

Bilkent University

Department of Computer Engineering

**CS 319 Term Project**

Section 2

Group 2I

A-day-in-Bilkent

Analysis Report

Project Group Members

1-Enes Varol

2-Mahammad Askari Iqbal

3-Asaf Kağan Bezgin

4-Alper Kılıçaslan

5-Imran Hajiyev

Supervisor: Uğur Doğrusöz

**Contents**

1. **Introduction**

**2. Overview**

**2.1 Game Play**

**2.2 Characters**

**2.3 Companions**

**2.4 High score**

**2.5 Shop**

**2.6 Settings**

**2.7 Credits**

**3. Functional Requirements**

**3.1 Play Game**

**3.1.1 Single Player Game**

**3.1.2 Multiplayer Game**

**3.2 High Score**

**3.3 Help**

**3.4 Settings**

**3.5 Credits**

**3.6 Characters**

**3.7 Companions**

**3.8 Shop**

**4. Nonfunctional Requirements**

**4.1 Game Performance**

**4.2 User-Friendly Interface**

**4.3 Extendibility**

**4.4 Responsiveness**

**5. System Models**

**5.1 Use case model**

**5.1.1 Play Game**

**5.1.2 View High Score**

**5.1.3 Select Companion**

**5.1.4 Settings**

**5.1.5 Shop**

**5.1.6 View Credits  
 5.1.7 Select Character**

**5.2 Dynamic models**

**5.2.1 Play Game**

**5.2.2 Select Character**

**5.2.3 Select Companion**

**5.2.4 Credits**

**5.2.5 High Score**

**5.2.3 Settings**

**5.2.4 Shop**

**5.2.5 Activity Diagram**

**5.3 Object and class model**

**5.4 User interface and screen mock ups**

**6. Conclusion**

**7. Improvement summary**

**8. Glossary & references**

**1. Introduction**

A Day In Bilkent is an Arcade Style SHMUP - or Shoot'em Up – game where the player's main mission is to destroy the incoming enemies which are trying to do the same. For to be specific, the game is inspired by the combination of "Asteroids", "EverWing" & "Space Invaders" but in the contrary of these games, A Day In Bilkent has different types of items, powers, etc... and it has a Bilkent Theme as the name refers and enemies, bonuses, extras, etc... are going to be related with Bilkent University. The player is going to experience the campus life in an arcade style shooting game.

**2. Overview**

A Day in Bilkent is an Arcade Style SHMUP game such as Asteroids, EverWing, and Space Invaders. The gameplay is not complicated. Every age of gamers can easily play this game and have some fun. In the game, user plays as one of the Bilkent student, fighting with TAs, assignments, quizzes, and bosses, which are Teachers. Teachers are chosen according to the courses. Player will face lots of enemies and bosses until they die. After they die, player can check their GPA, which is high score, and try it again to get a better GPA.

**2.1 Gameplay**

A Day in Bilkent is 2D shoot'em up game, and player plays as a student. In the game, player and companions always shoot projectiles. Enemies are TAs, assignments, quizzes, teachers. Some of the enemies cannot shoot but if collision happens, they damage the player. Teachers are bosses and they have some attack patterns. Player's companions have also their own attack pattern and they can also damage the enemies. Power ups are available during the game. There are 5 power ups available. 2 of them makes the player disadvantageous. Power ups are shield, double bullet, rage mode, Allnighter, and Mayfest. Shield bonus gives you a temporary shield for the enemy attacks. Double bullet doubles the player's bullet frequency. Rage mode gives the player a temporary attack bonus and movement speed. Allnighter slows the player’s movement speed. Mayfest reduces attack damage, movement speed, and health for a short time. If player dies, game will end. There are also 2 game modes. Player can play single player mode or multiplayer mode. In single player mode, player will choose the game type; survival mode and story mode. In survival mode enemies will spawn infinitely so that player tries to get their score as high as they can. In story mode player will face 8 different bosses, and enjoy the context. In multiplayer mode, 2 players can play the game at the same time using same keyboard. Players can change character before the game in multiplayer mode.

**2.2 Characters**

There are 4 different characters available. Back Bencher has high health, Quick Learner has high speed, High Achiever has high attack damage and Default character where all the stats are average. Player can choose character before the game.

**2.3 Companions**

Companions are player like characters. They can shoot and damage the enemies. Player has 2 companions. Player can change the companions through Select Companion screen. Companions have their own skills that affect the gameplay.

**2.4 Highscore**

After the game ends, player can see the high score, which is GPA in A Day in Bilkent.

**2.5 Shop**

Player can buy some items to enhance their gameplay. Items that sold in the shop are three time use only. Player has to have enough coins to buy items. Available items are MIPS Green Card, Coffee, YemekSepeti, and Cheat Sheet. MIPS Green Card gives player an attack boost. Coffee gives speed. YemekSepeti gives health. Lastly Cheat Sheet increases all the stats.

**2.6 Settings**

Player can change the difficulty and volume, also player can turn on or off the music.

**2.7 Credits**

Player can see the credits through the credits screen.

**3. Functional Requirements**

**3.1 Game**

The player is trapped inside a container / box referred to as the "Class Room". The player spawns in the room and is surrounded by different enemies with different powers. These enemies attack the player and if the attack hits, the player’s life decreases. When life decreases to 0, the player dies and the game ends. The player is also aided with an array of different companions, each having their own special abilities. Collectables are also dropped by killing enemies, and they spawn randomly to give special powers to the player. The game cannot be paused at any point. This features is to eliminate any unfairness caused by some players pausing the game at crucial times to better understand the situation at hand. The game aims to increase reaction time and quick thinking of the player. The player starts a single player game with a Character selected from the Main Screen - Character option - and goes through levels, each representing a Semester of Bilkent University in story mode. The player scores points with number of enemies destroyed and time survived. The main objective is to defeat the boss or the "Professor" at the end to pass the level or "Semester". Objective of this game is to get the highest score or “GPA” possible. Upon losing, the player is brought back to the end game screen. Same goes for the survival mode. Only difference between story mode and survival mode is survival mode is endless. It pushes the limits of the players.

**3.2 High Score**

This section allows for the player to view their top 10 highest scores from the game runs. Each player saves their high scores with their initials and this allows a sense of competition between different players. Not only will this want to make play more to score more using updates, they will want to compete with their peers and excel more in the game. This section can only be found on the main menu.

**3.3 Settings**

The Settings section enables the user to select a wide range of options to customize the game for their own liking. Properties that will be changeable include:

a) Difficulty

b) Sound and Music Volumes

These customizations would allow for a more personal environment to the game and would provide the player ease to play in a way that he so desires. This section can only be operated using the main menu.

**3.4 Credits**

This section is accessed independently from the main menu. It is a shout out to all the people involved to make this game possible.

