Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

Academic Warfare : The Conflict in Bilkent

Analysis Report

Doğukan Yiğit Polat, Selin Fildiş, Yasin Erdoğdu, Onur Elbirlik

Instructor: Uğur Doğrusöz

Analysis Report

Oct 15, 2016

Contents

Introduction

What are Tower Defense games? How do they work?

Tower defense games are mostly strategy games in which the player has to defend their territory which is their base tower from enemy invaders by planning defense strategies such as attacking with weapons, and defending using towers and shields in that territory.

Reasons for choosing to build a tower defence game

Tower defense games are fun to play and we found it interesting to build as well. We believe that this game will be a good Object-Oriented Software Engineering project.

Report Overview

This report consists of an overview of our game objects, functional and non-functional requirements and the use-case diagram and scenarios.

Overview

This tower defense game will implement a wave system. The wave system is groups of enemies entering and moving like a wave such as group 2 enters 10 seconds after group 1. The waves will continue as long as the player keeps on playing and after each wave the player will gain coins according to the number of enemies eliminated in that particular wave. Thus, the coins earned will increase as the game continues as the waves will get harder.
To defeat these waves the player will build weapons to stop their enemies from reaching and destroying their base. The game will also have power-up's and the user will collect these power-up's to help them defeat their enemies.

The game has 3 different maps and the second and third maps will unlock if the user completes the waves in the previous map successfully. The user can choose which map they want to play with after unlock.

The game can be paused, saved and continued. The saved game can be loaded in the beginning as well.

The game will be a desktop application written in java and will use a mouse for the user to interact with the game.

List of weapons

There are four weapons that the user can choose from, some of these weapons can be unlocked in-game as the player goes further and bought the coins the user gains after each wave. Weapons fire to the nearest enemy.

- *Tekman's Cannon:* Tekman's cannon is the base weapon in this game, the cannon shoots with 1 ball per second and deals 10 damage. To place the gun the user has to pay $20.

Figure 1: A cannon, Tekman's cannon will look like this [1]

- *Oktel's Double Trouble*: This gun is one of the unlockable guns in the game, can be unlocked for $200. Deals 7 damage but its fire rate is 2 shots per second. So it is a much better weapon than Tekman's Cannon. Build cost of Oktel's Double Trouble, after unlock, is $150.

Figure 2: Oktel's double trouble will be the 2D version of this [2]

- Doğramacı's Rail Gun: This is the most powerful weapon in the game. The gun deals 20 damage and fires at rate 5 shots per second and the build cost is $350 after unlock. The unlock fee is $2000.

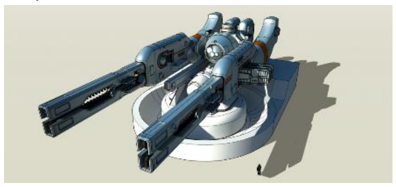


Figure 3: Doğramacı's rail gun will be simillar to this [3]

- Bilka's Bazlamaya Ayvalık Spitter: This weapon slows down the target by %10 on its each shot and shoots at rate 5 shots per second like the Doğramacı's Railgun. The unlock fee is $250 and in-game build cost is $150.

Figure 4: Bilka's Bazlamaya Ayvalık Spitter will look similar to this [4]

Enemies

There will be an academic and assistants accompanying the academic in every wave. In further waves the possibility of the department chair, the dean of engineering and the rector appearing as an academic will increase.

The Academics

A wave will consist of (wave #) / 4 number of academics as enemies. To eliminate deterministic strategies, there’s a %15 chance that an extra academic will also take place in a wave. Academics also move 0.5 cells per second. Eliminating an academic will provide $20. Academics have 80 health points.

The Assistants

In the each wave of enemies, each academic will have (wave #) of assistants with them. Assistants will have 15 health points and move 0.5 cells per second. Eliminating an assistant will provide $5.

List of Power-ups

There will be 4 kinds of power-ups.

- Mines: User can locate this bonus to any cell in the game screen to destroys enemies within its cell.
- Life Steal: With this bonus, user can increase tower health by %1 for each enemy killed in during 5 seconds from its activation.
- Freeze: User can freeze all enemies for 5 seconds by using this bonus.
- Thunder Ball: This bonus enables user to explodes each grid cell with probability %10, so if enemy is inside the cell that has exploded then it will die.

List of maps

There will be different settings of speed in every map. There are three terrains for this game and in each, the enemies will move in different speeds.

Grass Terrain

The grass terrain is the basic terrain type. It does not have any affect on neither the enemies.

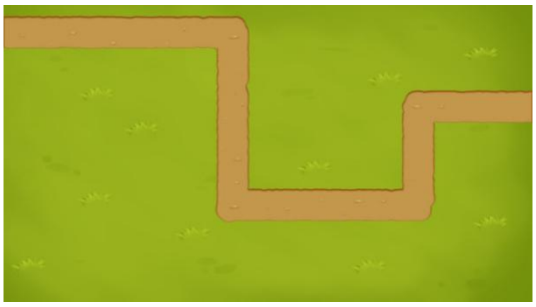


Figure 5: The Grass Terrain [5]

Snow Terrain

The Snow terrain's road is icy and is hard for the enemies to move, so their speed is dropped by %10 in this terrain.

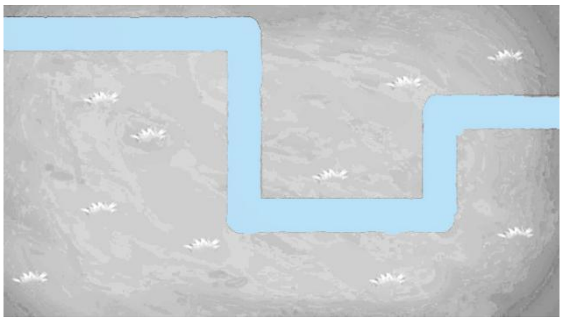


Figure 6: Snow Terrain [5]

Forest Terrain

The forest terrain is similar to the snow terrain but in reverse. In this terrain, the speed of the enemies are increased by %10.

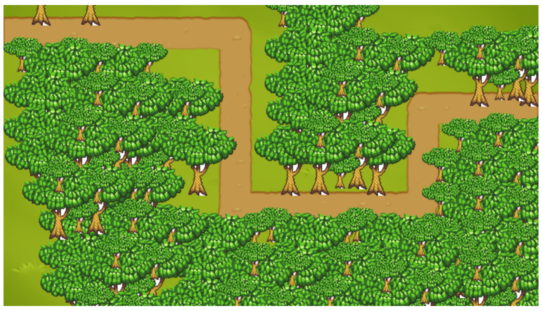


Figure 7: Forest Terrain [5]

Requirements

Functional Requirements

Help

User will able to access information about game dynamics and game elements such as enemies, weapons and bonus types via the help menu. It will contain game rules and detailed description of bonus types and weapons. It will also include information about developers of the game.

