|  |
| --- |
| Bilkent University |
| Computer Science Department |
| CS 319 - Object-Oriented Software Engineering |



Analysis Report

Project Name: Protect the Factory

|  |
| --- |
| Group Members: Ahmet Ay, Alper Eroğlu, Tolga İlikli, Kaan Yorgancıoğlu  Instructor: Dr. Hüseyin Özgür TAN |
|  |

**Table of Contents**

1. Introduction ………………………………………………………………………………….. 2
2. Requirements Analysis …………………………………………………………………. 2
   1. Overview …………………………………………………………………………... 2
      1. Evil Machines ………………………………………………………………. 3
      2. Towers ………………………………………………………………………… 4
      3. Workshops ……………………………………………………………….…. 4
      4. Factory ……………………………………………………………………..…. 5
      5. Spare - Parts ……………………………………………………………..…. 5
   2. Functional Requirements …………………………………………………… 6
   3. Non-Functional Requirements …………………………………………… 7
   4. Constraints ………………………………………………………………………… 7
   5. Scenarios …………………………………………………………………………… 7
      1. Play Game ……………………………………………………………………. 7
      2. Put in Item …………………………………………………………………... 8
   6. Use-Case Models ……………………………………………………………….. 9
      1. PlayGame …………………………………………………………………….. 9
      2. Mute …………………………………………………………………………… 10
      3. PauseGame ………………………………………………………………… 11
      4. ViewHelp ……………………………………………………………………. 11
      5. ViewCredits ………………………………………………………………… 12
   7. User Interface …………………………………………………………………… 12
      1. Navigational Path ……………………………………………………….. 12
      2. User Interface Mockups ……………………………………………... 13
3. Analysis ………………………………………………………………………………………… 17
   1. Object Model ……………………………………………………………………. 17
      1. Domain Lexicon ………………………………………………………….. 17
      2. Class Diagrams ……………………………………………………………. 20
   2. Dyanmic Models ………………………………………………………………. 21
      1. State Chart …………………………………………………………………. 21
      2. Activity Diagram …………………………………………………………. 22
      3. Sequence Diagram………………………………………………………. 23
         1. Starting new game with a code ………………………. 23
         2. Muting the game ……………………………………………. 24
         3. Building a new tower ……………………………………… 25
4. Conclusion …………………………………………………………………………………… 25
5. **Introduction**

Protect the Factory is a simple tower defence game. The purpose of the game is to build defensive structures in order to defend against hordes of evil machines marching to the factory. Defensive towers require a resource called “spare parts” to be built. Towers fire projectiles that reduce the health of the evil machines. Workshops produce the “spare parts” that will be used in the building and repairing of the towers. The player must place these structures on the lanes that the evil machines are using to reach the factory.

As the player progresses in the game, the levels become harder to complete with difficult enemies. However, the quality and power of the player’s defensive structures also increases with increasing levels. The game is a desktop application.

Protect the Factory is inspired by the game “Plants vs Zombies” (2009) by PopCap Games.

Plants vs Zombies:

<http://www.popcap.com/games/plants-vs-zombies>

1. **Requirement Analysis**
   1. **Overview**

Protect the Factory is a tower defense game like “Plants vs Zombies” by PopCap Games. In the game, the player must protect the factories from evil machines that are attacking it. Like the games of the same genre, the player must place obstacles to accomplish this task. The level is won when all the evil machines are destroyed by the obstacles before they reach the factory.

The game consists of 10 levels which become harder to complete as the player progresses. Also, the obstacles and towers the player can place in the way of the evil machines become more powerful and harder to destroy. There are also small workshops in the game that are placed as well that produce spare-parts for the towers. These help the player to repair his damaged towers.

There are 5 horizontal lanes that the evil machines attack from and the obstacles are also placed in these lanes. Each tower is assigned to destroy the machines that are attacking from its lane. The workshops are also placed on these lanes and provide spare-parts for the towers that are in the same lane. The towers and the workshops are placed by the player to the desired location by drag-drop. The player chooses the item he wants to place by choosing the item from the small menu above and drags it to its location. As the player progresses in the game new items are applicable for him to choose.

The player can also keep on playing from his latest level. After each completed level, he is given a code that indicates he has completed the previous level. At a later time when he starts playing, he can enter the code and choose his level to continue where he left. The player can replay the levels he had passed after completing that level or by choosing it later on with the appropriate code.

* + 1. **Evil Machines**

Evil Machines, are the antagonists of the game. Their task is to destroy the towers, workshops and the factory. In each level their type changes and they become stronger. Moreover, with each level the amount of evil machines that are attacking increases. They have a value associated with them that indicates how much life they are left with. They also have a damage value that shows the impact of their attacks on the towers, workshops and the factory. The types of the machines are as follows:

* Code 1: This is the least powerful type of machine that is seen in each of the levels. It has 20 life points and 20 damage points.
* Code 2: This is seen after level 3. It has 50 life points and 50 damage points.
* Code 3: This is seen after level 6. It has 100 life points and 100 damage points.
  + 1. **Towers**

Towers are one of the items that can be placed on the lanes by the player. They are assigned to protect the factory from the attackers. They fire projectiles and try to decrease the life points of the evil machines. The towers also have life and damage points. If the evil machines attack, their life points decreases. As the player progresses in the game, new tower types are opened to him which are more powerful. In order to place a tower, the user must have enough spare-parts that are produced by the workshops and also energy that is produced by the factory that is being protected. The spare-parts also allow the player to repair his towers when they are damaged. The types of towers are as follows:

