CS 319 - Object-Oriented Software Engineering

Project Analysis Report

Feeding Bobby

Group 1-K:

Arda Kıray

Gülce Karaçal

Volkan Sevinç

Okan Şen

Table of Contents

Table of Figures

1. Introduction

Feeding Bobby is a progressive single-player game for computers in which the user will start off as a little fish. Throughout the span of the game, the main goal of the player will be to get bigger by eating smaller fish, to survive by avoiding from bigger fish and to defeat the final boss while facing some mid game bosses.

Feeding Bobby will provide an environment in which the user will encounter some power ups such as freeze time, speed boost, vitality and hostile fishing rods which will immediately kill the character. In addition to these, we will allow the player to alter its physical appearance by choosing from various skin options.

The game progresses by finishing levels on the world map. As the game progresses, it gets harder as well by sending more threats in the player's way. The player will build up scores by evolving and eating. The final score of each chapter will be converted and added to the user's in-game currency that can be used to buy various skin options.

Feeding Bobby is a local game on a Windows desktop PC. The player won't be interacting with the game other than controlling the mouse. The character will pursue its path according to the way the player uses the mouse.

2. Current System

Not available.

3. Proposed System

3.1 Overview

The game consists of many levels spread out on a world map. Each of these maps will have different tasks to succeed but basically the main task is not to be eaten while eating other fishes. With each fish eaten our score will increase and on specific score limits player's fish, Bobby, will get bigger in size.

By getting bigger, the character will evolve into bigger species. In each level, there will be two types of fish except the player whose sizes are respectively smaller and bigger than Bobby. As Bobby grows bigger by eating smaller fish, its size will increase enough to eat the bigger fish and then the player will be faced with even bigger fish. The growth rate of Bobby will depend on the type and the number of fish eaten.

Each level will have different tasks to accomplish in order to finish them. Some will want the user to reach a score and some will want the user to eat a specific fish. These tasks will get harder to accomplish with the increasing progression of the player.

In each level, the user will start by eating smaller fish and build up points to reach the next level of growth which is evolving in the game. For example, when the player eats enough fish and reaches 500 points Bobby will evolve and grow in size. This will give an extra life to the player as well.

There is a combo system which will let the player grow faster with each successive eaten fish by multiplying the points taken but the combo will be over once the player takes damage. The game also encourages to keep the combo on and evolve as many times as possible by giving points for evolving too and this evolving score will be multiplied higher by the combo number. Additionally, there will be a frenzy bar which will fill as the player eats fish. It will only increase by eating fish and will only be decreased by enabling the frenzy mode, taking damage will not affect this bar. Frenzy mode will activate once its bar is full and it'll decrease till it is empty. During the frenzy mode Bobby can eat any fish whether it is bigger, smaller or special fish that would normally kill Bobby. Anything eaten will give points.

Bobby will die if he is bit by any bigger fish in any level's beginning until he evolves by eating enough fish. Then Bobby will be able to be bit twice, first one decreasing his size to the former size and second one killing him as if in the beginning of the game. To make things easier for the user after being bit, Bobby will be invulnerable for a few seconds to get back to a safe spot on the screen and continue eating smaller fish.

After when a level is finished, whether by accomplishing the tasks or by being eaten, the final score will be added to the high scores table if it is high enough. There will be power ups, power downs, random events and special fishes in the levels.

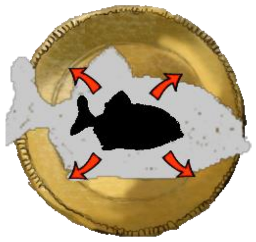
3.1.1 Power-ups

Figure 1: Speed Boost

- **Speed Boost:** This power up will slow all the fish on the screen and any fish that will come on the screen except Bobby, for a short duration.

Figure 2: Freeze Time

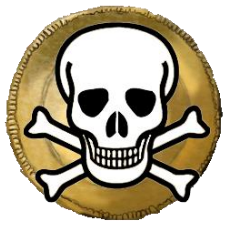
- **Freeze Time:** This power up will freeze everything on the screen except Bobby.



-

- Figure 3: Vitality/Health Boost

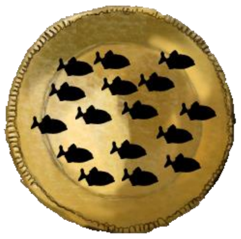
- **Vitality/health Boost:** This will grow Booby once in size and let him evolve.



-

- Figure 4: Kill Bigger Fish

- **Kill Bigger Fish:** As the name suggests this power up will diminish all the bigger fish on the screen immediately.



-

- Figure 5: Little Fish Swarm

- **Little Fish Swarm:** This power up will spawn and send a little fish swarm on the screen, possibly helping Bobby to have a boost on score and combo.

3.1.2 Power Downs

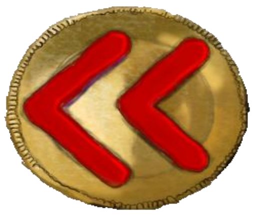


Figure 6: Slow

. **Slow:** This will make all the other fish on the screen and any other fish coming in screen faster except Bobby.

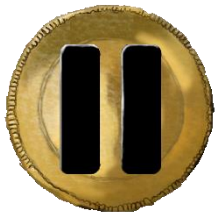


Figure 7: Stun

- **Stun:** This will stun Bobby for a short duration making him vulnerable for any incoming fish or random events.

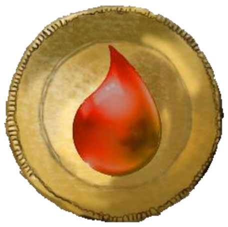


Figure 8: Health Decrease

- **Health Decrease:** This will immediately degrow Bobby and decrease his size once (de-evolve). This will not be available during first state of evolving of Bobby.

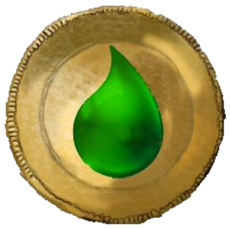


Figure 9: Poison

- **Poison:** This will make Bobby sick in the stomach and he will not be able to eat any other fish for a short duration.

3.1.3 Insta-Killers

- **Fishing Rod:** A fishing rod will come down from the top of the screen on a random side. This rod will stay there for a short amount of time and will pull back. If Bobby is close enough to be pulled back too, he will die instantly.
- **Sea Mines:** These mines will be deeper in the ocean so in the first levels they will not be very effective. If Bobby hits one of these mines, he will instantly die.

3.1.4 Random Events

- **Sea Currents:** In some levels sea currents will be available, randomly happening, it will push Bobby in its direction. Swimming against this current will be slower while going with it will be a lot faster than normal speed.
- **Geysers:** Hot geysers will randomly spawn at the bottom of the screen and will channel for a few seconds, exploding a hot stream of water vertically in its wake. If Bobby gets hit by this stream of hot water he dies.
- **Tornado:** With a low chance, a tornado will spawn and the game mode will change. The tornado will stay locked on the left side of the screen and the screen will start moving to the right in a sidescrolling manner. Bobby will have to evade all the incoming obstacles and threats while staying away from the tornado. If Bobby gets caught by the tornado he will die.

3.1.5 Special Fishes

A) Friendly

- **Astronaut Fish:** Eating this will grow Bobby once in size by evolving him. Astronaut fish has an oxygen tank strapped on its back and Bobby loves oxygen!
- **Enraged Fish:** This fish will increase Bobby’s rage and add boost to the frenzy bar.

B) Hostile

- **Pufferfish:** Eating this fish will cause Bobby’s death. The pufferfish will explode in Bobby’s stomach after a few seconds.
- **Drunk Fish:** Eating this fish will convert all the controls.
- **Jelly Fish:** Staying around these jelly fish will electrocute Bobby causing him to degrow if electrocuted for a few seconds.

3.2 Functional Requirements

* The user will be able to access the help menu that consists of information about how to play the game and descriptions of special power-ups.
* The player will be able to have some power-ups according to the type and the number of fish eaten.
* The player should be able to pause the game whenever he/she wants.
* The system should pause the game and should stop all movements when the pause button is pressed.
* The system saves the game after the player completes each level.

* The player should be able to load the game whenever he/she wants.
* The system should be able to show credits.
* The user can toggle music on and off.
* The player should be able to enter his/her name at the beginning of the game.
* The user should be allowed to play single player.
* The game should be controlled by the movements of mouse.

3.3 Non-Functional Requirements

* Control system of the game should be easy to understand and should feel natural.
* Concepts of the game should be easy to understand and react to.
* Game should not contain any game-breaking or progress preventing bugs.
* Load times should be minimal.
* Visuals such as menus should be easy to understand.
* Controls should contain the minimum delay possible.
* Concept art such as models and backgrounds should not harm or tire the player.

3.4 Constraints

* The game will be implemented in Java.
* The game will run on Windows desktop PC.
* Adobe Photoshop will be used for various designs such as fish and backgrounds.
* FL Studio will be used for custom music and sound effects.
* Player must have pre-installed Java 8 or higher versions in his computer

3.5 System Model

3.5.1 Use Case Scenarios

3.5.1.1 Start a New Game

Use Case Name: Start a New Game
Primary Actor: Player
Entry Condition: Player clicks “Start a New Game” button from Main Menu
Exit Condition: Player selects “Return to Main Menu” option from the game screen

Event Flow:
- A tutorial page displays onto the screen which contains all necessary information to prepare user for his first experience

- The main map displays on to the screen which contains a journey that consists of several chapters. Initially only the first chapter of the journey is available for the user and sequentially each next chapter will be opened after completing the previous one.

- Player selects the first chapter and game starts

- Player aims to eat smaller fishes to get bigger and complete the growth amount of the first chapter while simultaneously trying to survive.

- System keeps the growth amount of the user's character on the screen as "Growth bar"

- System keeps the score of the user

- System automatically saves the current progress after each chapter

Pre-condition: For the first use, game settings are set as default. If user changes any of game settings, adjusted settings will be used throughout the journey until the user makes another change.

Post-condition: If the score of the user is eligible to record on high-score table, high- score table will be updated by the system.

Success Scenario Event Flow:
1. Game is started by the system
2. Player starts a new game
3. A tutorial screen displays
4. Player selects the first chapter of the game
5. Game starts and player plays until according growth amount is satisfied
6. Player finishes the current chapter and the map of journey displays

Player repeats finishing consecutive chapters until the final chapter is complete or player dies and return to the journey of map.