High score

The game will keep top 10 scores with player name information. User will be able to see top 10 high scores of the game via clicking “High Score” button.

Play

User can start in the process of playing game with pressing Play button. User will be able to select the any game map before starting to play game. There will be three different maps with having different features.

Budget

The game will start with 200$ initial budget for user. With this budget, user will be able to have chance about how to use this budget. During the game, user will gain more money for each enemy which is killed.

Weapons

User will be able to select any weapons which they wants in order to unlock or use for tower defence with their earnings. However, user can use any locked weapon after it will be unlocked.

Bonuses

The game will supply four different types bonuses to user during the game. After having bonuses, User can use any bonuses which they want at any moment of the game.

Score and waves

User will be able to see current his/her own score and wave number at any time during the game.

Save and load

User will be able to save current state of game and user can load any saved games in order to continue to play where they left off.

Pause and resume

Player will be able to pause the game and resume the game afterwards.

Ready

After the selecting game map, user will have time to locate weapons into any cell which is not on the enemies' way. When user will be ready to play, User can launch the game by clicking ready button at the bottom.

Non-Functional Requirements

User Friendly interface

Since the user should be aware of every detail in the game, User interface should not be filled with useless and unnecessary information or details. This will lead user to understand the game easily and doesn't interrupt their gameplay with clutters in the user interface. Game will be easy to understand and details will be easily distinguishable.

Game Performance

Frame refresh delay should not exceed 17 milliseconds. This is approximately equivalent to 60 FPS (Frames per second) which is the native refresh rate of most display devices. This will give the users opportunity of smooth gameplay and low response time. Effective usage of objects and rendering will be also an important point of concern for the game. Keeping system requirements low will provide large amount of users to be able to play the game.

Graphical Quality

Since this project is a game, it should be good looking as well as good working.Higher amount of object, good looking texture and objects will require an efficient rendering of the graphics. Graphics and textures will be smooth.

Re-usability and extendibility

Game has different kinds of maps, weapons and waves, so that users can create their own replayability values. This kind of modular system can allow the game to be modified and extended in the future.

System Models

Use Case Model

Brief Senarios

Playing the Game (Complete New Game)

Player launches the game and chooses to play a new game. Player chooses one of the maps that he/she wants to play on. After the map selection, player places simple defensive entities with the initial budget, before the first wave of enemies attack. Player eliminates all of the enemies in the first wave. Builds more defense with the money earned during the wave, repeats this process for all waves. After each wave, there will be more and more enemies so it will be harder to defend the tower, eventually some enemies will make it to the tower and give damage to it.In time, tower will get more and more damage. Once the tower gets enough damage, the game will end. The score at the time when the tower is destroyed will be the final score.

Playing the Game (Save and Exit)

Everything except the exit condition is same as 5.1.1.1. But at any instance of the game, player may want to save the current state of the game. Player chooses to save the game state and quit playing afterwards.

PlayIng the Game (Load and Play a Previous Game)

Player chooses to play a previous game. Player chooses the save file he/she wants. Once the file is loaded game is continued from the point where user had previously chosen to save the game into that particular save file.
Rest is same as the exit condition of 5.1.1.1.

Viewing High-score Table

Player chooses to view high score table. Views the top scorers in a decreasingly sorted manner. User quits the game.

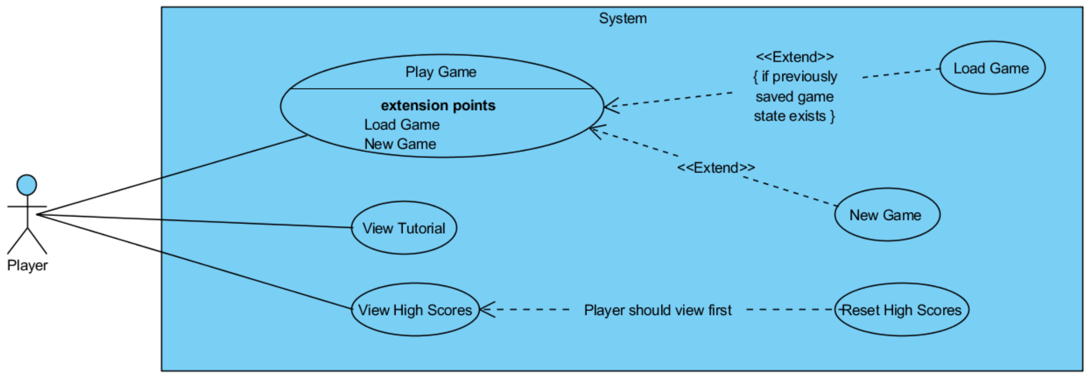
Resetting High-score Table

Player chooses to view high score table. Player may want to get rid of the entries and make a clean start. Player chooses to reset high score table, entries gets deleted. Player quits the game.

Viewing the Tutorial

Player chooses to view a tutorial before giving the game a try. A tutorial session will begin. Next, player reads the rules, goes through the descriptions and finishes the tutorial. After finishing the tutorial, player quits the game.

Use-Case Diagram



Use-Case Descriptions

Play Game

Primary Actor:

Player

Stakeholders and Interests:

Player wants to play the game, have fun and earn as much points as possible.

Pre-condition:

In the first run, player can only choose one of the maps. If player does well and unlocks other maps, he/she can choose one of the unlocked maps as well.

Post-condition:

If the tower stood still for enough time to unlock a new map, a new map will be unlocked. If the score earned is enough to get into leaderboard, it will be recorded there.

Entry Condition:

Player chooses to play the game when the game program starts.

Exit Condition:

Tower will take some particular amount of damage and game ends. Or player chooses to exit the game directly.

Success Scenario Event Flow:

1. Game progam execution initialized.
2. Player makes the map selection, chooses one of the available maps.
3. Player starts to play the game.
4. Player plays the game until the tower gets defeated.
5. Player gets enough scores to unlock the new map.
6. Player's score is eligible for the leaderboard.
7. Player records his/her score.
8. Player chooses to exit game program.