* Turret 1: This is the least powerful type of tower. It is open from the beginning. No energy is required for it. It only requires 1 spare-part to be built. It has 10 life points and 10 damage points.
* Turret 2: This is seen after level 2. 20 units of energy and 5 spare-parts are required to build it. It has 50 life points and 50 damage points.
* Turret 3: This is seen after level 5. 50 units of energy and 10 spare-parts are required to build it. It has 200 life points and 200 damage points.
  + 1. **Workshops**

Workshops are another type of items that can be placed on the lanes by the player. The player uses them to produce spare-parts. The spare-parts are later used to build and repair the towers. They cannot damage the evil machines, however the evil machines must destroy them to continue in the lane. They have life points like towers that indicate how long they can keep up with the attacks. They also have spare-part points that show how much spare-parts they can produce in 30 seconds of time. As the levels increase more durable and efficient workshops are opened for the players use. The player must have enough energy units to build these workshops. The types of workshops are as follows:

* Local Shop: This is the least efficient and durable workshop. It is open from the beginning. No energy is required for it. It has 10 life points and 1 spare-part points.
* National Shop: This is seen after level 3. 10 units of energy are required to build it. It has 50 life points and 10 spare-part points.
* Global Shop: This is seen after level 7. 20 units of energy are required to build it. It has 100 life points and 20 spare-part points.
  + 1. **Factory**

The factory is the main item of the game. All of the lanes end up at the factory and the objective of the player is to protect it against the attackers. It produces a certain amount of energy that allows the player to build items. It has 200 life points that decrease when the evil machines come near it. In start of each level, the life points is reset to 200. If the factory is destroyed the game ends and the player loses. However, if he is able to stop all attackers he progresses to the next level. The amount of energy produced by the factory increases in each level. The factory begins with 10 energy points in the first level of the game. The energy producing points are increased to the double of the old value in each level.

* + 1. **Spare-Part**

The spare-parts are produced by the workshops that allow towers to be built and repaired. The amount of spare parts are located in the above menu. When a spare-part is produced the amount on the menu changes.

* 1. **Functional Requirements**
* Player can start a new game from the 1st level.
* Player can continue the game from the last level he completed with the required code.
* Player can replay an old level he completed by choosing it from the levels menu and providing the appropriate code of that level.
* Player can get a code after completing a level that indicates he can play the next level.
* Player can mute the game’s sounds by choosing the mute option from the 1st menu.
* The game will have 5 horizontal lanes that the items can be placed and the evil machines can move.
* The towers will hit the closest evil machine in the same lane.
* The workshops will provide spare-parts for building and repairing the game.
* Player cannot play the level if he cannot provide the appropriate code.
* Player can replay a level if he has lost in that level.
* The player can choose to quit the game after he has lost or completed a level.
* The player can choose the pause option in the game menu and restart the level, quit the game or resume playing the game.
* The game will give the option of continuing into the next level if the player has completed the current level.
* The player can choose Help option from the main menu and documentation about the game will be shown.
* The player can choose Credits option from the main menu and the developers’ name will be seen.
  1. **Non-Functional Requirements**
* The game will have a user-friendly interface. The menus will be as simple as possible and the in-game menu will be easy to use so that the player is not distracted from the game by the complexity of the menu.
* The game will give 30 seconds before the first enemy appears in each level.
* The game will have smooth graphics. The actions of the projectiles fired and also the movement of the evil machines will be similar to real life movements.
* The game will be extendable to have different difficulty levels in the future. The evil machines will come faster or more powerful machines will come if the difficulty is increased.
* The game will be extendable to be reused as a mobile game.
  1. **Constraints**
* The game will be implemented in Java.
* The game will use jdk 7.0.
* The game will be written in the eclipse luna IDE.
* The game will not be developed with frameworks.
  1. **Scenarios**
     1. **Play Game**

Scenario 1: Player chooses the “Play Game” option from the Main Menu. Player chooses “Start New Game” from the next menu in order to start from the 1st level. System initializes the screen for the game to begin. System starts the game with the appearance of the first evil machine. System continues to update the screen for the game to proceed.

Scenario 2: Player chooses the “Play Game” option from the Main Menu. Player chooses “Choose Level” from the next menu. Player enters the appropriate code to the system, the code will allow him to choose from the levels he passed. System provides the player with a list of levels he can choose from. Player chooses the level he wants to play. System initializes the screen for the game to begin. System starts the game with the appearance of the first evil machine. System continues to update the screen for the game to proceed.

* + 1. **Put in Item**

Scenario 1: Player continues playing the game. Player decides to put a Tower to one of the Lanes. Player chooses a Tower type from the menu above and drags and drops it to the location on the Lane. System updates the screen. System updates the spare-part and energy values on the menu. Tower starts firing projectiles to the nearest Evil Machine.

Scenario 2: Player continues playing the game. Player decides to put a Workshop to one of the Lanes. Player chooses a Workshop type from the menu above and drags and drops it to the location on the Lane. System updates the screen. System updates the energy value on the screen. Workshop starts producing spare-parts.

* 1. **Use-Case Models**

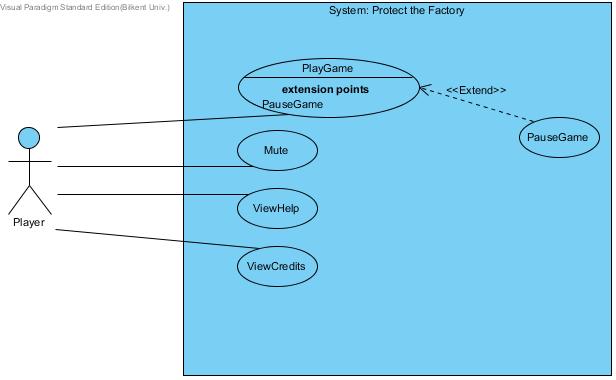


Figure 2.1. Use Case Diagram of Protect the Factory

* + 1. **PlayGame**

Use Case Name: Play Game

Participating Actor: Player

Entry Condition: Player selects “Play Game” from Main Menu.