7. System records players gathered points to his in game currency and high-score table if the resulting score is eligible for high score table

Alternative Flows:

Player will face with various power-ups and threats during the game and encountering one of them can change the flow of events as:

1. Player collects a special power-up

2. System does the necessary changes for applying the power-up

3. The power-up vanishes due to time out or death of the character

3.5.1.2 Pause Game

Use Case Name: Pause Game
Primary Actor: Player
Entry Condition: Player selects the pause icon from the game screen
Exit Condition: Player re-selects the pause icon from the game screen
Event Flow:
- Player pushes the pause icon from the game screen

- System stops the game

- Player pushes the pause icon from the game screen

- System unfreezes the game

Pre-condition: User must be on the game screen

Post-condition: -

Success Scenario Event Flow:
1. While in game player selects the stop game icon
2. System freezes the game until player selects to unfreeze the game by using the same icon

3.5.1.3 Load Game

Use Case Name: Load Game
Primary Actor: Player
Entry Condition: Player selects the “Load Game” button from the Main Menu
Exit Condition: Player selects the “Return to Main Menu” button from the game screen

Event Flow:
- Player selects the “Load Game” option from the Main Menu

- A list of auto-saved progresses displays on the screen

- Player chooses the desired one

- System loads the chosen saved progress’ game screen on the display

Pre-condition: There must be at least one saved game for loading that progress

Post-condition: The achieved progress will be over-written on to the same save file after loading game

Unsuccessful Scenario Event Flow:

1. Player selects the “Load Game” option from the Main Menu

2. A list of auto-saved progresses displays on the screen

3. Player chooses one to load

4. Due to corruption of saved data, game cannot be loaded and related error message displays on the screen

3.5.1.4 Change Settings

Use Case Name: Change Settings
Primary Actor: Player
Entry Condition: Player selects the “Change Settings” button from Main Menu
Exit Condition: Player selects the “Return to Main Menu” button from Change Settings display
Event Flow:
- Player selects the “Change Settings” button from Main Menu

- A list of game settings displays on the screen

- Player adjusts desired settings from the list by filling the according boxes

- Player selects the “Return to Main Menu” button from Game Settings screen

- System automatically saves the preferred settings before returning to the Main Menu

Pre-condition: For the first use, game settings are set as default. If user changes any of game settings, adjusted settings will be used throughout the journey until the user makes another change.

Post-condition: Adjusted game settings are updated.

Success Scenario Event Flow:
1. Player selects the “Change Settings” options from Main Menu

2. A list of game settings displays on the screen

3. Player fills or unfills the desired option’s check box

4. Player clicks the “Return to Main Menu” button

5. System automatically saves the changed settings

6. Main Menu screen displays

3.5.1.5 View Help

Use Case Name: Help
Primary Actor: Player
Entry Condition: Player clicks the question mark symbol at the bottom right corner of the Main Menu
Exit Condition: Player selects the "Return to Main Menu" button from the help screen

Event Flow:
- A tutorial screen consists of all necessary instructions to play the game displays
Pre-condition: -
Post-condition: -
Success Scenario Event Flow:
1. Player clicks the question mark symbol at the bottom-right corner of the Main Menu

2. A tutorial screen consists of all necessary instructions to be able to play the game displays

3. Player clicks the "Return to Main Menu" button

4. System automatically saves the changed settings permanently until the events 1-4 occurs again

3.5.1.6 View High Scores

Use Case Name: High Scores
Primary Actor: Player
Entry Condition: Player selects the "High Score" button from Main Menu
Exit Condition: Player selects the "Return to Main Menu" option from high scores table page
Event Flow:
- A list of high scores table corresponds to each saved players displays on the screen
Pre-condition: -
Post-condition: -

3.5.1.7 Exit Game

Use Case Name: Exit Game
Primary Actor: Player
Entry Condition: Player selects the "Exit Game" button from Main Menu
Exit Condition: -

Event Flow:
- System terminates itself

3.5.2 Use Case Model

Below is our Use Case Diagram and Use Cases as described in *Section 3.5: Use Case Scenarios.* Our primary intention was to keep the diagram as comprehensible as possible for the client.