**3.5 Characters**

This section is a display of all the different Characters that are present for the player to choose from. The name of the character and a small bio of the character is visible for each character that is on display. Along with that, each character has an option to be "equipped" or chosen and that specific character will be used for the game by the player. Characters have certain stats that will be displayed on the bio. This section can only be accessed through the main screen.

**3.6 Companions**

This section is a display of all the different companions. Companions have their own power to help the player. Player can choose only two companions. This section can only be accessed through the main screen. In multiplayer mode, there are no companions.

﻿**4. Non-Functional Requirements**

**4.1 Game Performance**

The game mechanics aim to be fluid. All Game Objects that are movable including the character, companions, enemies, collectable etc's movements will be fluid in motion. They will be produced in a manner that shall conserve the games frame rates hence to reduce lagging of the game to a minimal. Sound as well be dealt with and shall be run on another thread hence to allow for quicker response to the player. The stimulus of sound and sight are crucial to the games success and thus shall be optimized to work effectively on even a Graphically Low end Computer. A smooth running game will entice the user to play more and have not rage quit on losing because of no fault of their own.

**4.2 User-Friendly Interface**

The Interface of the whole game will be simple and intuitive. The player will not have to work on guess work to operate the App. The help section which explains the player the mechanics of the game are coupled with pictures to allow the player to understand the mechanics in a glance. During the game run, the position of player health, score and other details will be positioned in such a manner that it shall not interfere with the main game. Hence, the player will not be distracted from these vital information blocking the visuals of other game objects.

**4.3 Extendibility**

The characters can be used to enhance the game. Characters allow us to improve a new game mode, which is Story mode. Due to the theme chosen for the game, a lot more depth can be given to it in future releases. This depth can be:

a) An enhanced story line

b) Story lines for Other departments of Bilkent

c) Different types of Collectables, Enemies, Companions etc.

d) Increase in weapons for all enemies.

e) More Shooting patterns for bosses.

f) Different powers of Companions and Characters.

With progress of releases and input from players, we can modify the game to add the suggested features and more in future releases while keeping the game mechanics intact. With our design, different game modes can also be introduced like:

a) Multiplayer

b) Time Attacks

c) Survival Mode

d) Story Mode

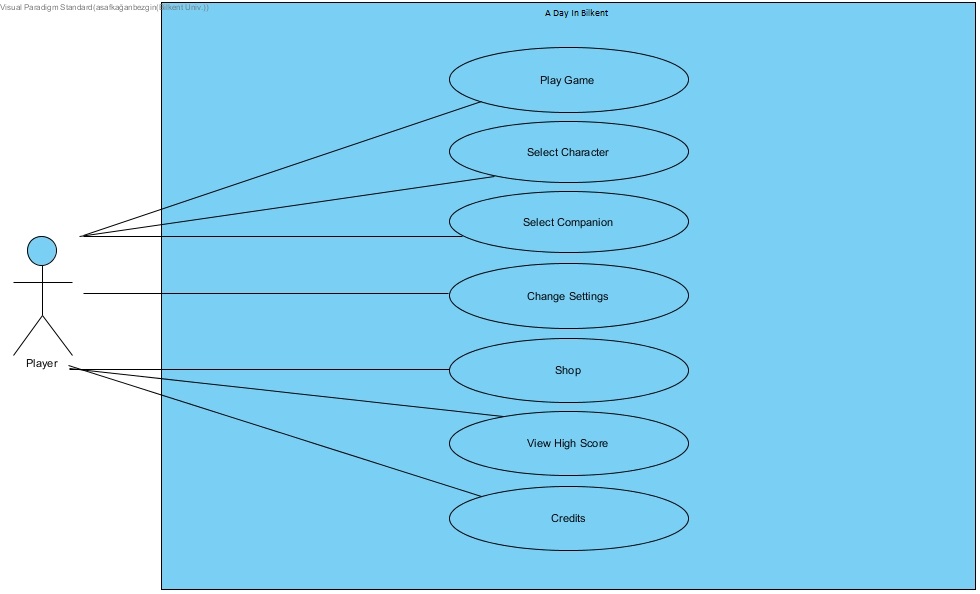
The responses from the playing community shall guide in the future extendibility of the game. All of this is a result of the theme of the game and the design of the game to divide all features as a separate and independent component.

**4.5 Responsiveness**

The game loop that will be designed - the heart of the whole game - will be such that shall maintain a steady frame rate and shall not be unresponsive. It shall be designed in such a manner that during gameplay, the player will not have to face any disadvantages because of a lag. The buttons for the main menu and other sections that have User Input shall be designed in such a way that they will be fluid in operations and not become jittery whilst in use. This serves the purpose to reduce any cause of frustrations for the player so they can have a relaxing experience.

**5. System Models**

**5.1 Use case model**



**Use Case: Play Game**

**Primary Actor**: Player

**Stakeholders and Interests:**

Player wants to play the game.

**Pre-conditions:** Player must be in the main menu.

**Post-conditions:** When the game ends, player will see and "End-Run" pop-up window.

**Entry conditions:** Player selects "Play Game" button from the main menu.

**Exit conditions:**

1. Player presses "esc" button to exit the game.

2. Player dies in the game.

**Success Scenario Event Flow:**

1. Player selects "Play Game" button from the main menu.

2. Player plays the game.

3. Player dies and sees the "End-Run" pop-up window.

**Alternative Event Flow:**

1. If player want to return to main menu:

a. Player presses "esc" button.

**Use Case Name:** View High Score

**Primary Actor:** Player

**Stakeholders and Interests:**

Player may want to check their High Scores.

**Pre-condition:**

Player Should be on the Main screen.

**Post-condition:**

Player returns to the Main screen.

**Entry-condition:**

Player selects "High Score" button from the main menu.

**Exit condition:**

Player Selects the "Go Back" button from the High Score menu.

**Success Scenario Event flow:**

1. Player Selects the "High Score" button from main menu.

2. Player returns back to the main menu.

**Use Case:** Select Companion

**Primary Actor:** Player

**Stakeholders and Interests:**

1. Player wants to choose their Companion.

2. System displays all the Companions and information of their Powers.

3. System allows players to choose their companions.

**Pre-conditions:**

Player must be in the main menu.

**Post-conditions:**

Companion choice updated.

**Entry conditions:**

Player selects "Companion" button from the main menu.

**Exit conditions:**

Player selects "Go back" button from the view Companion screen.

**Success Scenario Event Flow:**

1. Player selects "Companion" button from main screen.

2. Player views the companions and makes a choice.

3. Player exits the companion screen using the "Go back" button.

**Alternative Event flow:**

1. Player does not want to change companions.

a. Player exists using "Go back" button without changing companions.