Exit Condition: Player selects “Quit” from Pause Menu or End of Level Menu to exit to Main Menu.

Flow of Events:

1. Player chooses “Start New Game” option from Menu
2. System initializes the level
3. Player destroys all the Evil Machines in the level
4. System asks if the user wants to continue and gives level code to player
5. Player chooses to continue to next level
6. *Steps 2-5 are repeated for 10 levels unless the Factory is destroyed*
7. System displays End of Game Menu
8. Player chooses “Quit”
9. System goes to Main Menu

Alternative Flows:

* 1. Player chooses “Choose Level” option from Menu
     + - 1. Player enters the code last given to him by the system
         2. System displays levels
         3. Player chooses level
         4. Steps from 2 are repeated

3.1. Player decides to put Tower onto one of the lanes

(a) Player chooses one of the Towers

(b) Player drag and drops the Tower to the desired location on the Lane

(c) System updates energy and spare-parts values

3.2. Player decides to put Workshop onto one of the lanes

(a) Player chooses one of the Workshop

(b) Player drag and drops the Workshop to the desired location on the Lane

(c) System updates energy value

* + 1. **Mute**

Use Case Name: Mute

Participating Actor: Player

Entry Condition: Player is in Main Menu or Pause Menu.

Exit Condition: Player select Mute icon on screen.

Flow of Events:

Player chooses the Mute icon in the screen

System turns of the sounds

* + 1. **PauseGame**

Use Case Name: Pause Game

Participating Actor: Player

Entry Condition: Player is playing the game.

Exit Condition: Player can continue game or quit to Main Menu or restart the level.

Flow of Events:

Player chooses the Pause icon in the game

System shows Pause menu

Player chooses “Continue” option

Player continues playing

Alternative Flows:

3.1. Player chooses “Restart” option

(a) System initializes to start of the level

(b) Player starts playing the game

3.2. Player chooses “Quit” option

(a) System exits to the Main Menu

* + 1. **ViewHelp**

Use Case Name: View Help

Participating Actor: Player

Entry Condition: Player is in the Main Menu.

Exit Condition: Player chooses “Back” option on the screen.

Flow of Events:

Player chooses the “Help” option in the Main Menu

System displays the guide of the game

Player chooses “Back” option

System returns to Main Menu

* + 1. **ViewCredits**

Use Case Name: View Credits

Participating Actor: Player

Entry Condition: Player is in the Main Menu.

Exit Condition: Player chooses “Back” option on the screen.

Flow of Events:

Player chooses the “Credits” option in the Main Menu

System displays the names of the game’s developers

Player chooses “Back” option

System returns to Main Menu

* 1. **User Interface**
     1. **Navigational Path**

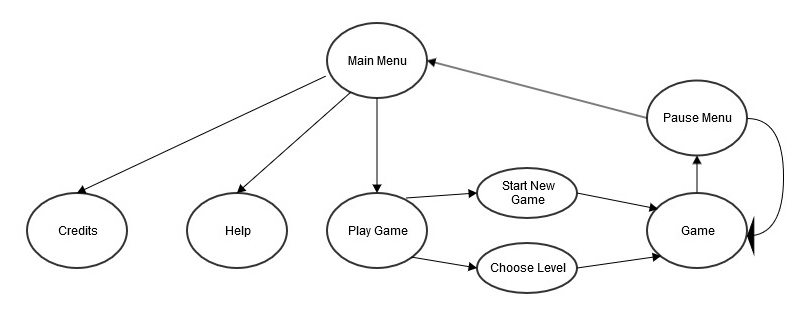


Figure 2.2. The navigational path of the user interface

* + 1. **User Interface Mockups**

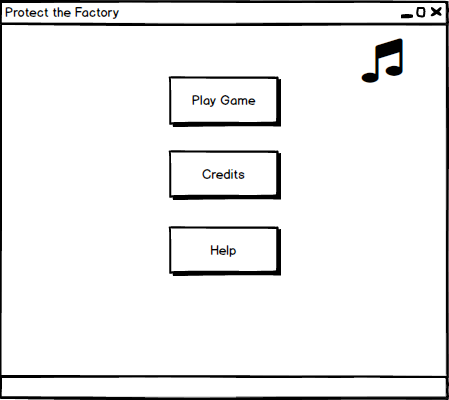


Figure 2.3. The Main Menu of the game

When the player starts the game, the first frame he will see is the Main Menu (Figure 2.3). In this Menu the player has 3 options to proceed. If he chooses the “Play Game” option he will proceed to the Play Game Menu (Figure 2.4). If he chooses “Credits” option he will go to the credits page that contains information about the developers of the game. If he chooses the “Help” option he will see a page that contains information about the game. These information will be about the gameplay, rules and other necessary documentation. In this Menu, the player also has the ability to mute the sounds of the game by pressing the sound symbol.