Alternative Event Flow 1:

1. Game progam execution initialized.

2. Player makes the map selection, chooses one of the available maps.

3. Player starts to play the game.

4. Player plays the game until the tower gets defeated.

5. Player could not get enough scores to unlock the new map.

6. Player chooses to play on the same map again.

7. Follow success scenario steps from step #3.

Alternative Event Flow 2:

1. Game progam execution initialized.

2. Player chooses to load his/her previously saved game state

3. Follow success scenario steps from step #4.

Alternative Event Flow 3:

1. Follow success scenario steps until step #4.

2. Player chooses to save his/her current game state.

3. Follow success scenario steps from step #8.

Load Game

"Alternative Event Flow 2" of "Play Game"

New Game

"Success Scenario Event Flow" of "Play Game"

View Tutorial

Primary Actor:

Player

Stakeholders and Interests:

Player wants to learn about the basics of the game.

Pre-condition:

Player does not know how to play a tower defense game or wants to understand the dynamics of Academic Warfare.

Post-condition:

Player knows about the basics of a tower defense game and has an idea about game dynamics of Academic Warfare.

Entry Condition:

Player chooses to view the tutorial when the game program starts.

Exit Condition:

Player chooses to exit from the tutorial session.

Success Scenario Event Flow:

1. Game progam execution initialized.

2. Player chooses to view the tutorial.

3. Player views the tutorial and learns about the game.

4. Player chooses to exit tutorial screen.

View High-Scores

Primary Actor:

Player

Stakeholders and Interests:

Player wants to view the leaderboard

Pre-condition:

Player wants to know about the situation of the leaderboard.

Post-condition:

Player acknowledged the high-scores that are on the leaderboard.

Entry Condition:

Player chooses to view high-scores when the game program starts.

Exit Condition:

Player chooses to exit leaderboard screen.

Success Scenario Event Flow:

1. Game progam execution initialized.

2. Player chooses to view leaderboard.

3. Player exits leaderboard.

4. Player chooses to exit game program.

Reset High-Scores

Primary Actor:

Player

Stakeholders and Interests:

Player wants reset the high-score table.

Pre-condition:

Player wants to make a clean start and chooses to delete all high-score table entries.

Post-condition:

Player reset the leaderboard.

Entry Condition:

Player chooses to view high-scores when the game program starts.

Exit Condition:

Player chooses to exit leaderboard screen.

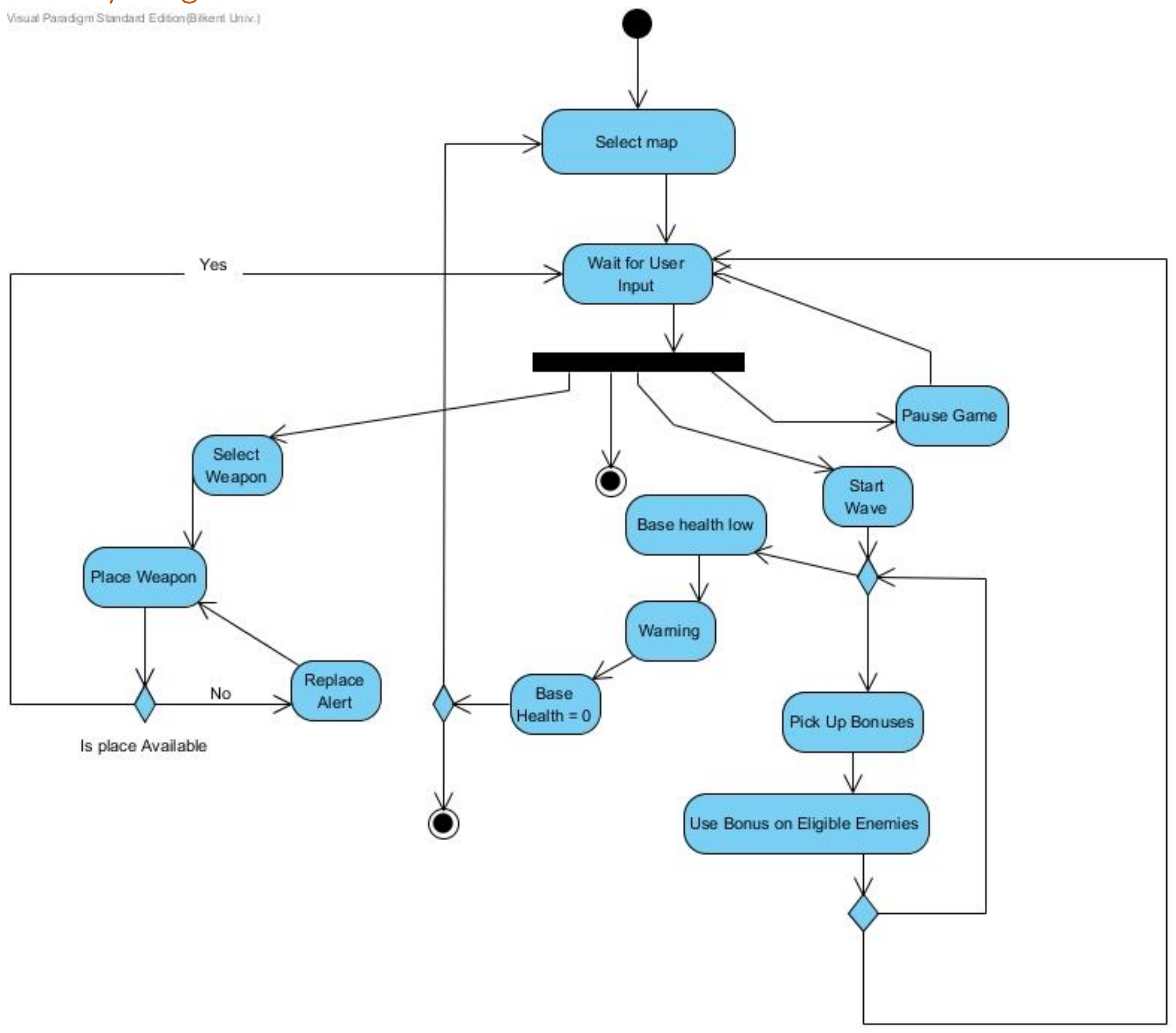
Success Scenario Event Flow:

1. Game progam execution initialized.
2. Player chooses to view leaderboard.
3. Player chooses to reset the highscores table.
4. Player chooses to exit leaderboard.
5. Player chooses to exit game program.

