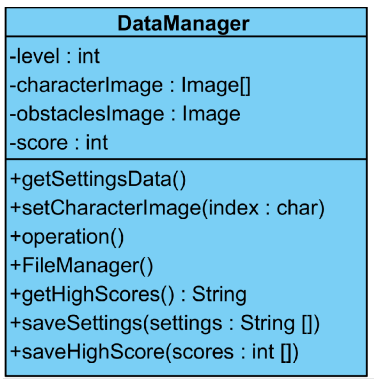


“InputManager” class is listener class of the system. It allows keyboard and mouse control to the user. “encodedButtons” keeps the information of activated buttons. “getButtonList()” method is used by “GameEngine” class to find pressed buttons by the users and changes the MainCharacters accordingly.

3.3 Data Management Subsystem

This subsystem write information to the users hard disk drivers. Also, it has information to help gameplay.

3.3.1 DataManager Class



“DataManager” class has level, character and obstacles images and score information. Also keeps the default settings of the system. When users changes any settings it keeps these changes and send these changes to “GameEngine” before user stars to play.