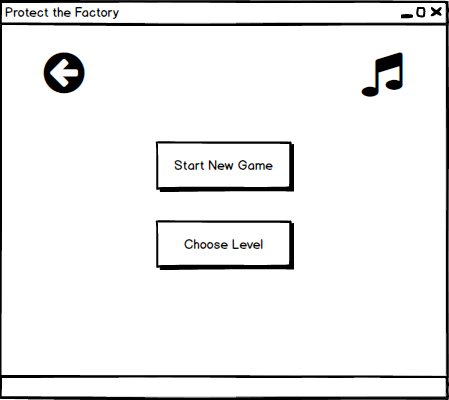


Figure 2.4. The Play Game Menu

In the Play Game Menu, the player again has 3 options and the mute button. If the player chooses the “Start New Game” he will directly start the game (Figure2.7). This option will start the game from the 1st level. If he chooses the “Choose Level” option he will proceed to the 1st Choose Level frame (Figure 2.5). The mute button has the same functionality as the one in the Main Menu. The back button can also be used to return to the Main Menu.

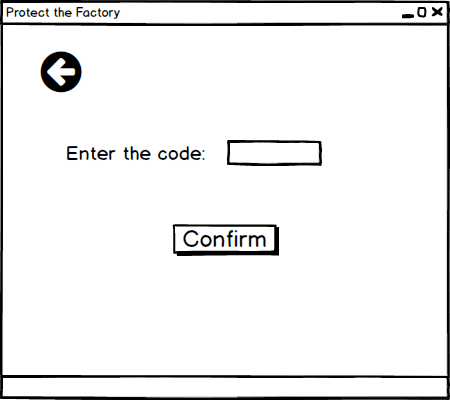


Figure 2.5. The 1st Choose Level frame

In the 1st Choose Level frame the system asks the user for a code. This code is given to the player at the end of a level if he has completed it successfully. He will enter the code to the given blank space. If the code is accepted by the system, the 2nd Choose Level frame (Figure 2.6) will be opened. The player can use the back button to return to the Play Game Menu.

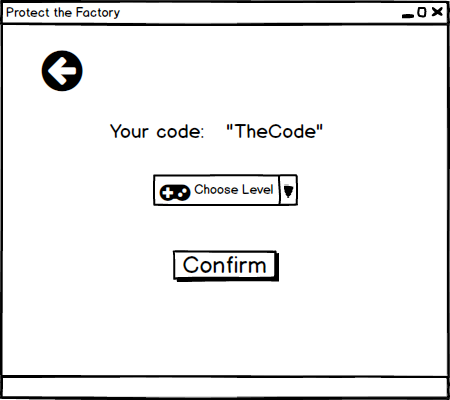


Figure 2.6. The 2nd Choose Level frame

In the 2nd Choose Level frame, the player is given the option of replaying a level that he completed before. The system will open the levels that the player can play in a combobox according to the code entered by the player. The player will choose the level he wants to play from the combobox menu. The system will proceed to the game screen (Figure 2.7). The game will start from the level the player had chosen. The back button in this frame can be used to return to the 1st Choose Level frame, so that the player can enter another code.

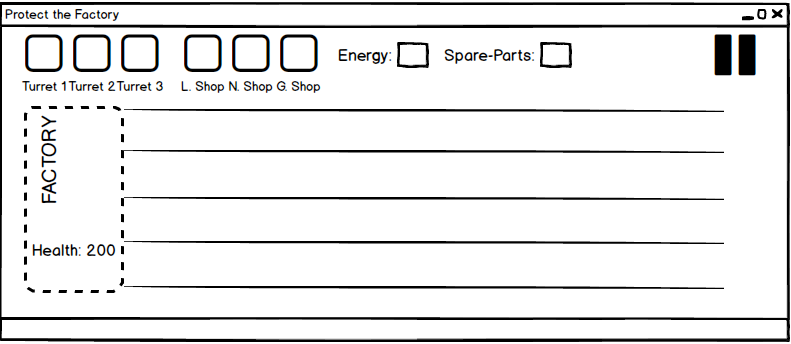


Figure 2.7. The Game screen

The game screen consists of the main game frame and the upper menu. The main game frame contains the Factory, towers, workshops, lanes and the evil machines. The main events of the game are depicted in this frame. The upper menu contains the items that the player can place on the lanes, the status of the energy and the amount of the spare-parts. It also contains the pause game button. When the player presses the pause button the Pause Game Menu (Figure 2.8) will be opened. If the player is able to complete the game with success the End Game Menu (Figure 2.9) will be opened by the system.

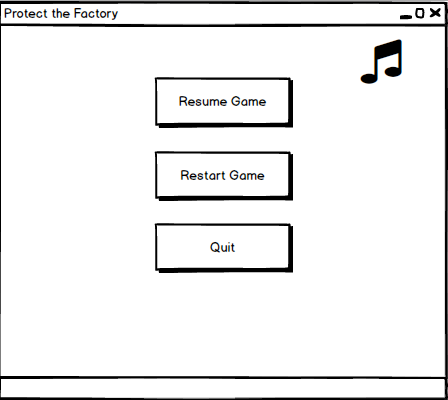


Figure 2.8. The Pause Game Menu

In this screen, the player can continue playing the game from the point he paused by pressing the “Resume Game” option. If he chooses the “Restart Game” option, the system will start the level from scratch. If the player chooses the “Quit” option the game will exit the game and open the Main Menu. The mute button can be used to silence of the sounds of the game.

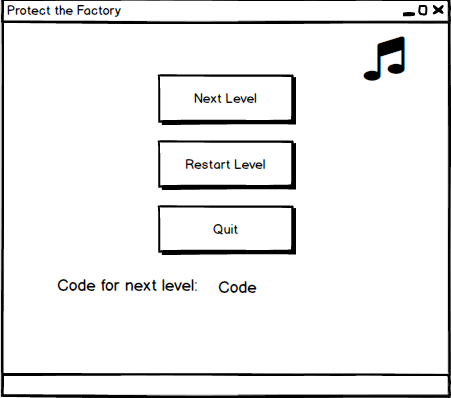


Figure 2.9. The End of Game Menu

In the End of Game Menu, player can choose to move to the next level by choosing the “Next Level” option. This choice will make the system start the game screen for the next level. If the “Restart Level” is chosen, the system will start the completed level from scratch. If the “Quit” option is chosen, the system will open the Main Menu. The next level code will be given in this frame, which will allow the player to play the next one and all previous levels when he enters it in the Choose Level frame.