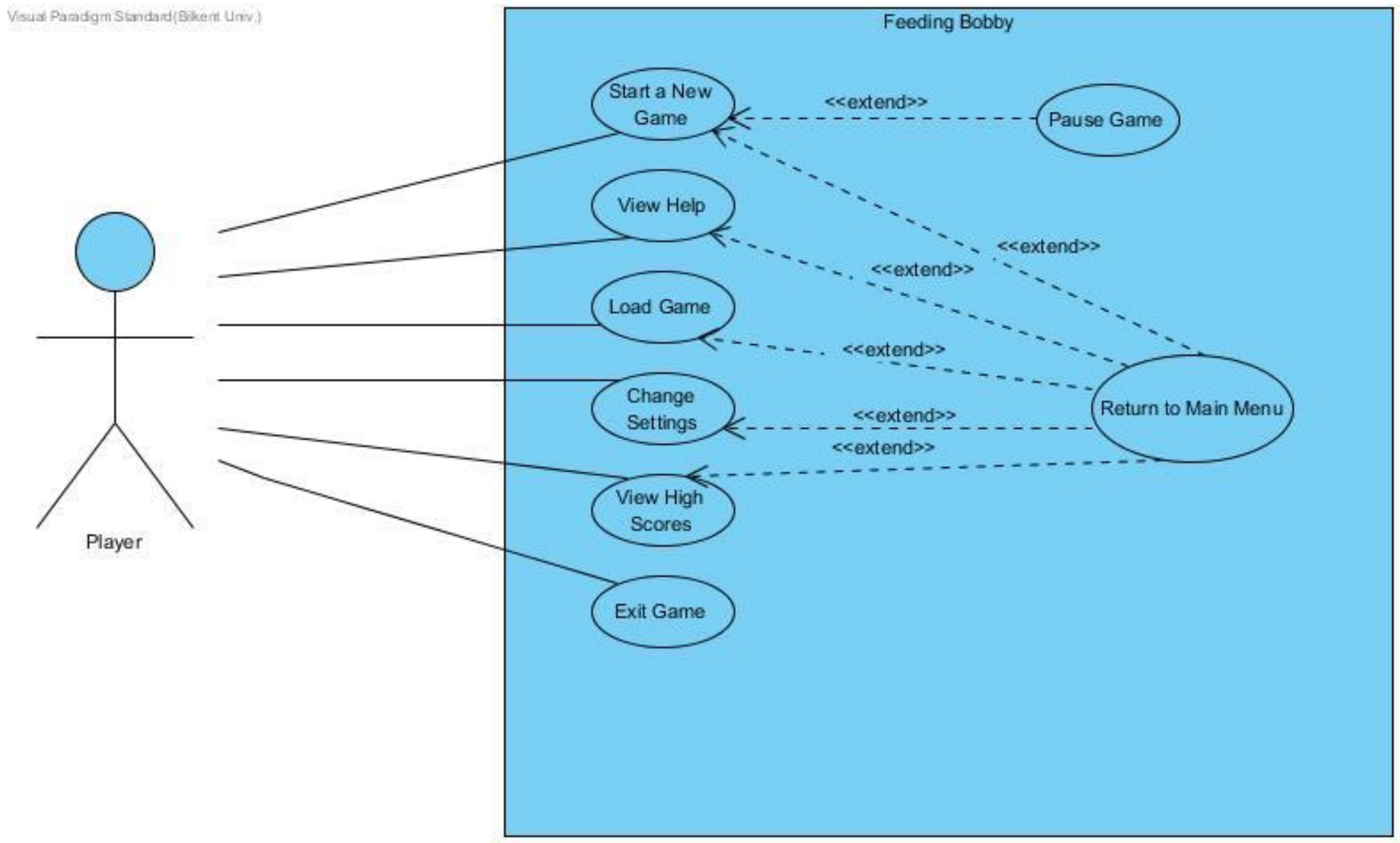


Figure 10: Use Case Model

3.5.3 Object Model

3.5.3.1 Class Diagrams

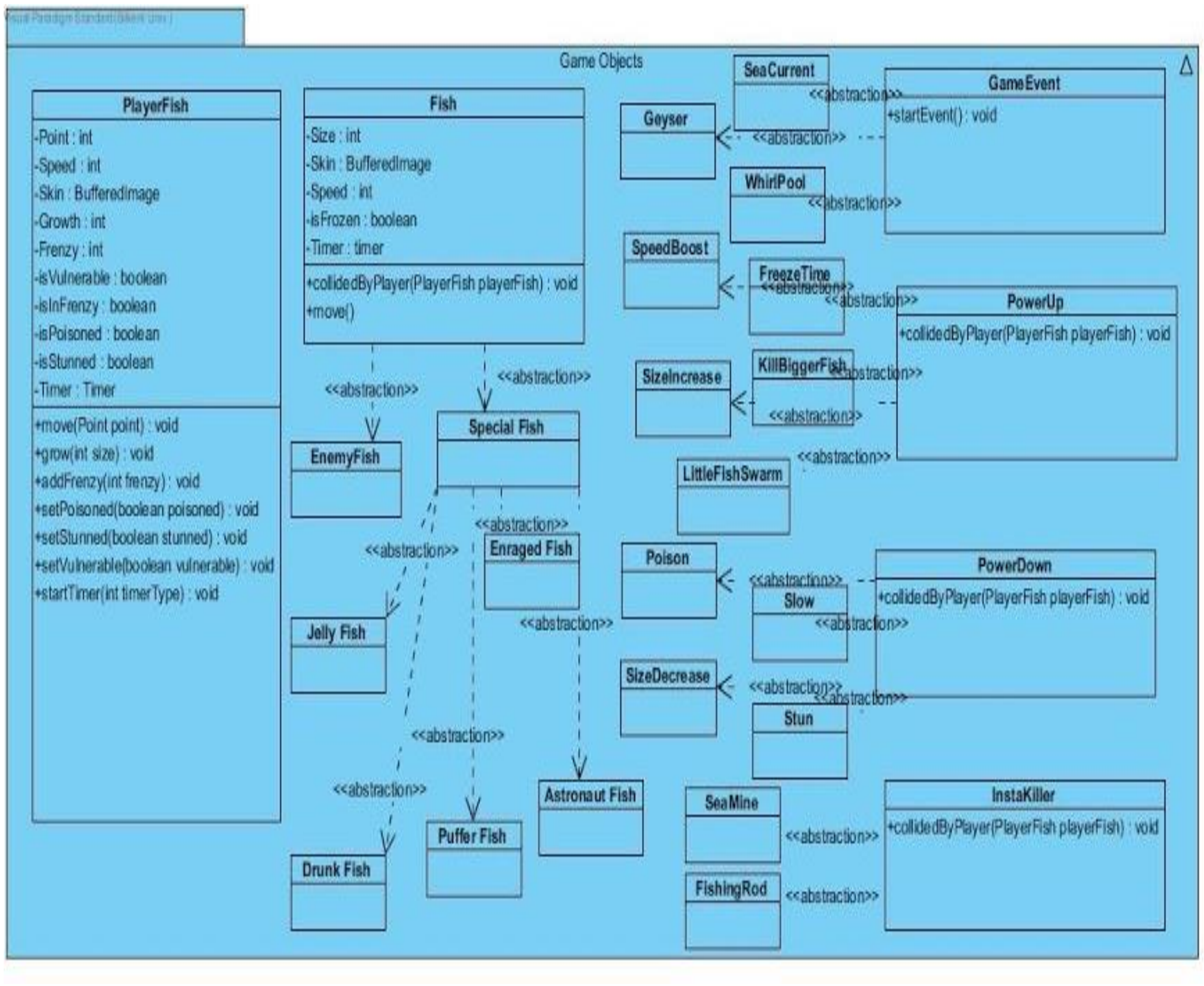


Figure 11: Class Diagram (Game Objects)

The above class diagram contains Game Objects and shows their relation. We have a Fish class which includes fish types that we have in our game such as Enemy Fish and Special Fish. PlayerFish class is the one containing attributes and operations of the user’s fish. Moreover, we have PowerUp class including SizeIncrease and SpeedBoost etc. Similarly, we implement decreasing in power in PowerDown class which includes Slow and SizeDecrease etc.

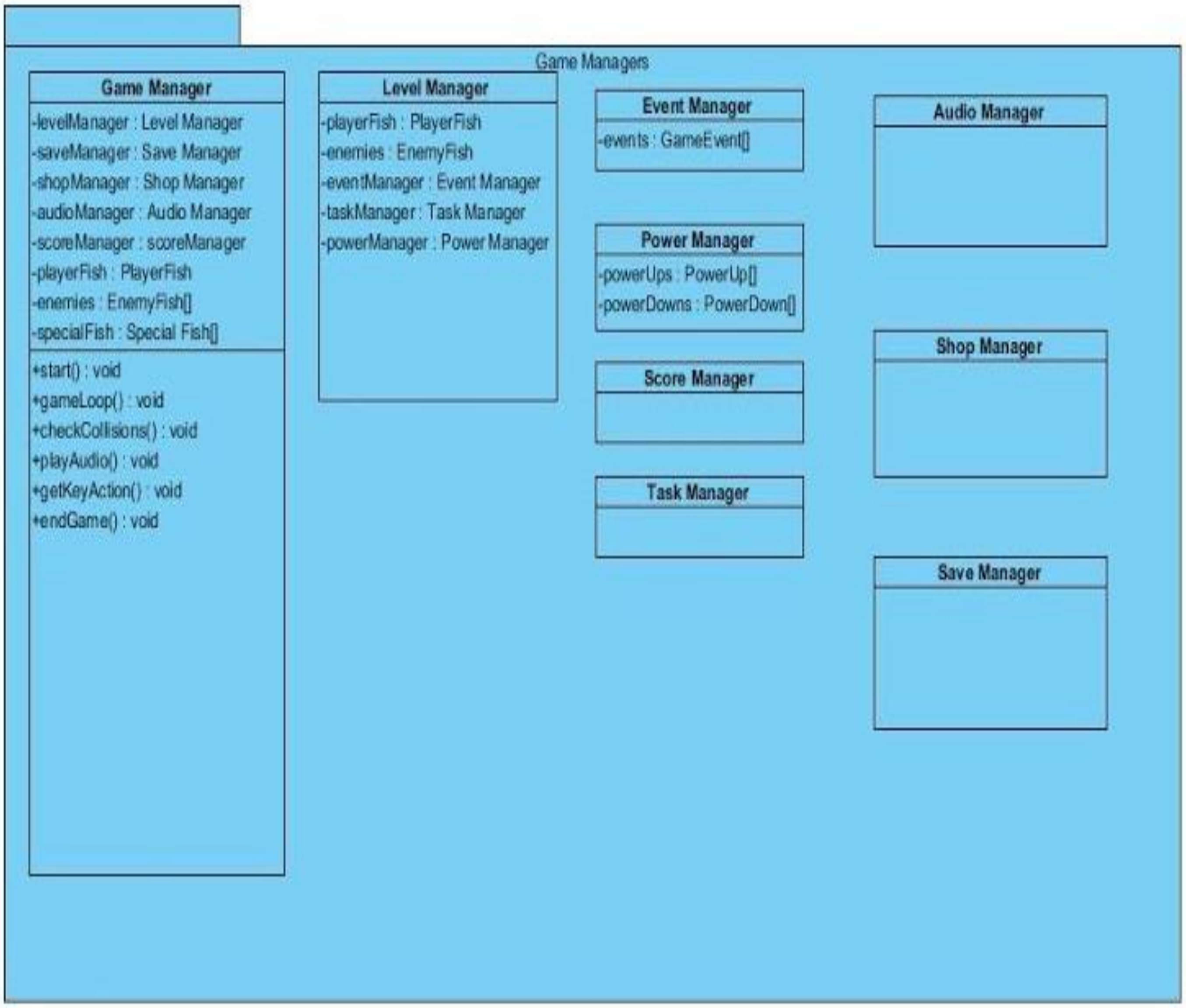


Figure 12: Class Diagram (Game Managers)

The above class diagram contains our Game Managers classes. We have decided to implement these classes separately because it is the easier to divide parts that implement crucial operations. GameManager class is the one where the main management of the game is handled. It manages the lifecycle of the game through working with other manager classes. FileManager class handles the connection between the files associated with the game. Furthermore, we have other classes which manages the features of the game such as PowerManager, SaveManager etc.

3.5.4 Dynamic Models

3.5.4.1 Sequence Diagrams

3.5.4.1.1 Start Game

Considering the flow of events, the game is quite straightforward. In the main menu, we will provide five options to user, either to choose help button and go to the help screen that explains the game in basic steps, or to select shop to buy the item that is affordable such as a skin, or to choose change settings button, or to select load, or to click start button to start a new game. When the user goes to the help activity, there is no other option but to get back to the main menu. This is the same with other options except start and load game. This is the reason why we decided to create only crucial scenarios' sequence diagrams to avoid using unnecessary diagrams.

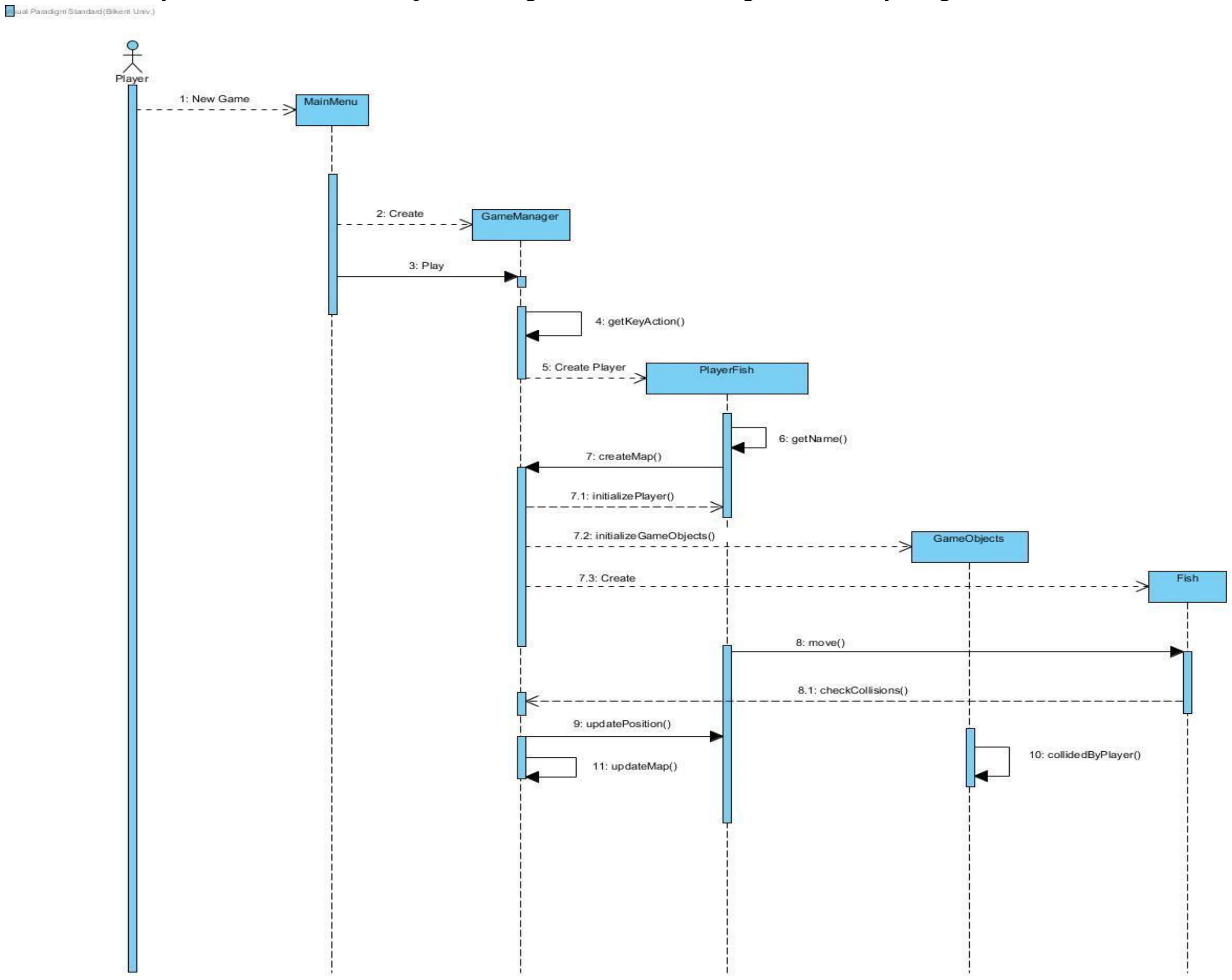


Figure 13: Start Game Sequence Diagram

Considering the above diagram, it can be said that the sequence begins when the user clicks on the game icon on the desktop. First of all, the main menu is reached and then the Game Manager class is created which is the one forming the Player Fish. And then, the user is required to enter a name. After entering the name, the map is created with all initial details including the start position of the player. Moreover, the position of the user is updated during the game.

