CS 319 - Object-Oriented Software Engineering

Analysis Report

HallwayRush

GroupD

Caner Bozkurt

Ezgi Ebren

Doğa Dikbayır

Mert Uygur

Course Instructor: Uğur Doğrusöz

Table of Contents

1. Introduction

Hallway Rush will be a high-score based arcade-survival video game in which the player's goal is to survive various types of enemies in an endless hallway. The player will be able to use a simple weapon to shoot the enemies. However, there will be different weapon types and bonuses appearing randomly through the game, which the player can choose to take or leave. The player can also choose to finish the game after killing the last enemy (the boss) or continue to beat their high score. Hallway Rush will be played by keyboard in the PlayHR mode and by mouse in other modes such as displaying the menu.

2. Overview

Game has three different modes. The enemy types will differ in speed, endurance, strength, movement and weapon choice in order to force the player to develop different survival strategies, in a sense. The player will also be able to pick up different bonuses (positive or negative effect), by shooting the chests appearing randomly in the game. There will be different types of weapons spawning according to the score and the player will be able to pick up these, or ignore them, just like the bonuses. The types of spawning enemies will change based on the player's score. Higher the score, it will be harder to survive the enemy waves. The game scenario will be beaten when the final enemy is killed, however the player will be able to continue playing in order to beat the high-score.

2.1. Gameplay

The player will need a keyboard and a mouse in order to play the game. He will be able to move on a horizontal line with the arrow keys, in order to shoot enemies. Various keys will be needed to use bonuses and weapons. The mouse will be needed to navigate through the menus.

2.2. Enemy Types

2.2.1. Mobs

Mobs are the enemies who are affected by all the bonuses and they spawn often, in large numbers, compared to other types of enemies.

Zombie: This mob is a simple zombie which has low health, medium speed, melee attack and vertical movement. If a zombie reaches the player, the player has to kill it in order to move again.

Armed Zombie: This mob has the same health and speed stats with the regular Zombie; however it has ranged attack and horizontal movement. So it attacks the player from a range with a gun.

Armored Zombie: This mob is a regular Zombie with higher health and slower speed.

Special Task Zombie: Armed Zombie with higher health and faster fire-rate.

Runner: Regular Zombie with faster speed and lower health. Once a Runner reaches player's horizontal line, it explodes. Therefore, if the player does not kill the Runner before it reaches the border, life points will be deducted from the player.

2.2.2. Non-mobs

Non-mobs are the enemies which are significantly harder to kill. They are completely or partially immune to player's bonuses. They have higher health than mobs and they have a high-score threshold in order to spawn.

Juggernaut: This enemy has very high health rate compared to regular mobs, it has a ranged attack. It can throw bombs which have a high area damage. If the player manages to kill it, it refills half of the health of the player.

Baba O' Riley: This unique enemy spawns only when the player is able to collect a large amount of score-points. It has the highest health rate in the game. It has ranged and high-damage attacks, it can throw bombs like the Juggernaut and it can spawn mobs to support itself. Once Baba O' Riley is killed, the player's health will be refilled and an extra life will be given.

2.3. Bonuses

2.3.1. Positive Bonuses

Health (10, 40, Full): It drops randomly in three states. It fills the player's health by 10/40/Max points, depending on its state.

Shield (3, 6, 10 seconds): It drops randomly in three states. It makes the player immune to all attacks for 3/6/10 seconds depending on its state.

Extra Life: It gives the player an extra life.

Hallway Rage: It takes down all the mobs in the screen. It damages Juggernauts and Baba O' Riley.

Push Back: It pushes back the vertically advancing mobs; Zombies, Armored Zombies and Runners.

Insta-Kill: The player is able to take down all the mobs in a single shot. Juggernaut and Baba O' Riley are not affected by this bonus.

Double Damage: It doubles the player's damage.

Freezer: It freezes the mobs for 6 seconds and non-mobs for 4 seconds.

Slow Down: It slows down all the enemies for 5 seconds.

Explosive Shots: The player is able to give area damage on every shot for 5 seconds.

2.3.2. Negative Bonuses

Damage (-10, -30): It drops in two states. It damages player's health by 10 or 30 health points.

Sticky Floor: The player is unable to move for 3 seconds.

Acid Puddle: The player's health is reduced by 20 points and he is not able to fire for 3 seconds.

Spawn Mob: Spawns a random mob.

2.4. Weapons

The player will have three slots of weapon and he will be able to navigate through the weapons by choice.

Pistol: This is the default weapon of the player. It has medium damage and slow fire-rate.

SMG: Picked up by the player during the game. Submachine gun has low damage and very fast fire- rate.

Shotgun: Picked up by the player. It has high damage and slow fire-rate.

Assault Rifle: Picked up by the player. This is the most long-term effective weapon in the game. It has fast fire-rate and medium damage.

Grenade Launcher: Picked up by the player. It has very slow fire-rate but very high area damage. It has limited ammo (5 capsules).

2.5. Game Modes

Standard Mode: In this game mode, the player will have the standard 100 points health bar and three lives. When the player has no health points, all the bonuses and weapons drop and one life is deducted. When all the lives are spent, the game is over and the user is asked to enter a nickname to enter his score to the system. If the score is among top ten scores, it is registered on the high scores table with the user's nickname.
If the game ends after Baba O' Riley is killed, the game scenario is also beaten and the player's score is recorded with a gold medal. Once Baba O' Riley is killed, the enemies will spawn in larger numbers in order to challenge the player exponentially.

God Mode: In this mode, the player has unlimited health. So it is a mode for exploring all the enemy types, weapons and bonuses. The game is over when the player quits it. No high score count.

Insane Mode: This mode will challenge the player more, compared to the standard mode. In this mode, the player will have a single life and only the health bonuses. The negative bonuses are still present in this mode. The ending conditions are same as in the standard mode.

3. Functional Requirements

I. The player will be able to start, pause and end a game.

When the user chooses the "Play Hallway Rush" menu, the system starts a game based on the previously selected game mode. Once a game is started, the player can pause the game. When a game is paused, the player can continue anytime they want (unless the application is closed).The player can end the game with the "Quit" option.

II. The player will be able to display help.

If the user has trouble with the controls or gameplay, he can access the help section. Help section consists of three subsections:

- Scenario explanation

- User controls
- Game rules

So, the player will be able to read the game scenario, learn the controls and learn about scoring and end conditions of the game.

III. The player will be able to change the settings of the game.

The "Change Settings" menu will allow the user to:

- Select main character's gender
- Turn on/off the music
- Select between day and night ambiance
- Change game mode (see 2.5.)

By default the background music is on.

The main character will have two texture files, one for male and one for female. So the user will be able to select character in this menu. By default the character is male.

Also the game's ambiance could differ between day and night based on user's selection. By default the ambiance is day time.

The user must apply the settings after selecting them in order to update the system. The user will also be able to use the default settings by clicking 'Default Settings' and applying the changes.

IV. The player will be able to view credits.

An information screen about the developers will appear when the user selects the "View Credits" menu.

V. The player will be able to display the high-score table.

The highest ten scores are displayed in a table with the nicknames of the users. A gold medal symbol will appear near the score if a user beats the game scenario too (see 2.5.1).

4. Non Functional Requirements

I. **Design of game graphics will be well-made.**

The graphics of Hallway Rush will be minimalistic and aesthetic at the same time to provide the users a modern and fun gameplay. We will try to reduce flickering and other graphical problems to minimum to create a smooth gaming experience.

II. **There will be background music.**

The background music is going to be an old-school 8-bit track, to give the player a nostalgic arcade-game atmosphere.

III. **Controls and response time will be optimum.**

The response time of the game will be reduced to a minimum level in order to provide the players with a smooth controlling. The control keys' positions will also be arranged to ease the gameplay.

IV. **Menu instructions will be simple and easy to understand.**

The menus will include clear and short instructions to allow younger players to play the game with ease.

