

Central Asian University

Center of Information Technology and Scientific Computing

Department of CS

**AI BOT FOR A 2D GAME**

Software Design Specification

Team Members

Sukhrob Sotiboldiev

Advisor: My Coursemates

December 2023

Table of Contents

[List of Tables ii](#_Toc514976868)

[List of figures iii](#_Toc514976869)

[Definitions, Acronyms, Abbreviations iv](#_Toc514976870)

[1. Introduction 1](#_Toc514976871)

[1.1 Purpose 1](#_Toc514976872)

[1.2 General Overview 1](#_Toc514976873)

[1.3 Development Methods & Contingencies 1](#_Toc514976874)

[2. System Architecture 2](#_Toc514976875)

[2.1 Subsystem decomposition 2](#_Toc514976876)

[2.2 Hardware/software mapping 4](#_Toc514976877)

[3. Object Model 5](#_Toc514976878)

[3.1 Class Diagram 5](#_Toc514976879)

[3.2 Sequence Diagram 6](#_Toc514976880)

[3.2 State chart Diagram 12](#_Toc514976881)

[4. Detailed Design 12](#_Toc514976882)

[References 20](#_Toc514976883)

# List of Tables

[Table 1: General description of Game class 13](#_Toc514976938)

[Table 2: Attribute description of Game class 13](#_Toc514976939)

[Table 3: Operation description of Game class 14](#_Toc514976940)

[Table 4: General description of Player class 14](#_Toc514976941)

[Table 5: Attribute description of Player class 15](#_Toc514976942)

[Table 6: Operational description of Player class 15](#_Toc514976943)

[Table 7: General description of Settings class 16](#_Toc514976944)

[Table 8: Attribute description of Settings class 16](#_Toc514976945)

[Table 9: Operational description of Settings class 16](#_Toc514976946)

[Table 10: General description of Mob class 17](#_Toc514976947)

[Table 11: Attribute description of Mob class 17](#_Toc514976948)

[Table 12: Operational description of Mob class 17](#_Toc514976949)

[Table 13: General description of SpriteSheet class 18](#_Toc514976950)

[Table 14: Attribute description of SpriteSheet class 18](#_Toc514976951)

[Table 15: Operational description of SpriteSheet class 18](#_Toc514976952)

[Table 16: General description of Platform class 18](#_Toc514976953)

[Table 17: Attribute description of Platform class 18](#_Toc514976954)

[Table 18: General description of AI class 19](#_Toc514976955)

[Table 19: Operational description of AI class 19](#_Toc514976956)

# List of figures

[Figure 1: layer 1 component diagram 2](#_Toc514976964)

[Figure 2: layer 2 component diagram 2](#_Toc514976965)

[Figure 3: layer 3 component diagram 3](#_Toc514976966)

[Figure 4: deployment diagram 4](#_Toc514976967)

[Figure 5: class diagram 5](#_Toc514976968)

[Figure 6: play game sequence diagram 6](#_Toc514976969)

[Figure 7: settings sequence diagram 6](#_Toc514976970)

[Figure 8: pause game sequence diagram 7](#_Toc514976971)

[Figure 9: change audio sequence diagram 8](#_Toc514976972)

[Figure 10: change difficulty sequence diagram 9](#_Toc514976973)

[Figure 11: change keybindings sequence diagram 10](#_Toc514976974)

[Figure 12: exit game sequence diagram 11](#_Toc514976975)

[Figure 13: state diagram 12](#_Toc514976976)

# Definitions, Acronyms, Abbreviations

**Layered Pattern**: a pattern that can be used to structure programs that can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer.

**GUI**: graphical user interface

# Introduction

## 1.1 Purpose

The purpose of System Design document is to translate the business requirements and business processes into a technical design that will be used to develop the application.

## General Overview

The system shall follow a layered architecture to implement its functionalities. The system will follow the three-tier architectural style and be organized into three layers: the interface layer, application layer and the storage layer. The interface layer will be the graphical user interface that allows the users to interact with the system. It will contain all the menus and settings. The application layer will contain the system logic and rules for storing data and also retrieving it in accordance with the user’s needs. This is the layer that will allow controlled access to the data files. Finally, the storage layer will store the metadata required for the system.

The three-tier architecture style shall be used because it not only separates the user interface and the metadata, but also provides an application logic layer. The application layer provides a middle layer that allows the data files and the GUI components to be loosely coupled. The application layer has to be modified if there are any changes to the format of the data files and the interface layer will need little or no modification. This will make it easy for clients of this software to modify the data file format and attributes for further research purposes if they wish to do so. This layer makes the system more maintainable and reusable and also hides the complexity of processing data from the users.

## 1.3 Development Methods & Contingencies

The development of this system used a combination of prototyping and structured software designs. Structured design is handy, especially when used in junction with layered architecture. The developers of this system can group similar components together into a structured form while following the three layers as a guideline. In this particular case prototyping has been a big part of the design as users of the system are not able to specify their functional requirements at the initial draft. Rather, they add requirements as they use or visualize some parts of the system to be developed.