After these steps, the fish is created and it returns the result of checking collisions to the Game Manager because when two fish have a collision their positions are changed. The game manager checks this result to update the positions of fish and also to update the map.

3.5.4.1.2 Load Game

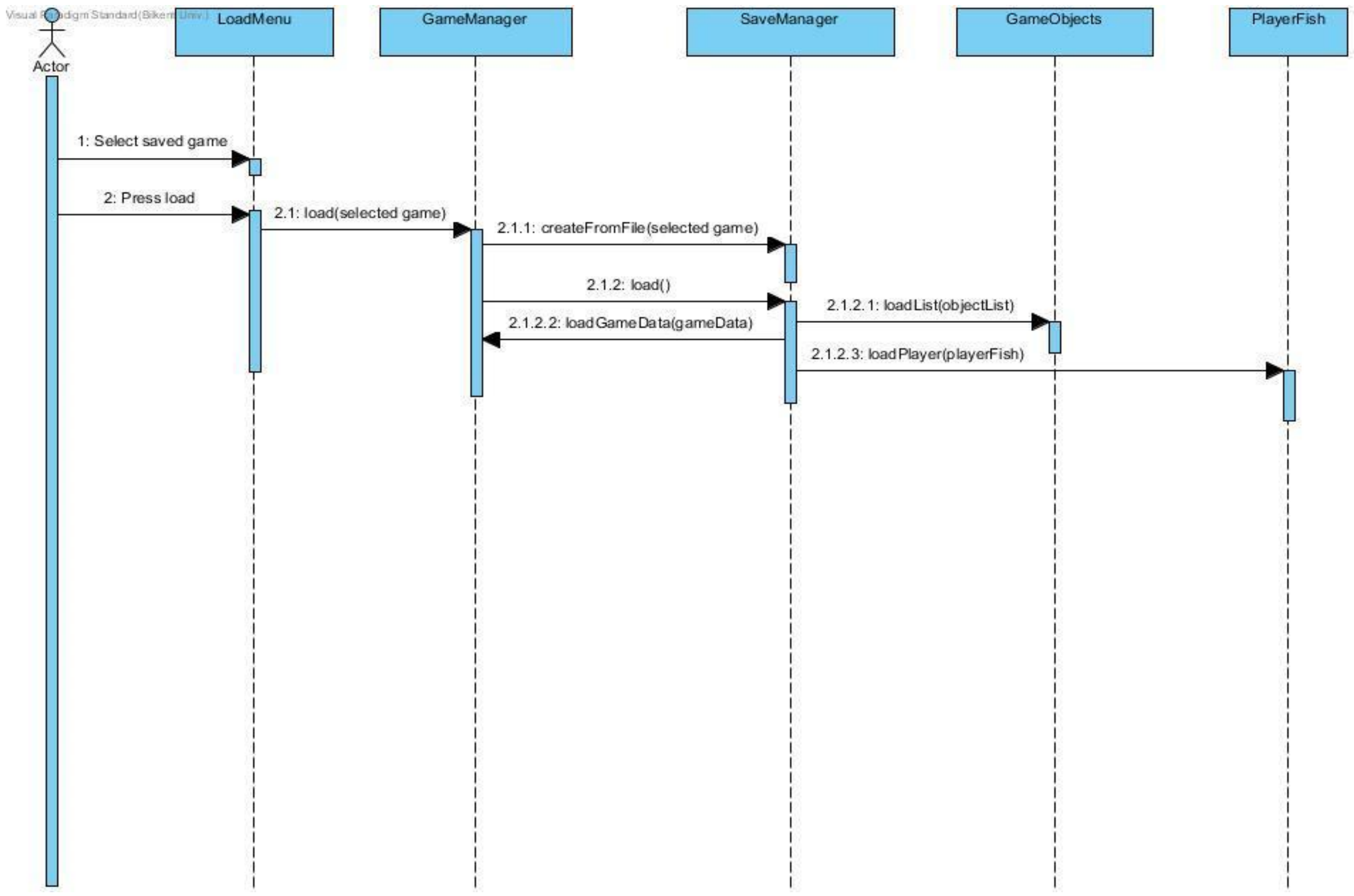

Visual Paradigm Standard (Bilkent Univ.)
Actor
LoadMenu
GameManager
SaveManager
GameObjects
PlayerFish
1: Select saved game
2: Press load
2.1: load(selected game)
2.1.1: createFromFile(selected game)
2.1.2: load()
2.1.2.1: loadList(objectList)
2.1.2.2: loadGameData(gameData)
2.1.2.3: loadPlayer(playerFish)

Figure 14: Load Game Sequence Diagram

When the user chooses their saved game the loading sequence is triggered. The game loads the selected game file by created files from the save manager. The game objects are taken back from the file as well as the player's fish's saved information.

3.5.4.2 Activity Diagram

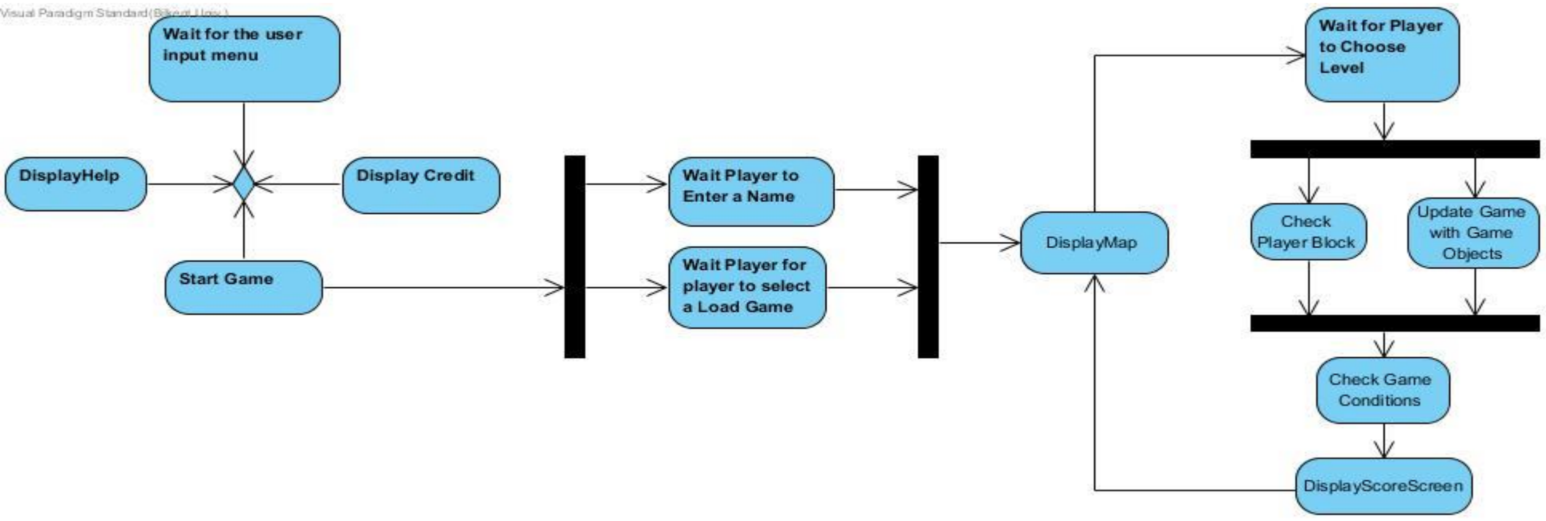
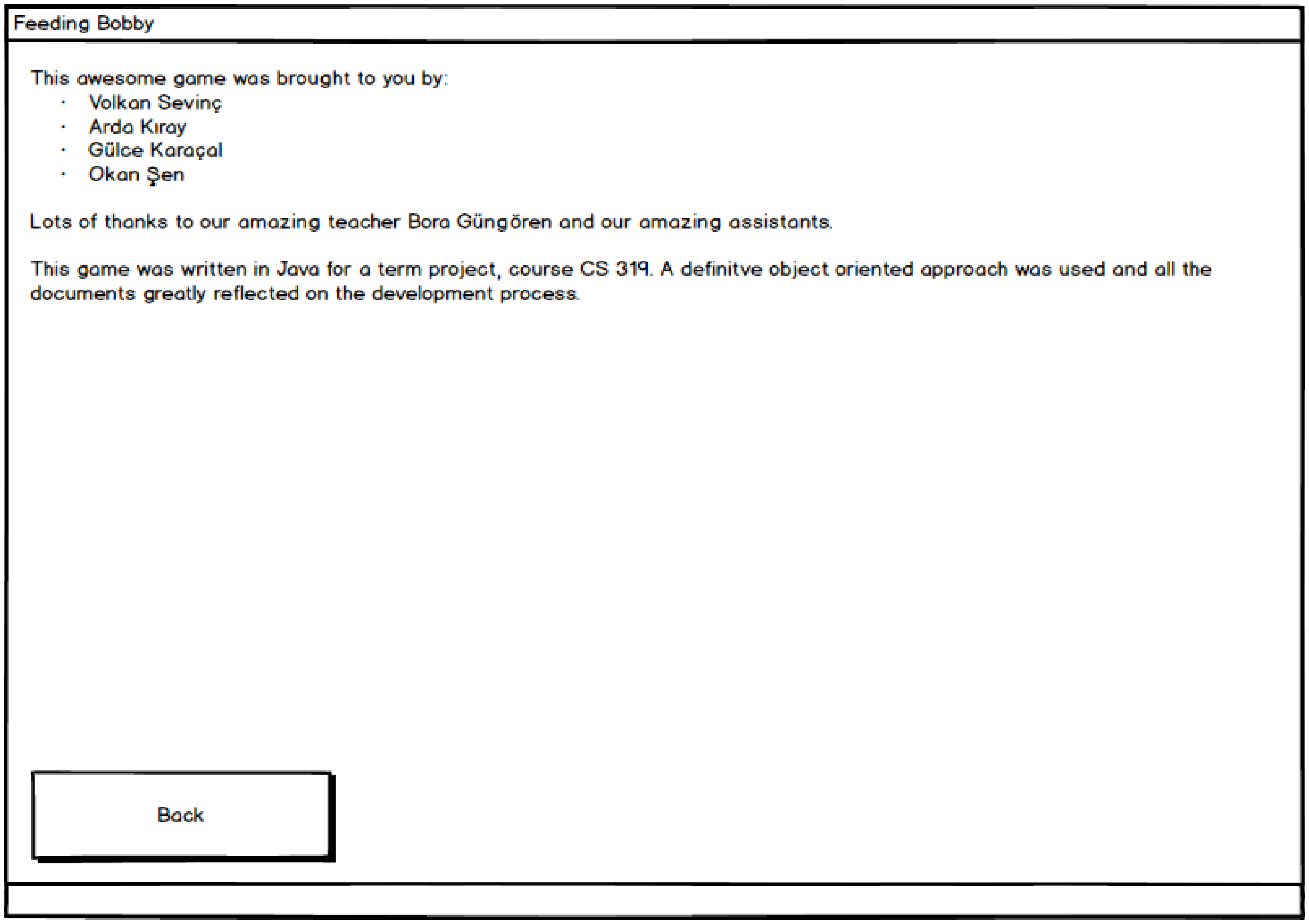