Dynamic Models

In this section, crucial parts of the AcademicWarfare game system will be explained via activity and sequence diagrams. GameEngine class and user interactions are main focus of our game so their mechanism will be explained here.

Activity Diagram



Sequence Diagrams

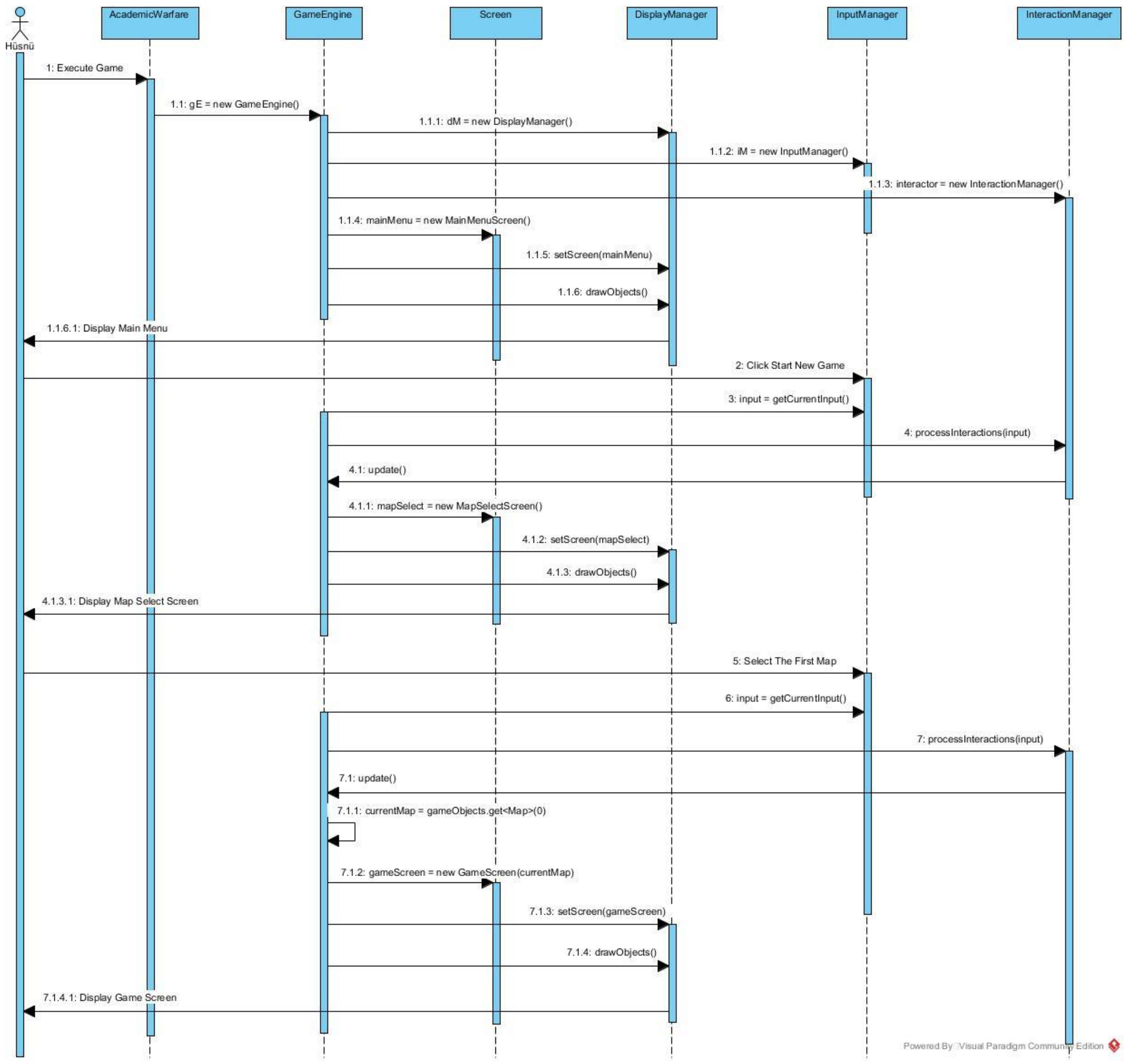
Start Game

Scenario:

Player launches the game and chooses to play a new game. Player chooses one of the maps that he/she wants to play on.

Description:

The main class is AcademicWarfare which initializes a GameEngine class. GameEngine class then takes control of the DisplayManager, InputManager, PhysicsManager etc. (You can check the class diagram to see what composes the the GameEngine class). User inputs are handled by InputManager class and GameEngine checks for any inputs if there are any. Then the GameEngine processes each GameObject and updates them on each frame to be generated. GameEngine sets and modifies any Screen object that is associated with DisplayManager.


Hüsnü
AcademicWarfare
GameEngine
Screen
DisplayManager
InputManager
InteractionManager
1: Execute Game
1.1: gE = new GameEngine()
1.1.1: dM = new DisplayManager()
1.1.2: iM = new InputManager()
1.1.3: interactor = new InteractionManager()
1.1.4: mainMenu = new MainMenuScreen()
1.1.5: setScreen(mainMenu)
1.1.6: drawObjects()
1.1.6.1: Display Main Menu
2: Click Start New Game
3: input = getCurrentInput()
4: processInteractions(input)
4.1: update()
4.1.1: mapSelect = new MapSelectScreen()
4.1.2: setScreen(mapSelect)
4.1.3: drawObjects()
4.1.3.1: Display Map Select Screen
5: Select The First Map
6: input = getCurrentInput()
7: processInteractions(input)
7.1: update()
7.1.1: currentMap = gameObjects.get<Map>(0)
7.1.2: gameScreen = new GameScreen(currentMap)
7.1.3: setScreen(gameScreen)
7.1.4: drawObjects()
7.1.4.1: Display Game Screen
Powered By Visual Paradigm Community Edition