# System Architecture

## 2.1 Subsystem decomposition

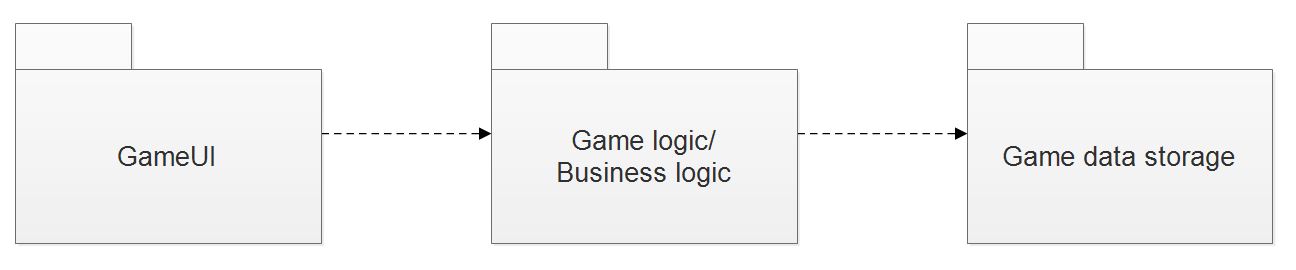


Figure 1: layer 1 component diagram

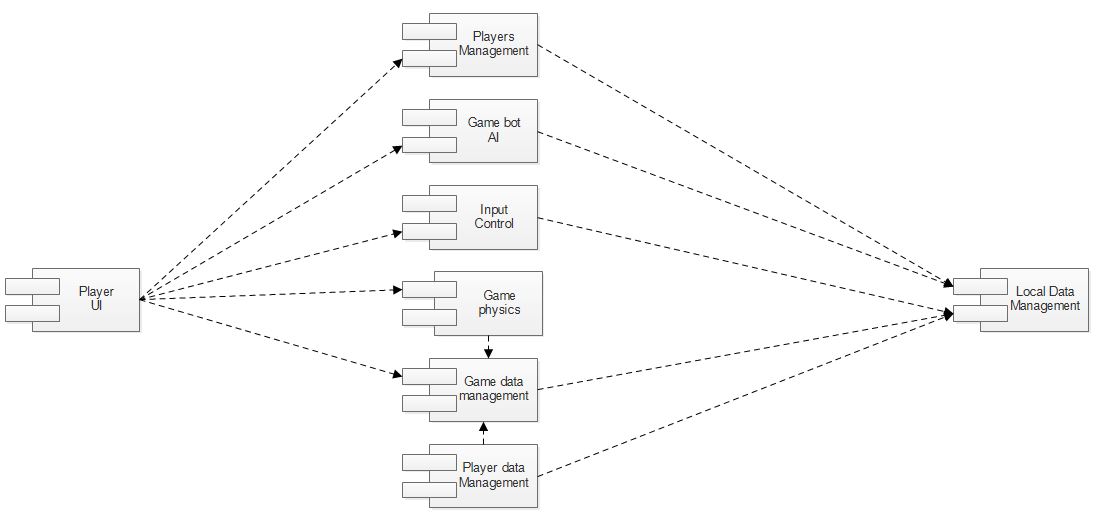


Figure 2: layer 2 component diagram

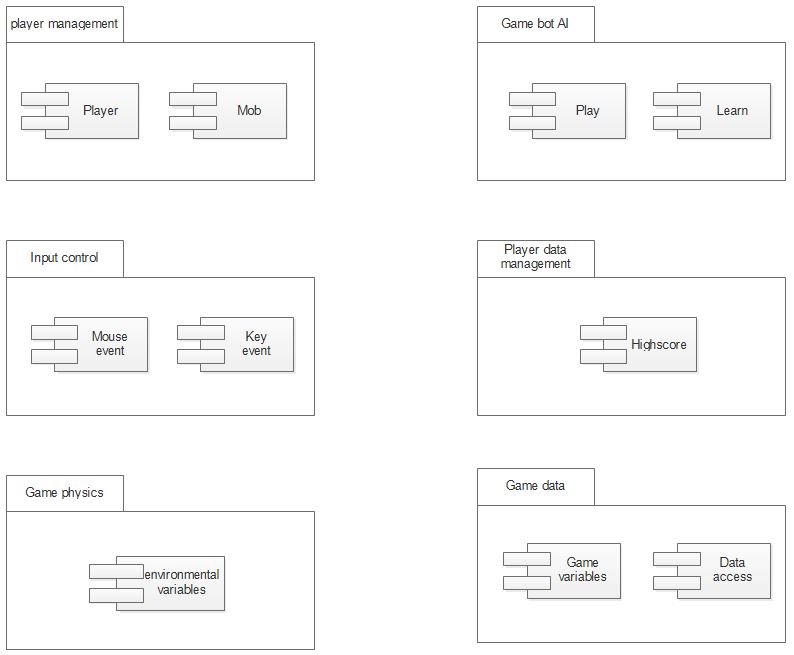


Figure 3: layer 3 component diagram

## 2.2 Hardware/software mapping

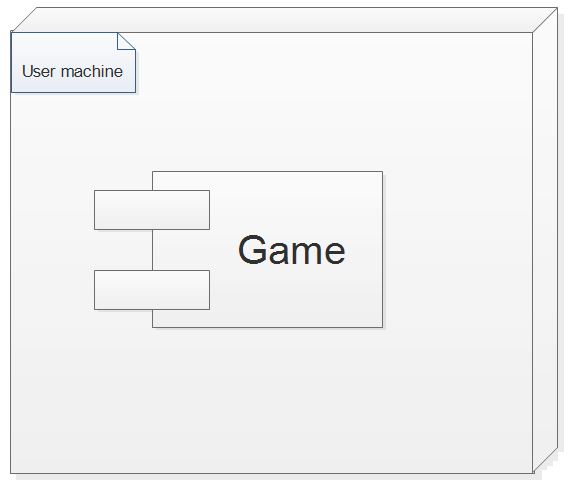


Figure 4: deployment diagram

# 3. Object Model

## 3.1 Class Diagram

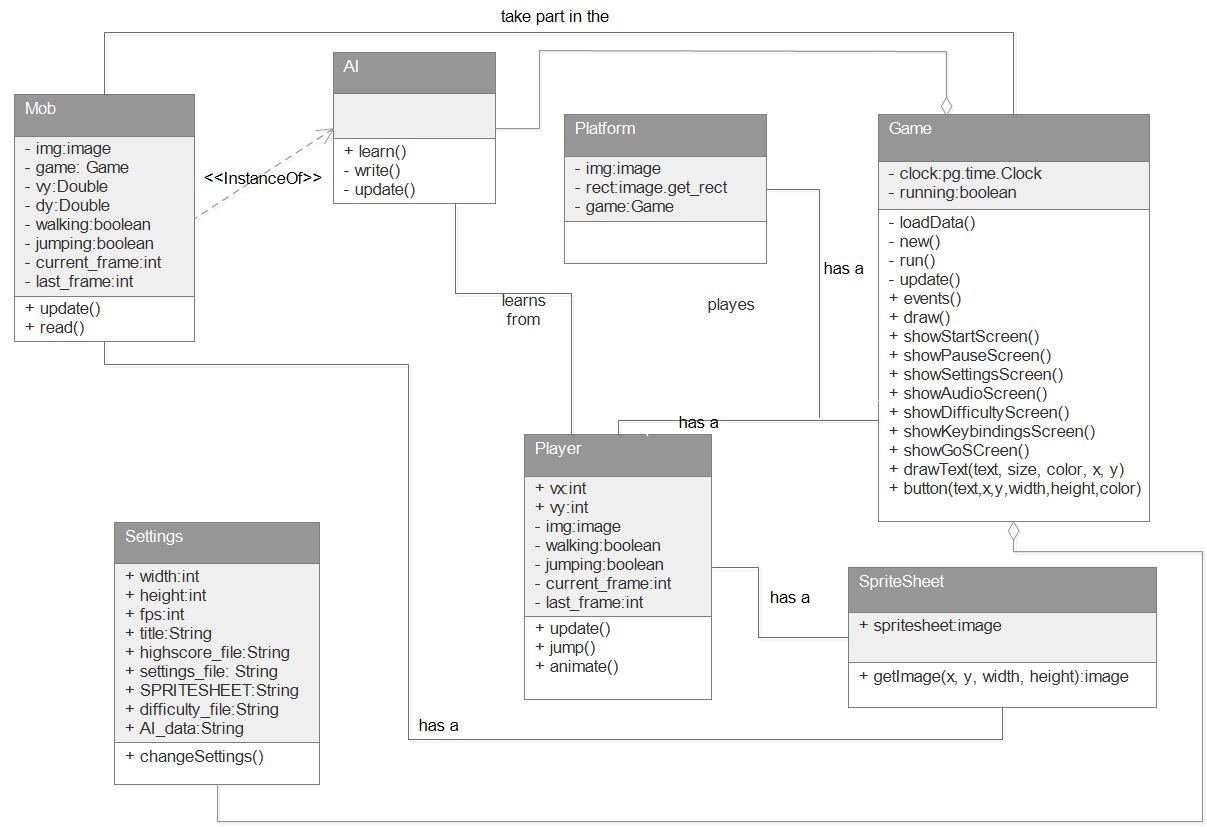


Figure 5: class diagram

## 3.2 Sequence Diagram

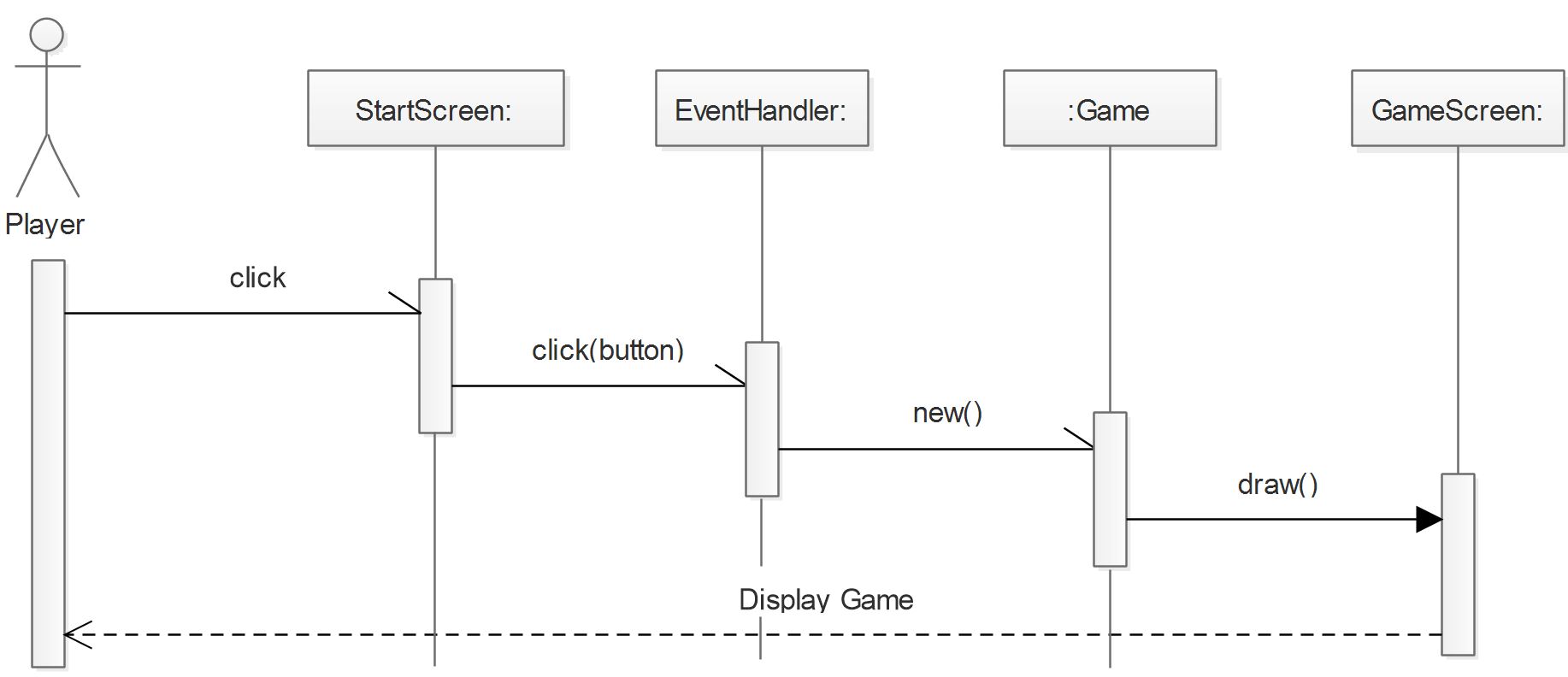


Figure 6: play game sequence diagram

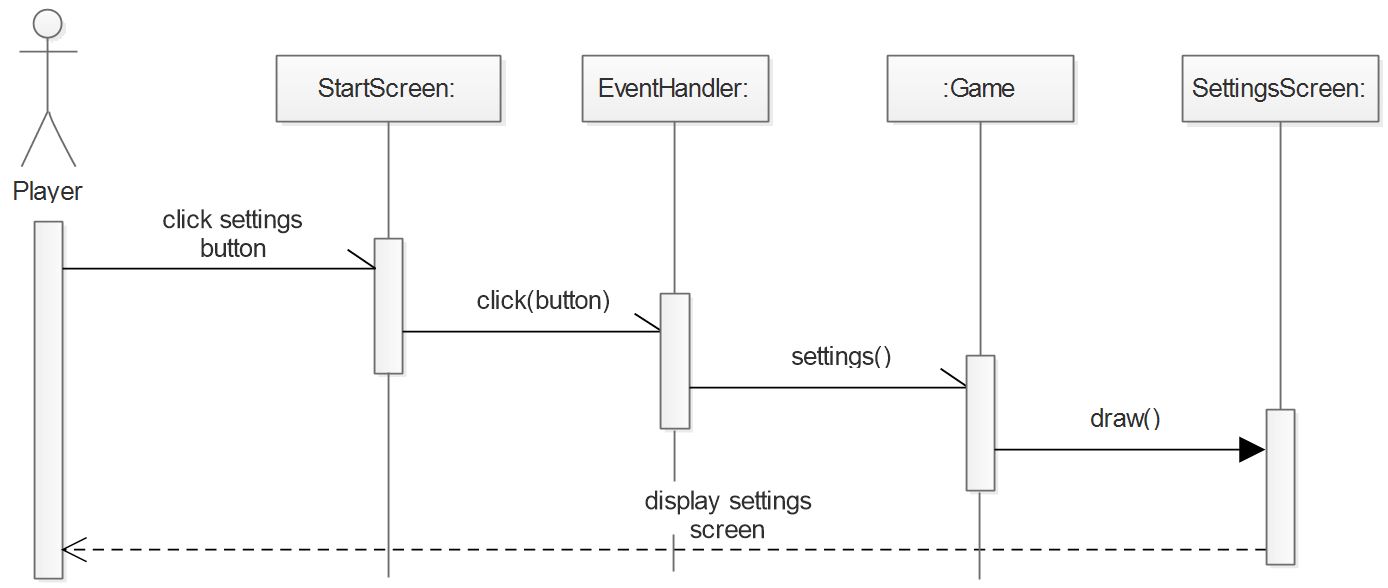