1. **Analysis**
   1. **Object Model** 
      1. **Domain Lexicon**

* **Spare Parts:** Spare Parts are the main currency in Protect The Factory. All buildings require spare parts to be built. By default there is a slow and steady supply of spare parts but it will not be enough to establish a sufficient defence therefore the player has to build workshops that provide larger quantities of spare parts.
* **Energy**: The factory provides a different amount of energy output for each level depending on the difficulty. All buildings use up some of the energy output when they are up on the field but the energy is refunded when the building is destroyed. The reason that this mechanic exists is to limit the number of buildings that can be built in a level and therefore encourage the player to make more strategic decisions instead of amassing towers.
* **Tiles**: Tiles are squares that make up the field. Only one building can be built on a tile but multiple enemies can be on the same tile.
* **Lanes:** Lanes are the horizontal lines that are made up of tiles. Enemies march along the lanes in the direction of the factory. If an enemy reaches a tile that is occupied by a building it will attack the building until it is destroyed, then it will continue to march. If there is an enemy on a lane that a tower exists, the tower will start to attack the enemy. Towers always attack the enemy that is closest to them.
* **Factory:** The factory is on the leftmost side of the game screen. Each enemy that reaches the factory reduces its hit points for a certain amount. When the hit points of the factory reach zero the player loses the game.
* **Workshop**: Workshops provide the player a steady flow of spare parts.
* **Tower:** Towers attack the enemies on their lane.
  + 1. **Class Diagram**