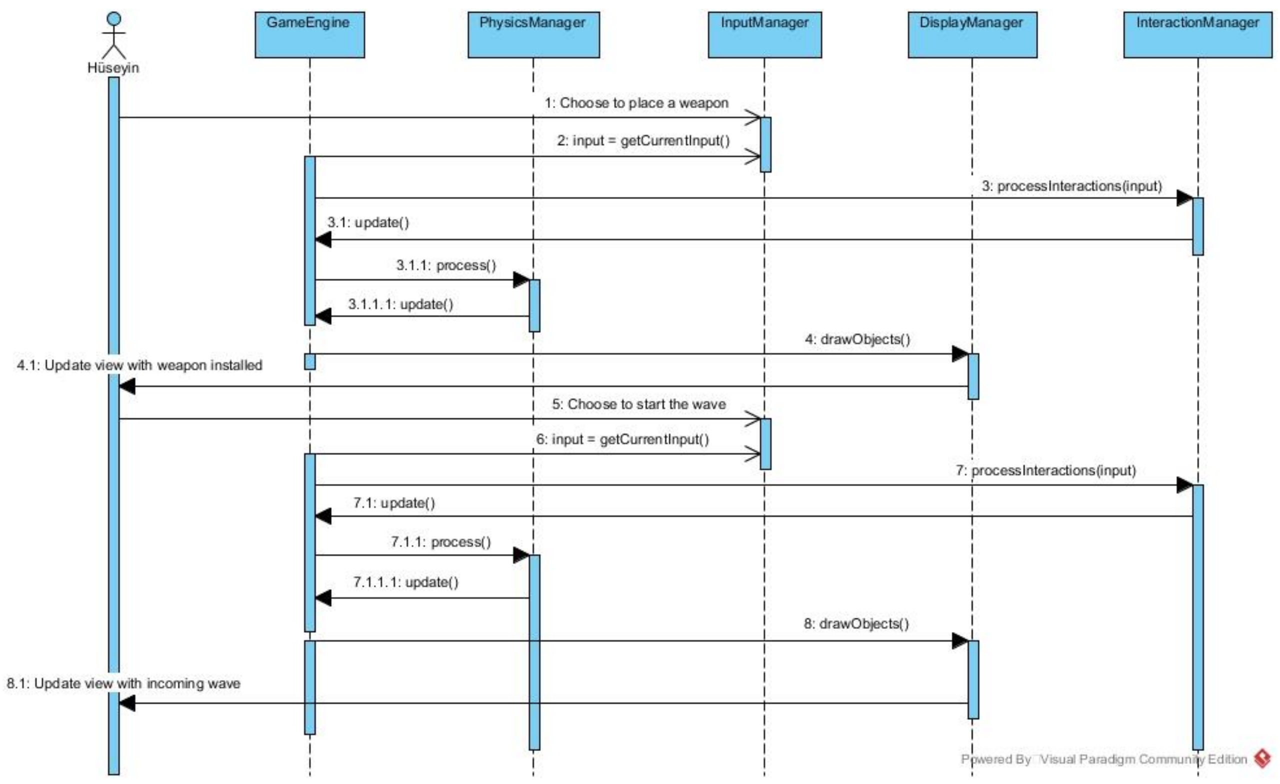
Play Game

Scenario:

After the map selection, player places simple defensive entities with the initial budget, before the first wave of enemies attack. Player eliminates all of the enemies in the first wave. Builds more defense with the money earned during the wave, repeats this process for all waves. After each wave, there will be more and more enemies so it will be harder to defend the tower, eventually some enemies will make it to the tower and give damage to it.

Description:

User clicks on the screen and causes an interrupt which is to be handled by InputManager. Once GameEngine gets the Input from InputManager, it sends it to the InteractionManager which is the responsible class for input interactions with GameObjects. It then updates any correlated GameObject accordingly. In this case Player chooses to place a weapon and once the weapo is placed on the map, Player chooses to start the wave. Input is handled with the same procedure but this time of course interpreted and resulted in an incoming wave.


Hüseyin
GameEngine
PhysicsManager
InputManager
DisplayManager
InteractionManager
1: Choose to place a weapon
2: input = getCurrentInput()
3: processInteractions(input)
3.1: update()
3.1.1: process()
3.1.1.1: update()
4: drawObjects()
4.1: Update view with weapon installed
5: Choose to start the wave
6: input = getCurrentInput()
7: processInteractions(input)
7.1: update()
7.1.1: process()
7.1.1.1: update()
8: drawObjects()
8.1: Update view with incoming wave
Powered By Visual Paradigm Community Edition

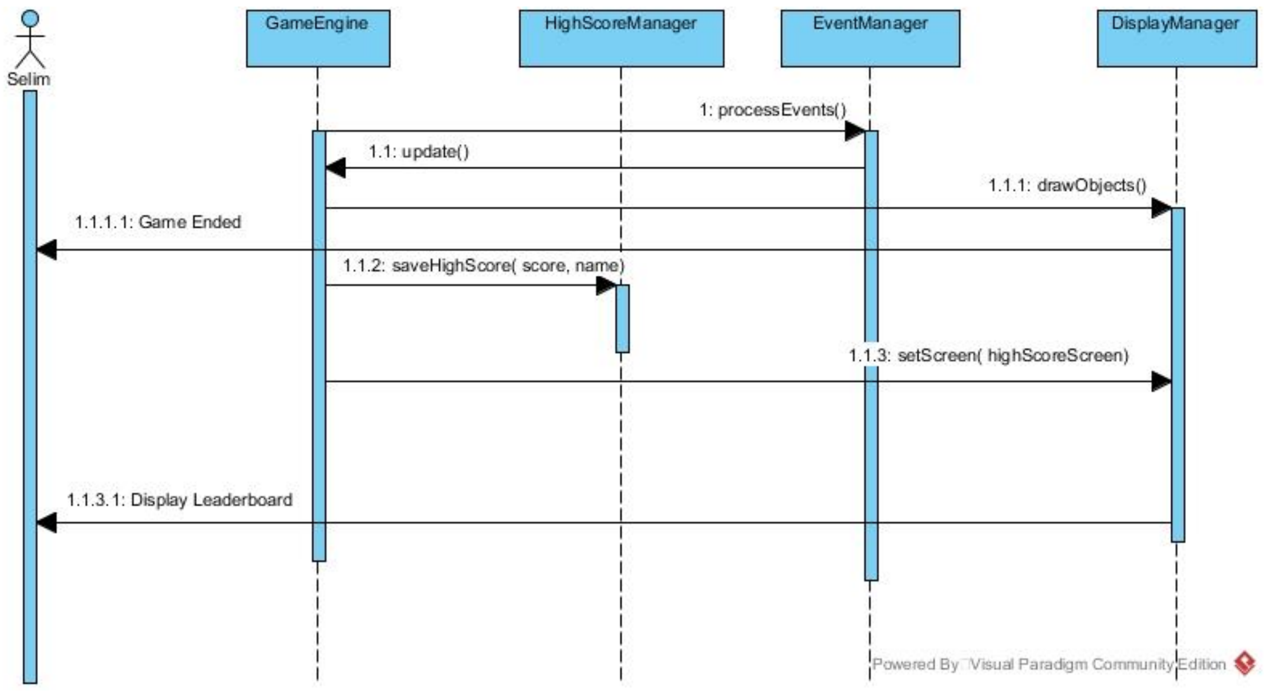
Game Ended - Enter Highscore

Scenario:

