İhsan Doğramacı Bilkent University

Departmant of Computer Engineering

CS319 Object-Oriented Software Engineering

Project Analysis Report

Group A

Tower Power!

Fall 2014

Özgür Öney	Serkan Demirci	Ahmet Küçük
21101821	21201619	21001069
Section 3	Section 3	Section 3

Table of Contents

Introduction

Tower Power! is a grid based and single player tower defense game. Tower defense is a kind of strategy game that player tries to stop enemies from crossing a map by building traps to slow them down and towers which shoot at them as they pass.

In Tower Power!, players sets traps and build towers in permitted areas in the map and tries to defend his unique castle which stays in the finishing line of the map. Towers that could be built by player have various abilities such as cost, attack damage, range and upgrade cost. And also, traps also have various abilities such as cost, effect area, damage and re-use cost.

On the other hand, enemies that are controlled by system's itself comes from the beginning part of the map wave by wave and tries to pass through the road of the map and reach the castle of player. Enemies in Tower Power! vary in type: Orcs, elfs, trolls and goblins. These four different races not only differs in name and looking but also differ in abilities such as stability, attack range, attack damage and speed. For example, while Orcs are durable, slow; Elfs are going to have an abilitiy attack player's towers from a long range and they also are going to be much faster than Orcs.

In Tower Power!, player will gain points from each enemy that killed and could use these points to either for his tower or his traps. After each wave, player will be able to construct new towers or traps as well as upgrade traps and towers in allowed time. This allowed time is going to decrease while levels are increasing.

Tower Power! consist of 16 levels that not only differ in map design but also differ in toughness of the waves and allowed time periods. The game starts from the level 1, and as player advances in his progress successfully, new levels are unlocked. Also, playing past levels are possible and in order to increase playability of the game, players success in past levels directly effects the situation in current level: remaining gold after all construction in one level transfers from one level to another.

In these report, as Group A in class CS 319 Object-Oriented Software Engineering, we are going to introduce detailed case description of the game Tower Power! as well as requirement analysis and analysis models.

Case Description

Since object-oriented programming is commonly practiced approach in creation of these kind of games, we have choosen to create such a tower defense game. The goal of us as Group A is creating an enjoyable and easy-to-play tower defense game in a sistematic way.

Game Overview

The game is about defending the castle, which could carry maximum ten enemy units. When ten enemy unit reach the castle, the game is over. Enemy units that comes from the beginning of the map try to cross the map and reach the castle. In this situation, player's aim is to defend these "fragile" castle from enemies by constructing towers, setting traps and upgrading these units in allowed time given after end of level.

Only peripheral that the game requires is mouse that is used in constructing and upgrading. Camera is positioned at top of the game world, so the player is accepted as if he looks from the top. The game has saving and loading features which allows the player to continue from where he/she left off. The game will save itself whenever the player finishes a "level".

The game has 16 levels to increase the difficulty as the player progresses. Since a play in one single map and constant enemy units is boring, levels that differ in map design are created. Each level has its own properties and each levels difficulty is different: a path that forces player by making enemy units' job easier, increasing number of units in every wave and changing in races in every single wave.

Also, game can be paused anytime that player wants and player gets no penalty for this. Saving the game at some particular point and loading from this point is also possible. Also, on settings menu, player will be given a chance to activate autosave ability, which gives permission to the system to save the game after each wave and disburdens player in saving.

In the game, a currency is used to determine price of every tower and trap and is named gold. Gold can be gained by killing enemy units and at the end of the every level, extra gold will be added to player's wallet depending on the number of units entered the castle of player. For example, if 4 enemy units could pass through the map and reach the castle, player will get 600 bonus gold at the end of the level but if 5 enemy could pass through the map and reach the castle, this number will be decreased into 500. Gold is used to construct new towers, upgrade the existing towers, set new traps, upgrade the traps and lastly expand the capacity of your castle.

Towers in the game have five different power level: bronze, silver, gold, diamond and Barad-dûr. Each of these towers have different health, attack range, attack speed and attack damage and these abilities increase from bronze tower to Barad-dûr. At the end of each level, player is going to have a chance to upgrade his towers in this order but in order to create the tower Barad-dûr, player also have to reach preconditions such as killing specific number of enemies and construct specific number of defence units. However, price of construction of

towers also increases from bronze to Barad-dûr; while a bronze tower needs 500 gold to be constructed, a Barad-dûr needs 10000 gold to be constructed.

Enemies in the game also varies, there are Orcs, Goblins, Elfs and Trolls. There are six main categories while classifying these races; health, speed, agressiveness, attack range, attack speed and attack damage. Orcs are accepted as durable, slow, soft and melee; Goblins are easy to kill, fast, soft and melee; Elfs are easy to kill, fast, agressive and ranger and lastly Trolls are durable, agressive, fast and ranger. In each wave that comes from the Forgotten Lands, number of these races differs and as could be guessed, number of powerful races will increase while player's level increases.

Lastly, there is a castle of player named as Dol-Guldur. As default, this castle has capacity to hold maximum 10 enemy units, which means that if player becomes unsuccessful to specific number of enemy units and these units reach the castle by passing through the map, player will not be considered as he lost. This castle, Dol-Guldur, also has different abilities to upgrade in exchange for money such as durability which increases the maximum number of enemies to be held and traps that decrease the number of enemies entered by killing them.

Map

In the game, we use a grid map and units will be set or move on these grids. Also, since camera is positioned in top and looking down, player will have panoramic vision through map. In Figure 1, an example of a map can be shown. While the black part demonstrates the path that will is used by enemies, others parts are reserved for player's tower constructions. Also, since traps are constructed on the path, black part could be used for constructions too. The left side of the map is accepted as a place where enemy waves come and on the right of the map, our castle Dol-Guldur will be set.

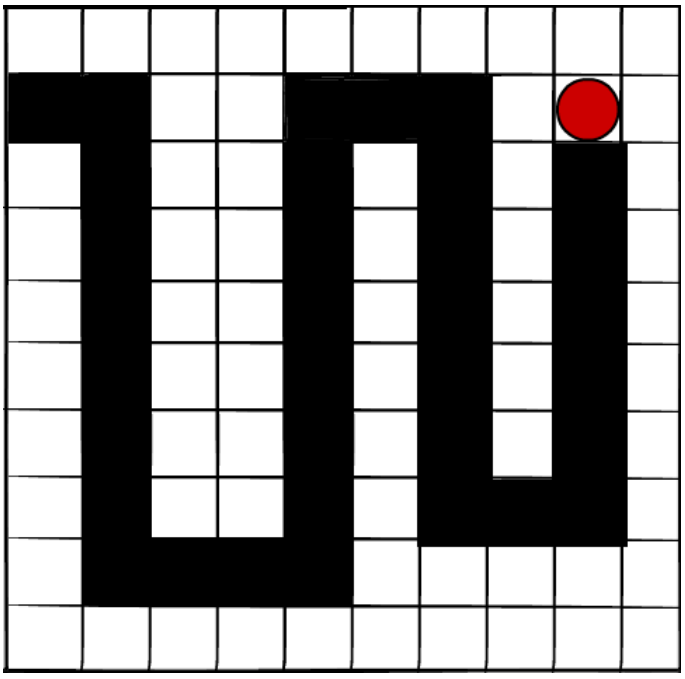


Figure 1: The map of the game

Towers

Towers are one of the in-game choices of the player to stand against enemy waves. They are upgradable, destructable and re-positionable units so that player has lots of things to do with them.

There are 5 types of towers: bronze, silver, gold, diamond and Barad-dûr. In the beginning of the game, player is only able to construct bronze tower with his gold. However, he could construct silver, gold and diamond towers as he passes the levels. Since the last one, Barad-dûr, is a special tower; in order to construct it, player must have kill certain number of enemies, reach a certain level and have certain number of constructions on the map.

Each tower has different health, attack range, attack speed and attack damage. All these abilities increases from bronze tower to Barad-dûr.

Tower / Ability	Bronze	Silver	Gold	Diamond	Barad-dûr
Health	5	10	15	20	40
Range	5	10	15	20	40
Speed	10	15	20	25	40
Damage	10	20	30	40	50

Figure 2: Attribute table of towers

Traps

Traps are another in-game choice of the player to defend his castle from enemy waves. They are also destructable and re-positionable units but they have differences from towers: they are not upgradable, they are disposable and they are positioned on the path that enemy units follow.

There is 4 types of traps: booby, mine, dynamite and Little Boy. Player is only able to set booby trap in the beginning of the game and as level passes, he is able to set other types of traps. Also, if player unlocks the Little Boy by reaching adequate level, he is only be able to set only one Little Boy for current level.

Each trap has different effect area and damage. All these abilities increases from booby trap to Little Boy. Booby trap blocks only 1 enemy units. Mine causes death of all enemy units in 1 square and dynamite causes death of all enemy units in 2 square. However, Little Boy are much more fatal then other traps; it not only causes death of all enemy units in 4 squares but also creates an fatal fallout and causes the death of all units that pass by from explosion area for 3 seconds.

Trap Ability	Booby	Mine	Dynamite	Little Boy
Effect Area	0	1	2	4
Damage	50	50	50	50

Figure 3: Attribute table of traps

Enemy Units

Enemy units are non-playable characters of the game, which means that they are under control of the system. Their purpose is simply to reach the castle of player, Dol Guldur. They comes from the beginning of the map as waves, passes through the path exists and if they still live, they enters the castle. Since castle can stand against limited number of enemy units, in order to keep going in the game, killing the enemy units before they reach the castle is important.