**Use case name:** Settings

**Primary actor:** Player

**Stakeholders and Interests:**

Player can change the settings of the game:

1. Changing buttons to control the character

2. Changing difficulty of the game to be played

3. Enabling or disabling in game sounds and music.

Game (system) updates the settings which are changed by player.

**Pre-condition:** First time the game is opened settings will be set as default: medium difficulty, sounds and music are on and default controls. If Player changes some of the game settings, those adjusted settings will be saved by the game (system).

**Post-condition:** Updated game settings.

**Entry Condition:** Player clicks on the "Settings" button on the main menu screen.

**Exit Condition:** Player clicks on the "Go Back" arrow-button to return the main menu screen.

**Success Scenario Event Flow:** 1. Player presses "Settings" button on the main menu screen to make the necessary changes in the game settings.

2. Game settings are displayed to the Player in "Settings" screen by the game (system).

3. Player clicks on the "Go Back" arrow-button from the "Settings" screen.

4. Pop-up window pops up and prompts Player with the question whether to save his custom settings or clear everything to default.

5. Player presses button.

6. Game settings are updated by the game (system).

7. Player returns to the main menu screen.

**Alternative Flows:**

1. Player does not want to save changes.

a. Player presses cancel button from pop up window and returns to the main screen.

**Use Case Name:** Shop

**Primary Actor:** Player

**Stakeholders and Interests:**

- Player wants to improve certain attributes by buying certain items.

- System updates player's attributes after the shopping made by player.

**Pre-Conditions:** For the first run of the game, nothing in the shop will be purchased as default.

If the player does shopping during the game, the items bought will be saved.

**Post-Conditions:** Items of the player is updated.

**Entry Condition**: Player selects "Shop" button from the menu.

**Exit Condition:** Player select "Return to Menu" button from menu.

**Success Scenario Event Flow:**

1. Player presses "Shop" button from the Menu to buy item(s).

2. All items are displayed to player.

3. Items which cannot be bought by player because of insufficient (game currency) is indicated.

4. Player does shopping if he/she wants.

5. System updates the changes.

**Alternative Flows:**

A. If Player desires to return to Main Menu at any time:

A.1. Player selects "Return to Main Menu" button from shop to return to Main Menu

A.2. System displays Main Menu.

**Use Case:** View Credits

**Primary Actor:** Player

**Stakeholders and Interests:**

1. Player wants to see the credits

2. System opens the credits and shows it.

**Pre-conditions:** Player must be in the main screen.

**Post-conditions:** When credits end player will go back to main screen automatically.

**Entry-conditions:** Player selects "View Credits" button from main screen.

**Exit conditions:** Player selects the "Go Back" button from the credits screen. Credits end.

**Success Scenario Event Flow:**

1. Player selects "View Credits" button from main menu.

2. Player sees the credits.

3. Credits end.

4. Player returns to main menu.

**Alternative Event Flows:**

1. If Player wants to return to main menu:

a. Player goes back to main menu by clicking the "Go Back" button.

**Use Case:** Select Character

**Primary Actor:** Player

**Stakeholders and Interests:**

1. Player wants to choose their character.

2. System opens the character list and displays them.

3. System allows player to decide their character of choice.

**Pre-conditions:** Player must be in the main menu.

**Post-conditions:** Character choice updated.

**Entry-conditions:** Player selects "Character" button from main menu.

**Exit conditions:** Player selects "Go Back" button from the Character screen.

**Success Scenario Event Flow:**

1. Player selects "Character" button from main menu.

2. Player views all Characters and makes a choice.

3. Player exits the Character screen using the "Go back" button.

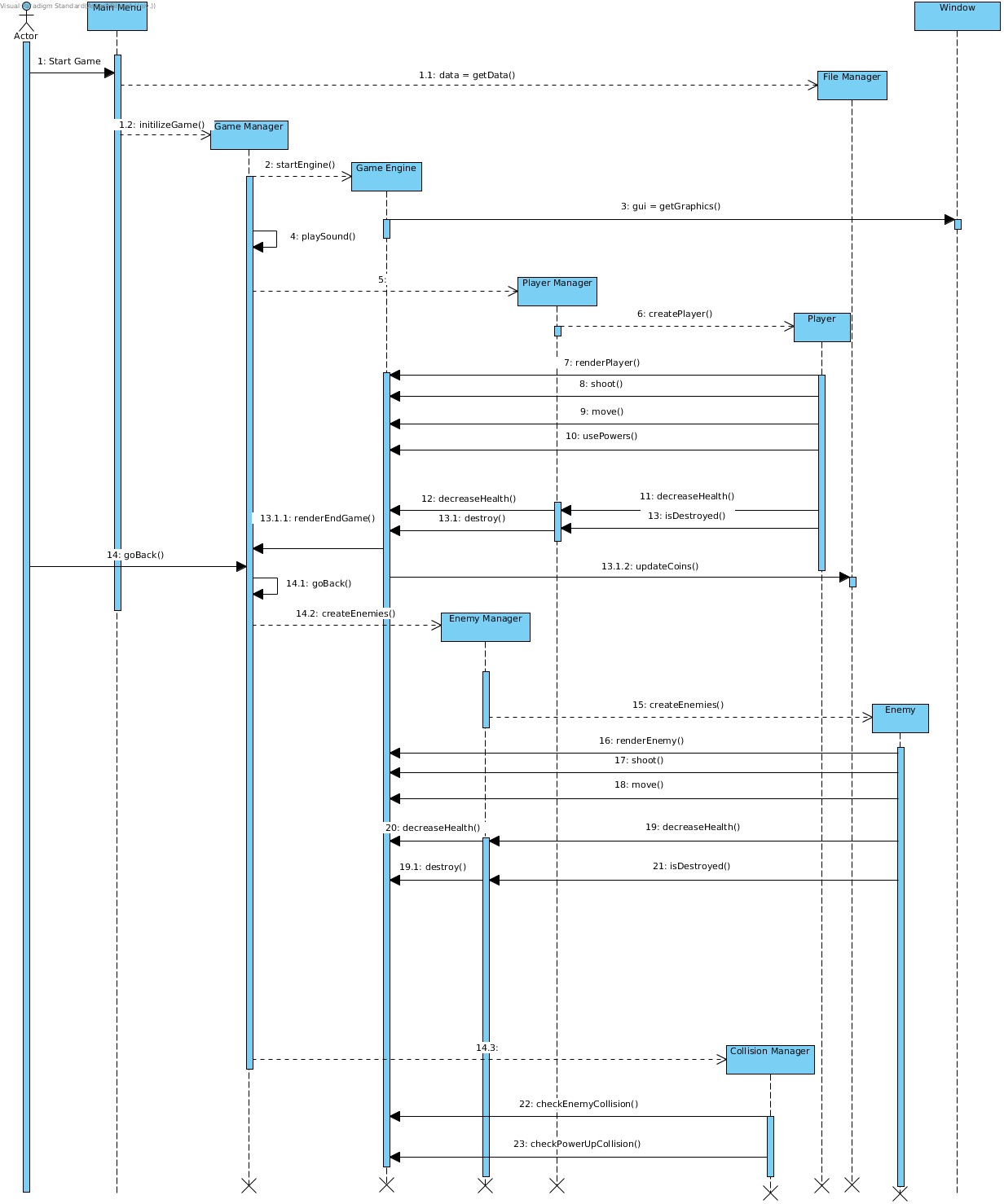
**Alternative Event Flows:**

1. Player does not want to change Characters:

a. Player goes back to main menu by clicking the "Go Back" button without changing characters.