5. System Models

5.1. Use case model

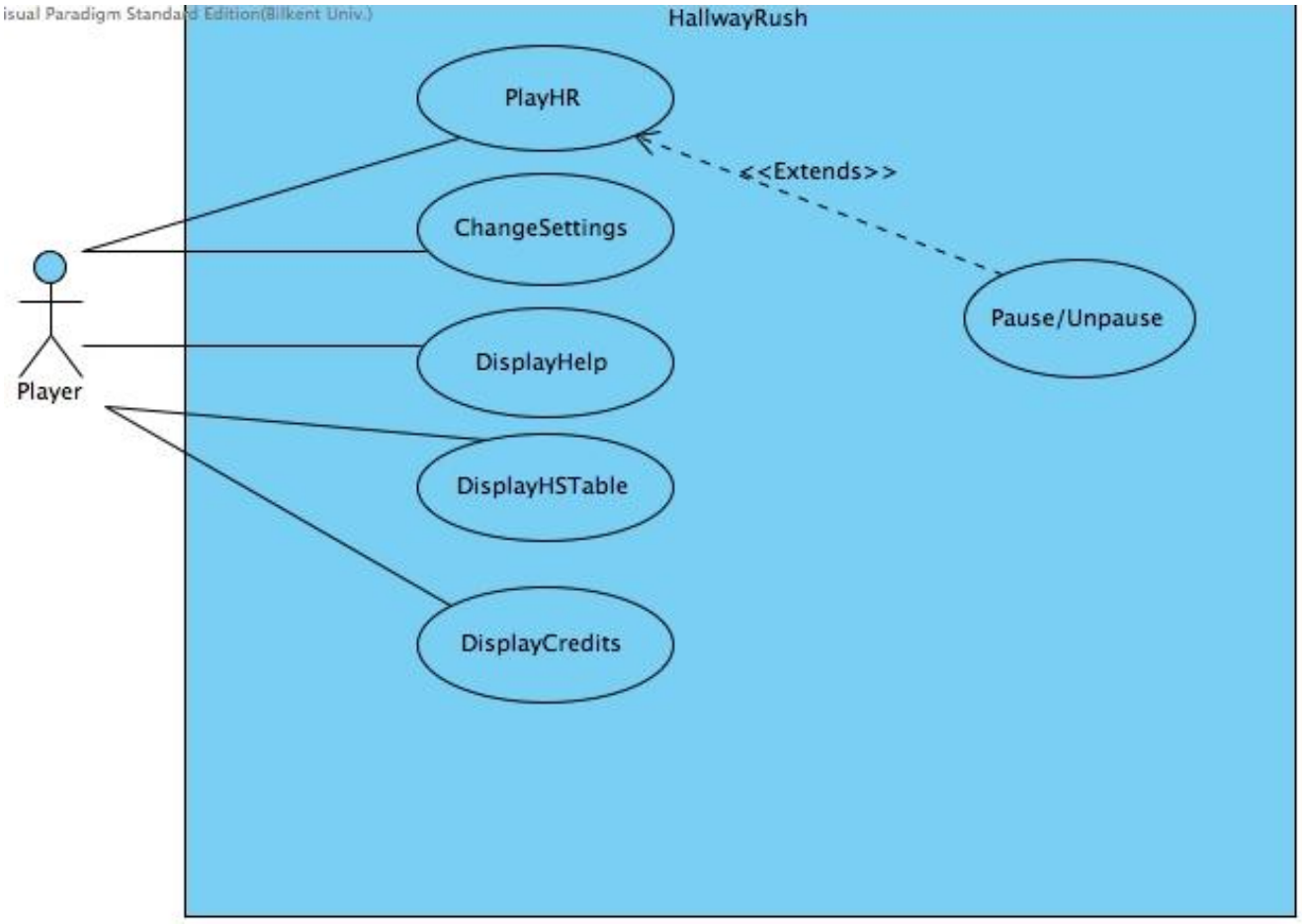


Figure1: Hallway Rush use-case diagram.

5.1.1. Use Case Descriptions

Use Case #1

Use Case Name: PlayHR

Participating Actors: Player

Pre-conditions: The user is on the opening screen.

Post-conditions: Either one of the two conditions below will end the game,

- The player has no lives remaining
- The player quits the game

If the player's score is high enough to enter the high scores list, the system updates the high scores list with the player's score.

Basic Flow of Events:

i. Player starts the game from main menu
ii. System constructs a game based on the default standard mode
iii. Player collects various bonuses
iv. As the player's score increases, new types of enemies and weapons spawn
v. Player collects the score needed to confront the game boss
vi. Player defeats the game boss and beats the game scenario
vii. System updates the high score status with a gold badge
viii. Player spends all of his lives
ix. System records player's score with a gold badge and a nickname, if it is a high score
x. System returns to the main menu

Alternative Flow of Events:

Alternative 1

i. Player starts the game in god mode
ii. Player quits the game
iii. System returns to main menu

Alternative 2

i. Player starts the game in standard mode
ii. Player collects various bonuses
iii. Player has no more lives before confronting the game boss
iv. System records player's score with his nickname but without a gold badge, if it is a high score
v. System returns to the main menu

Use Case #2:

Use Case Name: DisplayHelp
Participating Actors: Player
Pre-conditions: The player is on main menu
Post-conditions: System returns to the main menu when player clicks the return button

Basic Flow of Events:

i. Player clicks on 'Help' from the main menu
ii. Player reads the document and clicks 'Return to the main menu'
iii. System returns to the main menu

Use Case #3:

Use Case Name: DisplayHSTable
Participating Actors: Player
Pre-conditions: The player is on main menu
Post-conditions: System returns to the main menu when player clicks the return button

Basic Flow of Events:

i. Player clicks on 'High Scores' from the main menu
ii. Player looks at the scores and clicks 'Return to the main menu'
iii. System returns to the main menu

Use Case #4:

Use Case Name: ChangeSettings
Participating Actors: Player
Pre-conditions: The player is on main menu
Post-conditions: System returns to the main menu with the updated settings when the player clicks the return button

Basic Flow of Events:

i. Player clicks on ‘Change Settings’ from the main menu
ii. Player turns the sound off
iii. Player changes default ambiance(day time) to night time
iv. Player changes the main characters default gender(male) to female
v. Player changes the default standard mod to insane mod
vi. Player clicks ‘Apply’
vii. System updates the settings
viii. Player clicks the return button
ix. System returns to the main menu with the updated settings

Alternative Flow of Events:

i. Player clicks on ‘Change Settings’ from the main menu
ii. Player turns the sound off
iii. Player changes the character’s gender to female
iv. Player clicks ‘Apply’
v. System updates the settings
vi. Player clicks ‘Default Settings’
vii. System selects the default settings on the interface for the player
viii. Player clicks ‘Apply’
ix. System updates the settings as default settings
x. Player clicks the return button
xi. System returns to the main menu with the default settings

---

Use Case #5:

Use Case Name: DisplayCredits
Participating Actors: Player
Pre-conditions: The player is on main menu
Post-conditions: System returns to the main menu when the player clicks the return button

Basic Flow of Events:

i. Player clicks on 'Credits' from the main menu
ii. Player sees the credits and clicks the return button
iii. The system returns to the main menu

Use Case #7:

Use Case Name: Pause/Unpause
Participating Actors: Player
Pre-conditions: A game instance is running
Post-conditions: The game is on pause state or if it is already in pause, it returns to unpaused state

Basic Flow of Events:

i. Player clicks the pause button during the game
ii. System updates game's state to paused and it displays the pause screen
iii. Player clicks the unease button after certain time
iv. System updates game's state to unpaused and it resumes the game

---

5.2. Dynamic Models

Scenario 1: Starting a game with standard game mode

Player Ali selects the standard game mode from the main menu and starts a new game. The game frame creates a new game panel and the game engine creates the main character object with standard mode settings. Along with the main character, the game engine creates two zombie objects in the map. Game frame paints the objects on the panel and game engine starts the game loop