There is 4 types of enemy units, as mentioned above. These are orcs, goblins, elfs and trolls. All of these races have different abilities in gameplay in terms of

health, speed, agressiveness, attack range, attack speed and attack damage. Also, their frequency to be seen changes from one level to another to create challenging gameplay environment. Orcs are accepted as durable, slow, soft and melee; Goblins are easy to kill, fast, soft and melee; Elfs are easy to kill, fast, agressive and ranger and lastly Trolls are durable, agressive, fast and ranger.

Race/Att.	Health	Movement Speed	Agressiveness	Range	Attack Speed	Damage To Towers
Orcs	30	5	0	0	0	0
Goblins	15	15	5	0	10	1
Elfs	20	15	10	2	5	2
Trolls	25	10	10	2	5	5

Figure 4: Attribute table of enemies with name of enemy race.

Castle

At the end of the map, there is a castle which plays a direct role of game progress. By default, Castle the Dol Guldur can stand against maximum 10 enemy units and if this number becomes 11, game is over. So, player should create a defense project to protect his castle in most effective way. However, as player's level goes up, it is possible to upgrade this castle's capacity to stand against.

Level Mechanism

Level choice is asked before starting a new game. If the player is really a beginner, he only has a chance to start from level 1. However, if he advanced in his progress, he has chance to play past levels but still not able to choice next levels in progress since they are not unlocked. Finishing a level successfully is only way to unlock the next level. After finishing of the game, as expected, player can choose any level among 16 created level that he wants to play and play it again.

Each level consists of 3 enemy unit waves. These waves contain different types of races and the waves will be released in a planned order which challenges the player. At the very beginning levels of the game, waves will be formed from weak races orcs. In mid game, more powerful races, goblins and elfs, will be in companion with orcs and force player to be much more careful about spending his money and allowed time. However, as latest levels approach, orcs will be replaced with trolls which are very harsh and agressive enemies as well as hard to kill.

Requirements

Requirements Analysis

As expected, the game has requirements in itself and so it is important to determine them before creating the game. Game's requirements differ in type however, pseudo requirements, functional requirements and non-functional requirements are determined requirements of the game. It is possible to divide functional requirements into two as menu and in-game and non-functional requirements into four as salability, performance, easy to play and upgradeability.

Functional Requirements

1. **Loading Level**

   The game should load all the external elements like audio files, images and effects that are required for a level. Here, a loading screen should be displayed on the screen which contains hints about the game and show the loading bar.

2. **UI and HUD skin of the game**

   The map of the game covers biggest part in-game user interface. Map consist of the path, towers and traps that are set by player, enemy units that are under control of the system, the castle and some other environmental objects such as trees, rocks and grass. Towers and traps are interactive with mouse actions, which means that it is possible to see current abilities of tower by holding mouse on them and to see general information and upgrade menu by clicking mouse to them.

   On the other hand, left of the gameplay screen, there is a bar that show other information about the game such as number of remaining waves, number of units killed by player, remaning capaticy of player's castle and this tab allows player to create new traps and towers.

3. **Enemy Units**

   Each enemy unit differs in number and consists of stated races. In each wave, particular number of enemy unit with their graphical elements should be created and their other abilities (range, speed, health, aggressiveness and attack speed) should also be created by considering provided information. Role of enemy units are simple: They follow the path and try to reach player's castle. Their health points should be affected by towers by considering the connection between tower's attack damage and their health points. When enemy unit's health point becomes zero or drops to below zero, they are accepted as dead and they give bonus gold to player. Also, enemy units are simply are under control of the system and hereby they neither cannot be controlled nor get out of their routine.

4. **Towers**

   Towers play role in killing enemy units. They start attacking when an enemy unit enters their range and keep attacking a particular unit until it becomes dead or attacked unit gets out of range. A tower starts seeking a new target when its target is dead or out of range and applies this same loop for every target. They can be positioned anywhere in the map.

5. **Traps**

   Traps are also used in killing enemies and gold. They are located on anywhere in the path and become active when enemy units pass on it. They could counter at least 1 unit and at most all units in single enemy wave.

6. **Castle**

   Castle is located at very last of the path in the map and directly effect the player's success in one single game. It has a capacity to stand against particular number of enemies.Enemies that could pass the path reaches the castle and as default, when the number of enemies that can reach the castle becomes ten, player is accepted as unsuccessful in game. However, this capacity can be increased by player with upgrades.

7. **Construction Screen**

   When an enemy wave is successfully defeated, system lets player to modify his buildings for 30 seconds. In this time period, player can construct new buildings and set new traps as well as upgrade existing ones. However, in this period, a screen should be opened and give necessary information about current gold, available buildings to build and remaining time for construction.

8. **End Game**

   The game should end if the castle outnumber it's current capacity or all waves are successfully defeated. In this situation, an ending screen that contains a congratulation message for player and information about game ,such as duration of the game, number of killed enemies and current gold, should be displayed. Also, this screen allows user to go back to the Main Menu.

Non-Functional Requirements

a) Extendibility

Game should be available to be extended in order to meet new requirements from players in the future. For example; if the current number of levels or in-game content like number of different tower types, number of races etc. stays insufficient for players, our code is needed to be easy to upgrade in order to meet demands. Also; an upgradable design allow developer to interfere easily in the case of bugs, crashes or exceptions.

b) Easy to play

Game should have user-friendly and understandable interface in order to increase number of user and these users' in-game pleasure. Our design goal is creating a user-friendly interface that even a child can play and effective visual design that helps players to enjoy.

c) Performance

Game should be played with high performance even in old fashioned systems. System's currenct situation is not needed to limit player's enjoyment from the game and so, as designer of the game, our goal is combining smooth inputs and graphics with high performance as much as possible.

System Models

Use Case Model

Use Case -> StartPlaying
Actors -> Player
Flow of Events ->

1. The user hits “Play” button.
2. The game proceeds to the panel where player chooses Load or New Game
3. The game proceeds to the Level Selection Panel.
4. The user selects a level to play from Level Selection Panel.
5. The game starts loading the level.

Use Case -> ViewHelp
Actors -> Player
Flow of Events ->
1. The player hits the “View Help” button.
2. The game displays the guide supported with visuals.
3. The player hits the return button.
4. The game returns to the main menu.
Entry Conditions-> Player is in the main menu.
Exit Conditions -> Player is in the main menu.

Use Case -> ChangeSettings
Actors -> Player
Flow of Events ->
1. The player hits the “Settings” button.
2. The game displays the game settings.
3. The player changes settings.
4. The player closes settings.
5. The game returns to previous screen.
Entry Conditions -> Player is in main menu or in the level.
Exit Conditions -> Player is in main menu or in the level.

Use Case -> DiscardSettings
Actors -> Player
Flow of Events->

1. The player hits the “Settings” button.
2. The game displays the game settings.
3. The player makes changes in Settings.
4. The player decides to bring old setting preferences back.
5. The player clicks on “Discard Changes” button.
6. The player closes settings.
7. The game returns to previous screen.

Entry Conditions -> Player is in main menu or in the level.
Exit Conditions -> Player is in main menu or in the level.

Use Case -> SaveSettings
Actors -> Player
Flow of Events ->
1. User hits “Save” button in Settings screen.
Entry Conditions -> Player is in main menu or in the level.

Exit Conditions -> Player is in main menu or in the level.

Use Case -> BuildTower
Actors -> Player
Flow of Events ->
1. The player clicks “Build Tower” button.
2. The player selects a location.
3. The game adds the tower to the selected location.
Entry Conditions -> Player is in the game.
Exit Conditions -> Player is in the game.

Use Case -> SetTrap
Actors -> Player
Flow of Events ->
1. The player clicks “Set a Trap!” button.
2. The player selects a location.
3. The game adds the tower to the selected location.
Entry Conditions -> Player is in the game.
Exit Conditions -> Player is in the game.

Use Case -> InvalidLocation
Actors -> Player
Flow of Events ->
1. The player selects an invalid location.
2. The game exits from building mode
3. The game enters to free mode.
4. The game plays the error sound.
Entry Conditions -> This use case extends BuildTower and SetTrap use case. The game determined the invalid locations beforehand.

Use Case -> UpgradeTower
Actors -> Player
Flow of Events ->
1. The player clicks on a tower.
2. Sidebar displays details and options.
3. The player selects “Upgrade” option.
4. The game upgrades the tower.
Entry Conditions -> The tower should exist beforehand.

Use Case -> UpgradeTrap
Actors -> Player
Flow of Events ->
1. The player clicks on a trap set.
2. Sidebar displays details and options.
3. The player selects “Upgrade” option.
4. The game upgrades the trap.
Entry Conditions -> The trap should exist and not be exploded beforehand.

Use Case -> UpgradeCastleCapacity
Actors -> Player
Flow of Events ->

1. The player clicks on the Castle.
2. Sidebar displays details and options.
3. The player selects “Upgrade” option.
4. The game upgrades the capacity of the Castle.
Entry Conditions -> Maximum castle capacity should not be reached.