Figure 7: settings sequence diagram

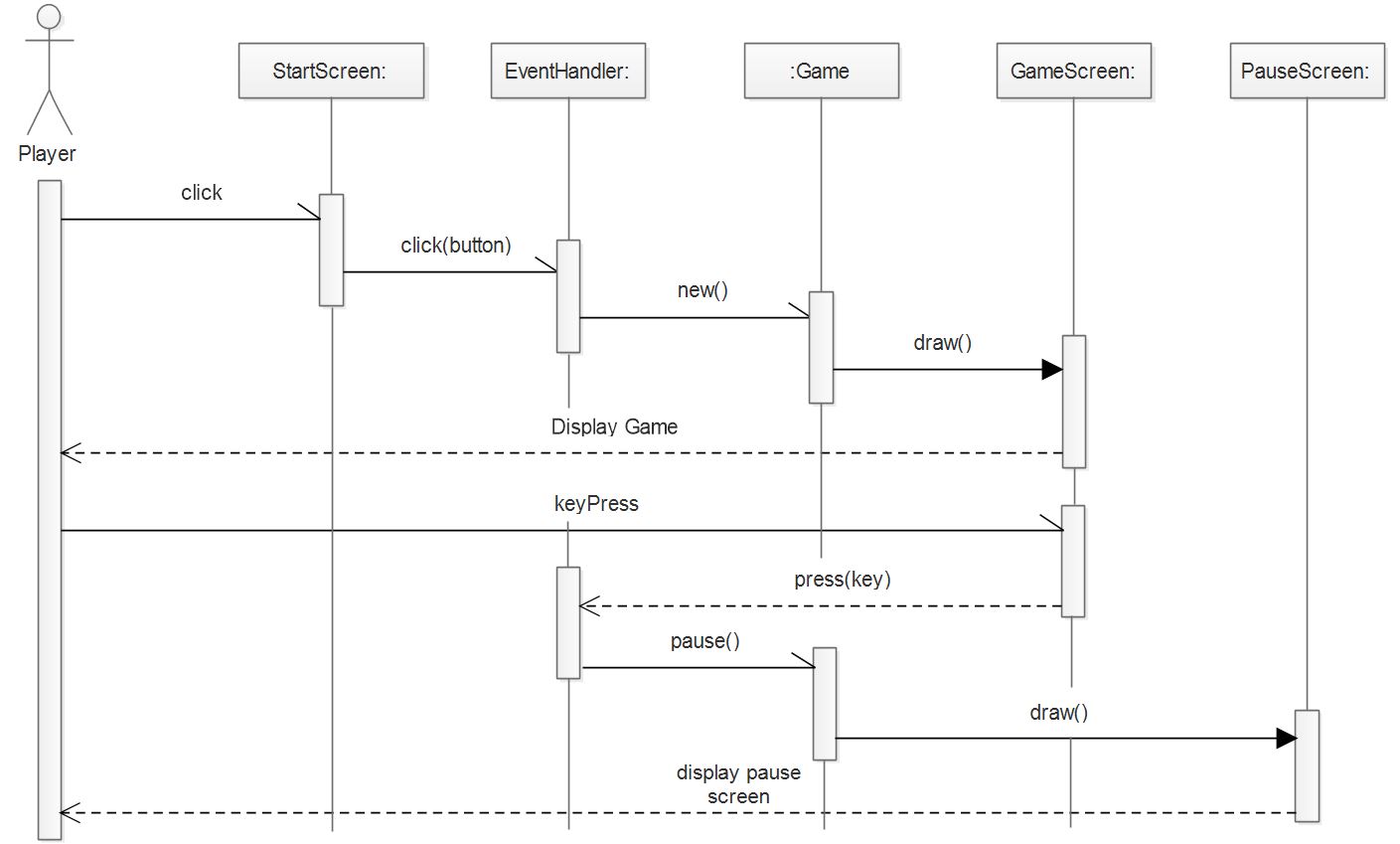


Figure 8: pause game sequence diagram

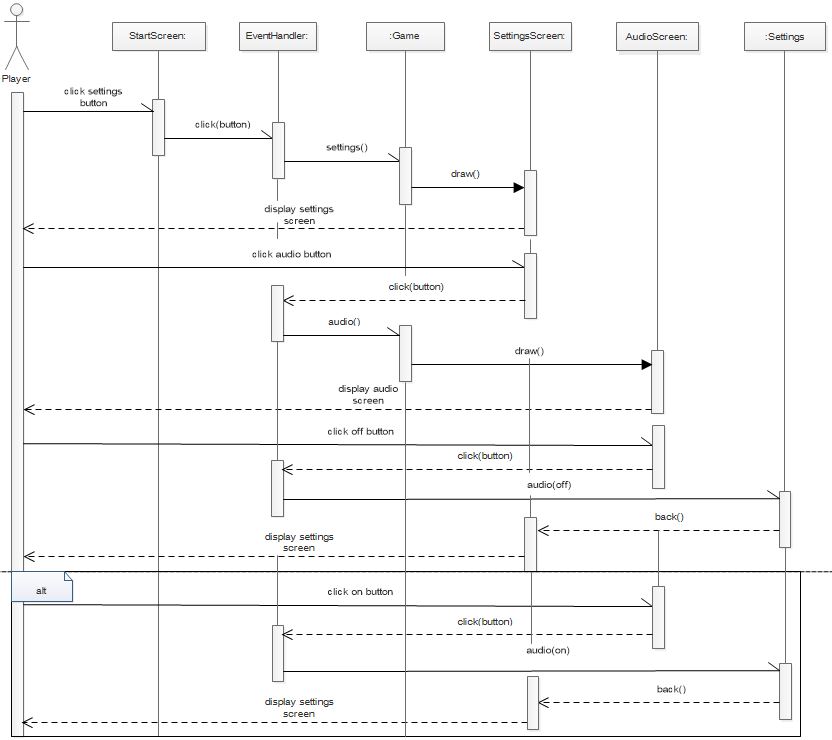


Figure 9: change audio sequence diagram

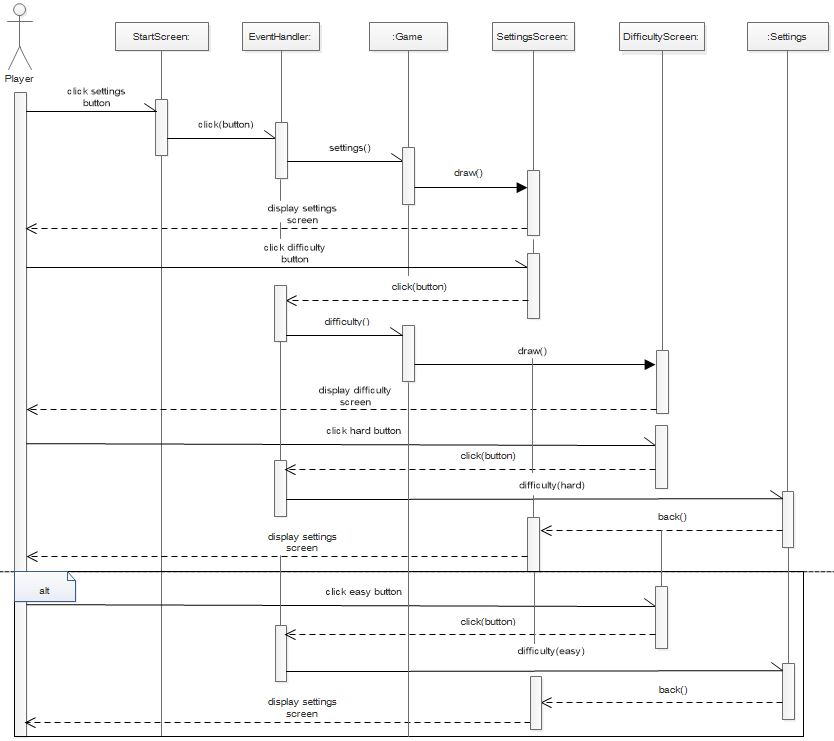


Figure 10: change difficulty sequence diagram

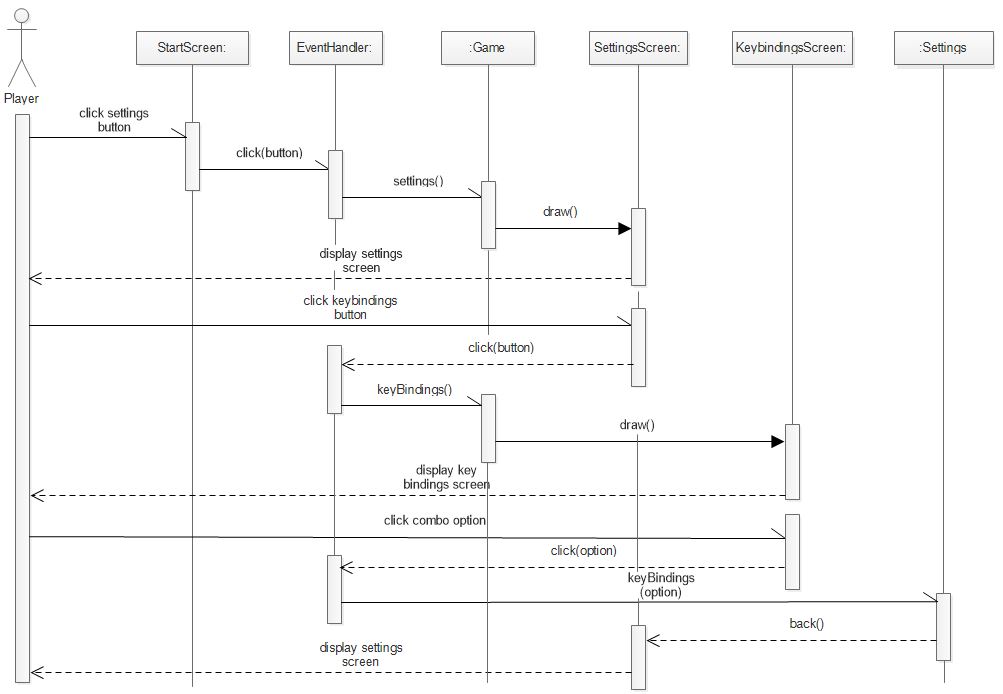


Figure 11: change keybindings sequence diagram

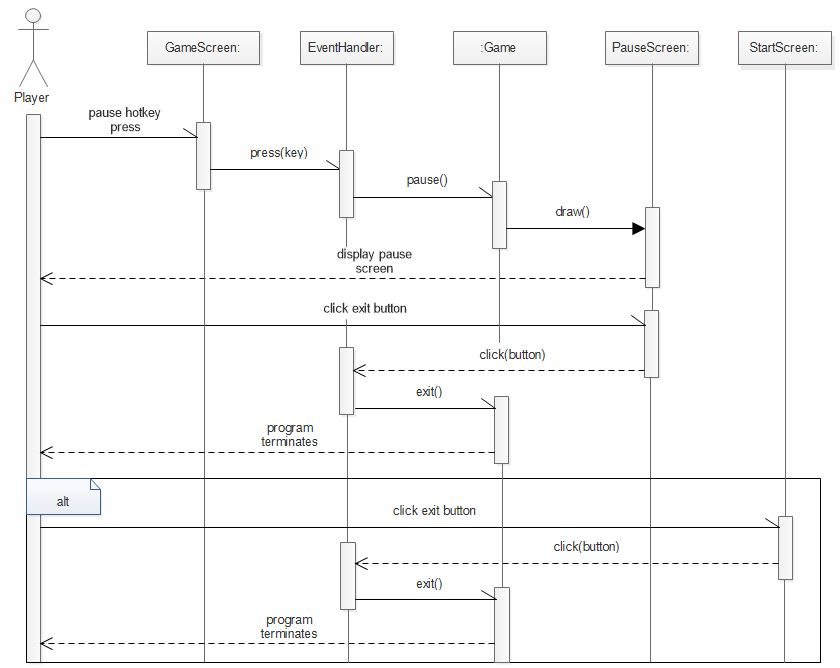


Figure 12: exit game sequence diagram

## 3.2 State chart Diagram

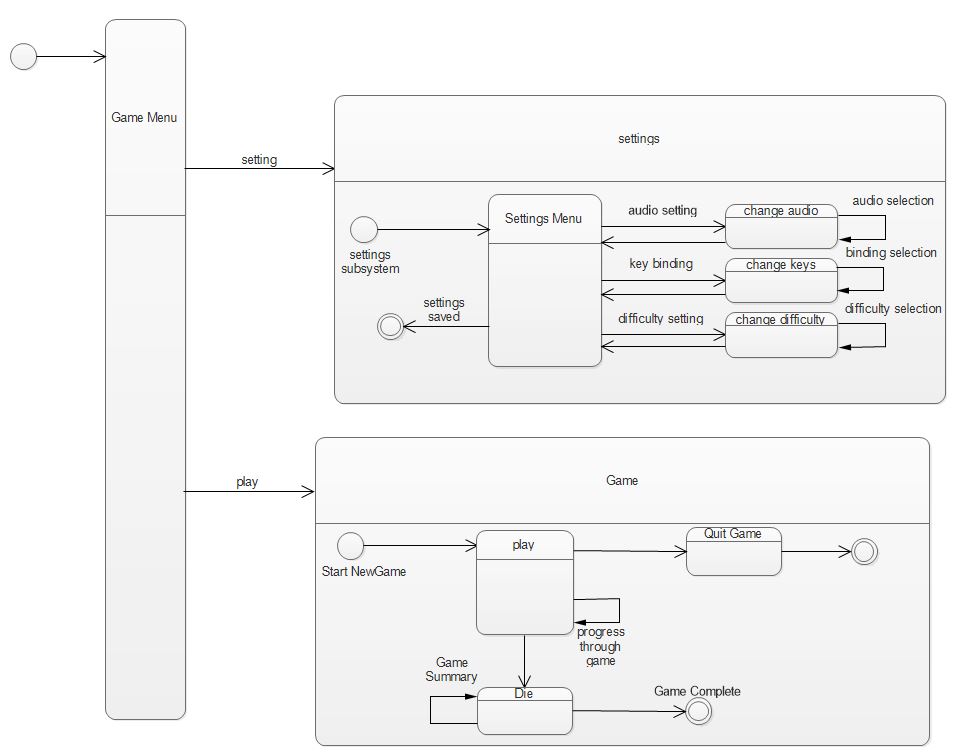


Figure 13: state diagram

# 4. Detailed Design

|  |
| --- |
| **Game** |
| - clock:pygame.time.Clock  - running:boolean |