**5.2 Dynamic models**

**5.2.1 Play Game**



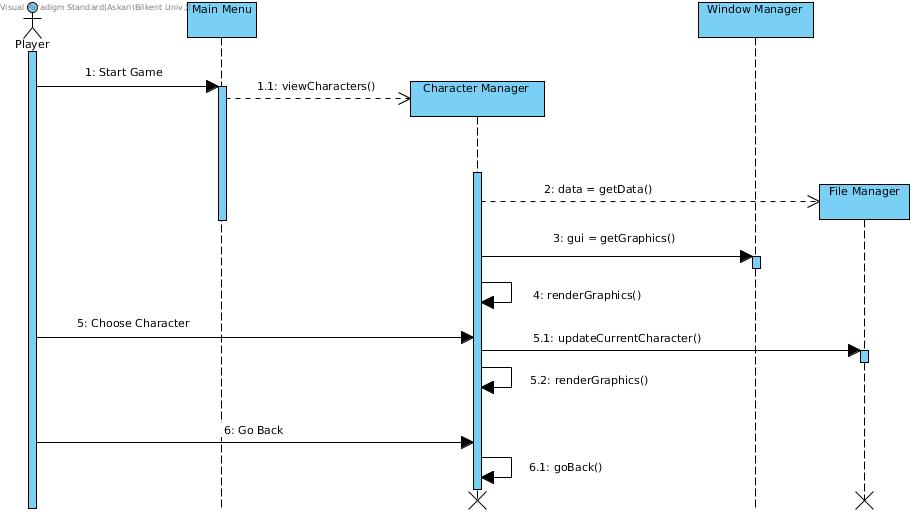
**Scenario:** Player starts the game

Player wants to start the game. Double clicks the desktop icon and main menu opens. Main menu calls file manager to get sound files and start them. Then player clicks the play game button and main menu initialize game manager instance. Game manager creates game engine instance which is the controller of the game play. Game engine calls window to get GUI after that game engine creates player manager instance. Player manager creates player instance player instance sends the player actions to the game engine player manager sends player stats to game engine. Game manager creates enemy manager enemy manager creates enemy instance and enemy instance and enemy manager sends data to the game engine. Game manager creates collision manager instance collision managers sends collision information to the game engine and game engine gathers all this information and manages the gameplay.

**5.2.2 Select Character**

Following sequence diagram illustrates the scenario explained below:

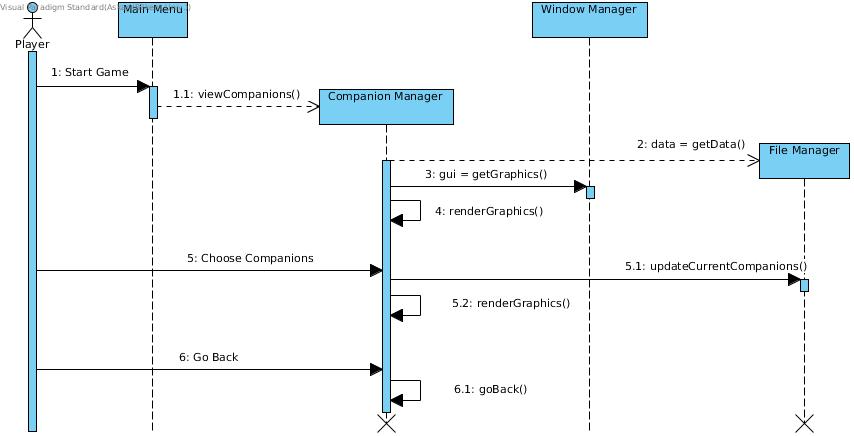
Scenario: Player presses the proper button from main menu in order to enter the select character page. The system initializes a new window where player has different options for changing the character he is using. Every character has different attributes, therefore player might want to change character in order to improve attributes of the character he is playing with during the game. All characters are going to be available to player. When the game starts initially, a default character will be assigned to the player therefore, the player does not need to select a character before he starts playing.



When player is in the select character page, the character manager gets information from file manager which holds the data of current character and draws them on the screen with the help of window manager. If the player changes his/her character, then the update is saved with the help of file manager. When player exist from select character page, main menu is displayed.

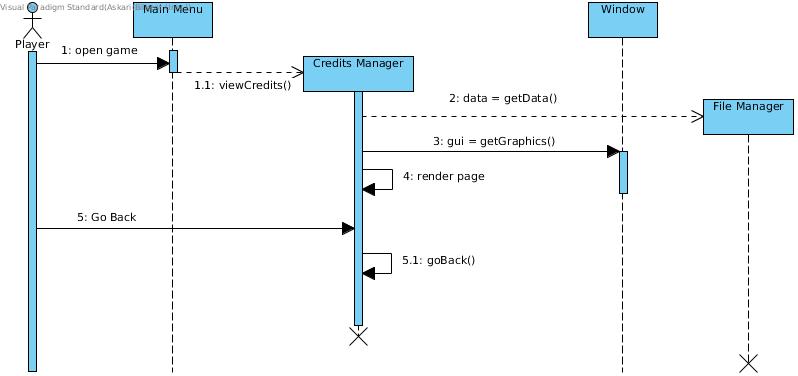
**5.2.3 Select Companion**

Following sequence diagram illustrates the scenario explained below:

Scenario: Player presses the proper button from main menu in order to enter the select companion page. The system initializes a new window and get graphics objects to draw on the screen where player has different options for selecting his/her companions during the game. Player has two companions while playing the game. When player is in the select companion page, he/she can either change one of his companions, change both of the companions or stick with the previous ones and can go back without making any changes. When game starts, system assigns two companions by default.****

When player is in the select companion page, the companion manager gets the information from file manager which holds the data of the current companions and draws them on the screen with the help of window manager. If any change occurs, then the system updates the data and saves it with the help of file manager. When player exits the select companion page, main menu is displayed.