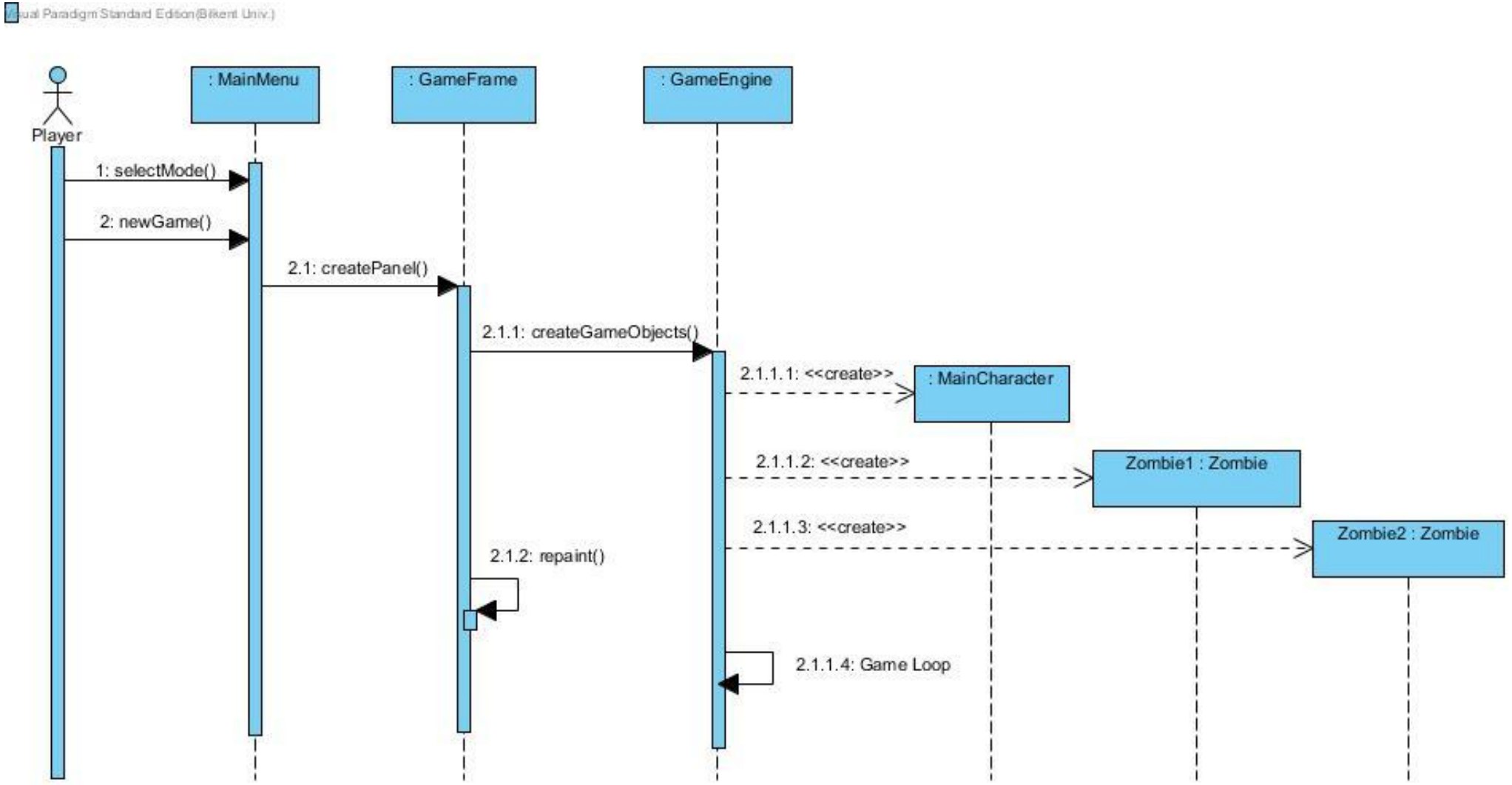


Figure 2: Sequence diagram of scenario 1.

Scenario 2: Firing a bullet

Player Ali presses the fire key on his keyboard, the keyboard handler detects which key is pressed and invokes the "fire()" method of the game engine. Then, game engine invokes the "createBullet()" method of the pistol – which is the current gun type of the main character – and pistol creates a bullet. Afterwards, the bullet moves once and the game frame class object "repaint()"s the main panel. Then the collision controller immediately checks that there is no collision between the bullet and another object. In this scenario, there is no collision so the bullet moves once more and game frame class object "repaint()"s again. Then, the scenario ends.

Scenario 3: Shooting a chest that has a health power-up

The collision manager checks whether there is collision between bullet1 and chest1 objects. In this scenario, there is. So, game engine destroys bullet1, invokes the "createRandomPowerup()" method of Chest. Chest creates a health power-up and returns it to the game engine so that it knows what type of power-up is created. Then, chest1 is destroyed and game frame "repaint()"s. The power-up moves once, game frame "repaint()"s again and the scenario ends.

Scenario 4: Getting a health power-up

The collision manager checks whether there is collision between mainChar and healthPU objects. In this scenario, there is. Game frame "repaint()"s the panel. Then, game engine invokes the "getHealthPoints()" method of Health Power Up and stores it as integer hp. Game engine destroys healthPU and sets the health of mainChar as the sum of its original health property and hp.

Scenario 5: Collision between a bullet and a zombie type mob

A bullet shot before by the player Ali's main character collides with a zombie and damages its health. Collision Controller checks the collision and returns true. game engine gets bullet1's damage – which is 20 in this scenario –, destroys it and sets zombie1's health as the subtraction of its original health by 20. Then, game

engine gets zombie1's health which is –5 and therefore destroys it too. Game engine adds 10 points to Ali's points. Game frame "repaint()"s.

Scenario 6: Collision between the main character and a zombie type mob

The main character collides with a zombie type mob. Collision Controller checks the collision and returns true. Then, game engine sets the move stats of the zombie and the main character false. Game frame "repaint()"s.

Scenario 7: Finishing the game with a gold medal

The player Ali has already beaten the game boss in this scenario and continued the game in order to beat the high score. He had a single life and he got hit so he lost his last life. In the beginning of the scenario, game engine checks player's health which returns 0. Then game engine gets the Ali's score and checks his medal status. Since Ali had beaten the game boss, main character object returns true to the game engine to note that Ali has a gold medal status. The game engine records Ali's score with a gold medal and ends the game.

