Crime City

By

Mehwish

INSTITUTE OF INFORMATION TECHNOLOGY

QUAID-I-AZAM UNIVERSITY

ISLAMABAD

2019

Crime City

Thesis submitted to the Institute of Information Technology, Quaid-i-Azam University, Islamabad, for the partial fulfillment of the degree of

Master of Science

In

Information Technology

By

Mehwish

Supervised

By

Ms. Robina Rashid

INSTITUTE OF INFORMATION TECHNOLOGY

QUAID-I-AZAM UNIVERSITY

ISLAMABAD

2019

STATEMENT OF SUBMISSION

This is to certify that **Ms. Mehwish** Registration No. **01161711026** has successfully completed the final project as ***"Crime City"*** Quaid-i-Azam University, Islamabad to fulfill the partial requirement of the degree "**Master of Sciences in Information Technology".**

External Examiner

Dr. Malik Ahmad Kamran
Assistant Professor

Department of Computer Sciences
COMSATS, Islamabad.

Internal Examiner

Ms. Robina Rashid
Lecturer

Institute of Information Technology,
Quaid-i-Azam University, Islamabad.

PHONE# 051-9064 3084 Email: robinarashid@hotmail.com

Dedicated To

My beloved parents & inspiring Teachers for their endless support & encouragement.

ACKNOWLEDGEMENT

Firstly, I would like to pay my gratitude to **Almighty Allah** who gave me courage and patience to complete this project. I am thankful to my parents, specially my mother for her prayers, and my beloved father for his prayers and financial support during my educational career, whose love and affection has been inspirational throughout my life. Also thankful to my brothers and sisters for their help and prayers, who were always there for me with support and encouragement.

I wish to express my deepest appreciation to my supervisor Ms.Robina Rashid. My supervisor, Ms. Robina Rashid, whose sounding advice helped me steer this project in the right direction. Her supervision guided and supported me from initial to the final level and enabled me to develop an understanding of the project.

Finally, my classmates whose technical and moral support throughout my stay at the IT department was of great help.

MEHWISH

Abstract

The main objective of this project is to make a web based game .A man who is retired from army as a commando name "Jacob ".The crime city game based on the city which is fully control by the bad boys. These bad guys had created their threat among people. The complete characters are shown in camera and their physical movement are clearly defined. All the actions player movement, player killing enemies , player buy guns , collecting coins are well defined in this game. Mini map support is available to see direction of enemies. Time freezing is advanced in this game in which the player can take advantage and kill the enemies. We built .apk file for android game development.

Thesis Organization

The thesis is organized in the following manner:

Chapter 1 gives the complete overview of the project, organization and project planning activities. Chapter 2 comprise of the system requirement analysis. Chapter 3 focuses on the design phase of the system with its characteristics. Chapter 4 contains the system implementation details. Chapter 5 includes the interfaces of the system. Chapter 6 provides details about testing carried out for this project. . In the last portion of the thesis includes references.

Table of Contents

List of Figures

List of tables

Chapter 1

1.1 Project Introduction

A man who is retired from army as a commando name "Jacob". There is a crime city which is fully controlled with bad guys. These bad guys had created their threat among people so that they follow their rules in city. That's why the people of this city called the Jacob to kill all the bad guys by using his abilities and clean up the whole city. The game is basically the third person perspective game (The complete characters are shown in camera and all their physical movements).The game is based on low poly graphics so that it will work fine on all the android devices without any fps(frames per second) lag.

1.2. Existing Examples / Solutions

There are a number of applications that provide such kind of functionalities. In the following section, some of these are listed.

Watch dogs is an open world game made by ubisoft .The story is to take revenge of peoples which was killed by a gang by using his hacking abilities completely intractable with all objects vehicles , person , guns traffic signals.

Just cause is an also open world or action adventure environment game. The FBI agency meets with the Sheldon and they believe on it. The work that Sheldon given is to kill all the drugs dealer from Chicago and sent them all to jail. In just cause player can perform stunts, change vehicles, and perform skydiving.

The far cry was the first person open world game based on the historical environment in where the player is need to survive against wild animals .In far cry you need to complete all the side missions and optional quests.