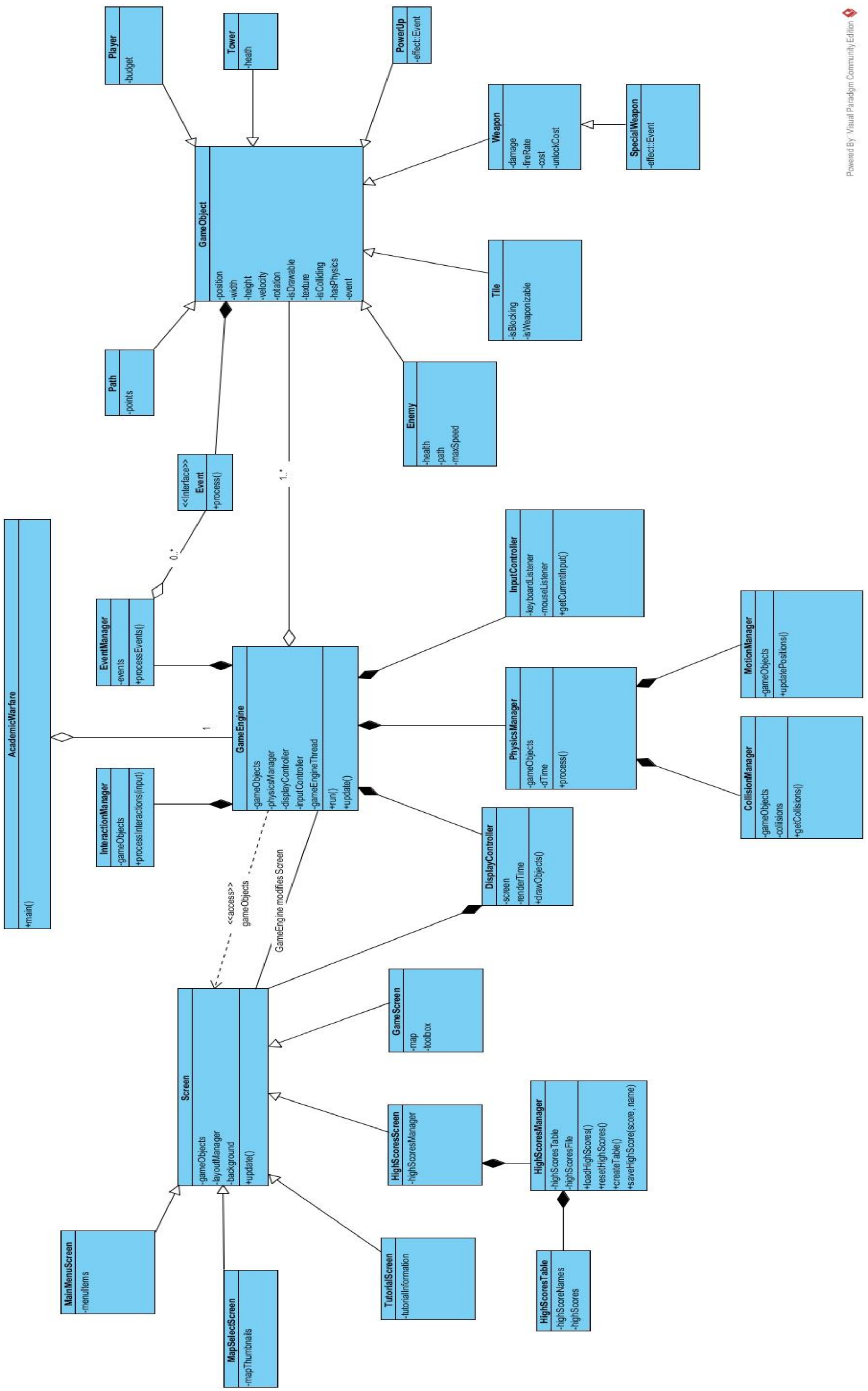
In time, tower will get more and more damage. Once the tower gets enough damage, the game will end. The score at the time when the tower is destroyed will be the final score.

Description:

At the end-game time, GameEngine will receive end-game message from the EventManager once GameEngine asked for what is happening on the game, to the EventManager. The end-game will cause proper end-game message to be displayed on the screen, to the Player. GameEngine will then call saveHighScore routine from HighScoreManager and everything else will be handled inside HighScoreManager. After saving high score, high score screen will be the current screen being displayed.


Selim
GameEngine
HighScoreManager
EventManager
DisplayManager
1: processEvents()
1.1: update()
1.1.1: drawObjects()
1.1.1.1: Game Ended
1.1.2: saveHighScore(score, name)
1.1.3: setScreen(highScoreScreen)
1.1.3.1: Display Leaderboard
Powered By Visual Paradigm Community Edition

Class and Object Models



Below are the descriptions of crucial classes of the Academic Warfare game.

AcademicWarfare class:

Main class to invoke execution of the GameEngine.

GameEngine class:

The central class where all game-dynamics and interactions are coordianted.

PhysicsManager class:

One of the core components of the GameEngine, enables physical interaction such as movements and collisions. Physics manager is composed of CollisionManager and MotionManager classes.

DisplayController class:

Again one of the most crucial GameEngine component. All of the graphics generation and rendering are handled through this class.

InputController class:

Provides user input information to the GameEngine.

InteractionManager class:

Handles user input related updates on the GameEngine.

Event interface:

There will be various kinds of events which provide different functionalities on different occasions.

EventManager class:

Provides event related interactions with the GameEngine.

GameObject class:

Every game-related object will inherit from GameObject class. It provides fundamental properties that each GameObject may require.

Screen class:

Provides general functionalities for various kinds of Screen objects that are going to be presented visually.

HighScoreManager class:

Handles leaderboard related functionalities such as saving into or loading from high score file.

HighScoreTable class:

Provides sorted high-scores to the HighScoreManager class.