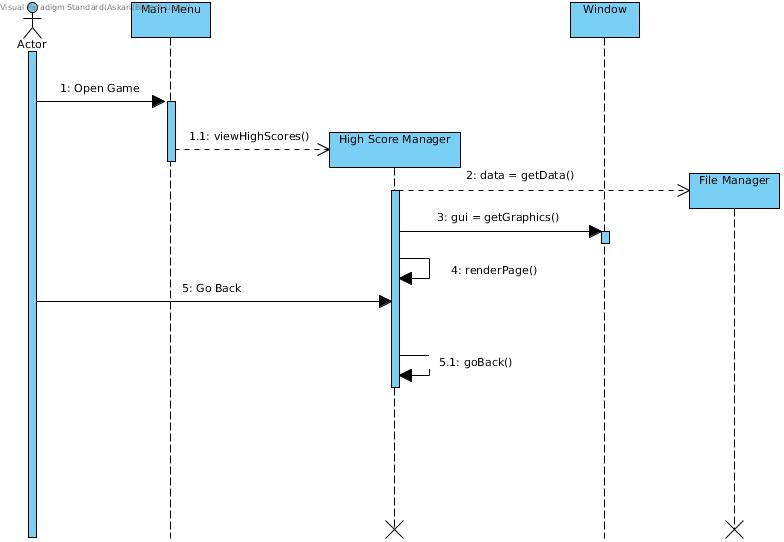
**5.2.4 Credits**



**Scenario:** Player wants to see the credits.

Player wants to see the credits, clicks the credits button. Main menu creates credits manager instance, credits manager instance gets the credits from file manager after getting the credits manager gets gui from window and renders the page until credits ends, when credits ends credits manager terminates the window and returns.

**5.2.5 High Score**



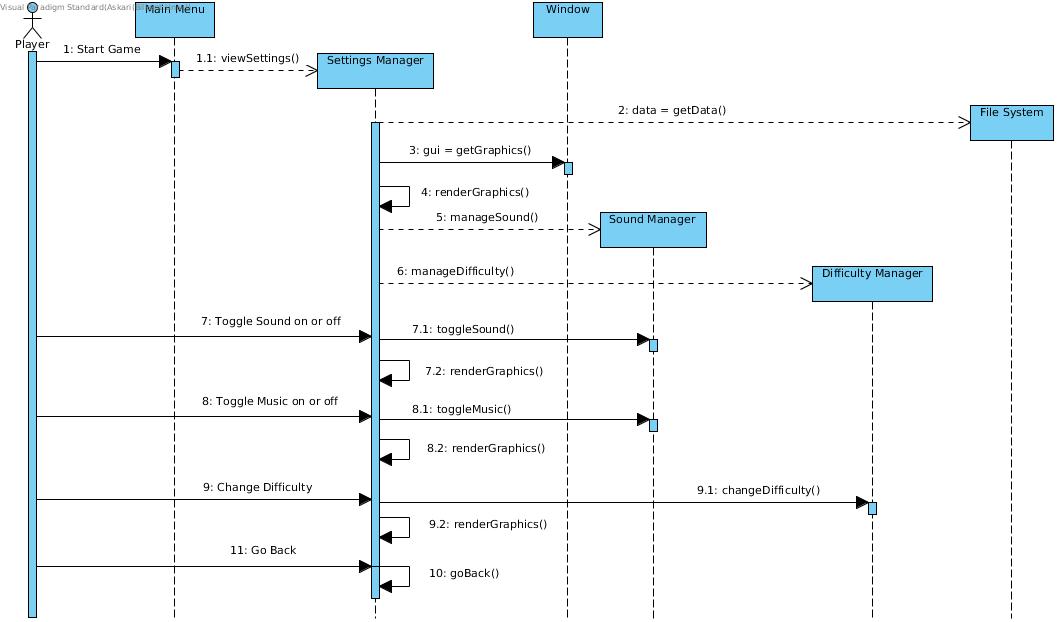
**Scenario**

Player double clicks the game icon in desktop. Then game opens. Player chooses “High Score” on the main menu. Player opens the “High Score” section. If player wants to go back to the main menu, player presses “Go Back” button.

**Description**

Player opens the game and main menu will be loaded. Then player goes to the “High Score” screen. High scores handled by High Score Manager. High Score Manager get the data from the File Manager and it gets graphics from Window, then renders the screen, so that player can see the high scores. If player wants to go back, it also handled by High Score Manager, presses go back button to return to the main menu.