| - loadData()  - new()  - run()  - update()  + events()  + draw()  + showStartScreen()  + showPauseScreen()  + showSettingsScreen()  + showAudioScreen()  + showDifficultyScreen()  + showKeybindingsScreen()  + showGoSCreen()  + drawText(text, size, color, x, y)  + button(text,x,y,width,height,color) |

Table 1: General description of Game class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| clock | pygame.time.Clock | Private | Clock should be a number with many digits to represent the system time. |
| running | Boolean | Private | Running should be a Boolean with the initial value of True. |

Table 2: Attribute description of Game class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| loadData | Private | void | . | A data should be requested | A data should be loaded |
| new | Private | void | . | The system should be on waiting or pause state | The system should begin the main game |
| run | Private | void | - | The new() method should be invoked | The system should be in a playing loop |
| update | Private | void |  | The run() method should be invoked | The system should update graphics continuously |
| events | Public | void |  | The system should be running | The system should execute the requested user event |
| draw | Public | void |  | The system should be running | The system should draw the specified graphics |
| showStartScreen | Public | void |  | Either the system should be at a first startup or the game playing loop should be aborted | The system should display the start screen |
| showPauseScreen | Public | void |  | The system should be in a game playing loop | The system should display the pause screen |
| showSettingsScreen | Public | void |  | The system should be on waiting loop | The system should display the settings screen |
| showAudioScreen | public | void |  | The system should be in settings screen | The system should display the audio screen |
| ShowDifficultyScreen | Public | void |  | The system should be in settings screen | The system should display the difficulty screen |
| showKeybindingsScreen | Public | void |  | The system should be in settings screen | The system should display the keybindings screen |
| showGoScreen | Public | void |  | The system should be in the game playing loop | The system should display the game over screen |
| drawText | Public | void | text, size,  color, x, y | A text draw should be requested | The system should draw a text on the screen |
| button | Public | void | text, x, y, width, height, color | A button should be requested | The system should draw a button on the screen |