Visual Paradigm Standard(Bilkent Univ.)
Wait for the user input menu
DisplayHelp
Display Credit
Start Game
Wait Player to Enter a Name
Wait Player for player to select a Load Game
DisplayMap
Wait for Player to Choose Level
Check Player Block
Update Game with Game Objects
Check Game Conditions
DisplayScoreScreen

Figure 15: Activity Diagram

3.5.5 User Interface



This awesome game was brought to you by:

- Volkan Sevinç
- Arda Kıray
- Gülce Karaçal
- Okan Şen

Lots of thanks to our amazing teacher Bora Güngören and our amazing assistants.

This game was written in Java for a term project, course CS 319. A definitve object oriented approach was used and all the documents greatly reflected on the development process.

Figure 16: Credits

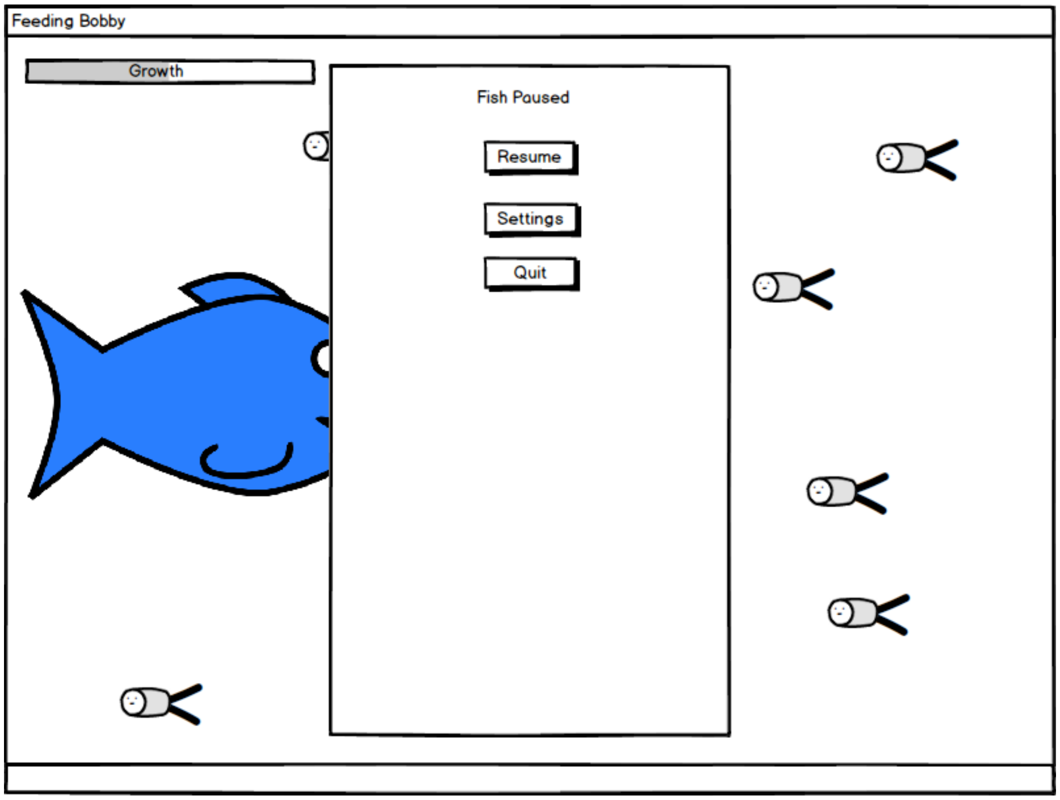


Figure 17: Pause Game

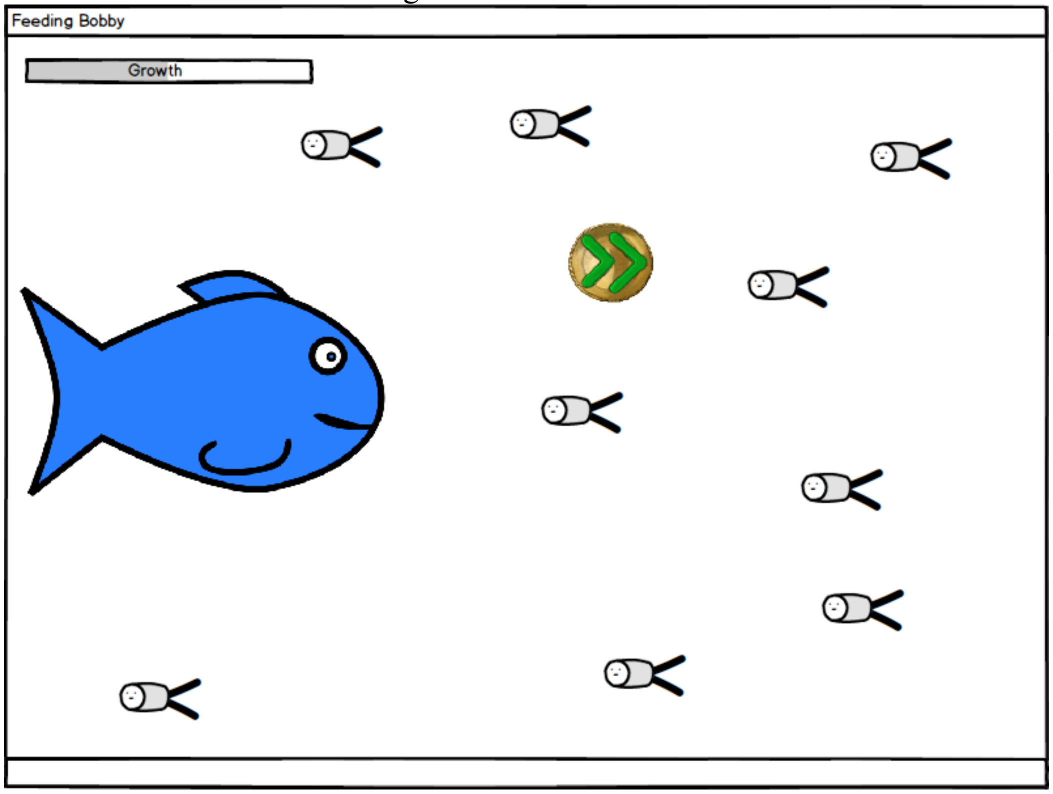


Figure 18: Game Screen

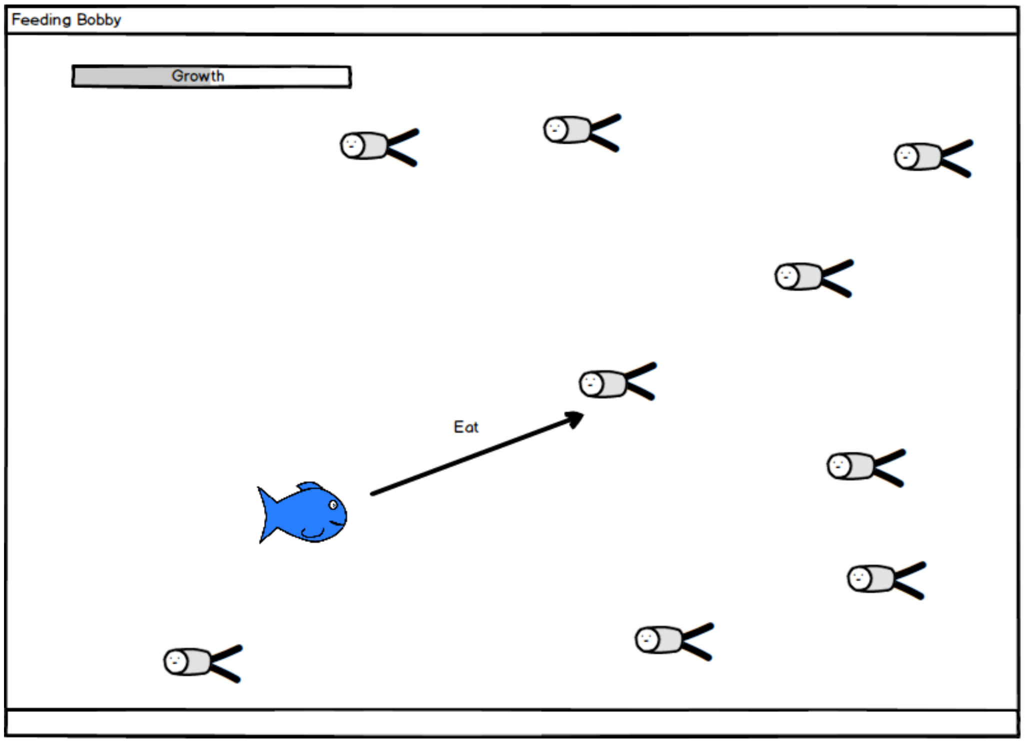


Figure 19: Before Eating

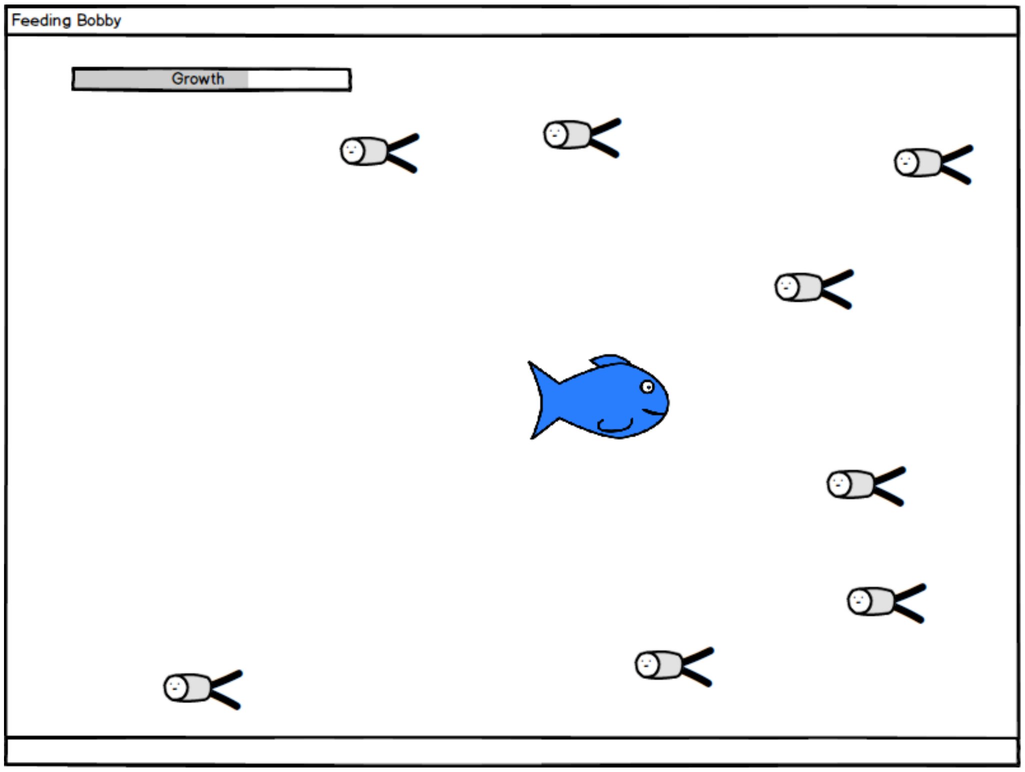


Figure 20: After Eating

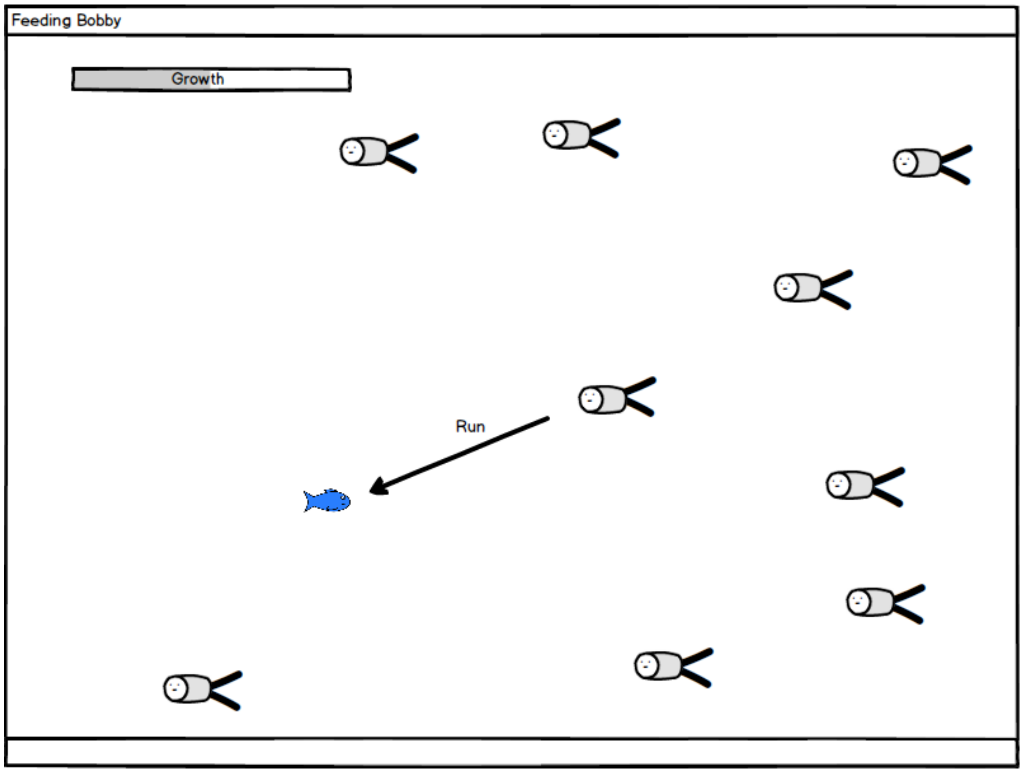


Figure 21: Escape

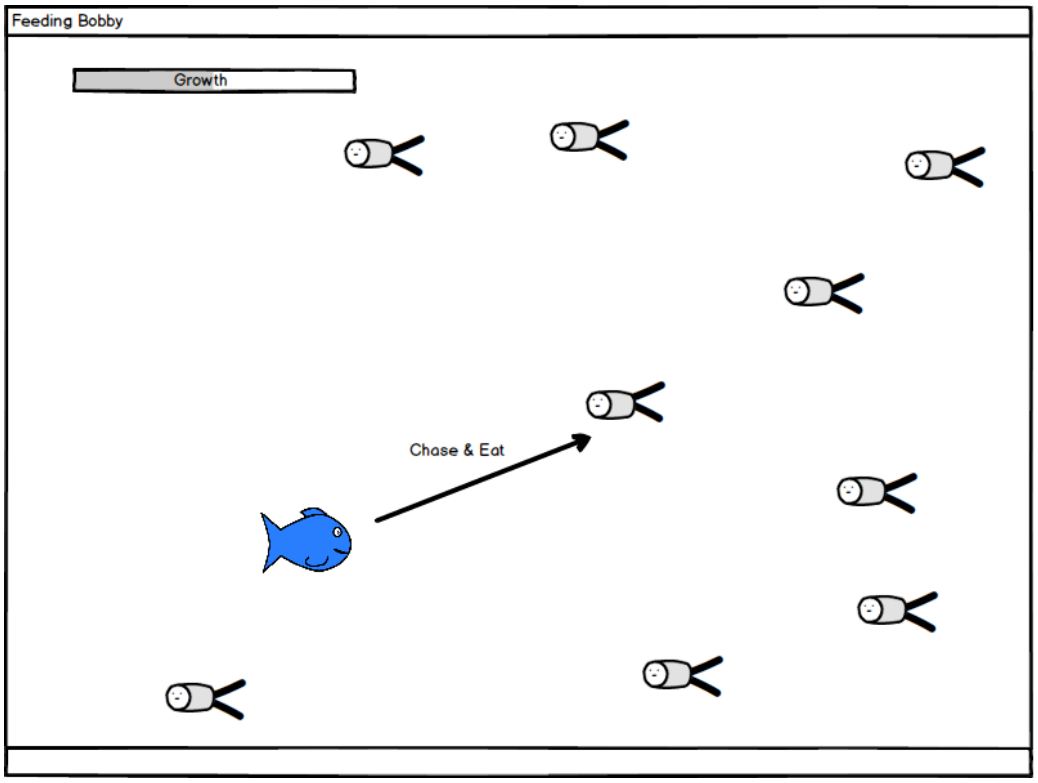


Figure 22: Chase

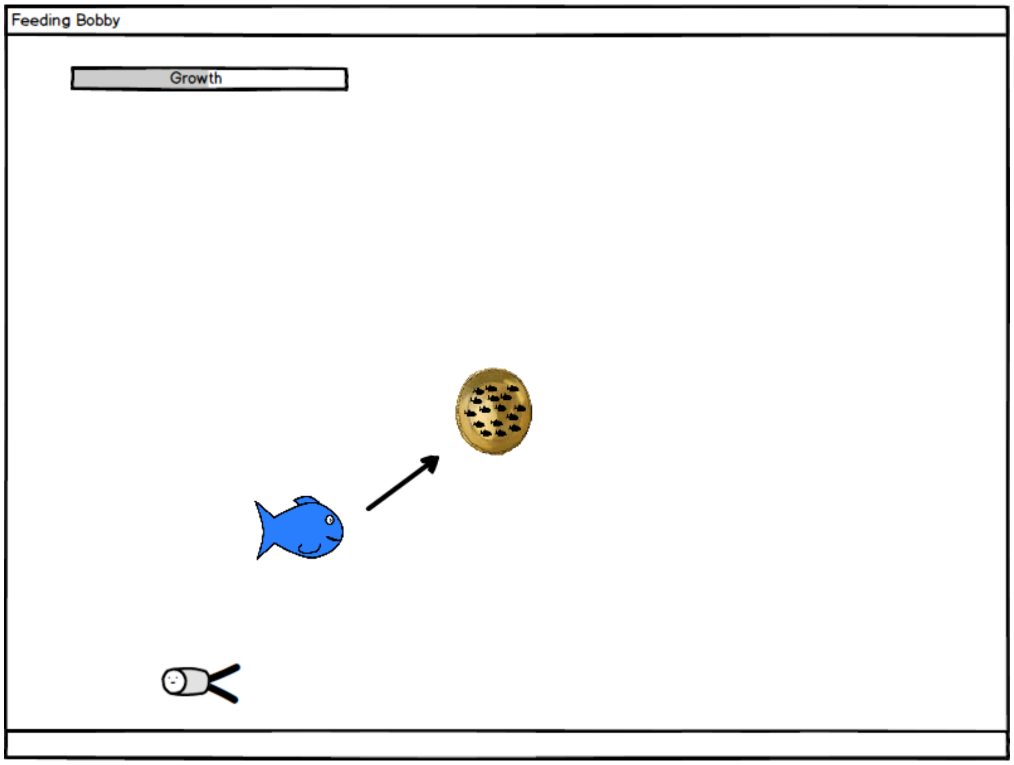


Figure 23: Before Swarm

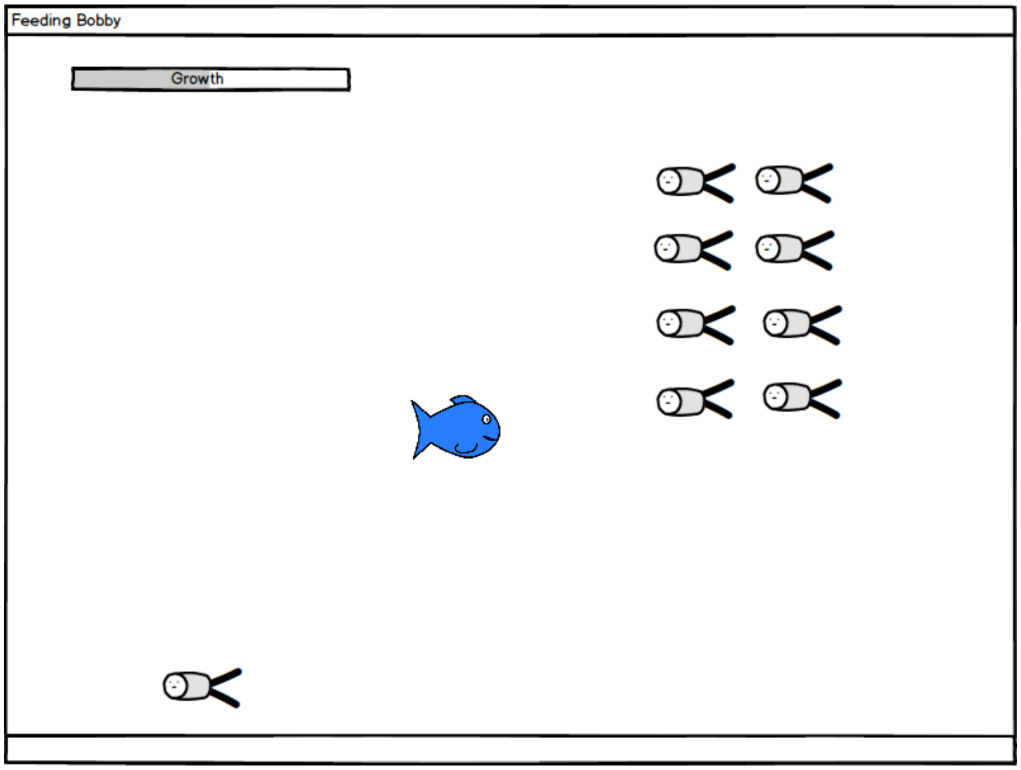