Use Case -> NotEnoughGold
Actors -> Player
Flow of Events ->
1. The player selects build or upgrade tower.
2. The game plays the error sound and shows a message that says “Insufficent Gold”.
Entry Conditions -> This use case extends BuildTower, SetTrap, UpgradeTower, UpgradeTrap and UpgradeCastleCapacity use cases. It is initiated by the game when the player’s current gold is not enough to build or upgrade the tower.

Use Case -> PauseGame
Actors -> Player
Flow of Events ->
1. The player hits “Pause” button.
2. The game pauses the level.
Entry Conditions -> The level is not paused.
Exit Conditions -> The level is paused.

Use Case -> Resume Game
Actors -> Player
Flow of Events ->
1. The player hits “Resume” button.
2. The game pauses the level.
Entry Conditions -> The level is paused.
Exit Conditions -> The level is not paused

Use Case -> ReturnToMainMenu
Actors -> Player
Flow of Events ->
1. The player hits “Return to Main Menu” button.
2. The game firstly saves and then closes the current level.
3. The game displays the main menu.

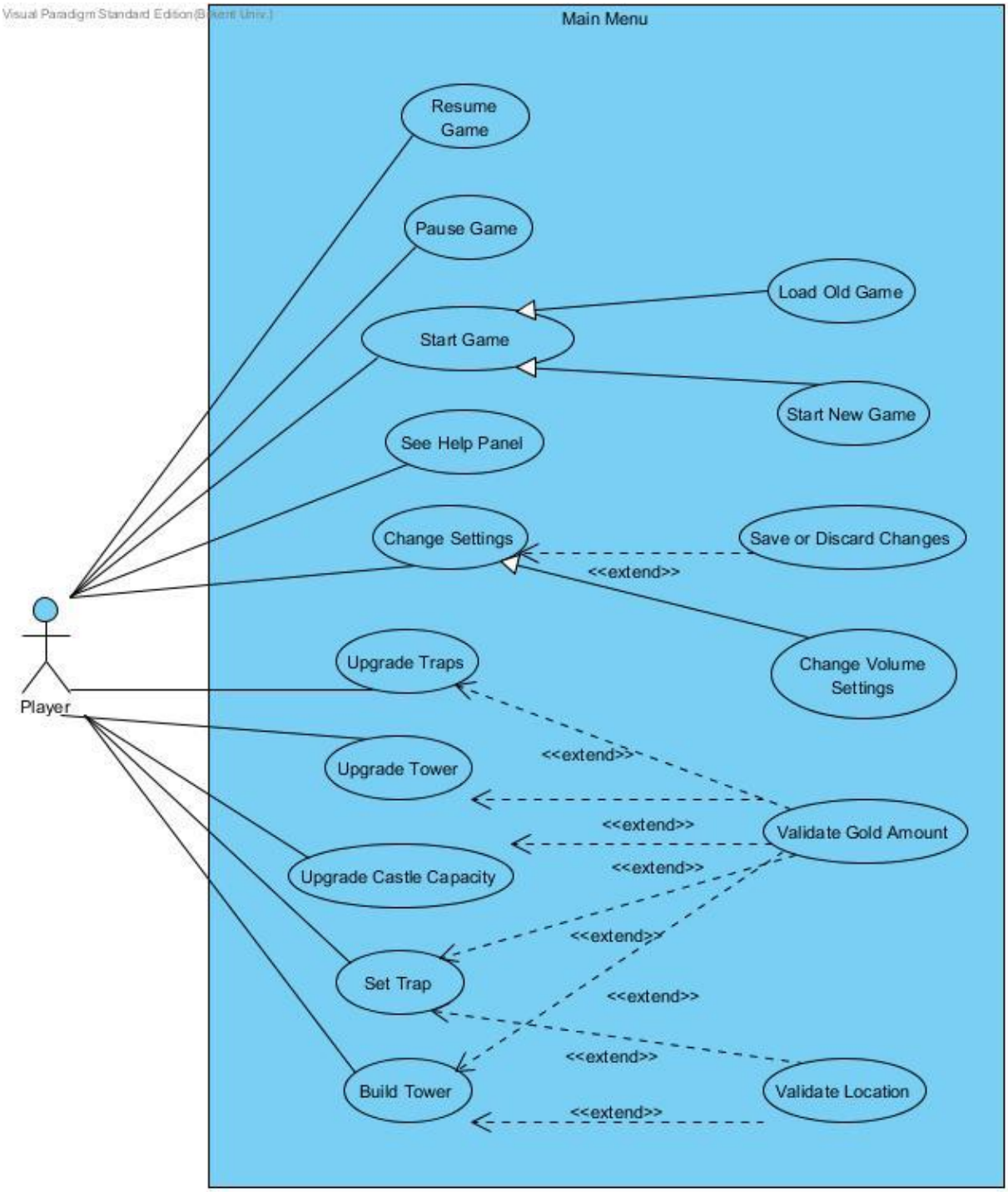


Diagram 1: Use case diagram

Object Model

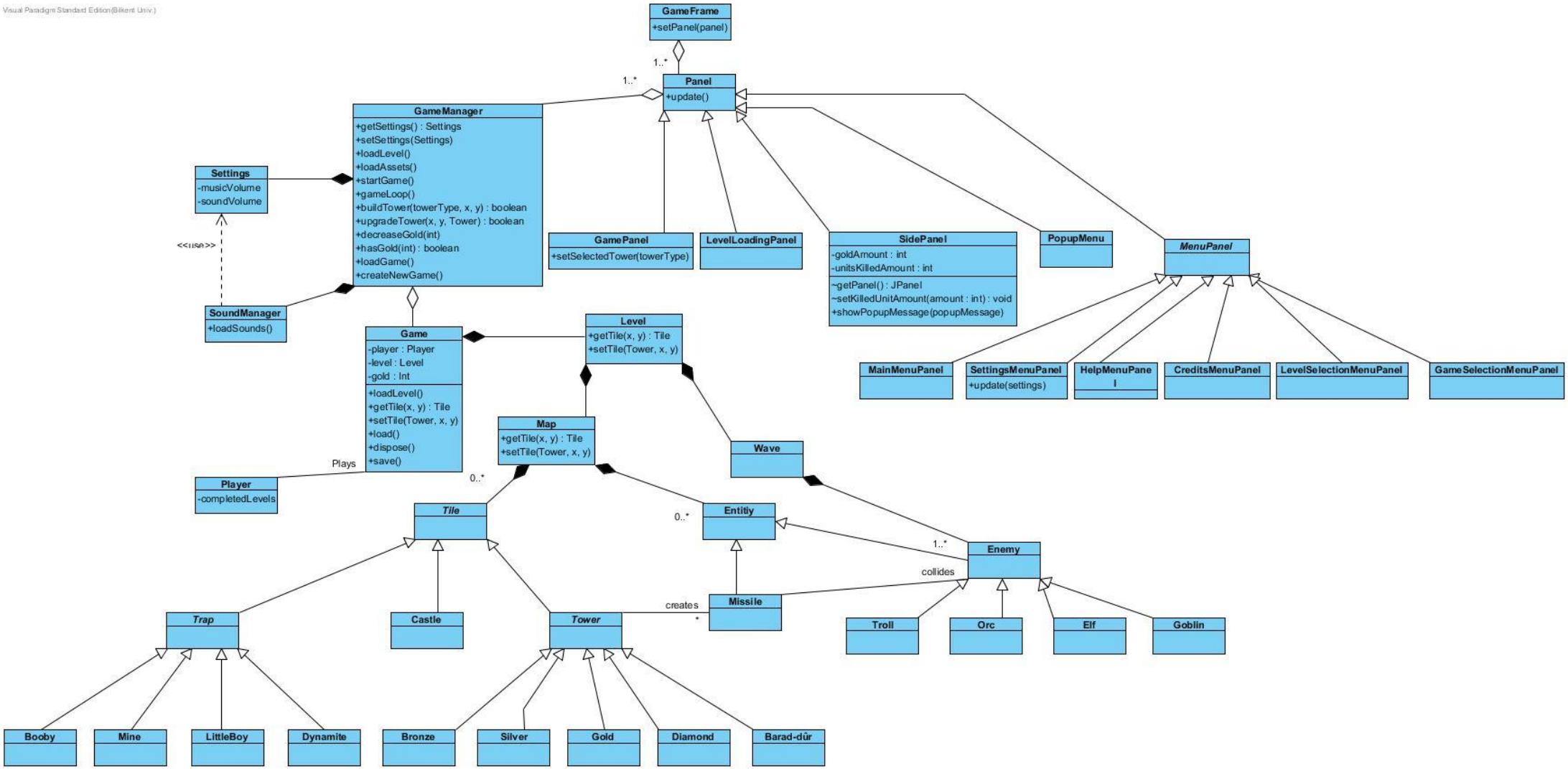


Diagram 2: Class Diagram

Dynamic Models

Change Settings

Following sequence diagram shows how the player make changes in settings of the game at Setting Menu.

Player enters Settings Menu using the “Settings” button in the main menu. Player presses “Default” button to set the volume levels to their default value. Player then presses “Discard Changes “ button the discard the changes he made and he just increases music volume through music volume slider.Then he uses “Save & Exit” button to exit the settings menu.