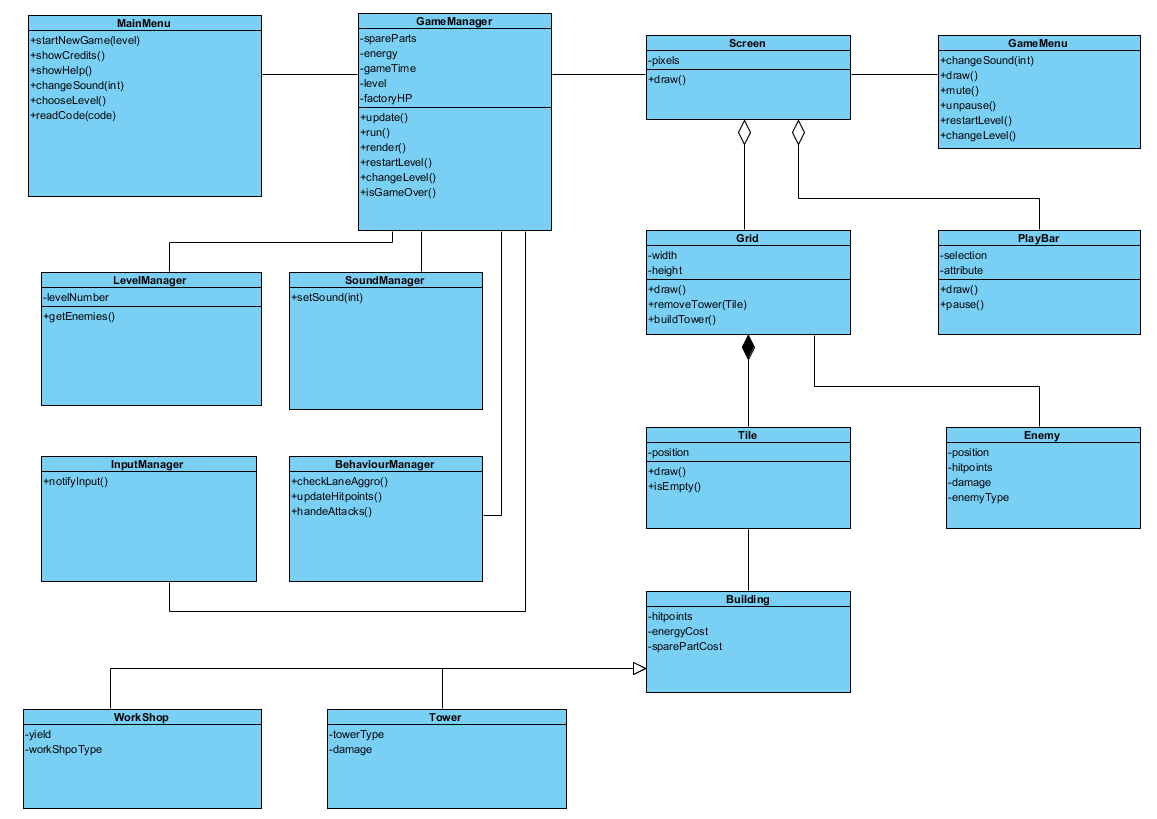
****

Figure3.1. The class diagram of the Game

**Explanations of Important Classes:**

* **MainMenu:** It is the class that handles the decisions that you make before starting the game such as selecting which level to play
* **GameManager:** GameManager is the main class that handles the game. It has many functions such as rendering of the game screen, updating the game objects and keeping track of the resources. It also contains other important manager classes.
* **SoundManager:** This class is responsible of generating all sound effects in the game.
* **LevelManager:** This class handles all mechanics that depend on the level that the user is playing such as when and which lane the enemies are generated.
* **InputManager**: The InputManager class listens to user inputs and notifies the game when the user provides an appropriate input.
* **BehaviorManager**: This class basically handles the behaviour of the towers and the enemies. It checks if there are any towers and enemies on the same lane that might be attacking each other. Then it refreshes the hitpoints of the entities.
* **Screen**: This class contains the elements of the game and it is responsible for helping the GameManager generate the graphics
* **GameMenu**: Also known as Pause Menu. This is the menu that appears when the pause button is pressed. It provides the user with options such as to restart the level. It then notifies the GameManager if the user makes a selection.
* **PlayBar:** This class handles all the actions of the player. It is the main way that the player interacts with the game. It provides options to build buildings, to destroy them and to pause the game. If the user pauses the game it notifies the GameManager and brings up the GameMenu.
* **Grid:** Grid is made up of tiles. It is where the enemies and towers are.
* **Tile:** This class is a container that might be empty or might be occupied by a building.
* **Building:** This class represents the actual buildings in the game. It contains valuable information about the building such as its cost and hitpoints.
* **Workshop:** Extends Building. It contains Information about the type of the workshop and its yield.
* **Tower:** Extends Building. It contains information about the Tower type and its damage.
* **Enemy**: Represents the enemies in the game. It contains Info about its position, type, hitpoints and its damage.
  1. **Dynamic Models**
     1. **State Chart**

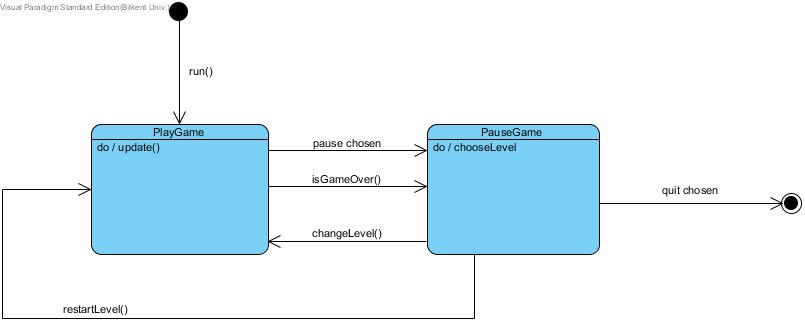


Figure 3.1. The state chart diagram of the GameManager object

GameManager is the most important object in the game. It manages the gameplay and helps the system function. The object has 2 states. When the player wants the game to start, the GameManager goes to the PlayGame state. In this state, the game is played by the player and the graphics are updated. If the game is paused by input of the player, by winning the game or if the game is over, the GameManager goes to the PauseGame state.

In the PauseGame state, the player can choose to restart the level, this will make the GameManager go to PlayGame state. The system will react in a similar way when the player wants to proceed in the game and change the level. The level is chosen in this state. If the player wishes to quit the game the GameManager object will end its states and the system will not require it until further notice.