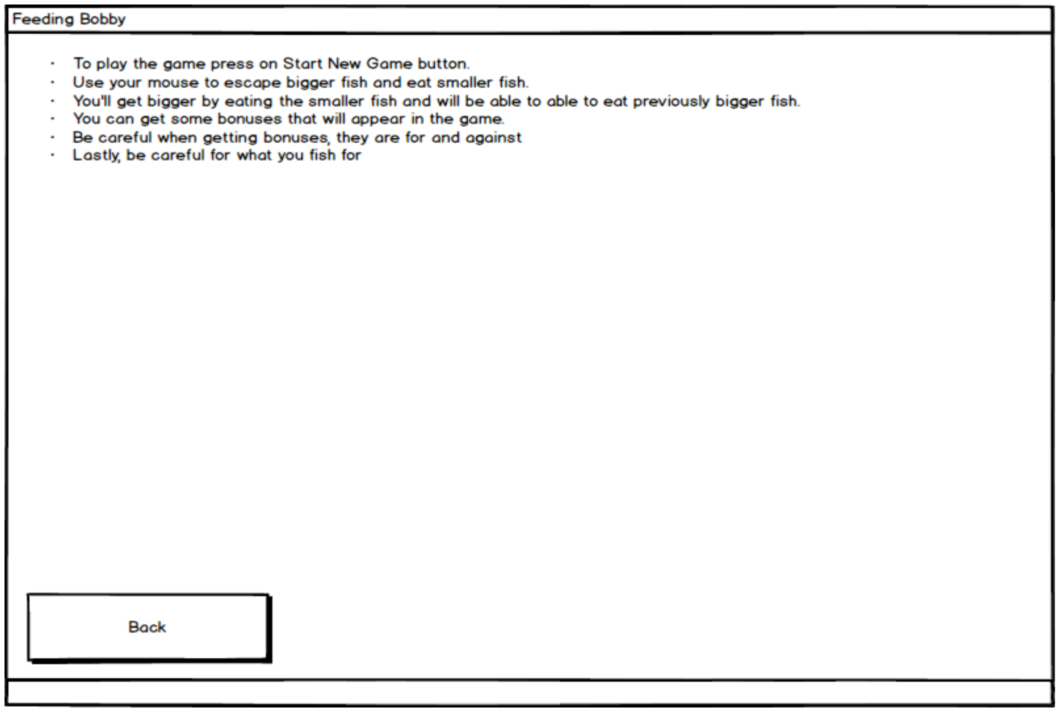
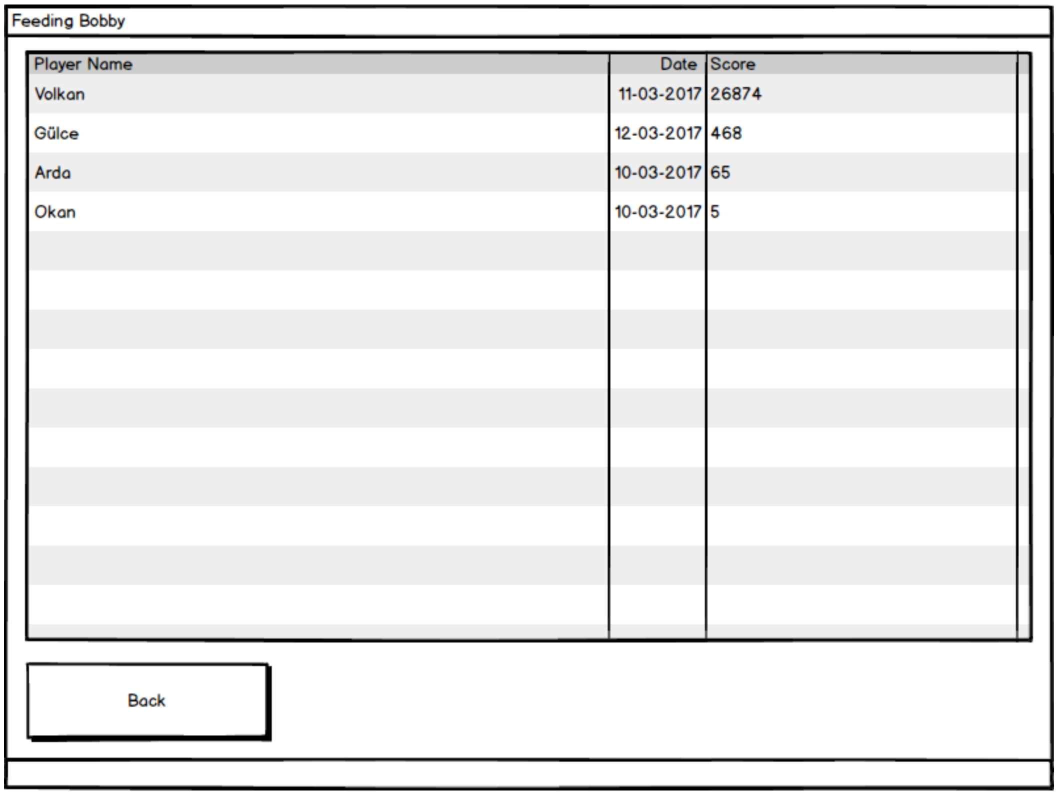


Figure 24: After Swarm


Feeding Bobby

- To play the game press on Start New Game button.
- Use your mouse to escape bigger fish and eat smaller fish.
- You'll get bigger by eating the smaller fish and will be able to able to eat previously bigger fish.
- You can get some bonuses that will appear in the game.
- Be careful when getting bonuses, they are for and against
- Lastly, be careful for what you fish for

Figure 25: Help Screen


Feeding Bobby

Player Name	Date	Score
Volkan	11-03-2017	26874
Gülce	12-03-2017	468
Arda	10-03-2017	65
Okan	10-03-2017	5

