|  |
| --- |
| **BILKENT UNIVERSITY**    **COMPUTER ENGINEERING DEPARTMENT**    **CS 319 – OBJECT ORIENTED SOFTWARE ENGINEERING**  **SECTION 03 – GROUP 3D**    **PROJECT TITLE: CASTLE DEFENSE**    **ANALYSIS REPORT**  **DRAFT**  **Team Members**  **Xhoana ALIU 21500429**  **Muhamed KETA 21503560**  **Onur KOCAHAN 21402013**  **Mehmet Can ALTUNTAS** |

[1. Introduction 3](#_Toc498812956)

[2. Proposed System 4](#_Toc498812957)

[2.1 Overview 4](#_Toc498812958)

[2.1.1 Gameplay 4](#_Toc498812959)

[2.1.2 Mapping and Leveling 4](#_Toc498812960)

[2.1.3 Types of Towers and Consumables 5](#_Toc498812961)

[2.2 Functional Requirements 5](#_Toc498812962)

[2.2.1 Play New Game 5](#_Toc498812963)

[2.2.1.2 Play New Game (Additional Requirement) 6](#_Toc498812964)

[2.2.2 Resume Game 6](#_Toc498812965)

[2.2.3 High Scores 7](#_Toc498812966)

[2.2.4 Options 7](#_Toc498812967)

[2.2.5 Tower Store 7](#_Toc498812968)

[2.2.6 Credits 7](#_Toc498812969)

[2.3 Nonfunctional Requirements 7](#_Toc498812970)

[2.3.1 The Animations and User-Friendliness 7](#_Toc498812971)

[2.3.1.1 User-Friendliness and Usability (Additional Requirement) 7](#_Toc498812972)

[2.3.2 Response Time and Game Plot 8](#_Toc498812973)

[2.3.3 Reusability (Additional Requirement) 8](#_Toc498812974)

[2.4 Constraints 8](#_Toc498812975)

[2.5 System Models 9](#_Toc498812976)

[2.5.1 Use Case Model 9](#_Toc498812977)

[2.5.2 Object Model 12](#_Toc498812978)

[2.5.2.1 Class Diagrams and Data Dictionary 12](#_Toc498812979)

[2.5.3 Dynamic Models 16](#_Toc498812980)

[2.5.3.1 Sequence diagrams 16](#_Toc498812981)

[2.5.4 User Interface 18](#_Toc498812982)

[4. Improvement Summary 20](#_Toc498812983)

[3. Glossary 20](#_Toc498812984)

[4. Disclaimer 23](#_Toc498812985)

# **1.** **Introduction**

We will implement a desktop application in Java, which is basically a video game called “Tower Defense”. “Castle Defense” will be a moderate version of strategy video games of the subgenre Tower Defense(TD). In this game, the aim is to defend the player's territories, so that the wave of the enemies cannot pass through the gates into the castle.

In the game there will be a castle and its gates. The waves of the creeps will be coming from different directions. In order to stop the creeps entering through the gates of the castle, the player should build towers along the roads. Each time an enemy is killed the player gains points and each time a wave is defeated the player gains coins which s/he can use later to buy towers.

There will be different types of towers and creeps. The towers will be upgradable. There will be boss creeps towards the end of the game. The boss creeps will be bigger, stronger and consequently harder to be defeated. There will also be some consumables like special powers which the player can use to defeat the creeps. The game will be set on a map which will be automatically generated so the player can play a different game each time s/he wants to play “Castle Defense”. The game will offer three levels: Easy, Normal and Hard. Depending on the level the player chooses, the number of waves and creeps will vary.

The player will have a different number of health pool depending on the level the player has chosen. Whenever a creep manages to pass through the gates, the player loses health pools depending on the strength of the creep that passed the gate. In order to win the game, the player should not lose all her/his health pools.

The high scores of the player will be recorded and the player can see her/his previous scores. The game will be played by using mouse in order to set the towers to defend the castle, use consumables and every other action inside the game. Additionally the keyboard will be used for the purpose of entering the player’s name into the high scores table.

# **2.** **Proposed System**

## **2.1** **Overview**

“Castle Defense” will be a Desktop video game. The game will be a reincarnation of the “Towers Defense” type of games. In this game the player will have a beautiful castle to defend. There will be some different kind of creeps that will try to enter the gates of the castle to invade and destroy it. In order to defend the castle, the player will have to put towers that have different characteristics, such as poison the creeps or slow them down, around the castle and along the roads. To help with her/his mission, the player will be able to use some other power ups of her/his own which we will put under “Consumables” name. The towers can be bought, sold and upgraded. There will be different types of creeps that will vary in their strength, speed and health meter. The game will have three levels; Easy, Normal and Hard, and the user will get to choose which level s/he would want to play. The towers will be bought, sold or upgraded. The player can buy or upgrade her/his towers by using the coins s/he earns after defeating waves of creeps. The aim of the player will be to not let creeps pass through the gates of the castle. The player will lose health pools for each creep that passes through the gates of the castles. If the player loses all her/his health pools s/he will lose the game.