Characteristics Detail:

- **Fully destruction:** All buildings which are shown in the map can be destroyed by shooting.
- **Mini-Map support:** The player spawn point , shop item logo are shown in map to easily identify the location.
- **Third person perspective:** The player and all physical movements are shown in camera.
- **Achievements:** The player will get reward after completing the certain objectives.
- **Shop items:** The player can buy the weapons from inventory depending upon the money. The inventory is static the player can only buy the weapons from inventory.
- **Time freeze:** The player use his ability to control the time to take advantage of killing enemies.

Table1.1 presents the comparison of features between watch dogs, just cause, far cry and Crime city.

Table 1.1: Existing Solutions features comparison

Sr no.	Characteristics	Watch dogs	Just cause	Far cry	Crime city
1	Fully destruction shown				✓
2	Mini-map support	✓	✓	✓	✓
3	Third-person perspective view	✓	✓		✓
4	use shop items	✓		✓	✓
5	Achievements.				✓
6	Time freezing				✓
7	Collectable items	✓		✓	✓

1.3 Business Scope

Mobile gaming's popularity has paralleled the widespread adoption of social media. Two-thirds of adults use social networks, and nearly 90% of cell-phone owners use smart phones. This game also would be developed by targeting the interest of youngsters. The game would be easy to play and be very interesting for action lover game players as it have low graphics, more stages and many acids in it at every stage which makes it interesting.

1.4 Useful Tools and Technologies

C#, java script, JDK unity3d, blender, Photoshop can be used to develop this game.

Android studio provides portability to build the android game.

Java JDK is use to build the game.

Unity3d is the game development platform to build the high quality game in 3d and 2d and deploy them on desktop, web, android, Desktop etc.

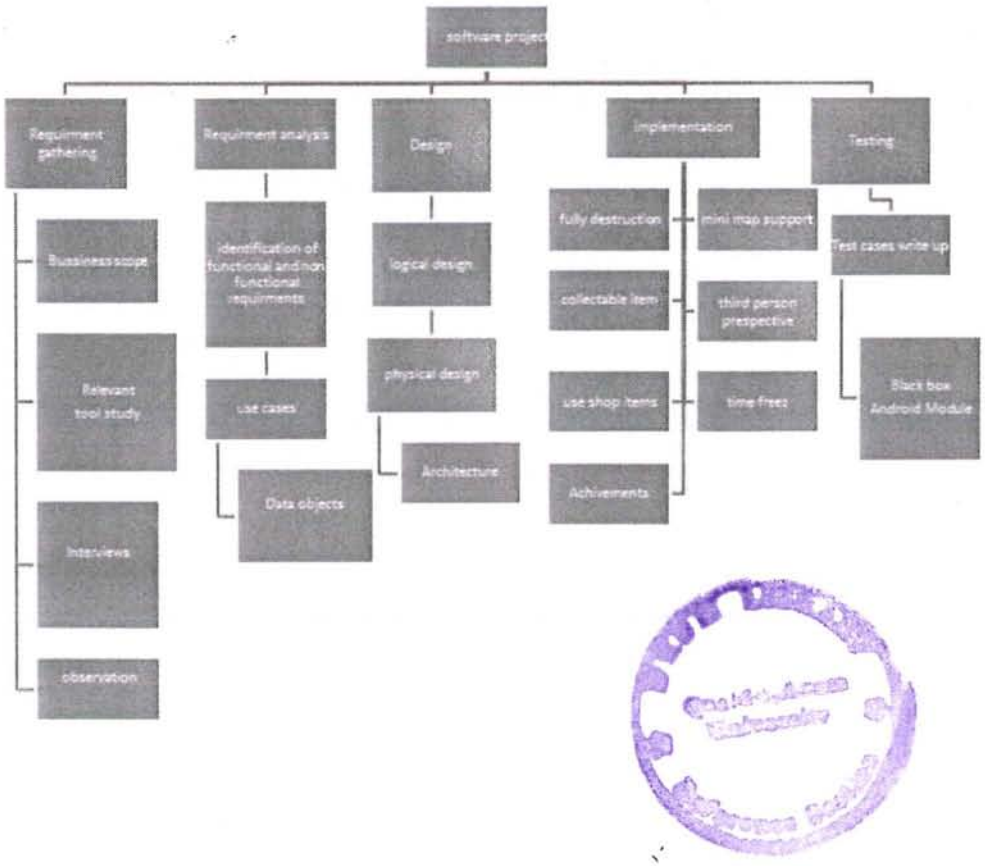
Visual studio 2015 supports different languages like c++, and c #and java script.