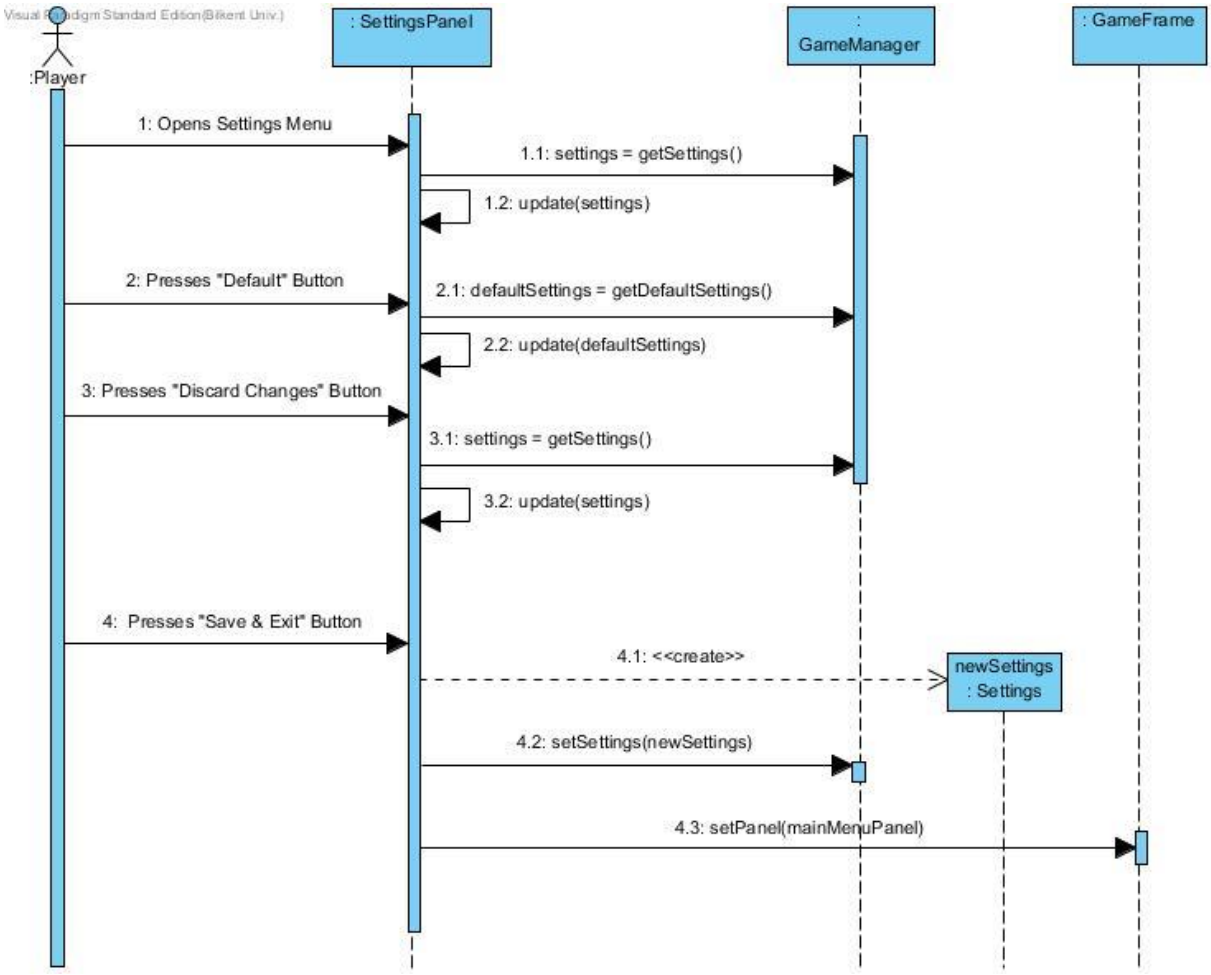


Diagram 3: Change Settings Sequence Diagram

Initialize and Load Game

Following sequence diagram shows how player loads an played game after clicking on play game.

Player enters Game Selection Menu from “Play Game” button in the Main Menu. At first, Player presses “Load Game” button to enter the last saved game. Then he sees that he finished all of the levels. Game is so enteraining that he wants to play the game once more again. He presses “Back” button to go back to the Game Selection Menu. Then he presses “New Game” button to create a new game.

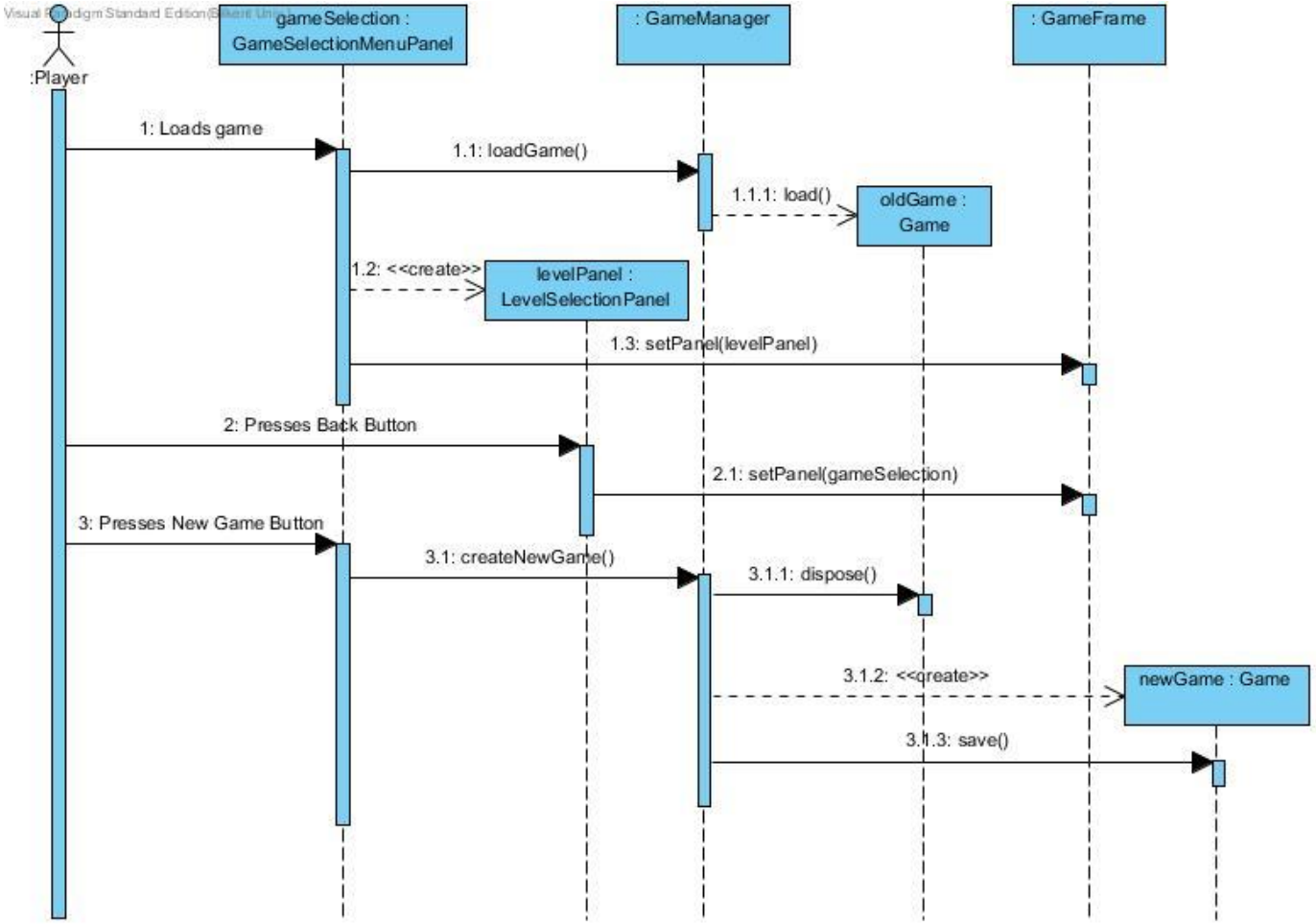


Diagram 4: Initialize and load game sequence diagram

Initialize Level

Following sequence diagram shows how the player selects a particular level from Level Selection Screen and how the game dynamics works in order to load a level.

Player chooses Level 10 in the Level selection menu. A loading screen is shown when game loads the assets for the level. Game loads serialized form of the level from a data file so that all objects that level has is loaded at the same time from this serialized data. Then game loads textures and sounds. After all loading is done, level starts.