### **2.1.1** **Gameplay**

The players use the mouse to play the game. Mouse will basically be used for using all the functionalities of the game. The player will navigate through the screens by clicking the appropriate function button on the screen. Moreover, the mouse will be used to set the building spot of the towers. It will also be used in order to let the player use the consumables. Additionally, the keyboard will be used in order to let the user enter her/his name in the “High Scores” table.

### **2.1.2** **Mapping and Leveling**

As stated earlier, the game will be based in Auto-generated map. The map will be randomly generated for each new game of the player. In this way the castle’s position, the gates and the roads where the creeps will pass, will be different every time the player starts a new game. The game will have only three levels; Easy, Normal and Hard. In the beginning of the game, the player is asked to choose her/his level. Additionally, the player can change her/his level in the options menu. The easy level will have less number of creeps and creep waves. The player will have a greater number of lives and the boss creeps at the end of the game will be weaker. However, if the player chooses to play the Hard level, there will be more creeps, more waves and less lives. Moreover, the boss creeps at the end will be much stronger and faster. Depending on the level the player has chosen, the waves that will be defeated will reward the player with more coins and the creeps will reward with more points.

### **2.1.3** **Types of Towers and Consumables**

The towers will have different types and will be upgradable. There will be Basic Towers, Ice Towers, Explosive Towers and Poison Towers. The Basic Towers will shoot arrows to the creeps while the Explosive Towers will shoot bombs. The Ice Towers will slow down each shot creep. The Poison Towers however will shoot the creeps with poison and will be more powerful than the Explosive Towers because they will shoot faster, but the poisoned creeps will be destroyed slower. We will try to make the equilibrate the powers of each type of tower so no specific tower will be the most powerful one and will make the player win the game immediately. Thus, it would take a combination of different types of towers to destroy all the creeps and defend the castle. The towers will be available to be bought on the game’s “Tower Store”. The player can upgrade the towers during the game, by using the coins s/he has collected. There will be three types of Consumables: Glue, Stun and Bomb. The Glue will make the waves move slower for a limited amount of time since it would “glue” them to the ground. The Stun as the name suggests will stun the waves of creeps and the Bomb will damage the area where it is thrown making a considerable damage to the creeps in that particular area. However, the consumables will have a weaker effect on the waves and creeps than the towers.

## 2.2 Functional Requirements

After the splash screen is displayed, the following buttons will be on the main menu of the game.

### **2.2.1** **Play New Game**

When the user chooses to play a new game s/he will be asked to choose the level of the game first. After choosing one of the levels; Easy, Normal or Hard, the player will start the game. A new randomly generated map will be the background map of the game. Before starting the actual game, the player will be given 10-20 seconds of time in order for her/him to set towers s/he can buy with the coins s/he has each start. For Easy level there will be 1000 coins, Normal level 2000 and Hard level 3000 coins. Moreover, the towers the player already has will be saved and also the coins the player gains after each win will be accumulated. After setting the first few towers up and building them (again using coins), the player can continue attacking the waves and the creeps that will come towards the castle to invade it.

There will be different number of waves and creeps according to the level the player chooses to play. Player will have a limited health pool according to the difficulty (25/15/5 for Easy/Medium/Hard respectively). This number will get less whenever some creeps get to the castle. Bosses cost 3 instead of 1 normal health pool.

During the game the player can also use consumables to help in defeating more creeps. There will be a limited number of consumables for each level, however they will be recharged after a certain number of enemies is killed. Towards the end of the game there will be a Boss fight where bigger, stronger and faster creeps come into the play against the castle. They will be harder to be defeated. The user can pause the game anytime by clicking on the pause button on the corner of the game window.

### **2.2.1.2** **Play New Game (Additional Requirement)**

Additionally, the player can save the game s/he is playing and resume it another time by pressing the menu on the toolbar of the window of the game.

### **2.2.2** **Resume Game**

In our game, the player will have the option to resume the game s/he has left unfinished. In case of the player leaves the game without finishing it, her/his progress is saved and the player can continue the game where s/he has left it by pressing the “Resume Game” option on the main menu.

### **2.2.3** **High Scores**

The scores of the user will be recorded in a table and the user can see her/his high scores whenever s/he wants. The High Scores option gives the player the opportunity to see her/his progress in order to encourage her/him to develop the game skills.

### **2.2.4** **Options**

At “Options” menu the player will be able to change the volume of the game since different sounds will be played during different attacks. The player can also change the level s/he wants to play in this menu.