Blender is free open world modeling tools to model high quality animation, characters, and 3D-Models.

The Photoshop is use to edit the effects and designs the good user menu interface.

1.3 Project Work Break Down

The project breakdown structure is shown in Figure 1.1



1.3 Project Time Line

The proposed project time line is shown in the following figure 1.2.

Requirement gathering

Oct 1, 2018

October	November	December	January	February
Requirement Gathering Start :2-2018	**Requirement analysis Start:7-2018**	**Designing Start:27-2018**	**Implementation Start: 3-2019**	**Testing Start:2-2019**

Chapter 2

Requirement Specification and Analysis

This chapter includes use case diagram, use case description and system sequence Diagrams for the requirements selected/revised in the current iteration.

2.1. Functional Requirements

The functional requirements are listed in the following table.

Table 2.1: Functional Requirements

S. No.	Functional Requirement	Type	Status
1	Player can move defender forward.	Core	Completed
2	Player can move defender backward.	Core	Completed
3	Player can move defender left.	Core	Completed
4	Player can move defender right.	Core	Completed
5	Player can move defender to jump.	Core	Completed
6	Player can aim buildings.	Core	Completed
7	Player can kill enemies.	Core	Completed
8	Player can use time freeze.	Core	Completed
9	Player can drive vehicle.	Intermediate	Completed
10	Buildings health status will be	Core	Completed

	shown to defender.		
11	Player health status will be view to defender.	Core	Completed
12	Enemies' health would be shown to defender.	Core	Completed
13	Player can usemenu to change setting.	Core	Completed
14	Player can use menu to start game.	Core	Completed
15	Player can use menu to quit game.	Core	Completed
16	Player can stop the game by pressing ESC button.	Core	Completed
17	Player can resume the game by pressing resume button.	Core	Completed
18	Player can restart the game by pressing restart button.	Core	Completed
19	Player can buy grenade from shop by coins.	Core	Completed
20	Player can buy jetpack from shop by coins.	Core	Completed
21	Player can select level 1.	Core	Implemented
22	Player can select level 2.	Core	Implemented
23	Player can select level 3.	Core	Implemented
24	Player can view mini map.	Core	Completed
25	Player can collect money by killing enemies.	Core	Completed
26	Player can view money on shop screen.	Core	Completed
27	Player can view money while playing.	Core	Completed
28	Player can change sound setting	Intermediate	Completed

2.2. Functional Requirements

Table 2.2: Functional Requirement

S. No.	Functional Requirements	Category
1	Guns will have unlimited bullets and require no reloading.	System level
2	Buying a weapon from shop will add weapon in weapon inventory.	System level
3	The player name entered by the user will be used in the game levels.	User level
4	The player will be able to move in all directions.	User level
5	The collected coins will be added in the score inventory.	System level
6	The player will die when hit by bullets.	User level
7	The enemies will be able to shoot if player is in front of them.	System level
8	The bullet fire and enemies movement will be slow down while player will press time freeze button.	System level
9	The level will automatically end when the player dies.	System level
10	The player name can be long of 16 characters maximum and 1 character minimum.	System level
11	The player will be able to switch weapons if available in the weapons inventory.	User level
12	Each level will consist of unique set of objectives to be completed.	System level
13	Restarting the level will result in starting the level from beginning with new score board.	System level

2.3 Selected Functional Requirements

List of selected functional requirements for current iteration.

Table 2.3: Selected Functional Requirement

S. No.	Functional Requirement	Type
1	Player can drive vehicle.	Intermediate
2	Buildings health status will be shown to defender.	Core
3	Player can buy grenade from shop by coins.	Core
4	Player can buy jetpack from shop by coins.	Core
5	Player can select level 1.	Core
6	Player can select level 2.	Core
7	Player can select level 3.	Core
8	Player can change sound setting	Intermediate

2.4 System Use Case Modeling

A use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as can actor) and a system, to achieve a goal .The actor will be human.