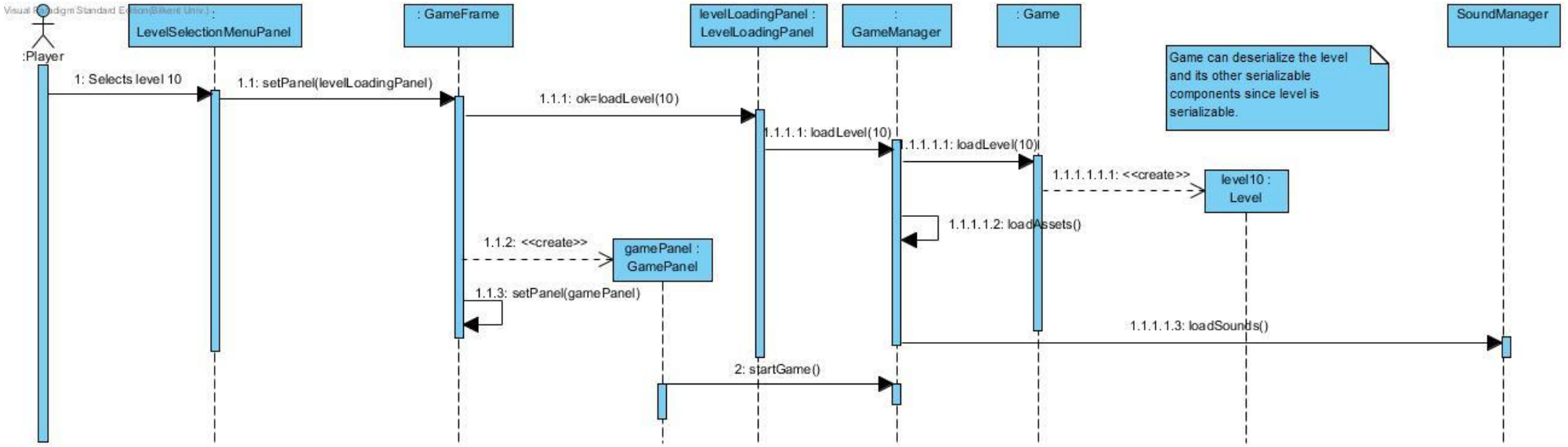


Diagram 5: Initialize level sequence diagram

Game Loop

The main function of the game loop is the update the tiles and entities in the map. The main loop also handles new wave events.

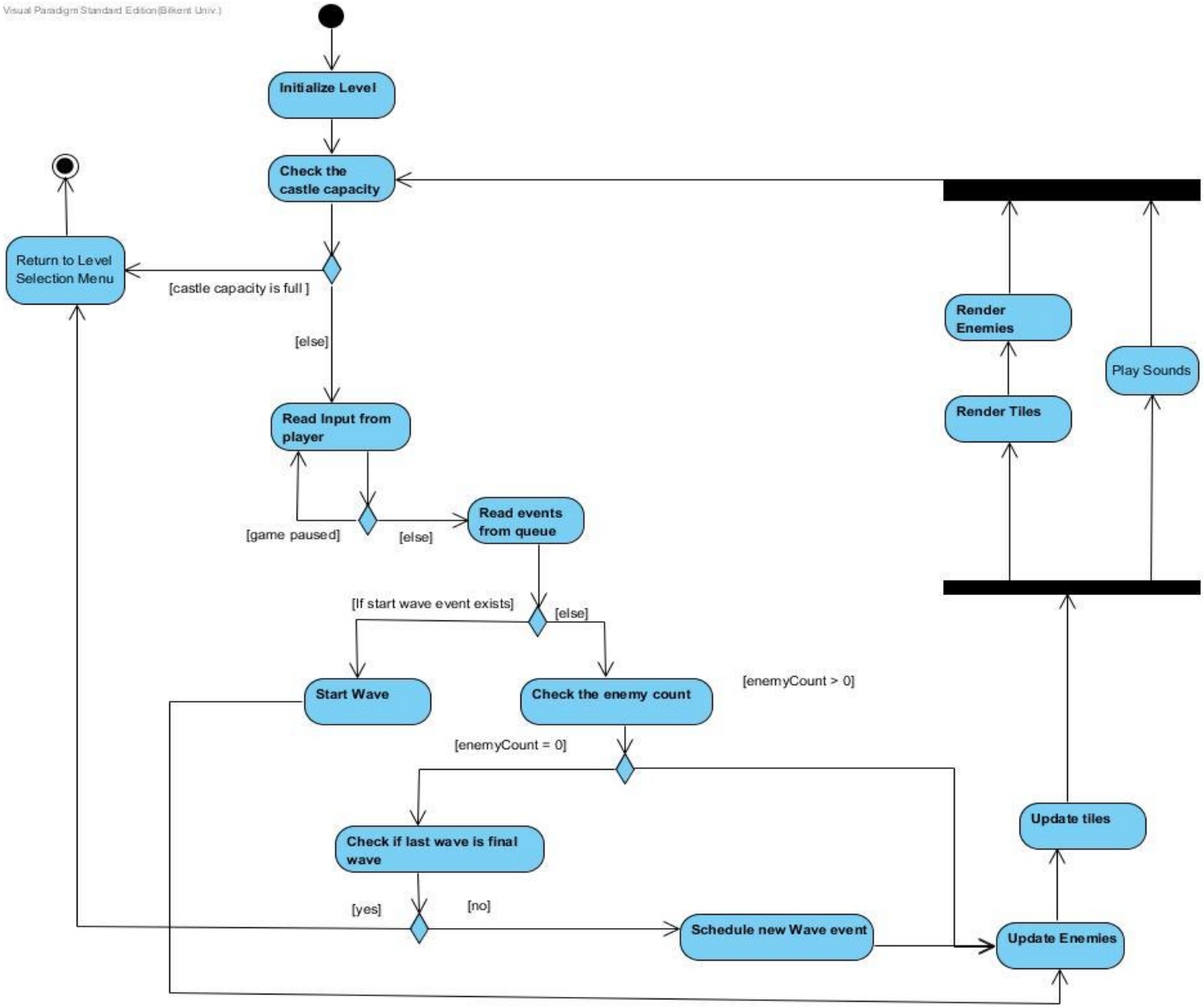


Diagram 6: Game Loop Activity Diagram

Build Tower/Trap

While playing the game Serkan he saw enemies coming towards the castle and he decides to build a tower to protect his castle. He chooses one of the available (according to amount of money that Serkan owns) tower types from left panel and click somewhere on road of enemy. A popup appears, which indicate that it is not possible to build a tower on road of enemy. He clicks another point on the map. Tower is successfully built. Gold of Serkan decreases.

*Same scenario are going to be applied to Set Trap, however, this time Serkan will choose trap from right menu and he will only be able to build trap in road.

** x,x1,x2,y,y1,y2 represents the position of the new units to be added to map.

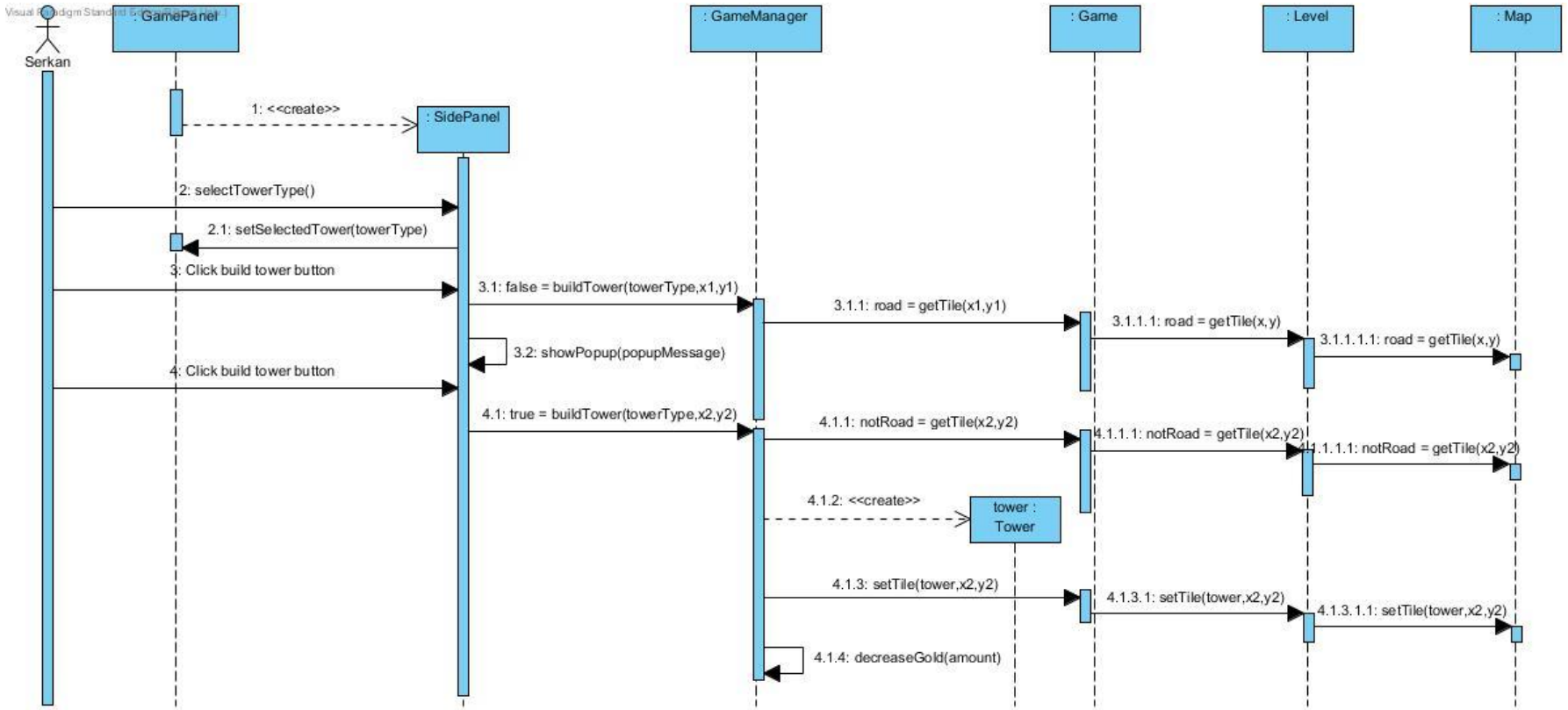


Diagram 7: Build tower/trap sequence diagram

Upgrade Castle/Tower

While playing game Serkan realizes that his tower is damaged a lot. He decided to upgrade that tower. He clicks on tower and saw a popup menu, which show type of upgrade available. He upgrades his tower into Diamond tower type and observe that gold is decreased.