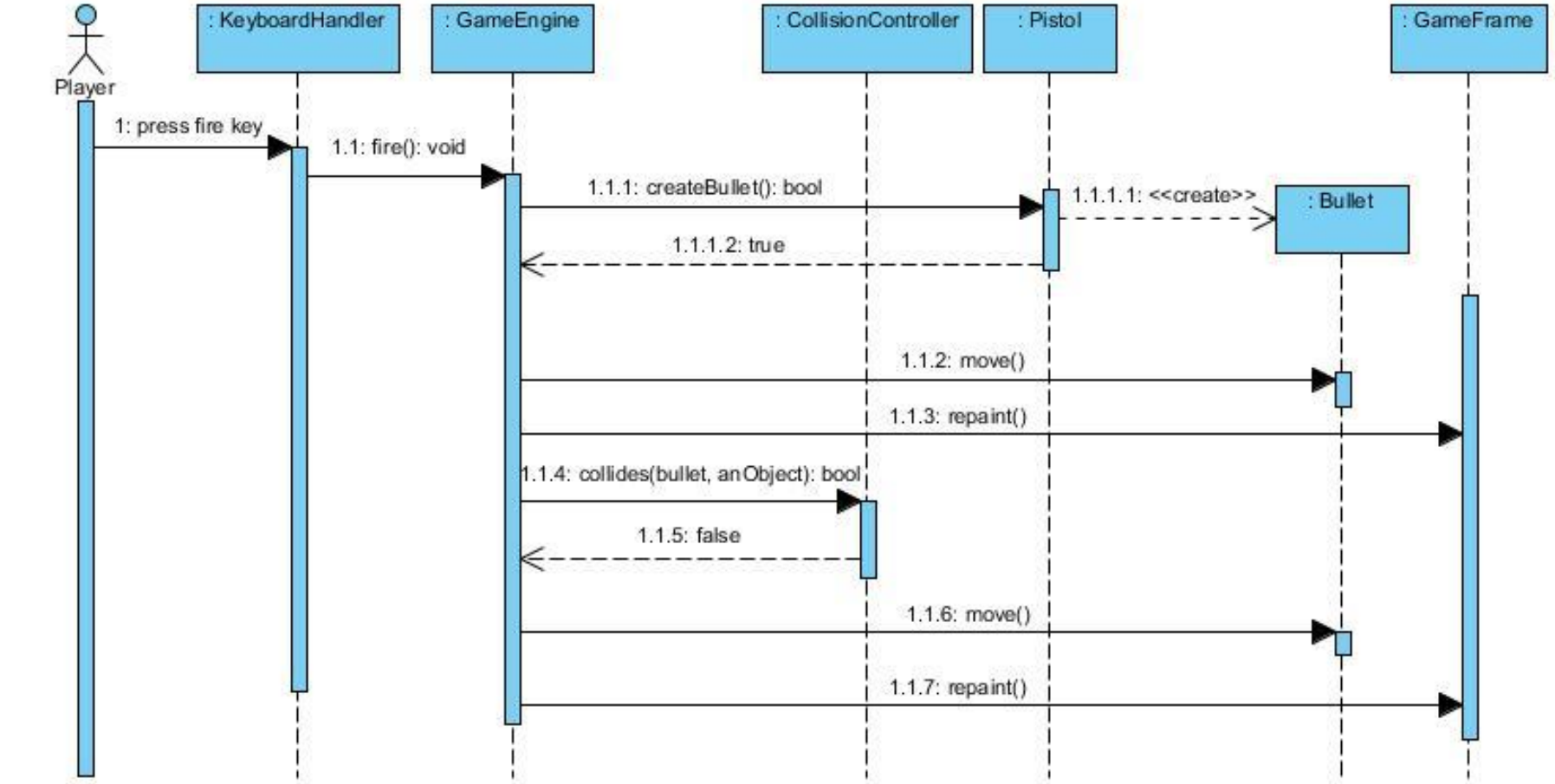


Figure3: Sequence diagram of scenario 2

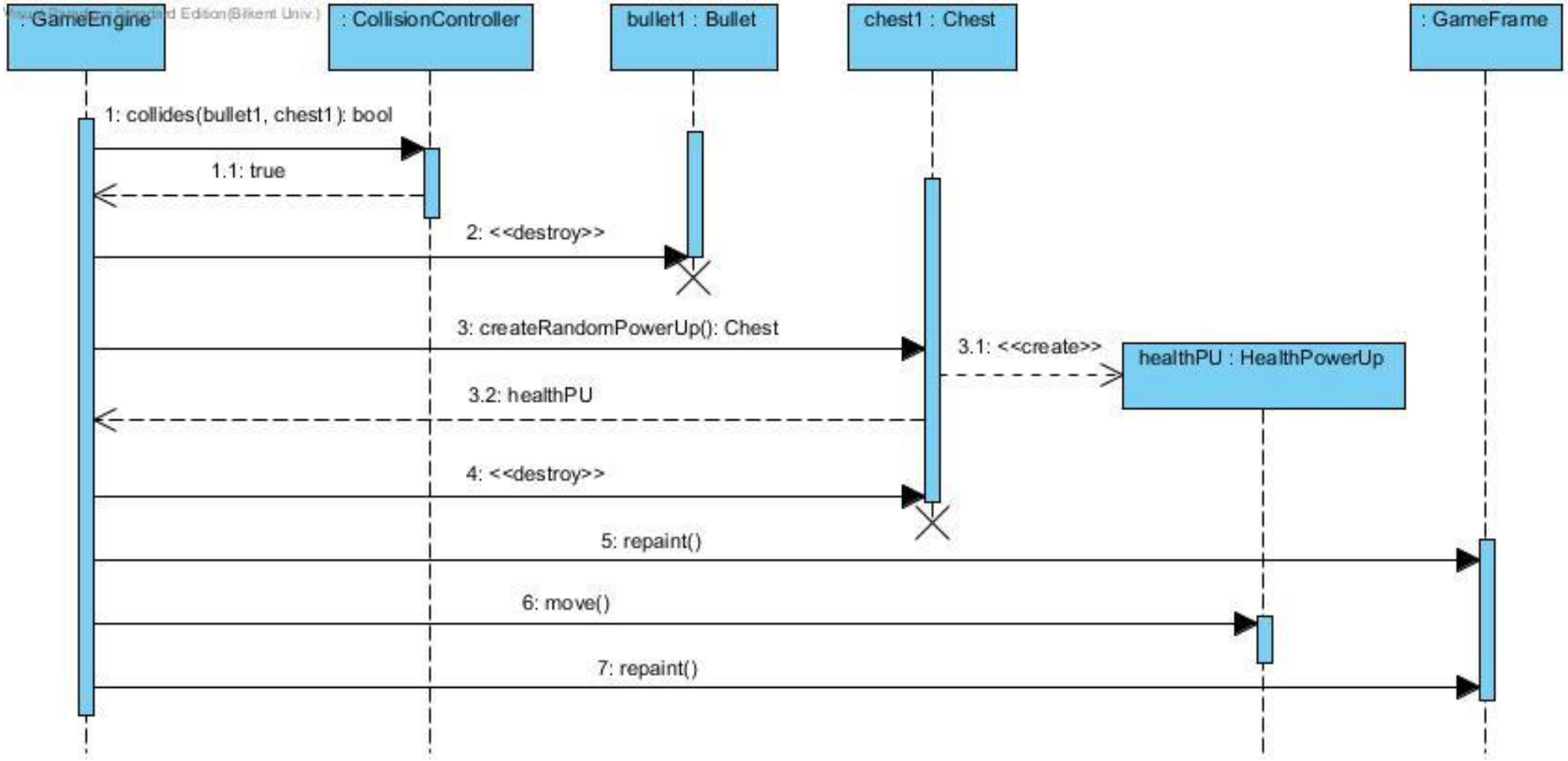


Figure4: Sequence diagram of scenario 3.

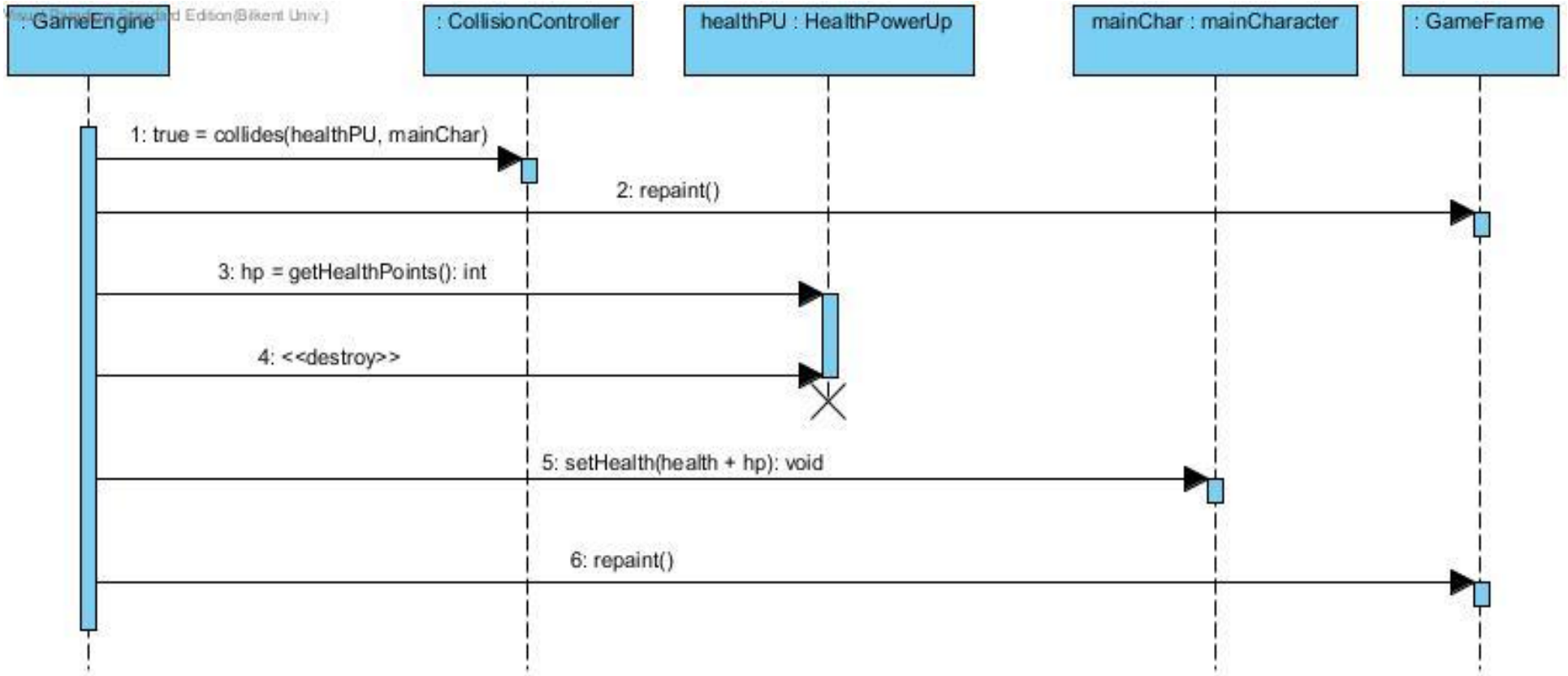


Figure5: Sequence diagram of scenario 4.