Figure 26: High Scores

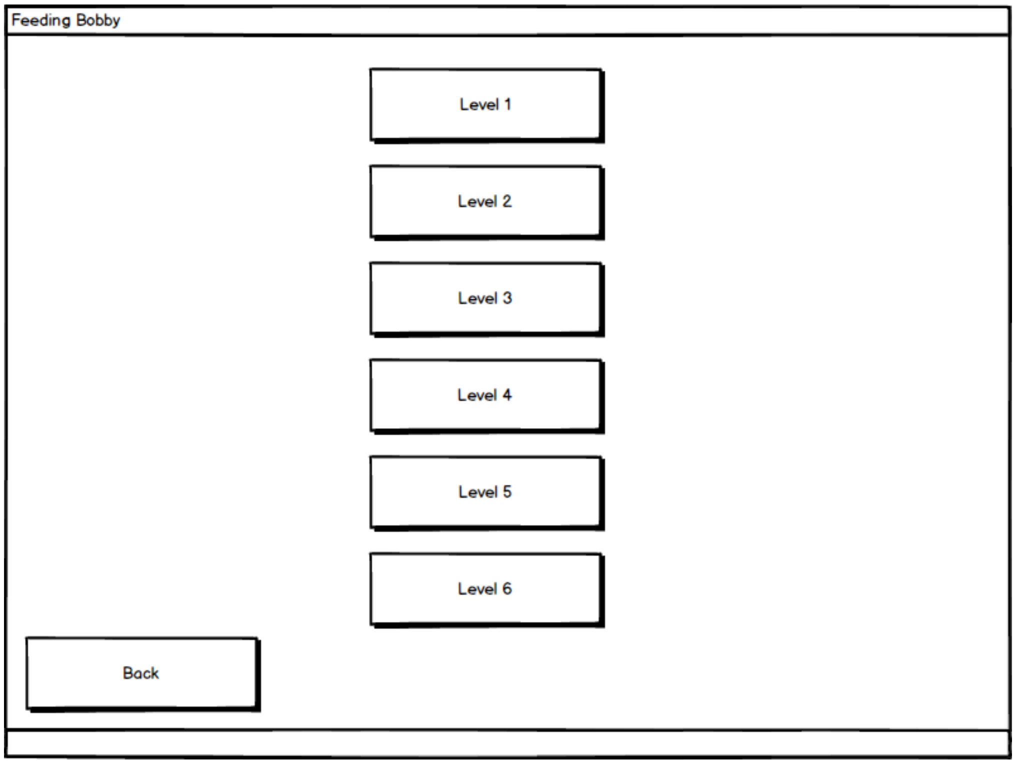


Figure 27: Journey Screen

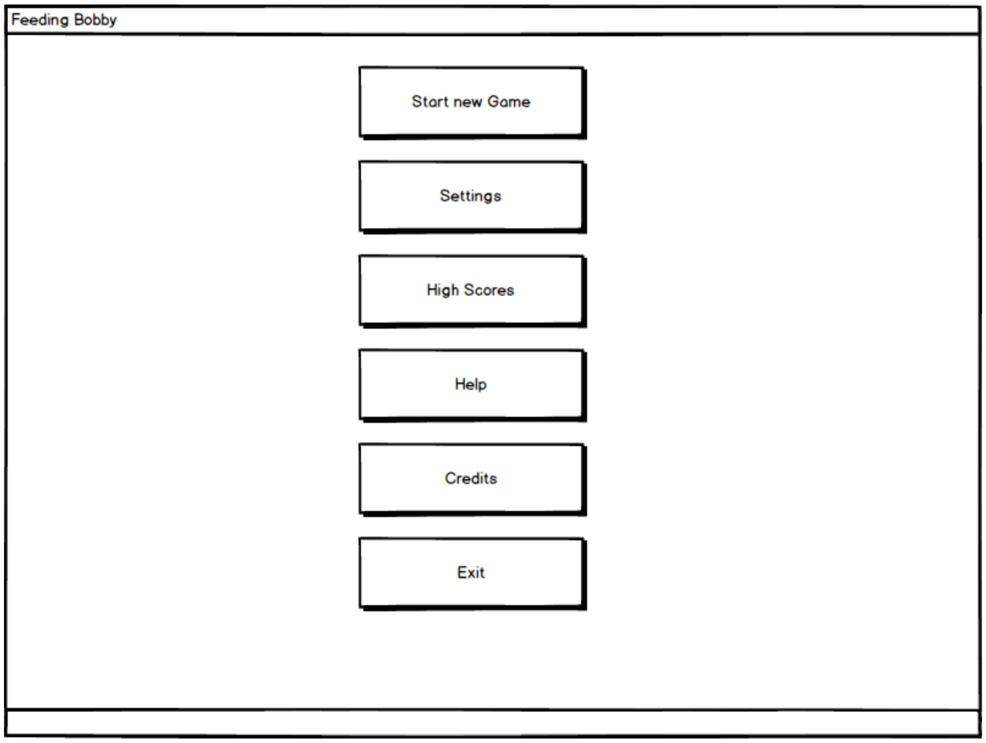


Figure 28: Main Screen

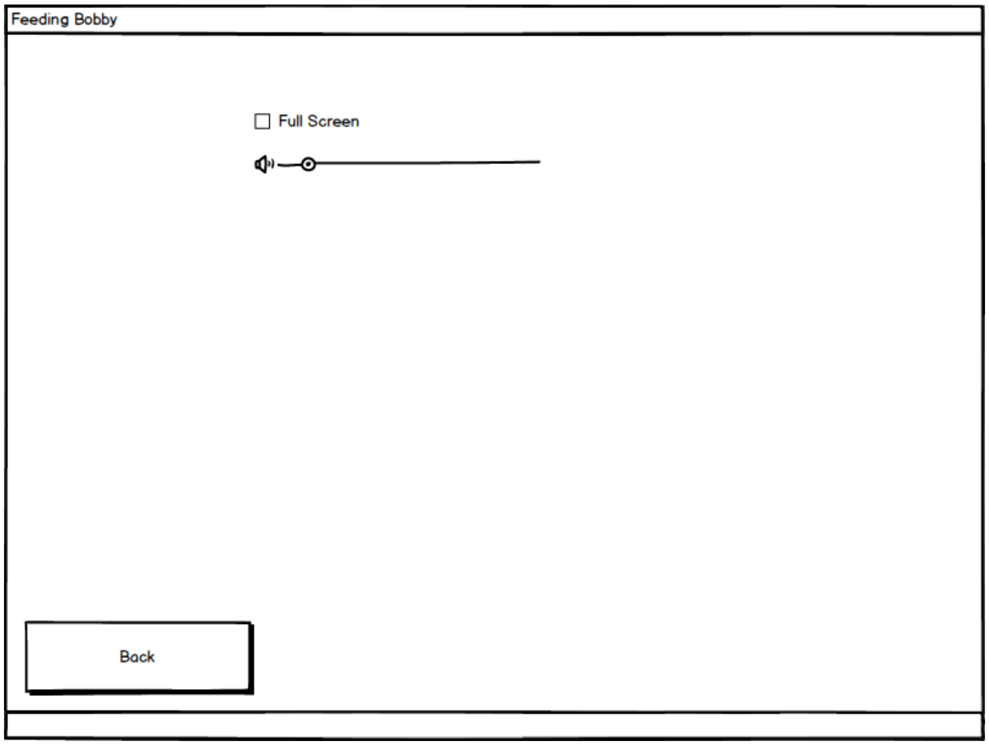


Figure 29: Settings

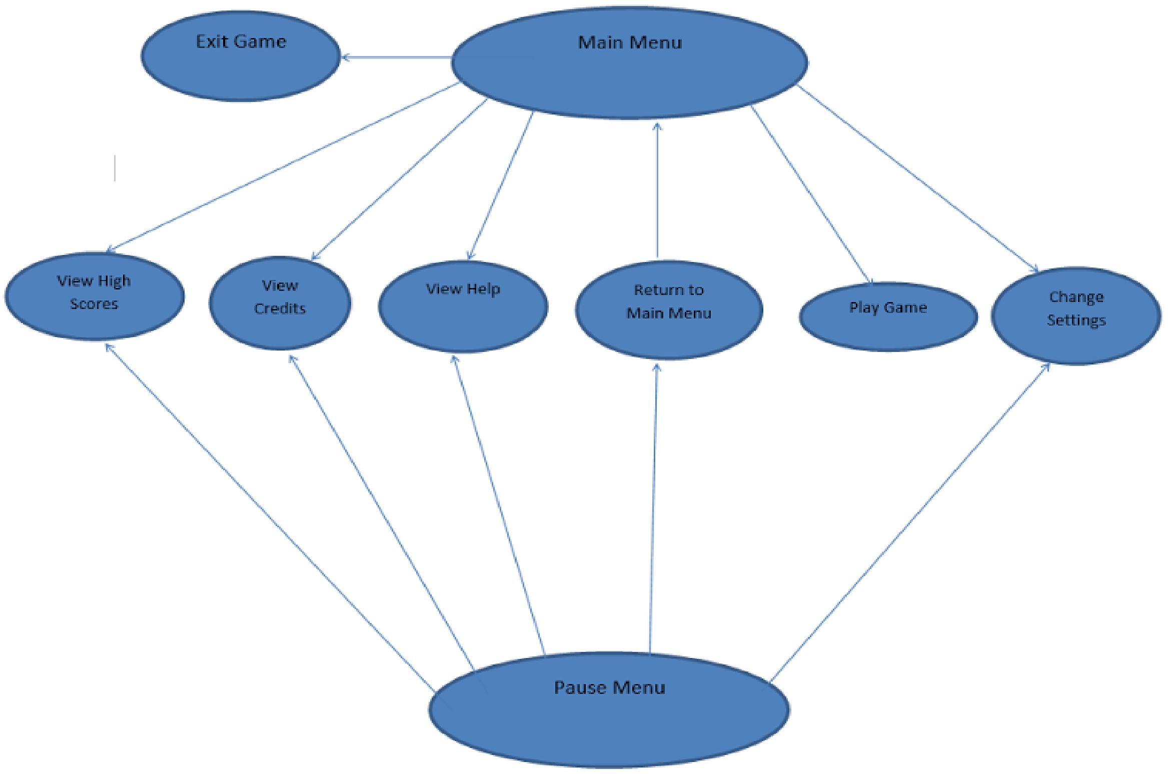


Figure 30: Navigational Path

4. Conclusion

In conclusion, in this analysis report we aimed to create an arcade game called Feeding Bobby. Our analysis report has two main sections which are Requirements Specification and System Model. In requirements specification, we analyzed the operations which are accessible to the user. Considering these actions, we defined our functional and nonfunctional requirements for the game.

System Model Section consists of following these four parts;

1. Use Case Model
2. Class Diagram
3. Dynamic Models
4. User Interface

First of all, for functional modelling, we created our use case scenarios. System design was the second part which we have formed our Sequence and Activity Diagrams. In sequence diagrams we have shown all the possible actions that a player can perform. Activity diagram demonstrates primarily the gameplay. It represents our game components such as fish and power-ups. Furthermore, our implementation is indicated by our class diagram.

To sum up, Feeding Bobby is a game that takes inspiration from a game called "Feeding Frenzy [1]" that shares the same concept as our game with the inclusion of our own ideas and implementations. The game will be unique in the case of its attributes, themes as well as its visuals and features.