* + 1. **Activity Diagram**

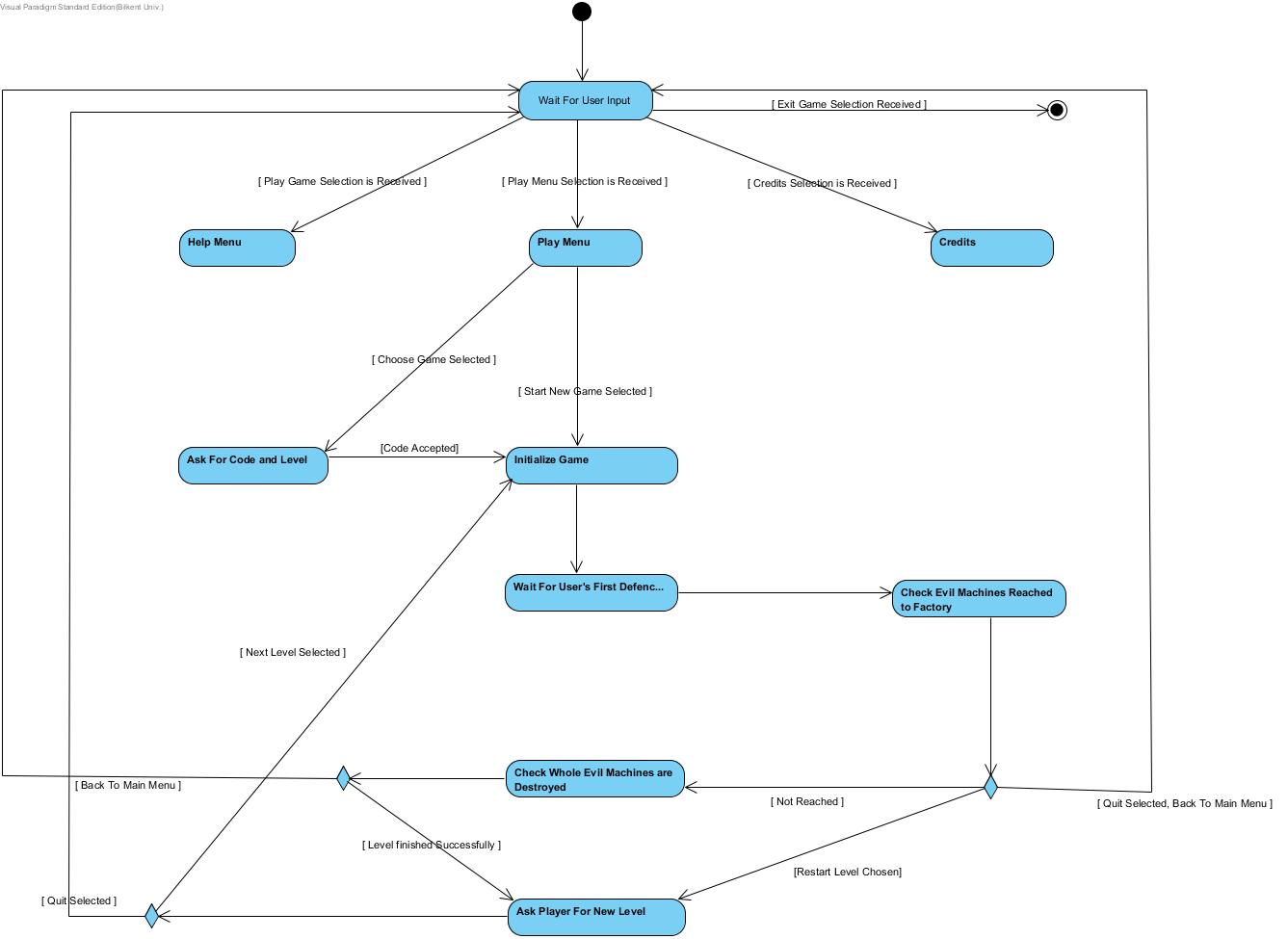


Figure 3.2. The activity diagram of Protect the Factory

When user selected Play Game selection, system redirects user to play game menu. In this menu, system presents 2 different options to user. When user selected Start New Game option, system automatically creates basic game objects and wait for user to place first defensive towers. After that process system starts to create random evil machines to attack towards factory. When towers destroy evil machines, user gets chance to build new towers against evil machines. System consistently checks whether evil machines reached to factory or towers destroyed the whole evil machines in the given time. If evil machines are completely destroyed, system finishes the level and ask user for next level. If any evil machine reaches and destroy the factory, system finishes the game and asks the user if he wishes to restart the game or quit to the main menu.

If the user selects Choose Level option in the Play Game menu, system asks the user for a code. This code is given to the player at the end of a level if he has completed it successfully. User enters the code to the given blank space. If the code is accepted by the system, the system gives the levels he can play to the user. User selects a level. The game is initialised by the system and starts from the level that the player wants to play.

* + 1. **Sequence Diagrams** 
       1. **Starting new game with a code**

In this scenario the player decides to play a specific level. Player presses “choose Level” button. The main Menu asks for the player’s code. Player enters the code. MainMenu displays which levels are available. Player chooses Level 3. LevelManager sets the level to 3. GameManager starts the game.

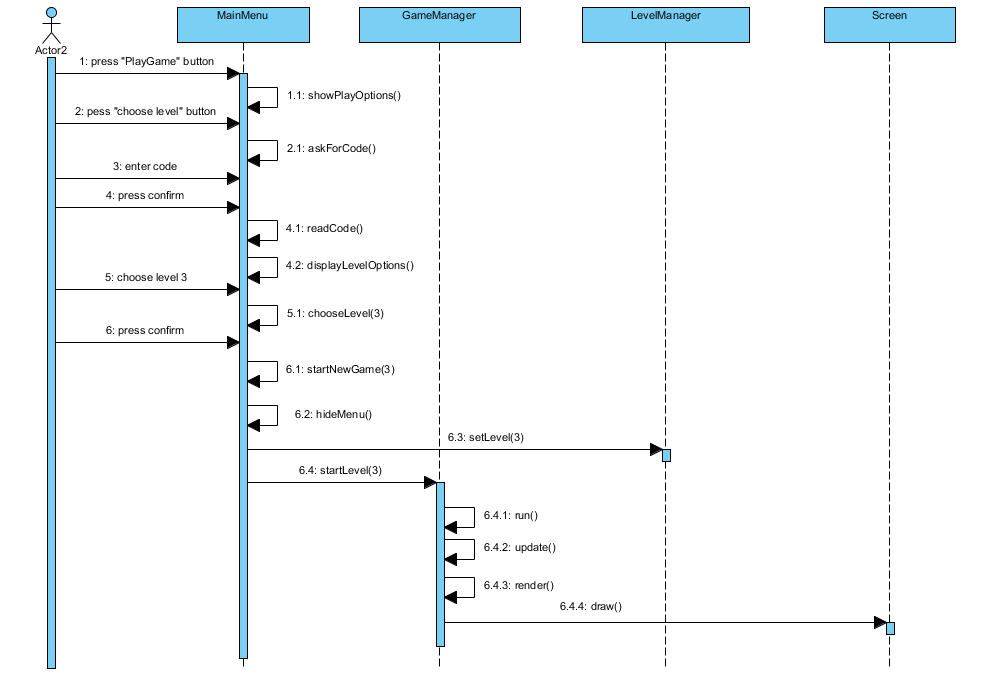
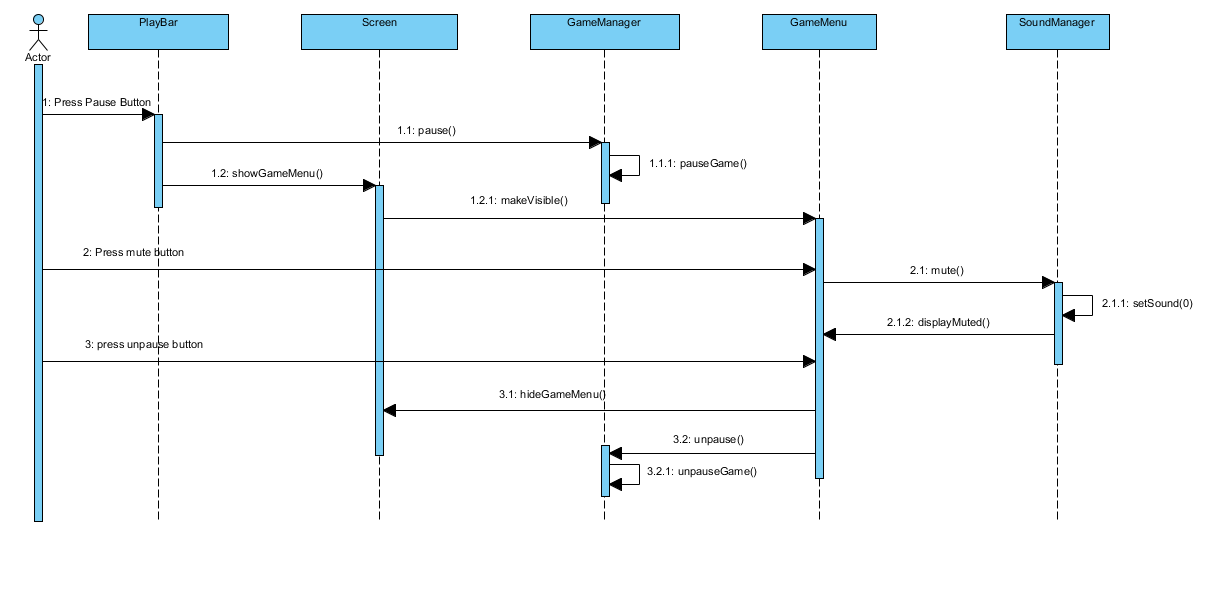
****