### **2.2.5** **Tower Store**

There is an option called “Tower Store” in the Main Menu. This option will direct the user into a screen where s/he can buy different towers. There will be a variety of towers with special characteristics and they will have different prices. However, the player must have collected enough coins to buy them. Additionally, the player can also sell her/his towers for a lower price.

### **2.2.6** **Credits**

In the Credits screen, the player will see more information regarding the game developers and the game version.

## 2.3 Nonfunctional Requirements

### 2.3.1 The Animations and User-Friendliness

The game graphics will be designed in such a way so the game will look appealing to the eye. Additionally, we will try to make the animations such as firing bombs or poisoning as smooth as we can.

### 2.3.1.1 User-Friendliness and Usability (Additional Requirement)

The user interface will be made as user-friendly as possible so that the game will be more beautiful and enjoyable. We will pay special attention to the User Interface in order make it easier to navigate through the game functions and also to make it look appealing.

### 2.3.2 Response Time and Game Plot

The game will have a low response time, which will enable the player to play without any significant delay. The game plot will be easy to understand and the user can easily learn all the functionalities of the game.

### 2.3.3 Reusability (Additional Requirement)

One of the most important non-functional requirements will be the reusability. We will try to make the code as readable as possible since we want to make use of it in later times in order to develop the game further. Thus, writing the code while keeping in mind that the code will be used in the future will make it easier for us to fulfill this requirement.

## 2.4 Constraints

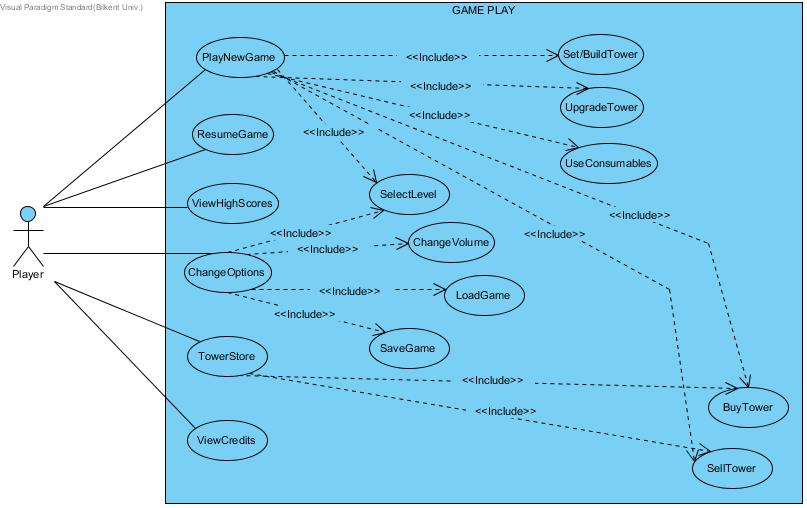
The game must be a Desktop application.

Implementation language must be Java.

Some of the graphics will be made by using Adobe Photoshop.

## 2.5 System Models

### 2.5.1 Use Case Model



Use case #1

Use case name: PlayNewGame

Participating actors: Player

Entry condition: Player has opened the game and s/he is on main menu

Exit condition:

* Player has won the game, OR
* Player has lost all of her/his lives and therefore lost the game, OR
* Player has chosen to exit the game.

Main Flow of Events:

1. Player starts the game.
2. S/he chooses the level s/he wants to play.
3. Player wins the game.
4. “You Won” message is displayed.
5. The points and coins the player has collected during the game are shown and recorded into the High Scores table.
6. The player can choose to “Play Again” and start a new game.
7. The system records the score and modifies the high scores table.
8. Player returns to the main menu.

Alternative Flow of Event:

* Player loses all her/his lives and loses the game.
* “You Lost” message is displayed. (go to step 5)
* At any time, player can exit the game. (go to step 8)

Use case #2

Use case name: ResumeGame

Participating actors: Player

Entry condition: Player has opened the game and s/he is on main menu

Exit condition:

* Player does not any unfinished game the last time s/he has played, OR
* Player has lost all of her/his lives and therefore lost the game, OR
* Player has chosen to exit the game.

Main Flow of Events:

1. Player resumes the game and it starts where it has been left.
2. Player wins the game.
3. “You Won” message is displayed.
4. The points and coins the player has collected during the game are shown and recorded into the High Scores table.
5. The player can choose to “Play Again” and start a new game.
6. The system records the score and modifies the high scores table.
7. Player returns to the main menu.

Alternative Flow of Event:

* Player loses all her/his lives and loses the game.
* “You Lost” message is displayed. (go to step 4)
* At any time, player can exit the game. (go to step 7)

Use case #3

Use case name: ViewHighScores

Participating actors: Player

Entry condition: Player has opened the game and s/he is on main menu

Exit condition:

* Player has chosen to return to the main menu, OR
* Player has chosen to exit the game.