User Interface

Navigational Path

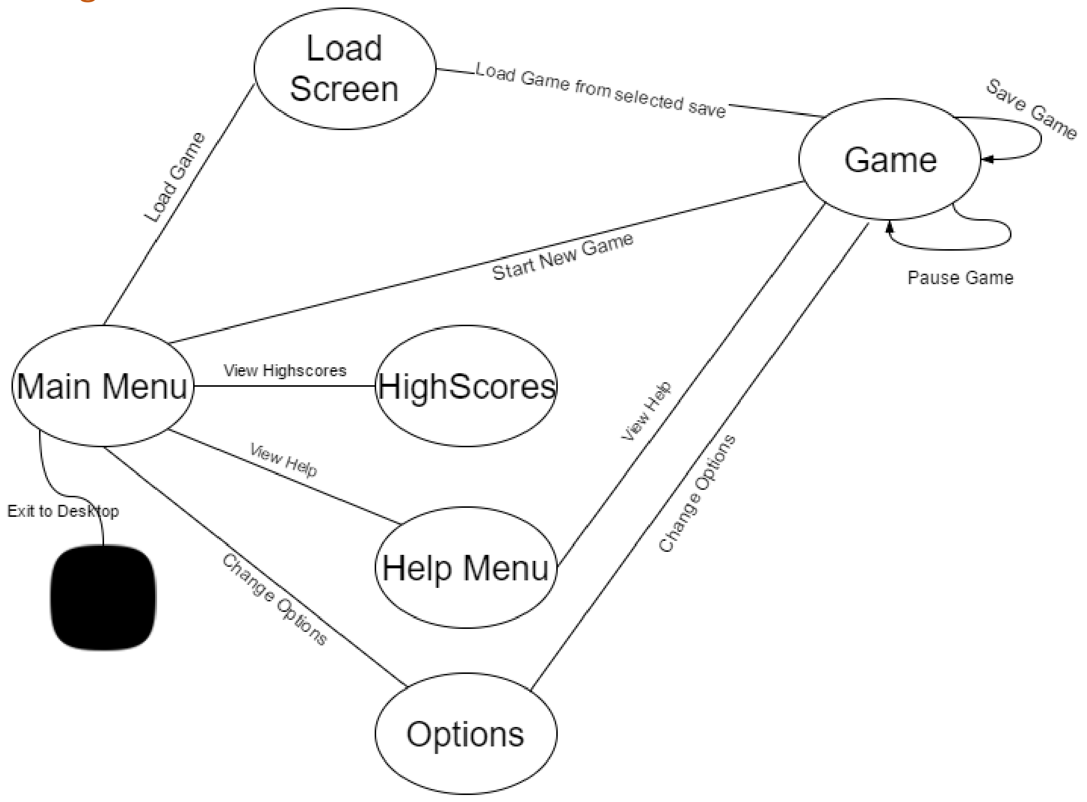


Figure 8: Navigational Path for the game

Mockups

Main Menu

Figure 9 is the template of the screen when the game begins to run. The main menu consists of buttons “New Game”, “Load Game”, “Tutorial”, “Options”, “High Scores”, “Exit” which navigates the user to the desired point.

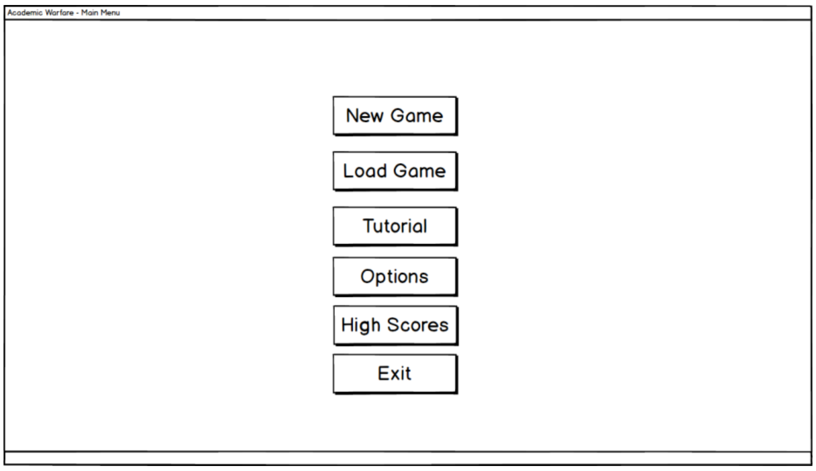


Figure 9: Main Menu

Load Game

In the load game screen shown in Figure 10, the user can select 3 of his previously saved game, and continue to play the saved game or he can return back to the main menu.

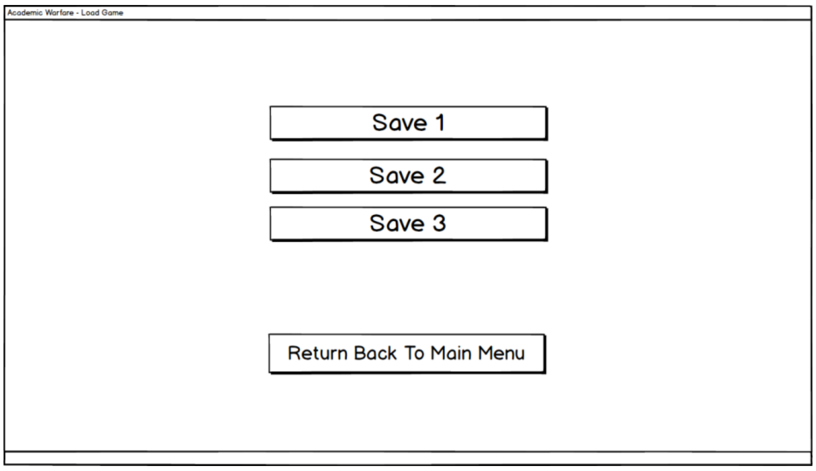
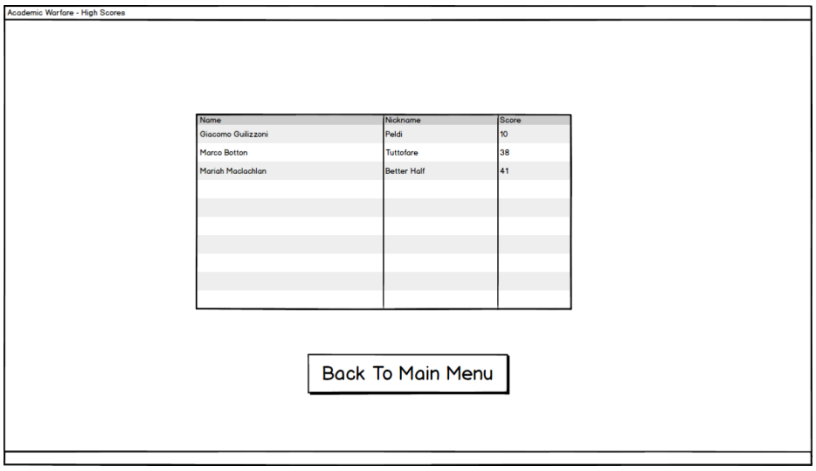


Figure 10: Load Game

Highscore Table

The user can view his/her highscores through this table shown in figure 11. Afterwards, the user can return to the main menu.