**5.2.5 Settings**

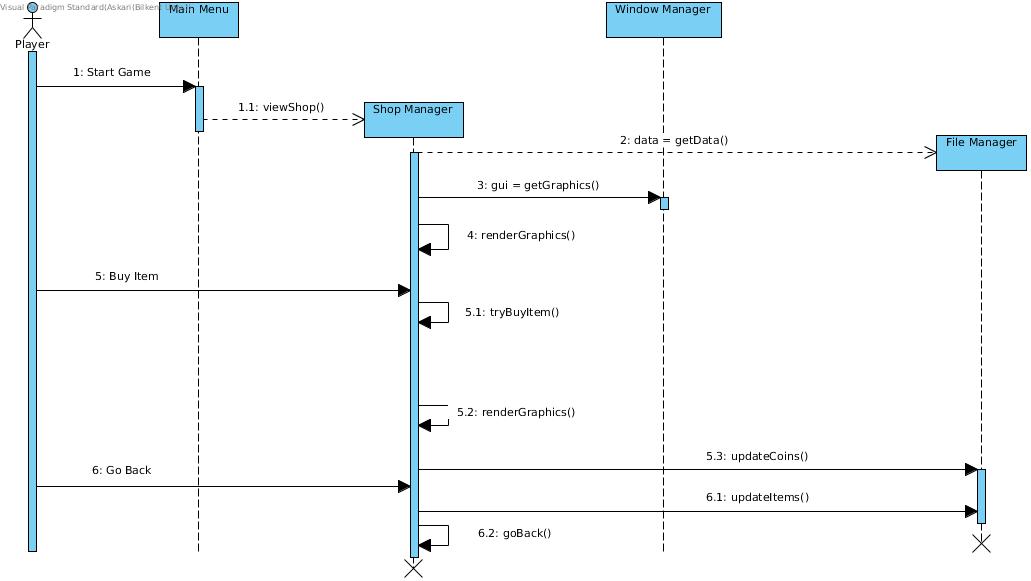


**Scenario:** Player wants to change the settings

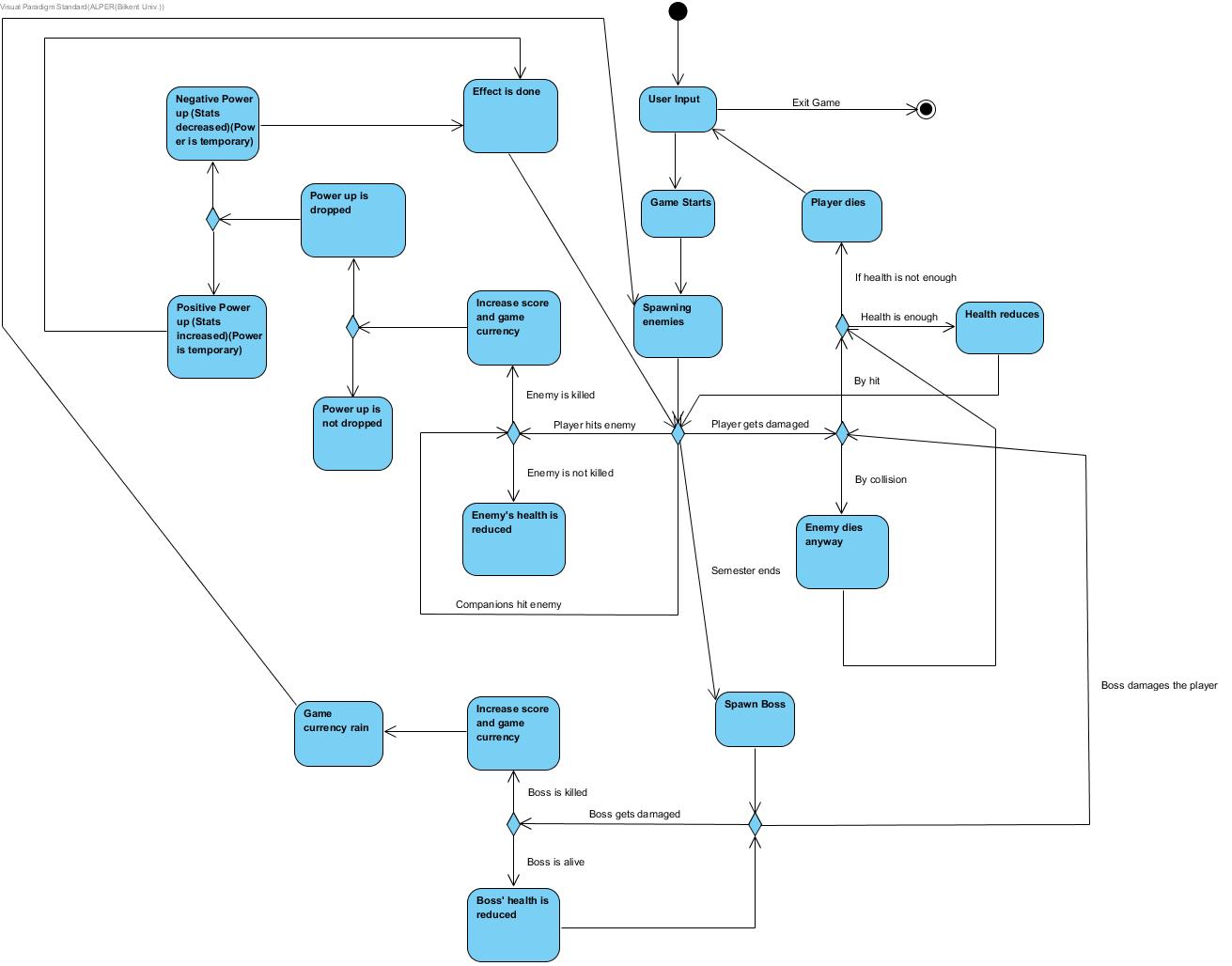
Player wants to change settings, clicks settings button from main menu main menu creates settings manager instance. Settings manager gets data from file system after that gets gui from window and renders the graphics. If player changes the sound level or the music level, then settings manager sends data to sound manager and sound manager updates the file system. If player changes the difficulty level settings manager sends data to difficulty manager and difficulty manager updates the file system. During the all of the changes settings manager renders the graphics.

**5.2.6 Shop**

Following sequence diagram illustrates the scenario explained below:

Scenario: Player presses the proper button from main menu in order to enter shop page. The system initializes a new window and get graphics objects to draw on the screen where player has different options to buy an item for improving the attributes of his/her character during the game if item is equipped. Items in the shop is being bought by the game currency which are obtained during the game by killing enemies or bosses. Initially the player has no items, by playing and gaining game currency, player might decide whether buying an item or not.

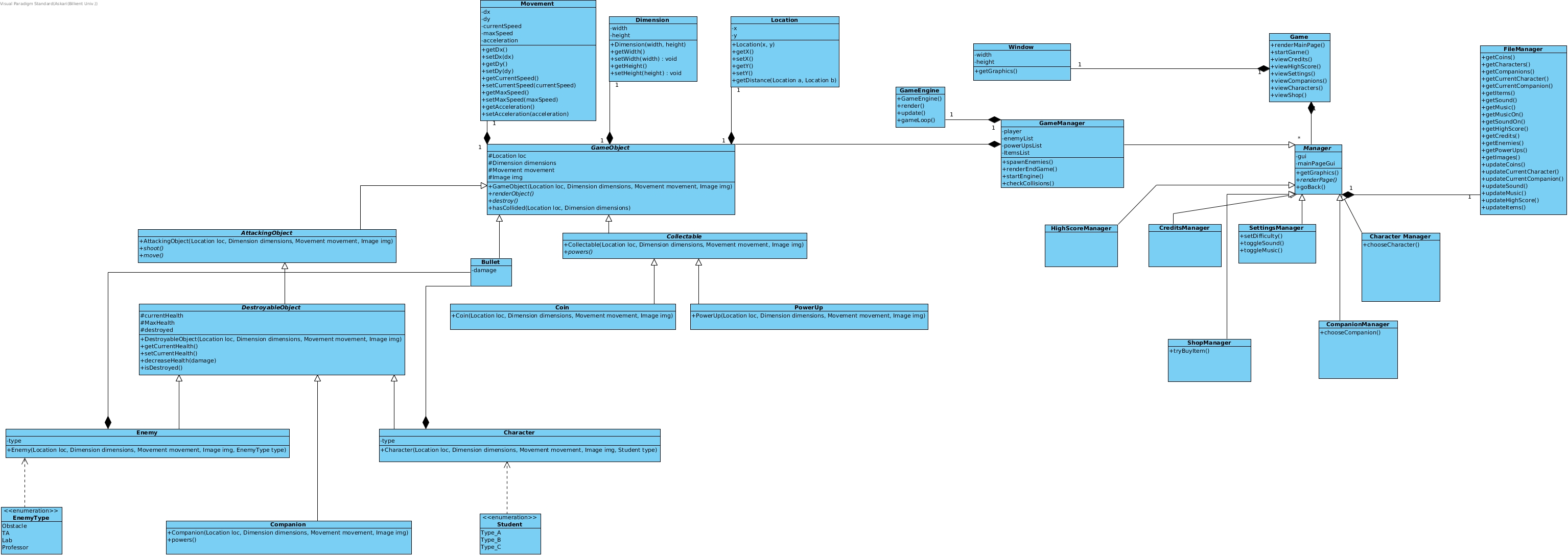
When player is in the shop page, the shop manager gets information from file manager which holds the data of the current items of the player and draws them on the screen with the help of window manager. Game currency is also saved in the file manager therefore, when user tries to buy an item which does not match with the balance of the player, then the system does not allow player to buy that item. Player has also an option to only look for items and go back without buying any, in that case if the player presses go back button, system displays main menu.