*Same scenario is going to be applied to Upgrade Castle, however, this time Serkan will click on castle instead of tower.

** x,x1,x2,y,y1,y2 represents the position of the new units to be added to map.

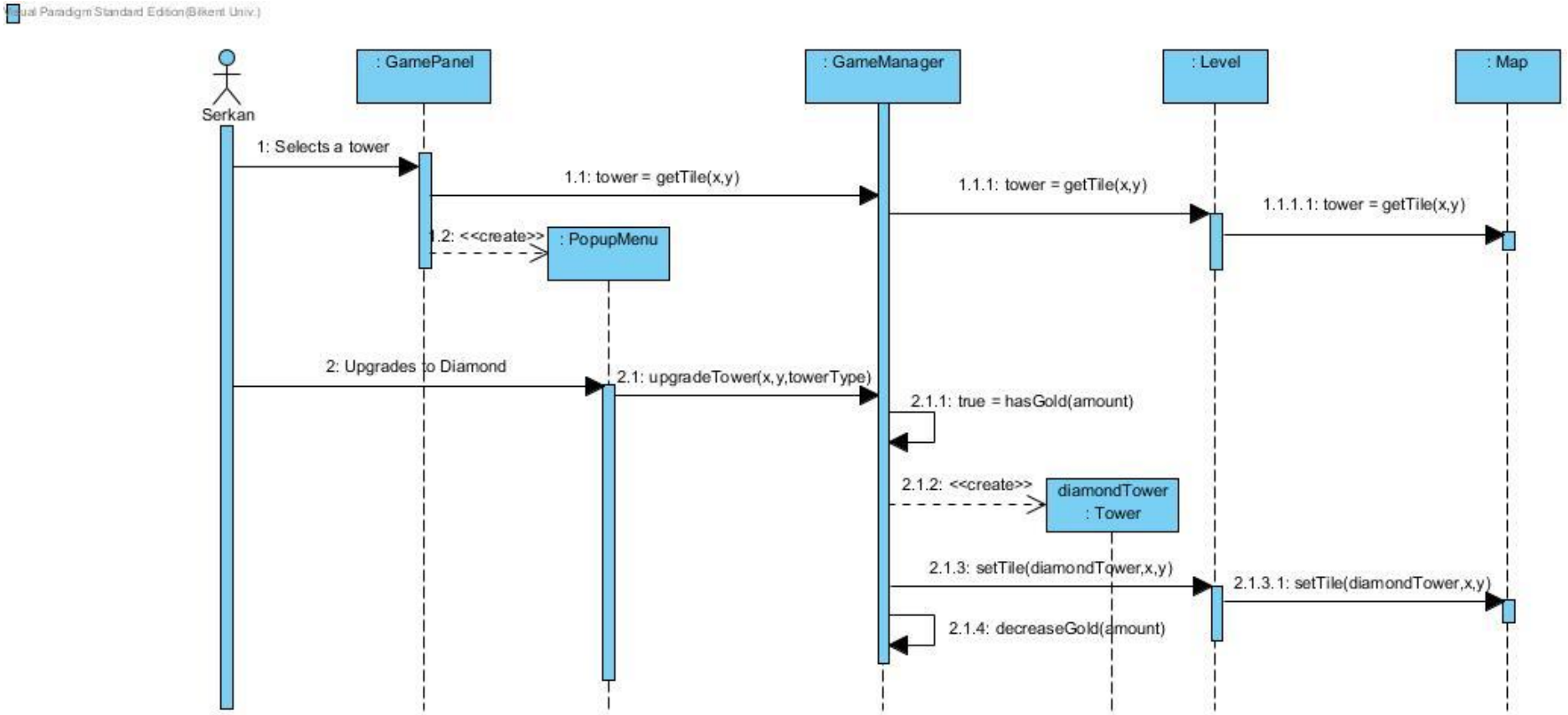
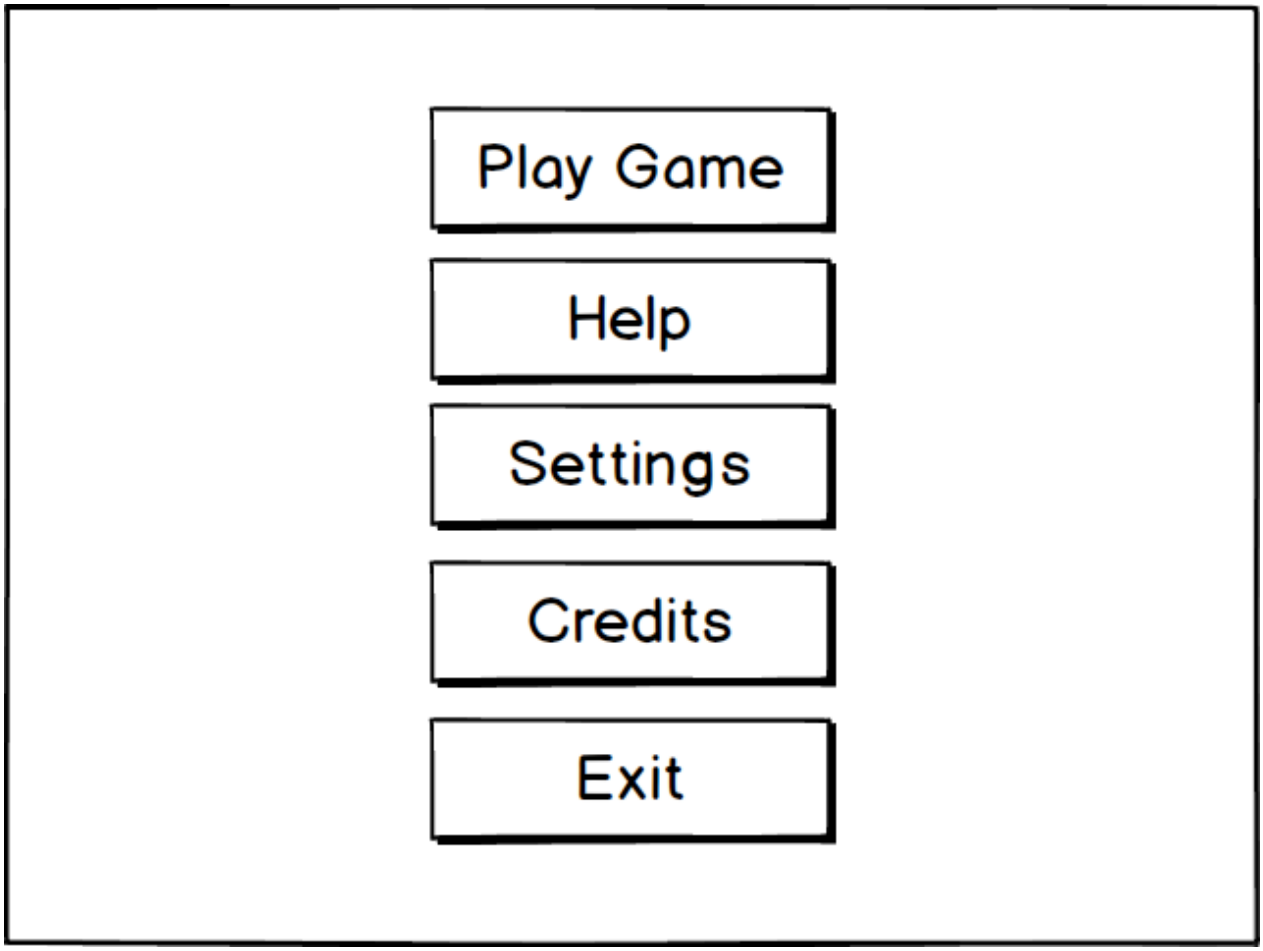