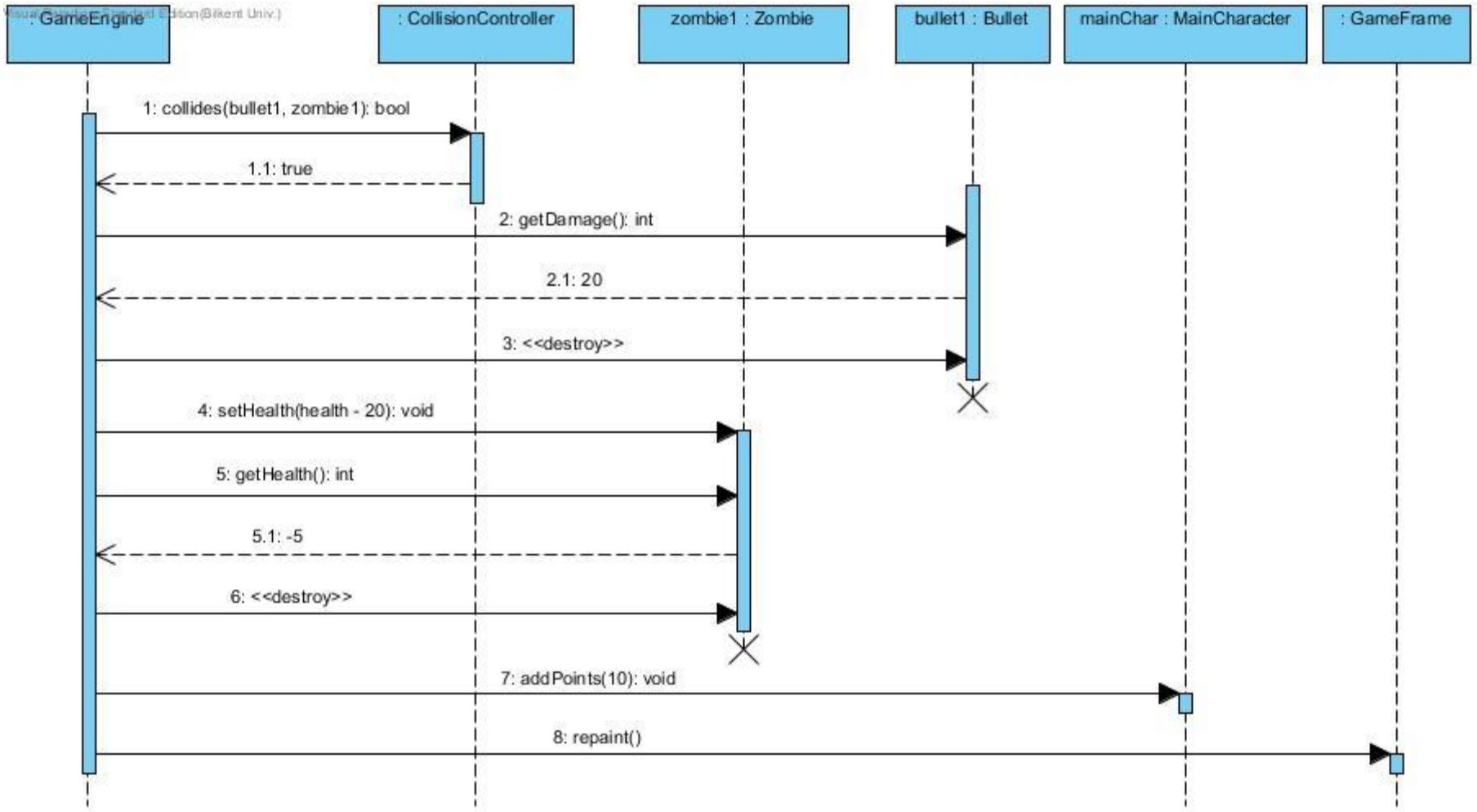


Figure6: Sequence diagram of scenario 5.

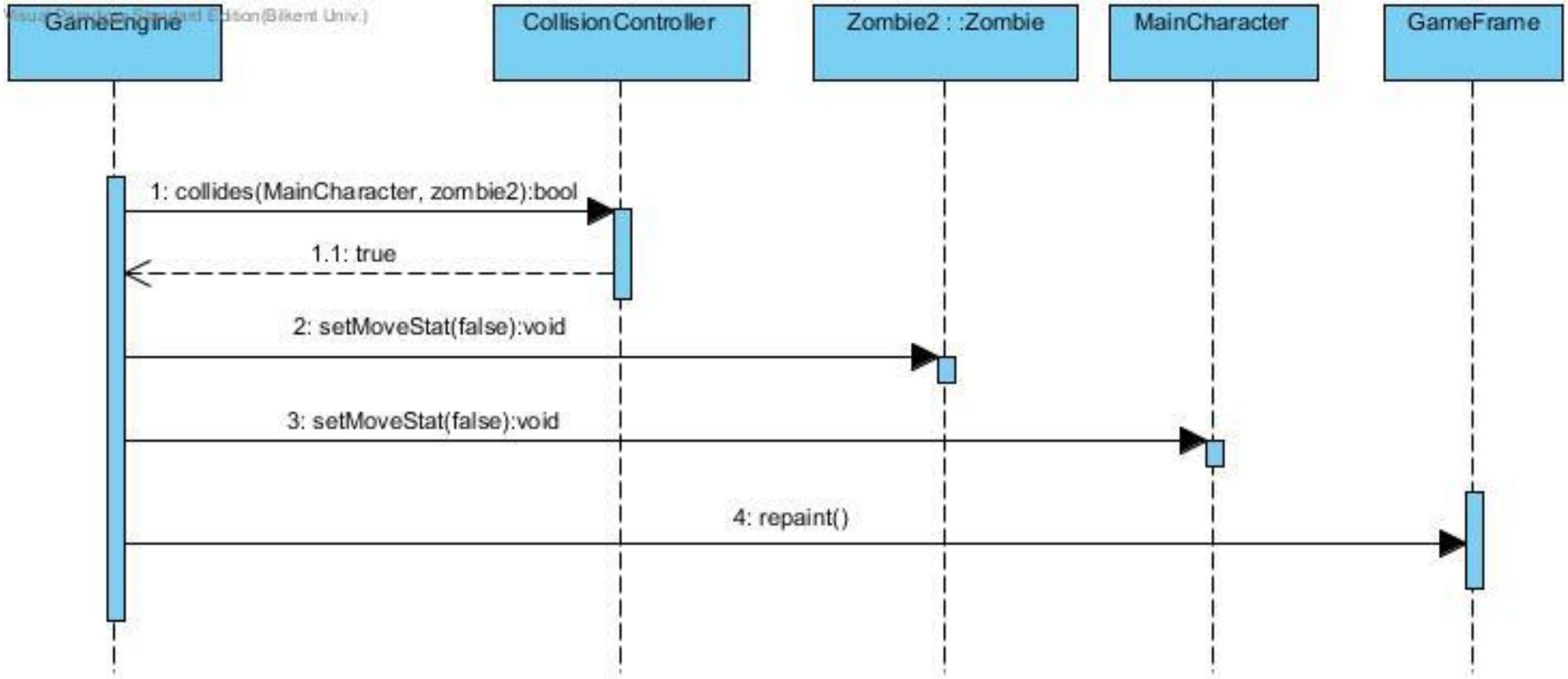


Figure7: Sequence diagram of scenario 6.