Main Flow of Events:

1. Player views her/his High Scores s/he has set for each of the three levels: Easy, Normal and Hard.
2. Player returns to the main menu.

Alternative Flow of Event:

* Player exits the game.

Use case #4

Use case name: ChangeOptions

Participating actors: Player

Entry condition: Player has opened the game and s/he is on main menu

Exit condition:

* Player has chosen to return to the main menu, OR
* Player has chosen to exit the game.

Main Flow of Events:

1. Player changes the volume of the background noises, OR/AND:
2. Player sets the level of the game: Easy, Normal or Hard. (If the player sets the level of the game via Options menu, then he does not have to choose the level each time s/he opens the game.)
3. Player can Load a previous game left unfinished.
4. Player can Save the current unfinished game
5. Player returns to the main menu.

Alternative Flow of Event:

* Player exits the game.

Use case #5

Use case name: TowerStore

Participating actors: Player

Entry condition: Player has opened the game and s/he is on main menu

Exit condition:

* Player has chosen to return to the main menu, OR
* Player has chosen to exit the game.

Main Flow of Events:

1. Player views the available towers he can buy.
2. Player can sell her/his towers for a lower price.
3. Player returns to the main menu.

Alternative Flow of Event:

* Player does not have enough coins to buy a tower. (A warning sign comes up.)
* Player exits the game.

Use case #6

Use case name: ViewCredits

Participating actors: Player

Entry condition: Player has opened the game and s/he is on main menu

Exit condition:

* Player has chosen to return to the main menu, OR
* Player has chosen to exit the game.

Main Flow of Events:

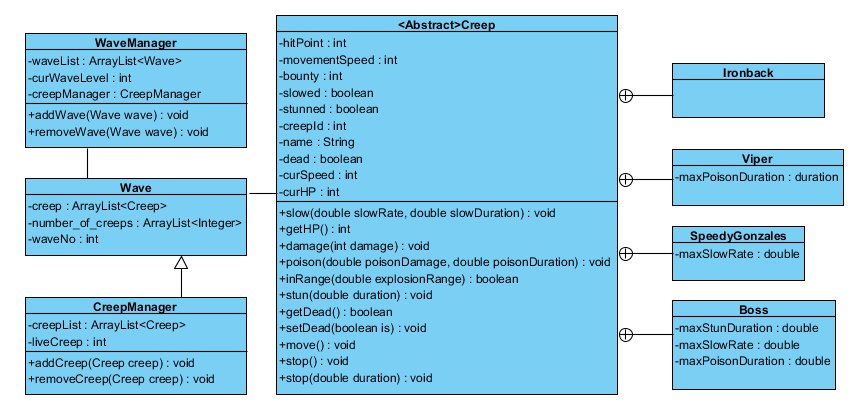
1. Player learns about the game developers and the game version.
2. Player returns to the main menu.

Alternative Flow of Event:

* Player exits the game.

### 2.5.2 Object Model

#### 2.5.2.1 Class Diagrams and Data Dictionary



This classes will be responsible for managing alive, dying, damaged and spawning creeps. Also, responsible for different types of creeps and their specific stats.

**<Abstract>Creep:**

hitPoint: How much hit points the creep has.

bounty: How much gold the user will get when the creep is dead

creepId : the id number of creep in order to there will be no confusion between same type of creeps

movementSpeed: It is the speed of the creep before it is slowed.

name : the name of the creep

dead : controls if the creep is alive or dead

stunned: controls if the creep is in effect of stun.

slowed: controls if the creep is slowed or not in order to not slowing the same creep twice.

curSpeed: current speed of the creep after slowing

curHP: current hit point of the creep

positionX , positionY: position of the creep on the map

**Viper:**

maxPoisonDuration: It shows how long the creep will be under the effect of poison.

**SpeedyGonzales:**

maxSlowRate: It shows how much the creep will be slowed.

**Boss:**

maxSlowRate: It shows how much the creep will be slowed.

maxPoisonDuration: It shows how long the creep will be under the effect of poison.

maxStunDuration: It shows how long the creep will be under the effect of stun.

**Wave:**

creep: list of creeps in the wave. It is parallel arraylist with number\_of\_creeps

number\_of\_creeps: Integer arraylist includes the number of creeps for the wave. It is parallel arraylist with creep.

waveNo: It is unique number of the wave.

**CreepManager:**

creepList : It contains all the creep on current wave.

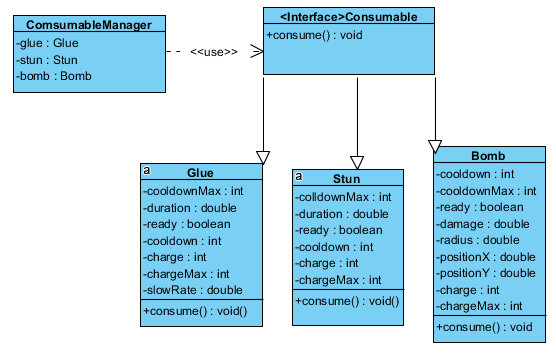
liveCreep: It shows number of alive creep on current wave.