Diagram 8: Upgrade tower/castle sequence diagram

User Interface

Main Menu

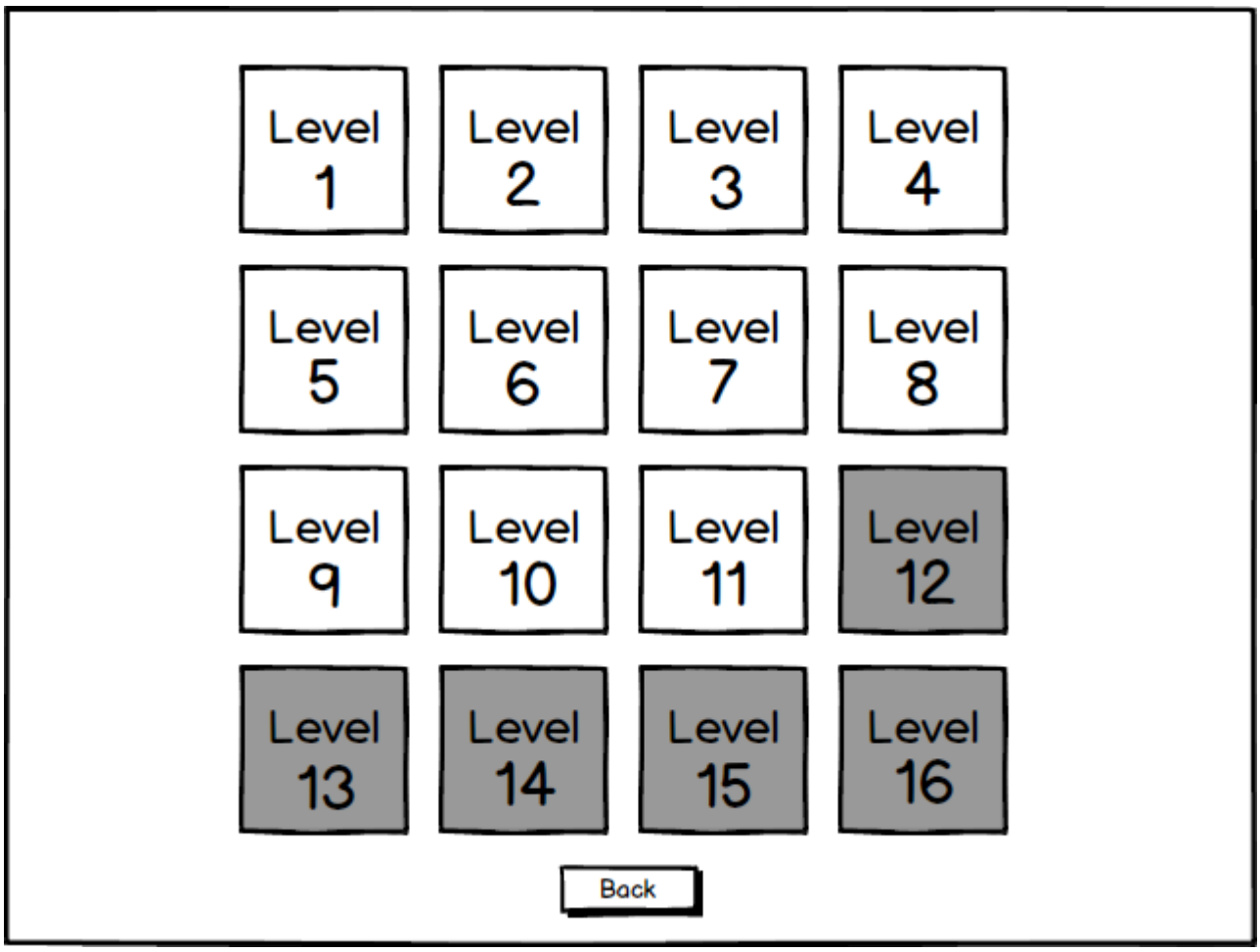
Player directly contact with Main Menu interface when he opens the game. Here, he may go into Options panel where he can manage his preferences, Start Game panel where he may load his past game or create a new game, Help Panel where he can get help about how to play game and Quit.



Mockup 1: Main Menu mockup

Level Selection Panel

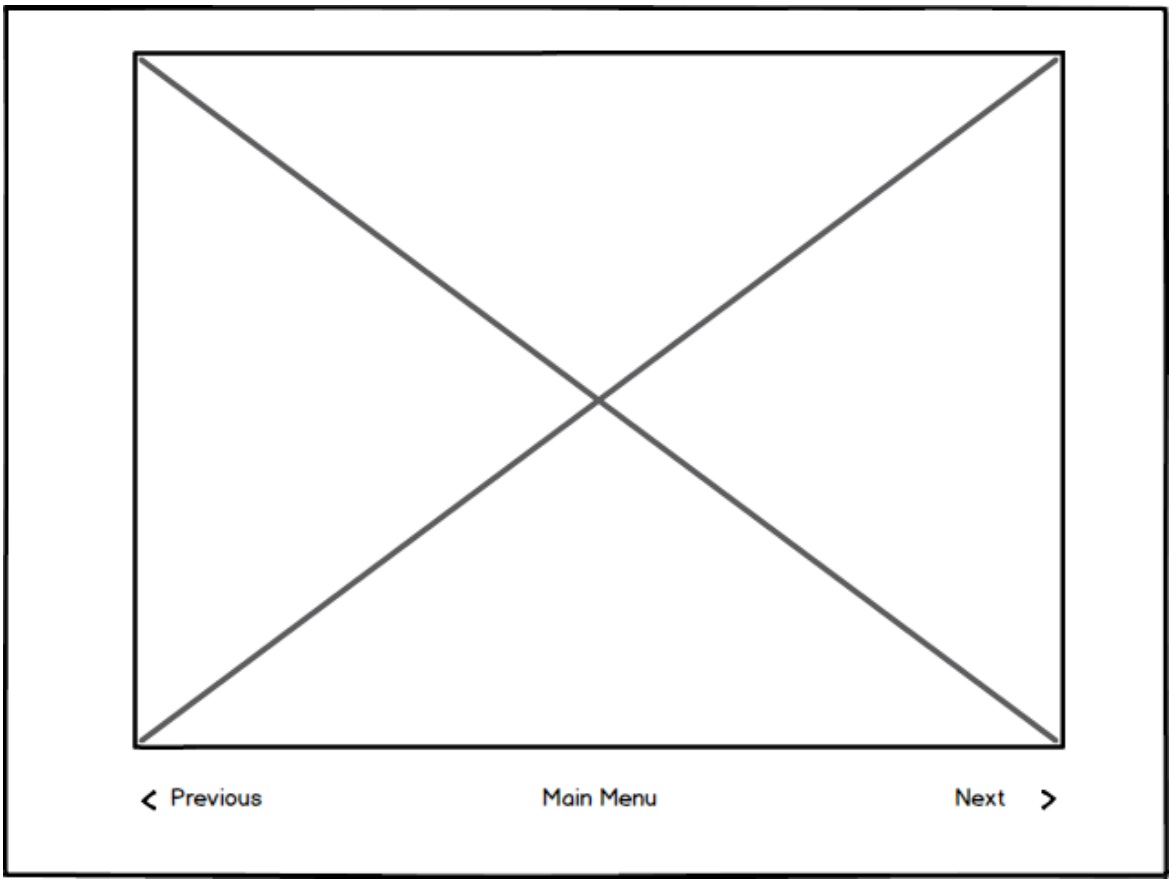
Here, player should see all levels in the game. However, player may only choose all levels that he played and successfully completed. These panel is directly connected to the game's itself; which means that after player's choice of level, the game starts.



Mockup 2: Level selection menu mockup

Help Panel

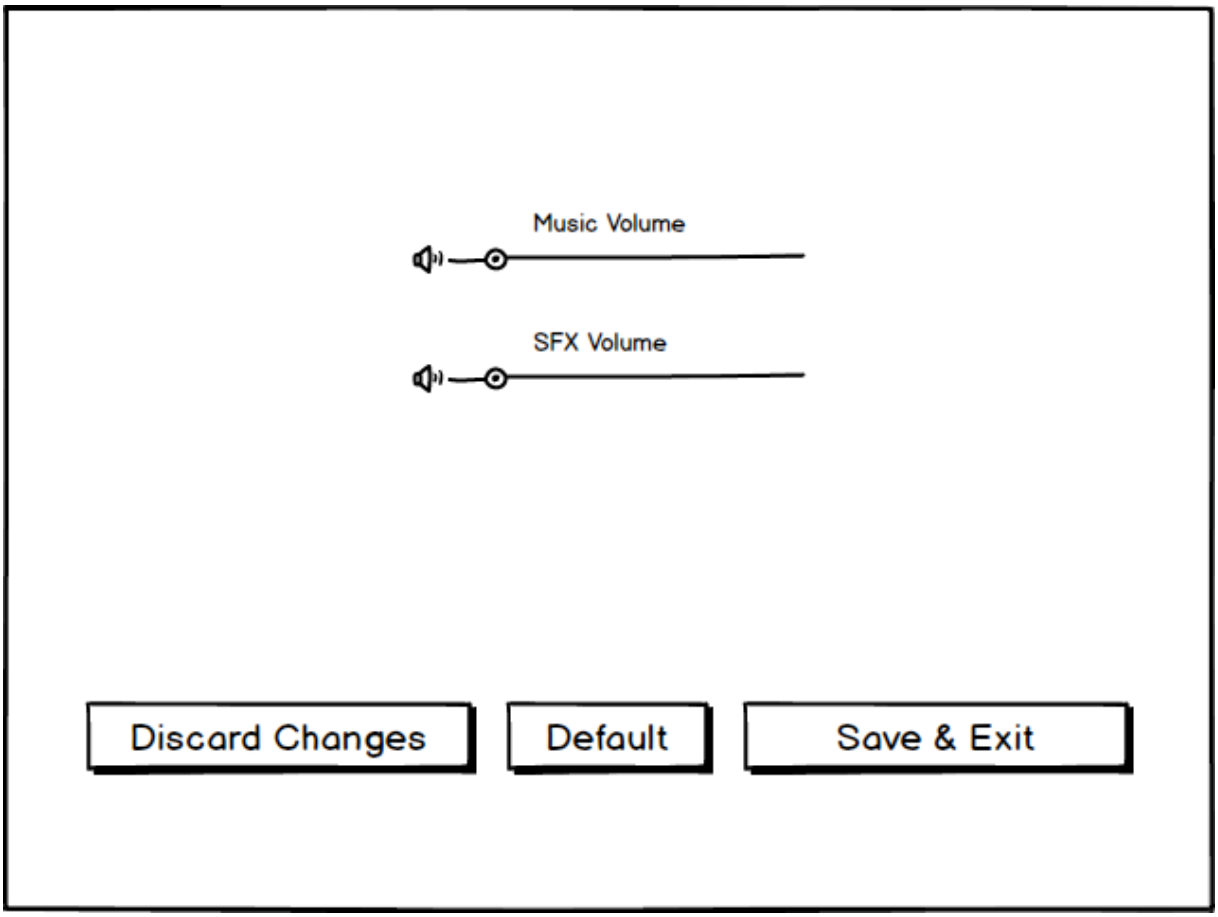
In Help Panel, player get information about how to play game. Main and most important function of this panel is this: it is formed as a image gallery and player learns how to play game step by step. These images are in-game images and supported by some sketches and notes to make gameplay understandable. In Figure 5, a mockup screen of panel is provided.