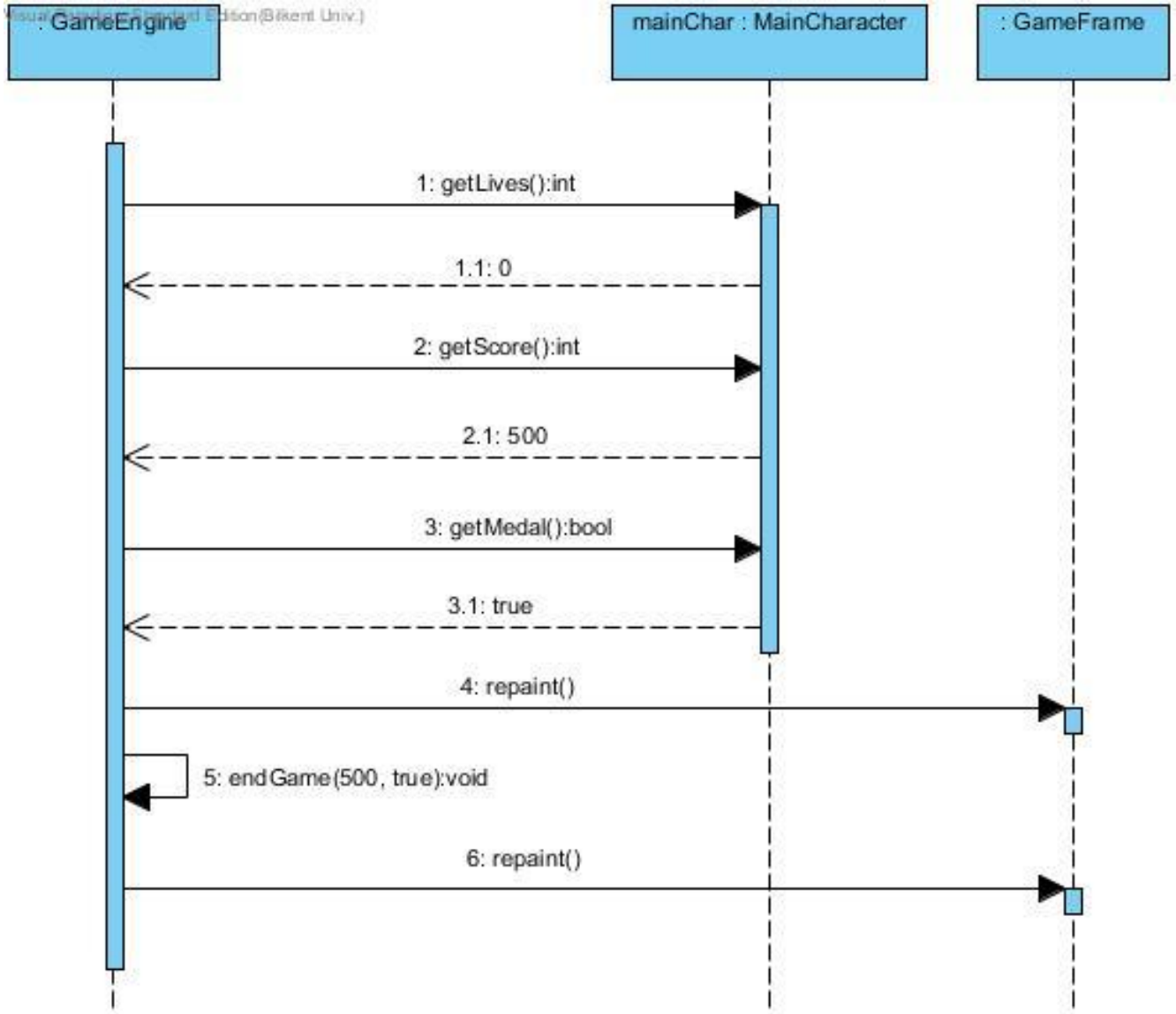


Figure8: Sequence diagram of scenario 7.

Below is an Activity Diagram that explains the workflow of Hallway Rush. It models the decisions and concurrencies in the construction of classes of the game.

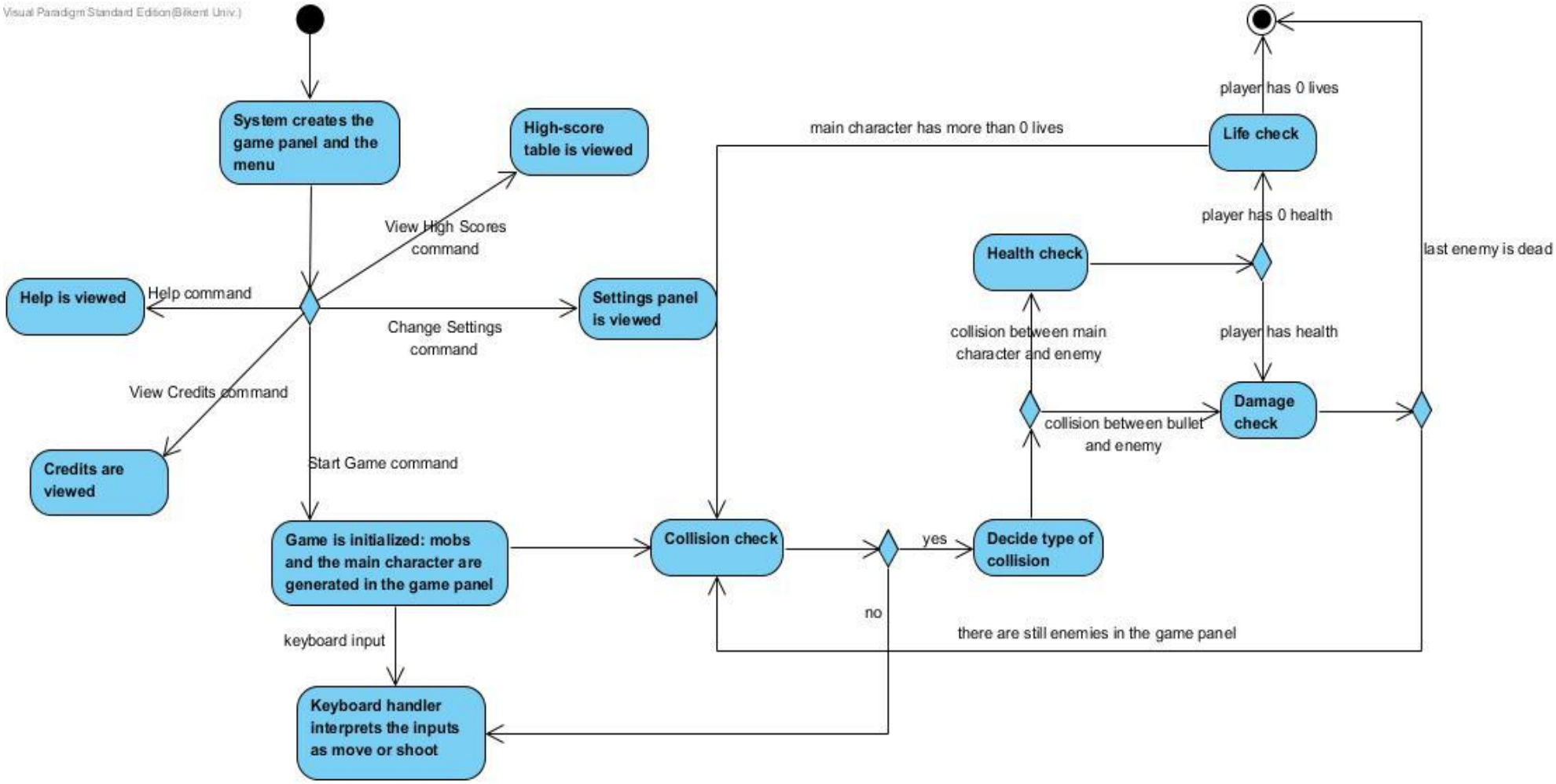


Figure9: Activity Diagram of Hallway Rush

5.3. Object and Class Model

The Class Model Diagram briefly shows relations between the classes of Hallway Rush.