**WaveManager:**

waveList: Contains all the waves in the episode.

curWaveLevel: It shows which wave the player is dealing with.

creepManager: Keep records of the creeps on the current wave.



This is responsible for effects that consumables have and their ‘consumption’.

**Glue:**

cooldownMax: It shows how much creep the player should kill for recharging after using it.

duration: It is duration of the effect of glue(slow).

ready: Shows if the power-up is available for using.

cooldown:It shows how much creep the player should kill to use it again.

charge: It shows how many time the player will use the power-up.

chargeMax:It shows how many time the player could use the power-up in current game.

slowRate: Shows how much the creeps in the wave will be slowed.

**Stun:**

cooldownMax: It shows how much creep the player should kill for recharging after using it.

duration: It is how long the creeps are in the effect of stun.

ready: Shows if the power-up is available for using.

cooldown:It shows how much creep the player should kill to use it again.

charge: It shows how many time the player will use the power-up.

chargeMax:It shows how many time the player could use the power-up in current game.

**Bomb:**

cooldownMax: It shows how much creep the player should kill for recharging after using it.

ready: Shows if the power-up is available for using.

damage: Shows how much damage the creeps in the area of bomb will get.

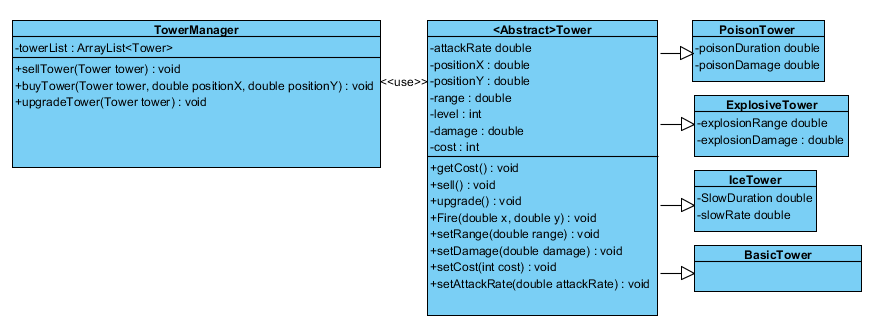
radius: Shows the radius of the explosion.

positionX, positionY: Is the cordinates that the bomb will be used.

cooldown:It shows how much creep the player should kill to use it again.

charge: It shows how many time the player will use the power-up.

chargeMax:It shows how many time the player could use the power-up in current game.



Responsible for tower selling/buying/upgrading and as well for the specifics on different tower types and their level placement.

**<Abstract>Tower:**

attackRate: It is how frequently the tower will be attack to the target creep.

positionX, positionY: It is the cordinates of the tower.

range: It is range of the tower

level: It is level of the tower. If player upgrades the tower, the level, range, damage and attack rate of the tower will be enhanced.

damage: How much damage the creep will get with one shot of tower.

cost: Total invested coin for the tower.

**TowerManager:**

towerList: It keeps the records of the tower in the episode.

**PoisonTower:**

posionDuration: It shows how long the creep will be under effect of poison:

poisonDamage: It show how much damage the creep will be taking per second while under the effect of poison.

**IceTower:**

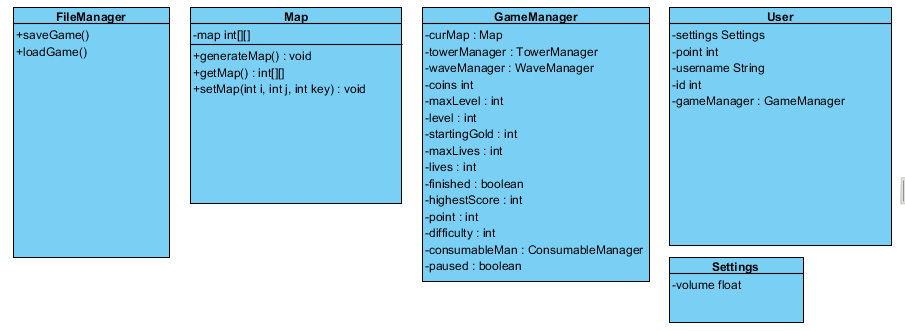
slowDuration: It shows how long the creep will be under effect of slowing.

slowRate: It show how much damage the creep will be slowed while under effect of slowing

**ExplosiveTower:**

explosionRange: It shows the range of explosion when explosive tower fires.

explosionDamage: It shows how much damage the creeps in the area of explosion range will take.