Mockup 3: Sample mockup of Help Panel

Settings Panel

In Settings panel, player is able to change the volume of background music in game and sound effects in game as well as mute the game completely. Also, these panel contains “Discard Changes”, “Default” and “Save & Exit Buttons” in order to fulfill player’s preferences completely.

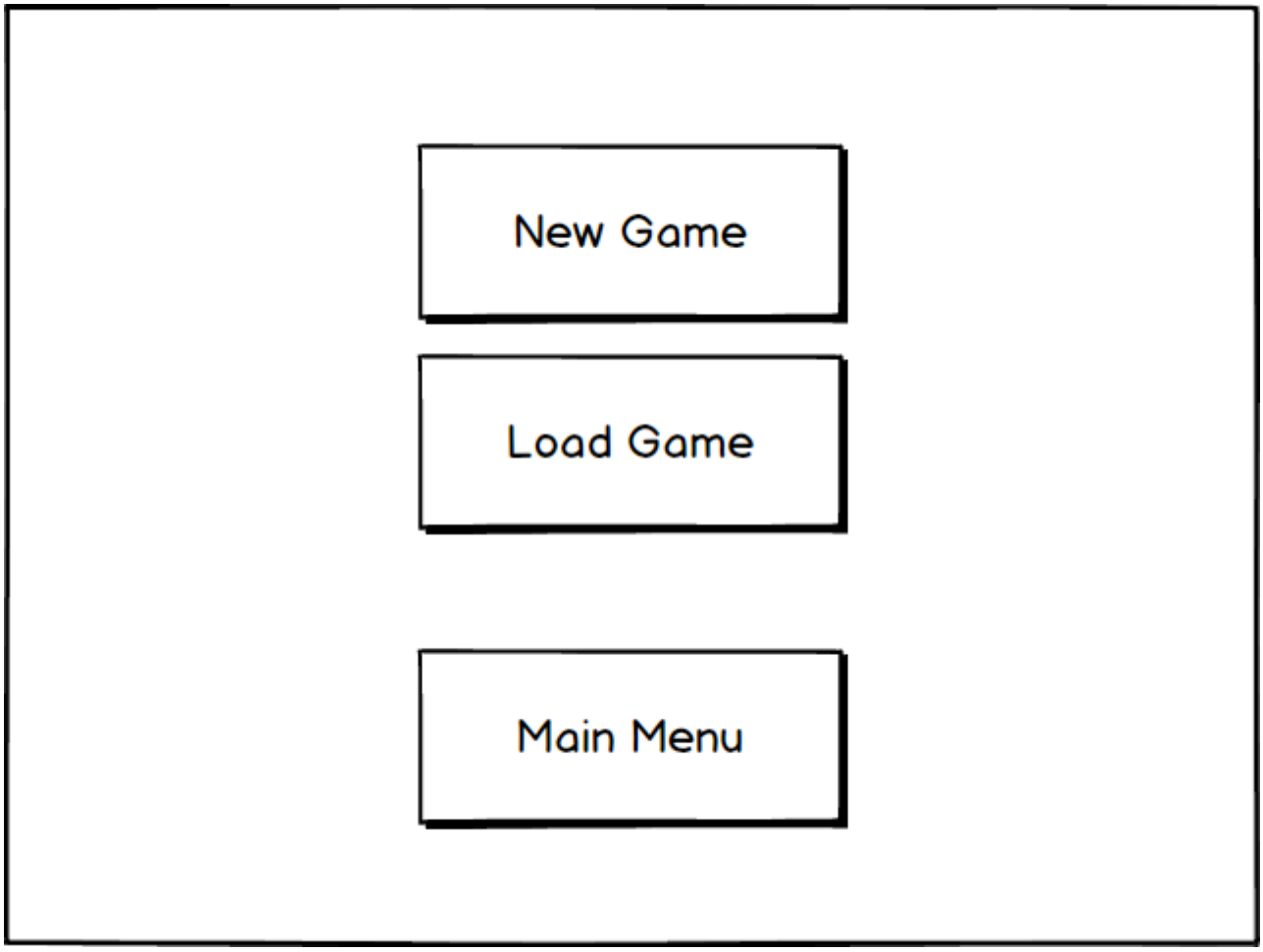


Mockup 4: Sample mockup of Settings Panel

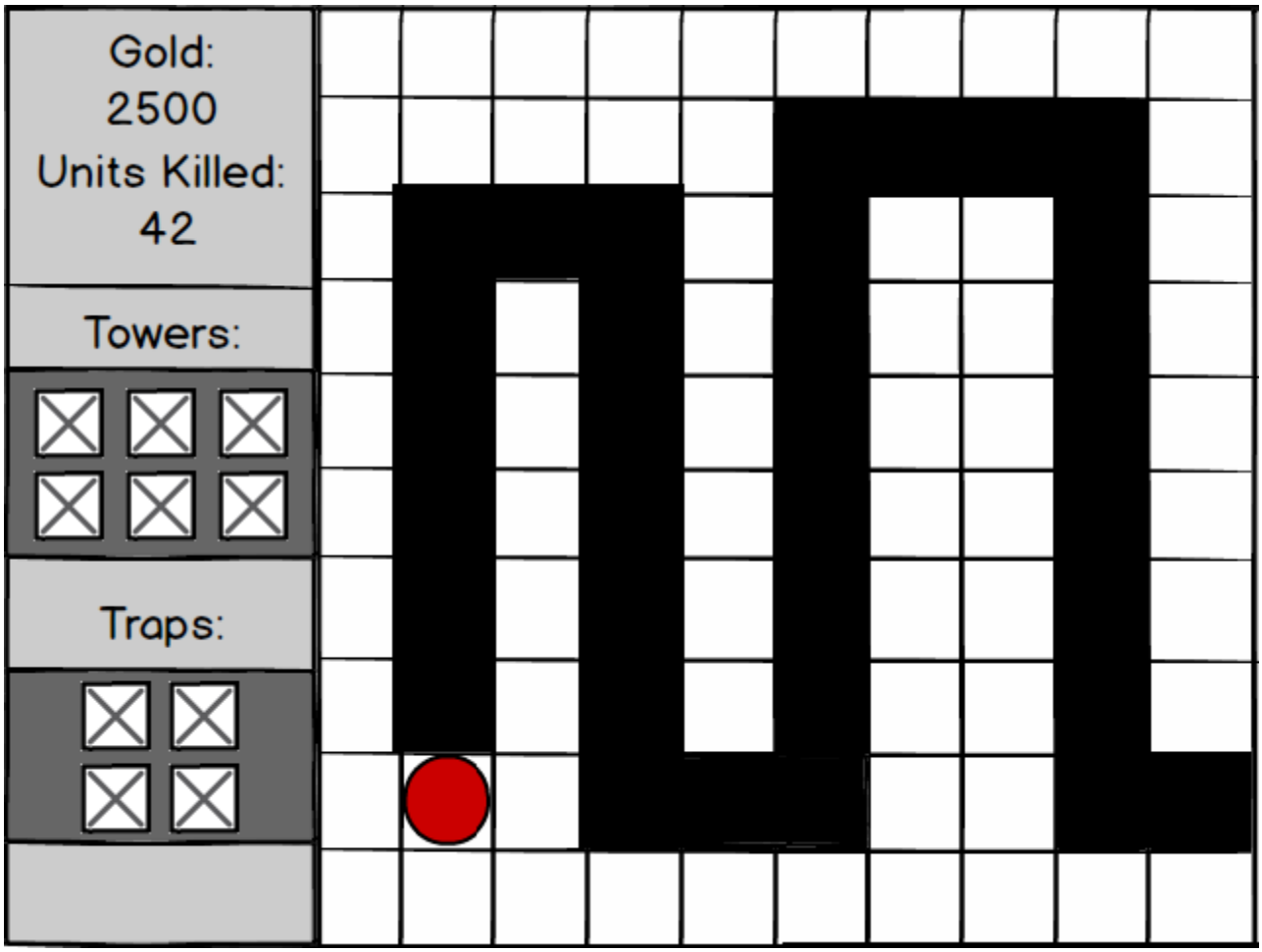
Credits Panel

Credits Panel contains information about developers of the game, time of development of the game and other information about development process. All these information on this panel is in scrolling text style.

Game Selection Menu



Mockup 5: Game Selection Menu Mockup



Mockup 6: Game panel mockup