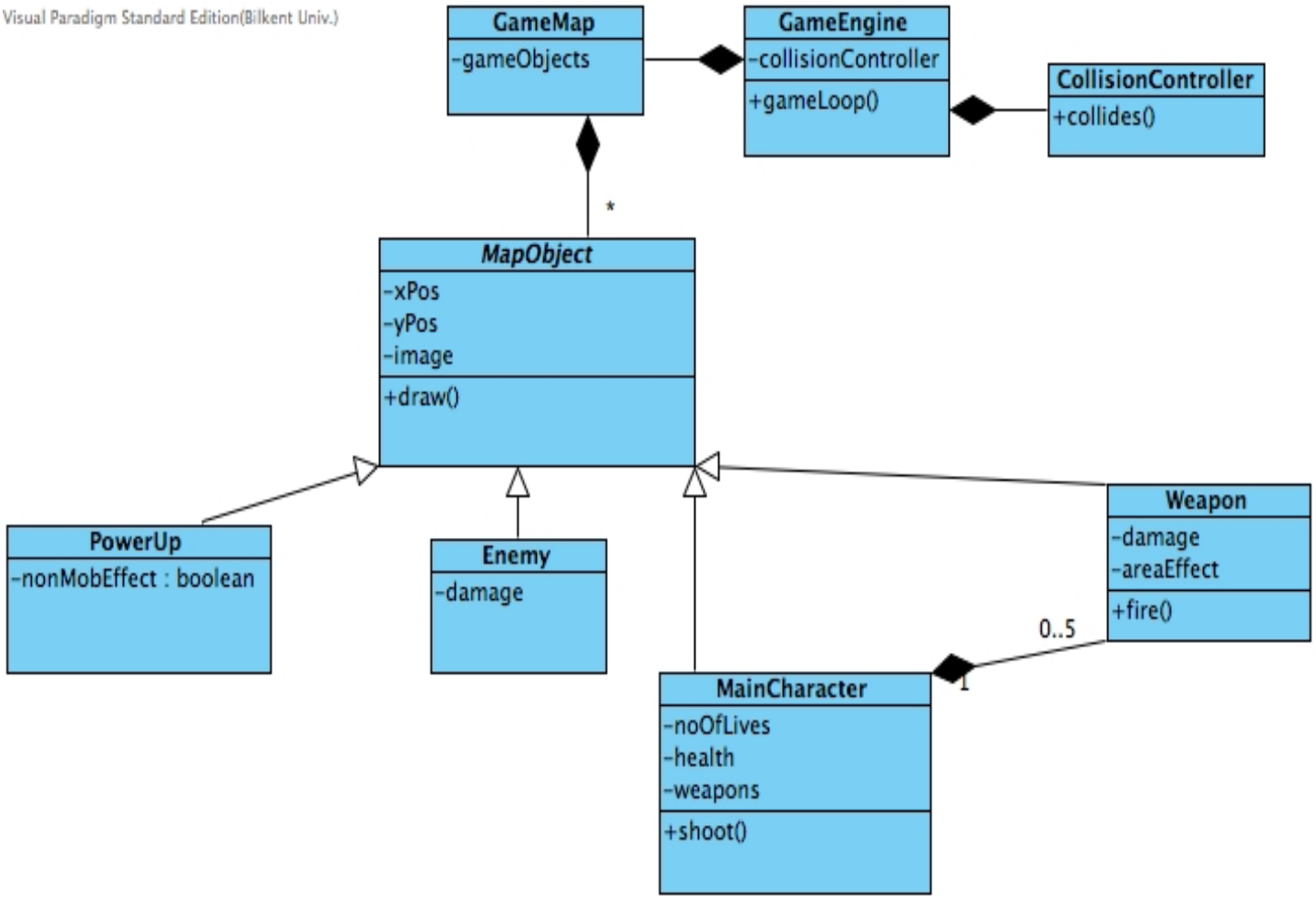


Figure10: Class Diagram of Hallway Rush.

5.4. User Interface

5.4.1. Main Menu

Main Menu welcomes player when game is launched. This screen contains a bunch of options such as start game, change game mode, view high scores, view help, change setting.

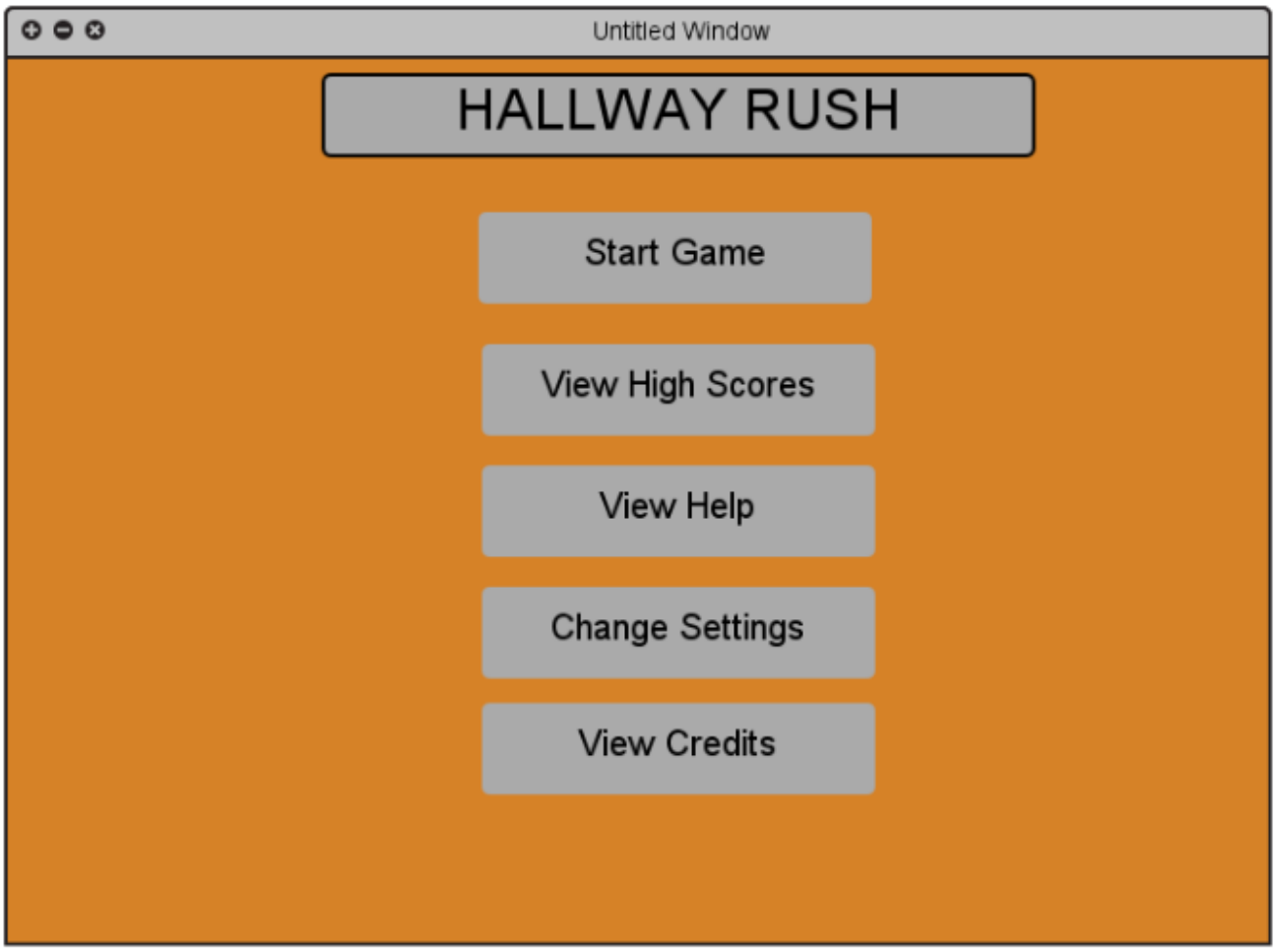


Figure11: Mockup of main menu.

5.4.2. Change Settings

Change Settings menu allows user to configure the main character of the game, game ambient and music of the game.