Information on the current state of the game, user settings and information and also the map generator.

**Filemanager:**

saveGame(): It saves game.

loadGame(): It loads last game.

**Map:**

map: It is an integer array for the path of the game.

generateMap(): It generates the map randomly.

**GameManager:**

coins: It shows how much coins the player has during game.

towerManager: It keeps records of the tower during game.

waveManager: It keeps records of all waves in the game and keeps records of the creeps in the current wave.

maxLevel: It is how much level the game has.

level: It shows which level(wave) the player is playing

startingGold: It is how much gold the player will start when s/he is sharting to a new game.

maxLives: It shows how much lives the player will have at the start of the game.

lives: It shows remaining lives the player has.

finished: Is shows if the game is finished or not.

highestScore: It shows the highest score in the game in its own difficulty.

Point: It shows current point of the player in the current game.

difficulty: It shows difficulty of the game.

consumableMan: It contains consumables such as Glue, Stun and Bomb.

paused:It shows if the game is paused or not.

**User:**

point: It shows how much points the player has.

gameManager: It is responsible for the keeping the records of the game.

username: It is username, nickname, of the user.

Id: It is unique id number of the player.

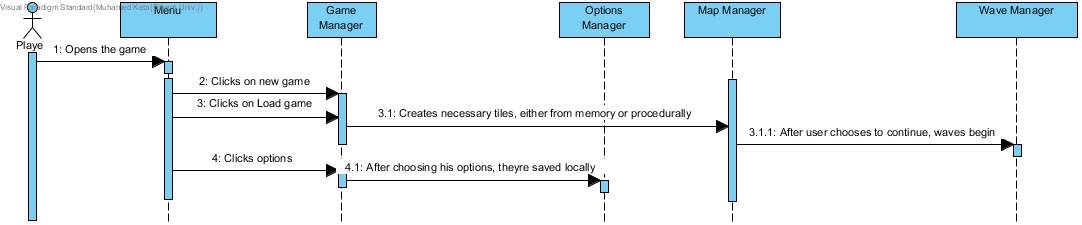
**Settings:**

volume: Shows how louder the game music will be.

### 2.5.3 Dynamic Models

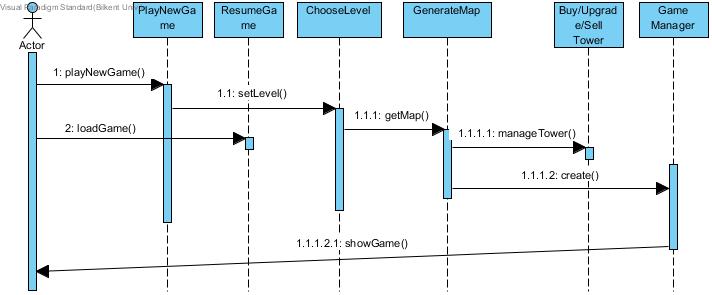
#### 2.5.3.1 Sequence diagrams

**Scenario:**

Player requests to start a new game from Main Menu. System initializes the window and generates the level design procedurally. If he wants at any time he can continue the level, save or load the game, therefore losing the current level. Clicking anywhere in the map will start the wave, continuing the game. 

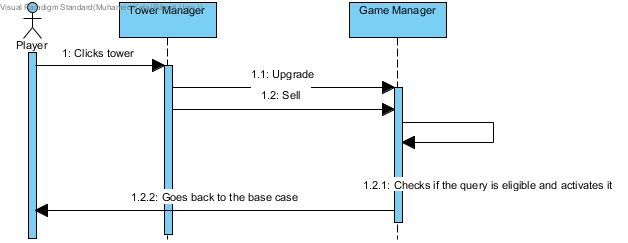
**Scenario:**

Player wants to continue a saved game. He clicks Resume Game from Main Menu and then clicks the respective game he wants. This brings him to the exact map, towers and gold amount he saved, but at the start of the wave (even if he saved mid-wave for example).