Figure 3.3: Sequence Diagram showing the player choosing to play level three with code

* + - 1. **Muting the game**

While playing the game, the player pauses the game and the pause menu pops up. Player presses the mute button. SoundManager is notified. The player unpauses the game.

Figure3.4: The sequence diagram that explains the process of muting the sound of the game.

* + - 1. **Building a new tower**

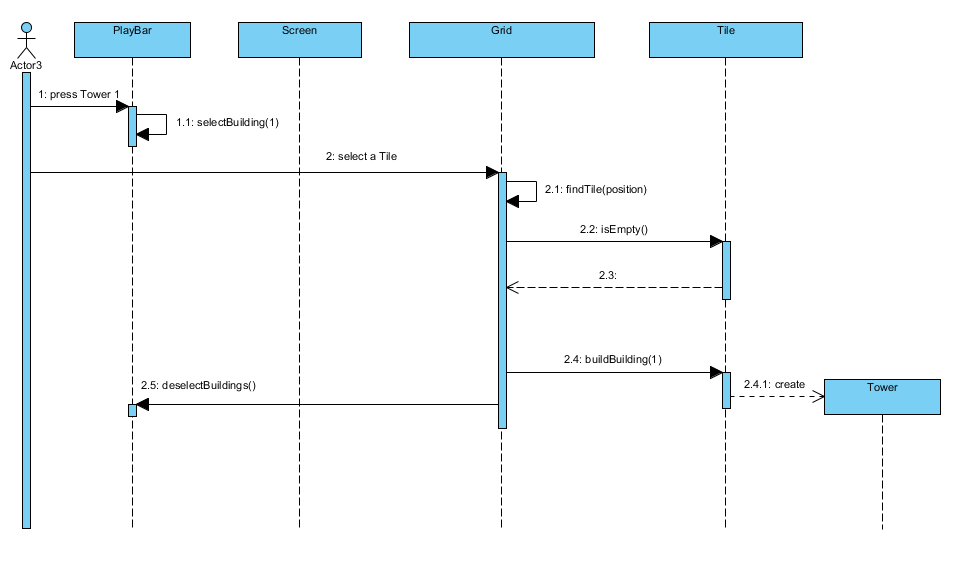
In this scenario the player clicks the tower icon on the play bar. Then he/she clicks a tile on the grid. The Grid Checks if the specific tile is empty then a new tower is created in that tile.

Figure 3.5: Sequence diagram showing the player building a new tower.

1. **Conclusion**

This report is the analysis report of the Protect the Factory game. This report will be highly valuable in the upcoming stages of design and implementation. The report is very important as it will be a guide to the designers and developers in the latter stages. The report concentrates on the requirements and analysis of the project.

In the requirements analysis part of the report, the main topics were the requirements of the project, scenarios of the system, the use cases and the user interface of the game. The functionalities were classified into functional, non-functional and constraints. These requirements are important as they show the capabilities of the system and also give guidance about how it will be implemented. This stage was done thoroughly to ease the work in latter design and implementation stages. The scenarios and use cases show how the system acts under certain conditions. These will be important in the latter stages as well, because all possible actions of the system must be addressed. The interface is intended and designed to be user-friendly. The mockups used are not the exact products of the systems interface, however they give the general concepts and capabilities expected from the interface.

The analysis part of the report is supported by UML diagrams. Dynamic and object diagrams are used in this part of the report. The activity, state-chart, sequence and class diagrams are used in this part. These show how the system acts under certain conditions and how it interacts with the user. The activity diagram will show how the game flows while working and what interactions result in what actions. This will be helpful in constructing the system and its internal operations. The state-chart diagrams show what stages the GameManager, which runs the game, goes through. It will be useful when implementing the actions of the GameManager as it is the most important and vital object in the system. The sequence diagram will be important in constructing the interactions between the actions and methods of objects. Constructing this relation early will help the developers implement the calls better. The class diagram also helps to see the interactions between the classes on an object base. This is also very vital for the implementation phase.

In conclusion, this report is a thorough analysis of the Protect the Factory. The system is analysed deeply so that the latter phases of development are passed with less problems. The latter phases will concentrate on other concepts and they will require a good understanding of the system. This report will deliver that requirement.