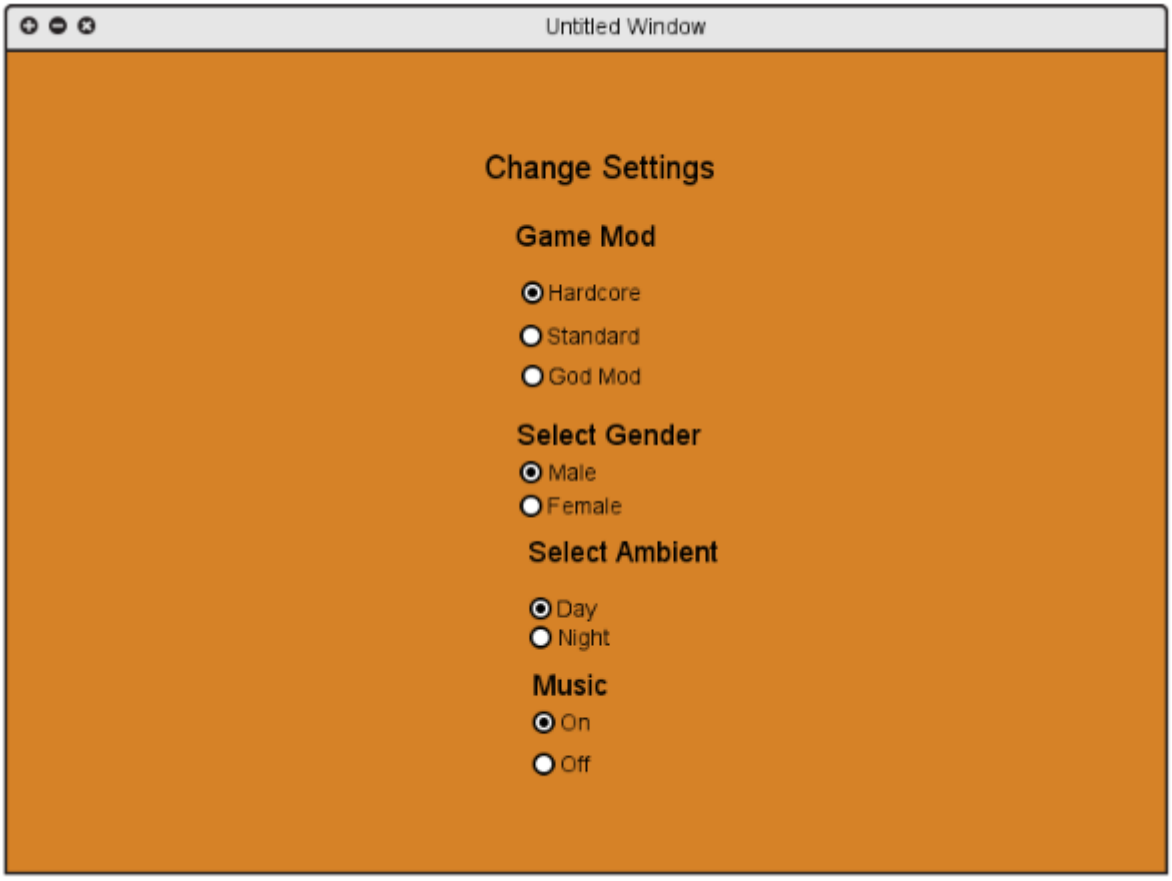


Figure12: Mockup of change settings

5.4.3. Gameplay

In gameplay screen, user can see main character's lives on the top left corner, main character's health indicator on top right corner. Also there is hud no at the bottom of the screen that shows available weapons to the main character.

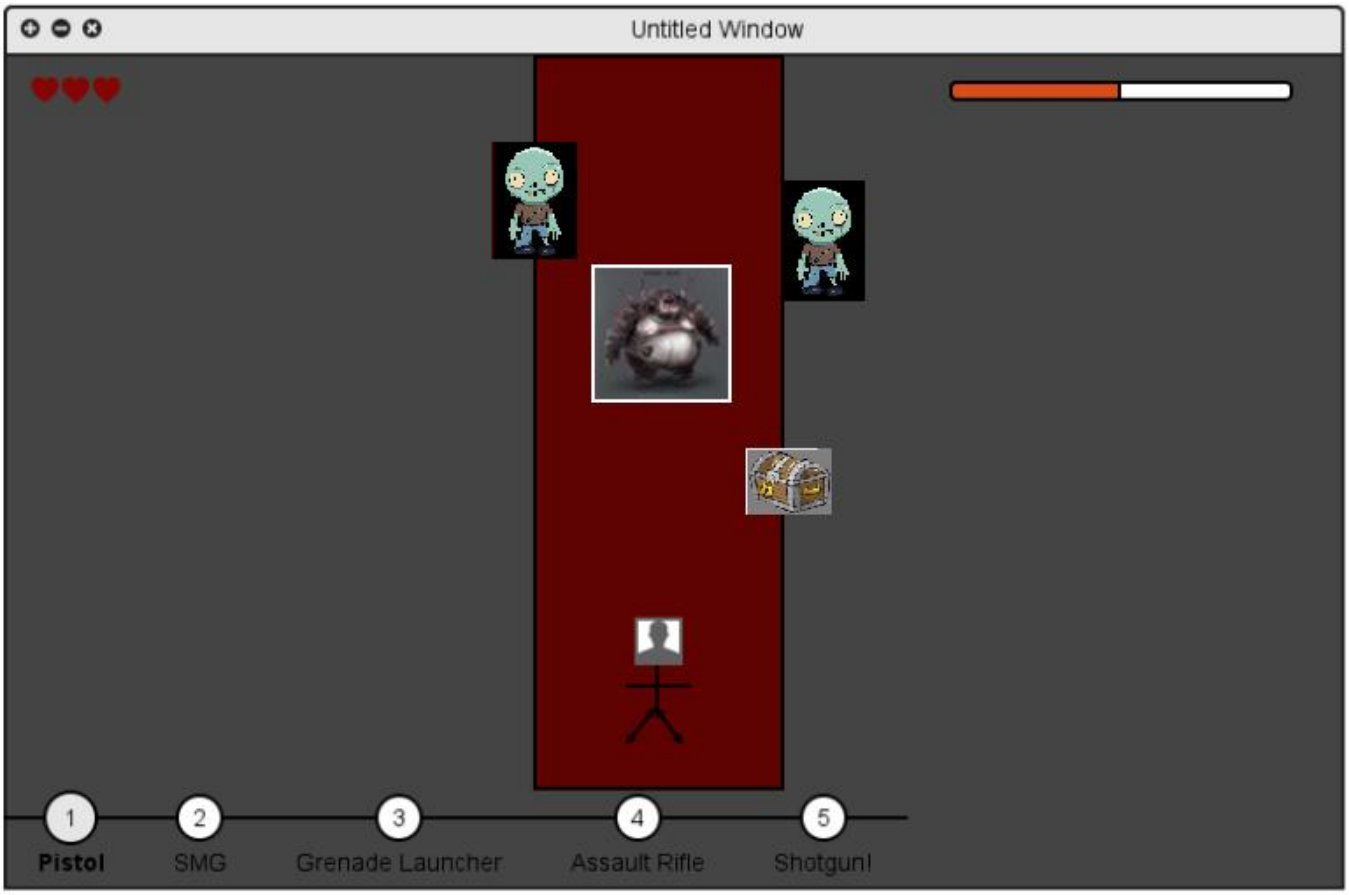


Figure13: Mockup of gameplay.

5.4.4. Game Images

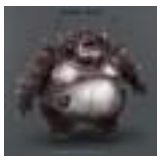
All the non-mobs will be photoshopped versions of the standard zombie in order to maintain the game consistent in sense of graphics.

[1] **Standard Zombie**

[2] **Juggernaut**

[3] **Armed Zombie**

[4] **Armored Zombie**



[5] **Baba O' Riley**

6. Glossary & references

[1] http://www.clipartlord.com/wp-content/uploads/2014/03/zombie8.png
[2] http://www.deviantart.com
[3] http://www.pinterest.com
[4] http://www.deviantart.com
[5] http://www.pinterest.com