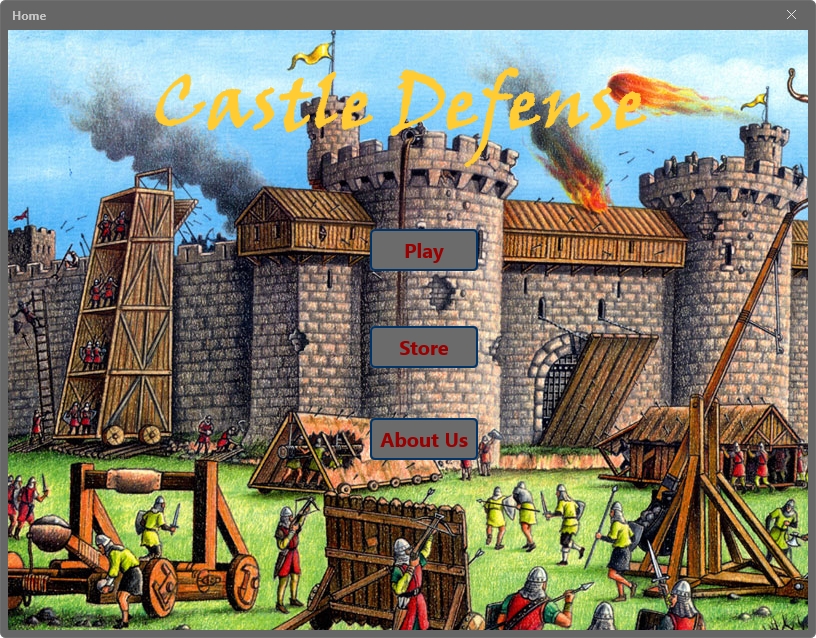
**Scenario:**

Player selects a tower, then he is shown options from which he can upgrade the tower (up to 2 times) increasing the stats according to fixed pre-stated values. He also can sell the tower for 75% of the price it was bought including upgrades. Selecting anywhere in the map (and therefore deselecting the current tower) will logically make the menu unusable.

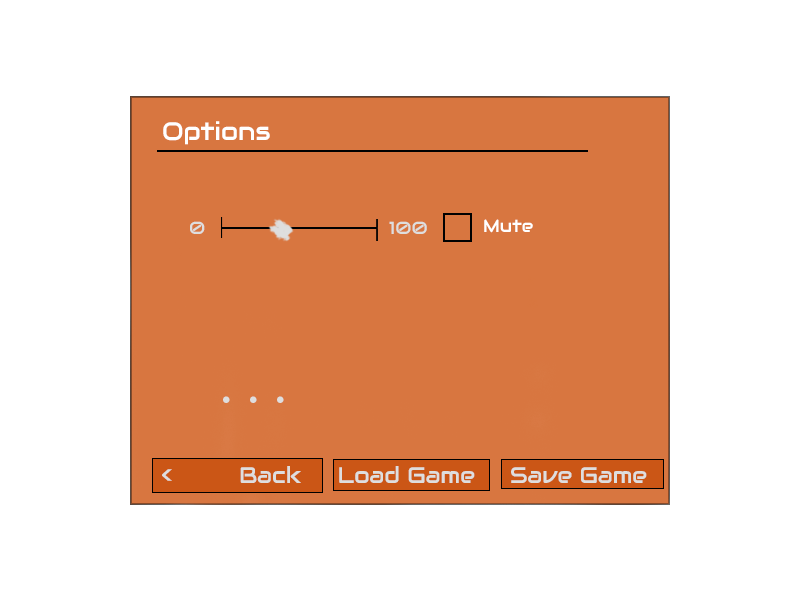


### 2.5.4 User Interface

Main Menu (There will be 6 options as mentioned above.)

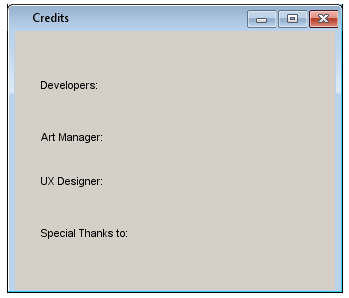


Options Menu



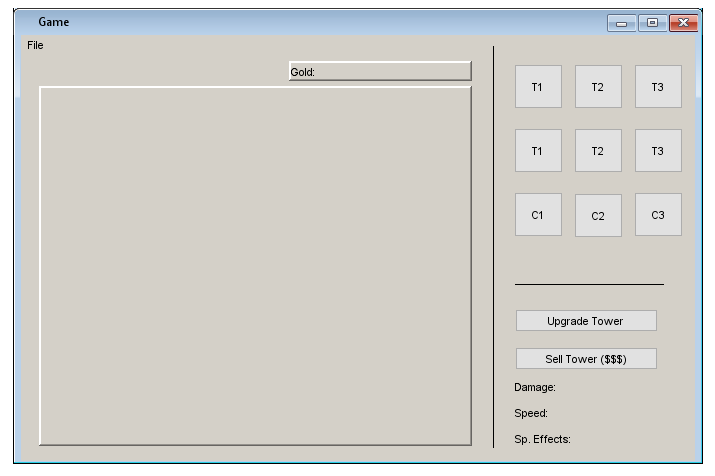
From where the user can Load, Save Games, mute, change the volume and set the level.

Credits:



Here we showcase our developers and people who worked hard on the game and also some brief information about the current version game.

Tower Management



Here the user can buy new towers, consumables, upgrade the towers or sell them. He can also see the stats the towers have currently.