Table 3: Operation description of Game class

|  |
| --- |
| **Player** |
| + vx:int  + vy:int  - img:image  - walking:boolean  - jumping:boolean  - current\_frame:int  - last\_frame:int |
| + update()  + jump()  + animate() |

Table 4: General description of Player class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| vx | Integer | Public | vx <> NULL |
| vy | Integer | Public | vy <> NULL |
| img | Image | Private | Img should have an some image for initial value |
| walking | Boolean | Private | walking should be a Boolean with the initial value of False. |
| jumping | Boolean | Private | jumping should be a Boolean with the initial value of False. |
| current\_frame | Integer | Private | current\_frame should be integer with the initial value of 0 |
| last\_frame | Integer | Private | last\_frame should be integer with the initial value of 0 |

Table 5: Attribute description of Player class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| update | Public | void | . | The run() method should be invoked | The player character should update accordingly |
| jump | Public | void | . | The system should be in a game playing loop | The player character should jump |
| animate | Public | void | - | The player update() method should be invoked | The player character movement should be animated |

Table 6: Operational description of Player class

|  |
| --- |
| **Settings** |
| + width:int  + height:int  + fps:int  + title:String  + highscore\_file:String  + settings\_file: String  + SPRITESHEET:String  + difficulty\_file:String  + AI\_data:String |
| + changeSettings() |

Table 7: General description of Settings class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| width | String | Public | width should be an integer with a specific value |
| height | Integer | Public | height should be an integer with a specific value |
| fps | String | Public | fps should be an integer with a specific value |
| title | String | Public | title should be a string with some value |
| highscore\_file | String | Public | highscore\_file should be a string with some value |
| settings\_file | String | Public | settings\_file should be a string with some value |
| SPRITESHEET | String | Public | SPRITESHEET should be a string with some value |
| difficulty\_file | String | Public | difficulty\_file should be a string with some value |
| AI\_data | String | Public | AI\_data should be a string with some value |

Table 8: Attribute description of Settings class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| chageSettings | Public | void | . | The system should be at settings screen | The system should change the requested setting |

Table 9: Operational description of Settings class

|  |
| --- |
| **Mob** |
| - img:image  - game:Game  - vy:Double  - dy:Double  - walking:boolean  - jumping:boolean  - current\_frame:int  - last\_frame:int |
| + update()  + read() |

Table 10: General description of Mob class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| img | Image | Private | img should have an some image for initial value |
| game | Game | Private | game should be an object of the class Game |
| vy | Double | Private | vy <> NULL |
| dy | Double | Private | dy <> NULL |
| walking | Boolean | Private | walking should be a Boolean with the initial value of False. |
| jumping | Boolean | Private | jumping should be a Boolean with the initial value of False. |
| current\_frame | Integer | Private | current\_frame should be integer with the initial value of 0 |
| last\_frame | Integer | Private | last\_frame should be integer with the initial value of 0 |

Table 11: Attribute description of Mob class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| update | Public | void | . | The run() method should be invoked | The enemy character should update accordingly |
| read | Public | void | . | The system should be in a game session with a player | The enemy bot AI should read from user collected data to be challenging |

Table 12: Operational description of Mob class

|  |
| --- |
| **SpriteSheet** |
| + spritesheet:image |
| + getImage(x, y, width, height):image |

Table 13: General description of SpriteSheet class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| spritesheet | Image | Public | spritesheet should have an some image for initial value |

Table 14: Attribute description of SpriteSheet class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| getImage | Public | Image | x, y, width, height | The system should request for a sprite sheet | The method should provide the system with a sprite sheet |

Table 15: Operational description of SpriteSheet class

|  |
| --- |
| **Platform** |
| - img:image  - rect:image.get\_rect  - game:Game |
|  |

Table 16: General description of Platform class

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Visibility | Invariant |
| img | Image | Private | img should have an some image for initial value |
| rect | Image.get\_rect | Private | rect <>NULL |
| game | Game | Private | game should be an object of the class Game |

Table 17: Attribute description of Platform class

|  |
| --- |
| **AI** |
|  |
| + learn()  - write()  - update() |

Table 18: General description of AI class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation | Visibility | Return type | Argument | Pre-Condition | Post Condition |
| learn | Public | void | . | The system should be in session with a player | The system should learn player moves |
| write | Private | void | . | The system should invoke the learn() method | The system should write information for later use |
| update | Private | void | - | The system should be in session | The system AI should update some information |

Table 19: Operational description of AI class

# References

Smartdraw; <https://www.smartdraw.com/component-diagram/#whatisComponentDiagram>, On May 17 2018

Smartdraw; <https://www.smartdraw.com/uml-diagram/how-to-make-uml-diagrams.htm>, On May 18 2018