Name	Nickname	Score
Giacomo Guilizzoni	Peldi	10
Marco Botton	Tuttofare	38
Mariah Maclachlan	Better Half	41

Figure 11: Highscore Table

Options

The user can change sound options through this menu shown in figure 12 and return to the main menü afterwards.

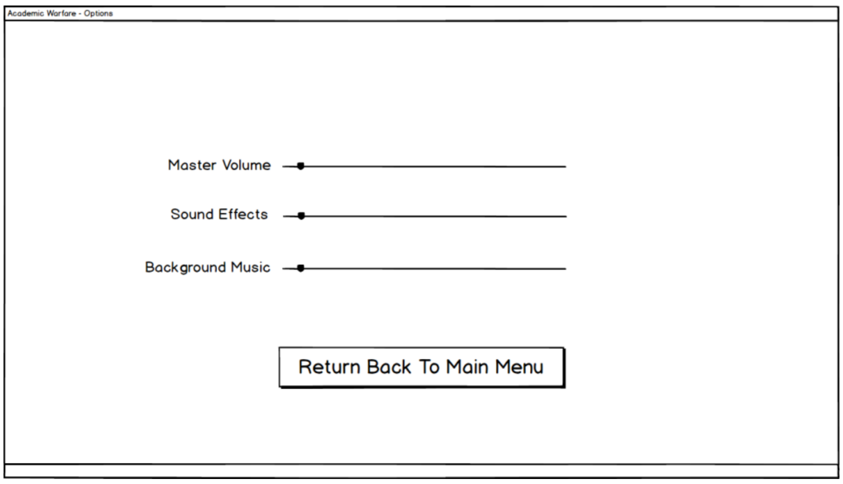


Figure 12: Options

Tutorial – Help

The user can view the tutorial whose template is shown in figure 13 to understand how the game works. The tutorial will consist of multiple pages.

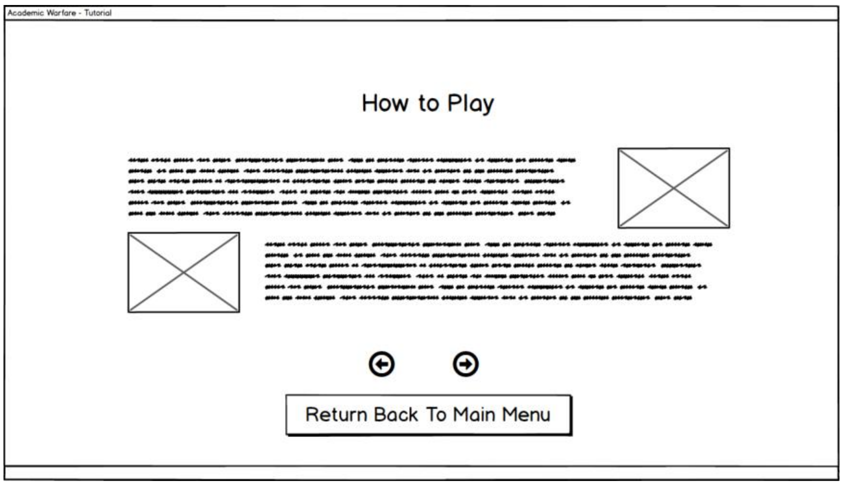


Figure 13: Tutorial

Gameplay

The template for the game play screen is shown in figure 14. In the larger square the game will be played while in the right squares the available weapons, list of used bonuses and stats regarding the wave will be displayed. Also there is a small menü on the upper left corner in which the user can return to the tutorial page, save game, pause and change options.

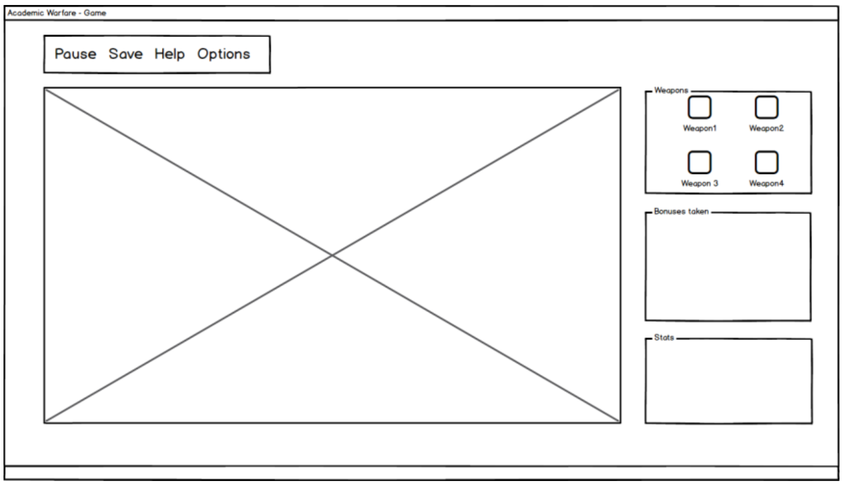


Figure 14: Main Game

References

[1] [Online]. Available: http://ask4asset.com/wp-content/uploads/2015/07/1-680x382.jpg.

[2] Autodesk, "Cannon Large," [Online]. Available: http://static.spark.autodesk.com/2012/08/15__06_09_43/14054_Top.png0e6cdf43-b612-4271-bbf0-ed8b281129bdLarge.jpg.

[3] Renegade Dynamics, "Double 105mm Cannon Turret," [Online]. Available: http://www.sharecg.com/images/medium/17142.gif.

[4] Pinterest, "Scale Models and Miniatures," [Online]. Available: https://s-media-cache-ak0.pinimg.com/564x/68/04/d4/6804d4943f4aea72e054959c2549b6a6.jpg.

[5] Pinterest, [Online]. Available: https://s-media-cache-ak0.pinimg.com/originals/fb/f4/2a/fbf42aec11598d2b8fe9606a8db5f33c.png.