# 4. Improvement Summary

There are not any significant changes on the analysis report of our project. However, there are some additional requirements made in order to improve the quality of the game. We added some functional requirements such as Save and Load the game. These two options now will also be available while the player is playing a new game on the toolbar of the window. Moreover, as non-functional requirements we decided to add User Interface and Usability which will have a high priority since we want for the game to be as appealing and enjoyable as possible. Another added non-functional requirement is the Reusability since we want to work on the project and develop it further, such as make an online version of the game, in the future. We should mention that adding audio effect in the game is not anymore aa must-have but rather a nice-to-have priority requirement.

Note: For the user interface part of this report we have used some mock-up screens and not the actual screens of the game at this moment of developing it.

# 3. Glossary

**Map**: It is generated randomly. The path of the creeps is nearly fixed length.

**Tower class** is an abstract class for the different type of towers which are Poison Tower, Explosive Tower, Ice Tower an Basic Tower. Each tower has common properties that tower class includes.

Common properties of towers:

**Attack Rate:** It is basically attack frequency of tower.

**Position :** On the map , each tower invades a place which is consist of component of x and y. A place can be used by only one tower.

**Range:** Range is how far the tower can fire.

**Damage:** It is how much destruction made by tower by one shot.

**Level:** Towers are upgradable which means by coins you can enhance the properties of towers. Enhanceable properties of towers are its attack rate, its range and its damage

**Selling:** Each tower could be sold. The coins you get by selling is %75 of its cost which equals coins that is invested for the tower.

**Firing:** Each tower fires the closest enemy. The firepower depends on the damage of the tower and its speed depends on attack rate the tower has , as well as its range. By upgrading, since range, damage and attack rate are enhanced, the firing will be more effective.

Types and characteristics of towers

**Basic Tower:** It fires to one enemy when it is fired. (Picture is of the base type, upgrades are different)

**Explosive Tower:** When explosive tower fires, it gives damage to target creep and the creeps that are in the explosion range. The damage given to target creep is depends on damage of the power and the damage given to the creeps that are in the range of explosion range is depends on explosion damage of the tower. (Picture is of the base type, upgrades are different)

**Ice Tower:** When ice tower fires, it slows the target creep for slow duration of the tower and creeps gets slower due to slow rate of the tower. (Picture is of the base type, upgrades are different)

**Poison Tower:** When poison tower fires to the target creep, it gives poison damage per second to the target creep which the poison effect longs due to poison duration of the tower. (Picture is of the base type, upgrades are different)

**Tower Manager:** It is the list of the towers on the map. By using tower manager the user can buy towers, upgrade the towers and sell them.

**Creep class** is an abstract class for different type of creeps. Each type of creep has common properties.

**Hitpoint:** Each creep has hit point. If the creep has higher hitpoint, it is harder to kill.

**Movement Speed:** Each creeps moves due to its movement speed. The higher movement speed the creep has, the faster it moves and harder to kill.

**Bounty:** Each creeps gives coins to the player when it dies.

**Creeps can be slowed, poisoned and stunned.**

There are 4 different type of creeps:

**IronBack:** It has no peculiar property besides the common properties of creeps.

**SpeedyGonzales:** This type of creep can be slowed up to its maximum slow rate.

**Viper:** The poison effect lasts for maximum poison duration of the Viper.

**Boss:** Bosses are harder to kill, which means has higher hitpoint. Also, bosses can be slowed up to its maximum slow rate, be under effect of poison for maximum poison duration of boss and be stunned up to its maximum stun duration.

**Wave:** Each wave will consist of combination of different types of creeps which are different amount.

**Creep Manager:** Keeps record of the alive creeps on current wave.

**Wave Manager:** Wave manager keeps records of the waves for the game. When the wave is cleared, it selects the next wave.

**Consumable:** It is an interface for the consumable power ups, which are glue, bomb and stun.

**Stun:** It stuns the wave up for its stun duration and once it can be used up to its maximum charge times. Once it is used, it must be recharged. For recharging, the player must kill certain number of creeps.

**Glue:** It slows the wave up to its slow rate for its slow duration. It can be used up to its maximum charge times. Once it is used, it must be recharged. For recharging, the player must kill certain number of creeps.

**Bomb:** It makes an explosion to the player specified position. It gives a damage to the enemy within range of the bomb. It can be used up to its maximum charge times. Once it is used, it must be recharged. For recharging, the player must kill certain number of creeps.

**Consumable Manager:** It is used for controlling consumable power ups.

**Game Manager**: Game manager keeps records of the game. It contains tower manager, wave manager , consumable manager, difficulty, coins, points, highscores, remaining lives, and level.

**File Manager** saves the game and user properties.

**User**: User has its game manager to keeping records of unfinished games, current points and highscores. User also has username and its settings.

**Settings**: Settings are used for adjusting music volume.

# 4. Disclaimer

Information on this paper may be subject to change in the later revisions, be it written or drawn.