**5.2.7 Activity Diagram**

**Description**

Player chooses to play the game. In order to start the game player must press “Play Game” button, where system needs a user input. Then game starts. Enemies start to spawn. User moves with keyboard inputs. When user hits an enemy, system checks whether it is dead or not. If it is dead, game score and in game currency will increase according to the enemy. After killing an enemy, there is a chance to drop a power up. Power up may be a positive one or a negative one. If player gets the power up, timer starts for the effects. If enemy does not die, its health is reduced. If an enemy crashes to the player, player gets damaged and system checks if enemy dies. System also checks player’s current health. If there is not enough health, the game will end and player sees their high score, if there is enough health, player continues to play. After several enemies semester ends, and boss came. If the boss is dead, it drops lots of coins, and player has to collect them. If boss hits the player, system checks it is whether by hit or by collision, then health is updated according to the damage.

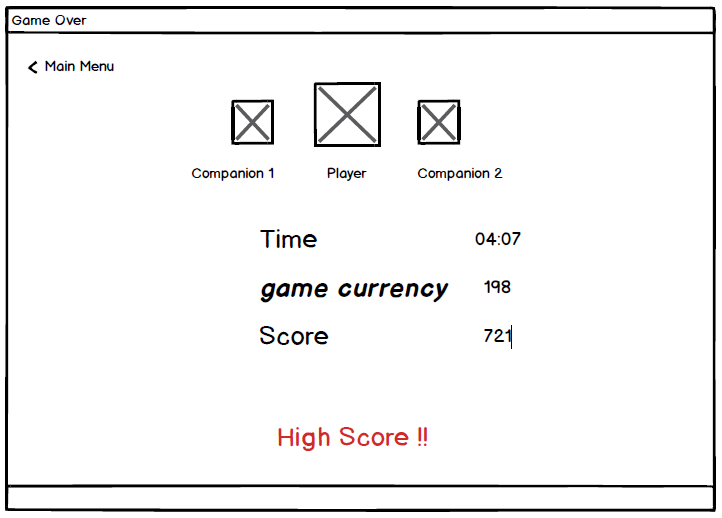
**5.3 Object and Class Model**



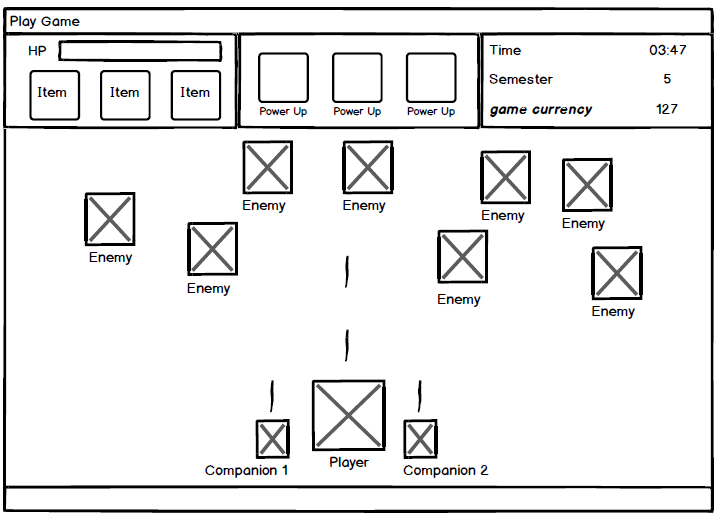
We think that, those classes will be enough for the first iterations. Extra classes will be added according to the project. For now, these classes is sufficient enough to implement the game itself, and we can upgrade the game easily.

**5.4 Screen Mock-ups**

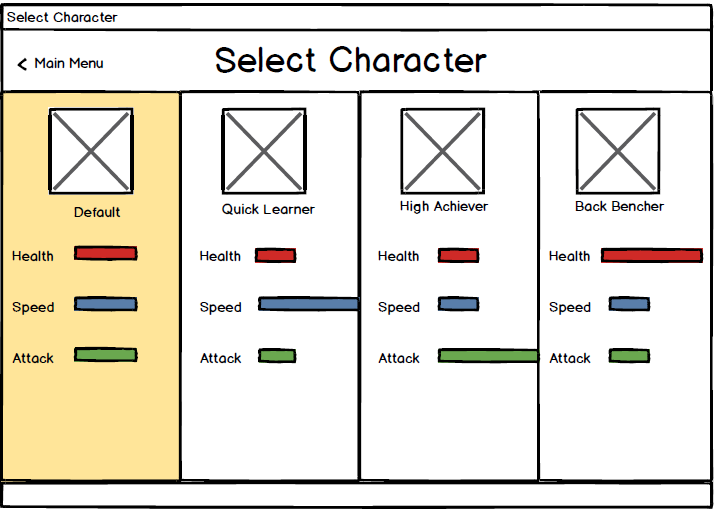
**Game Over**



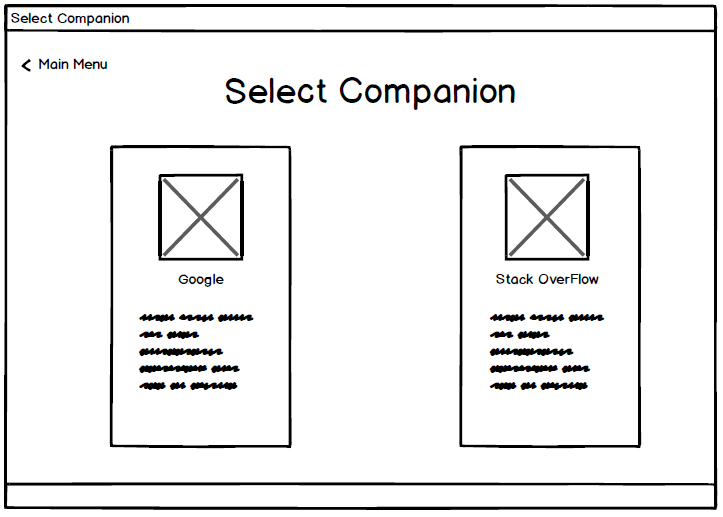
**Play Game**



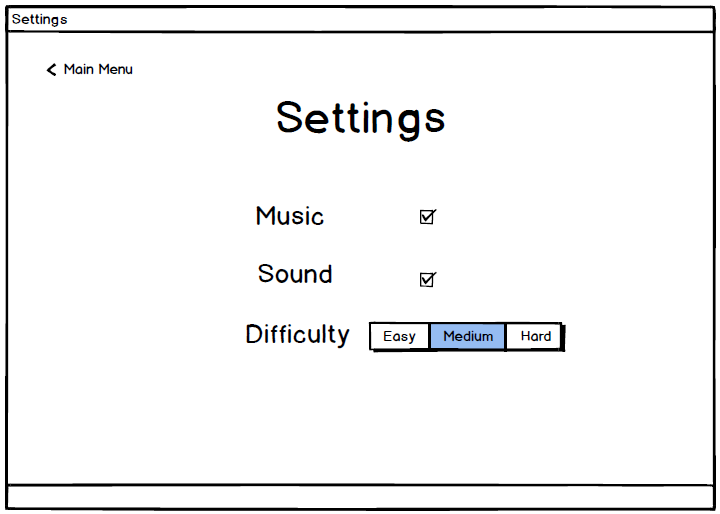
**Select Character**



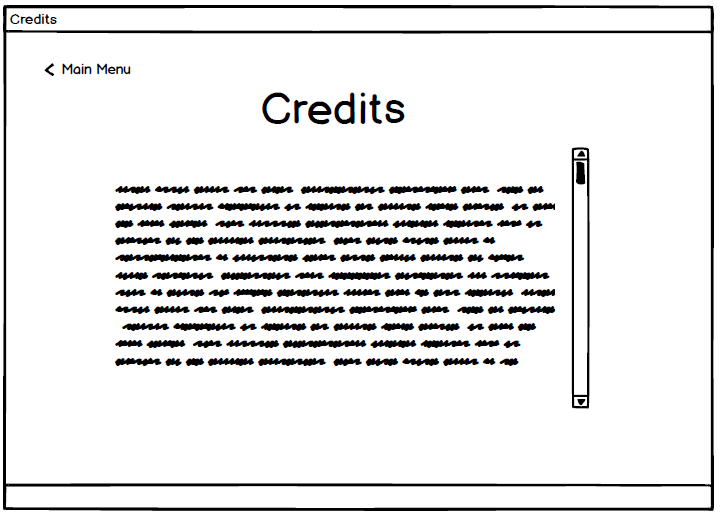
**Select Companion**



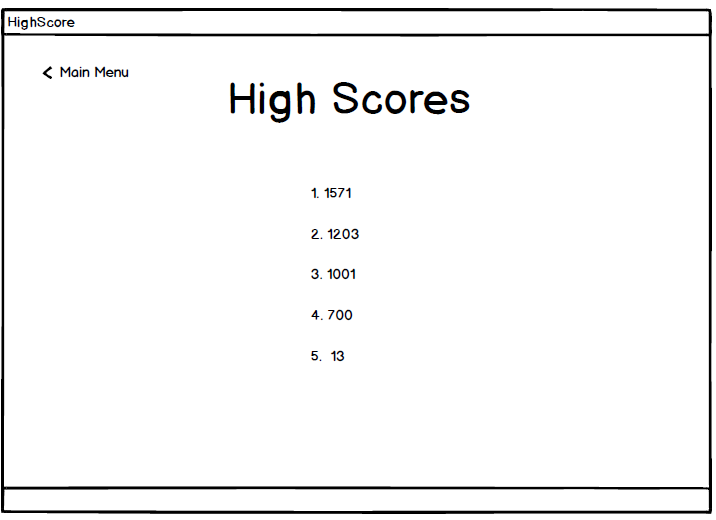
**Settings**



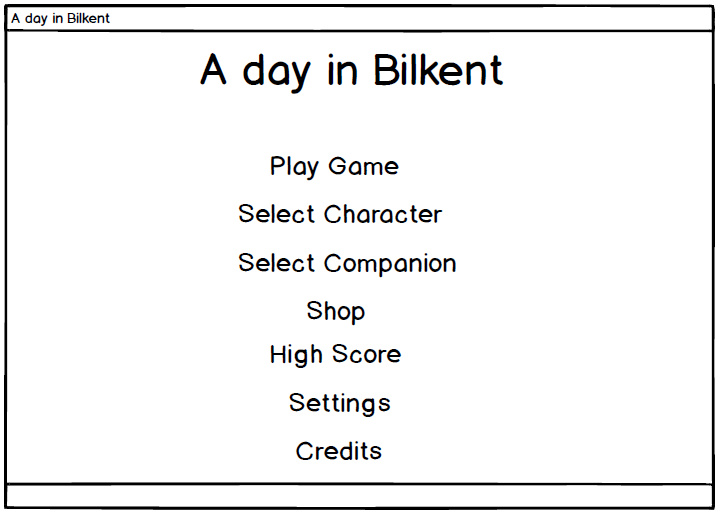
**Credits**



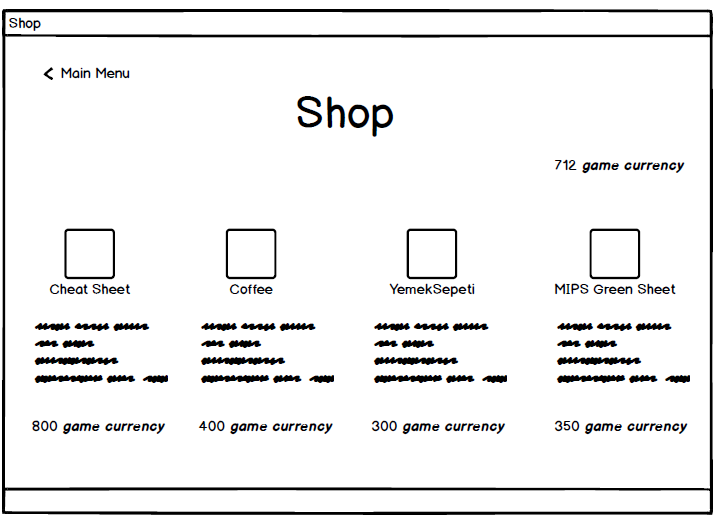
**High Scores**



**Main Menu**



**Shop**



**6. Glossary**

1- Asteroids: <http://www.freeasteroids.org/>

2- EverWing: <http://www.playeverwing.com/>

3- Space Invaders: <http://www.pacxon4u.com/space-invaders/>