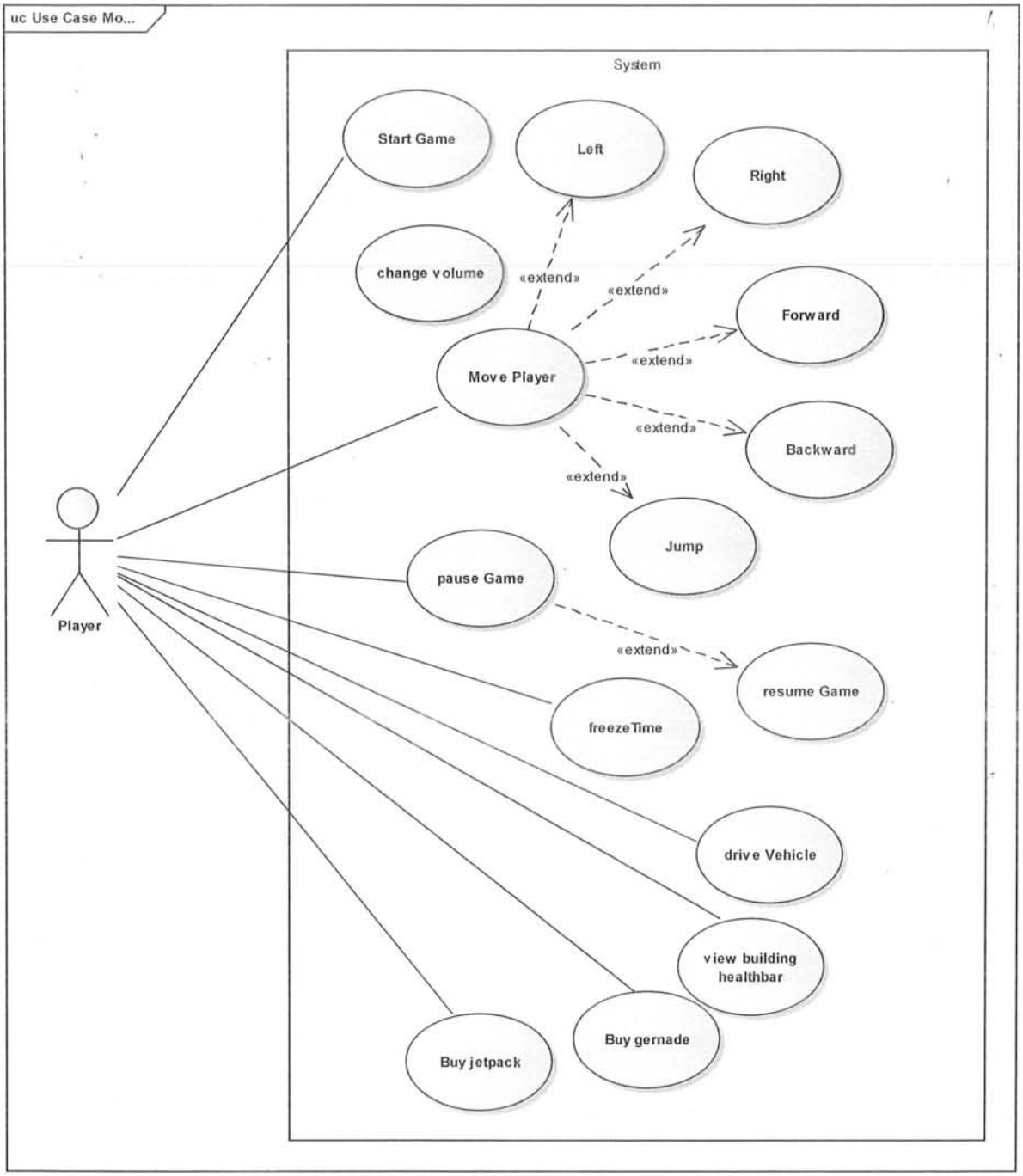


Figure 2.1(a): Sample Use case Diagram

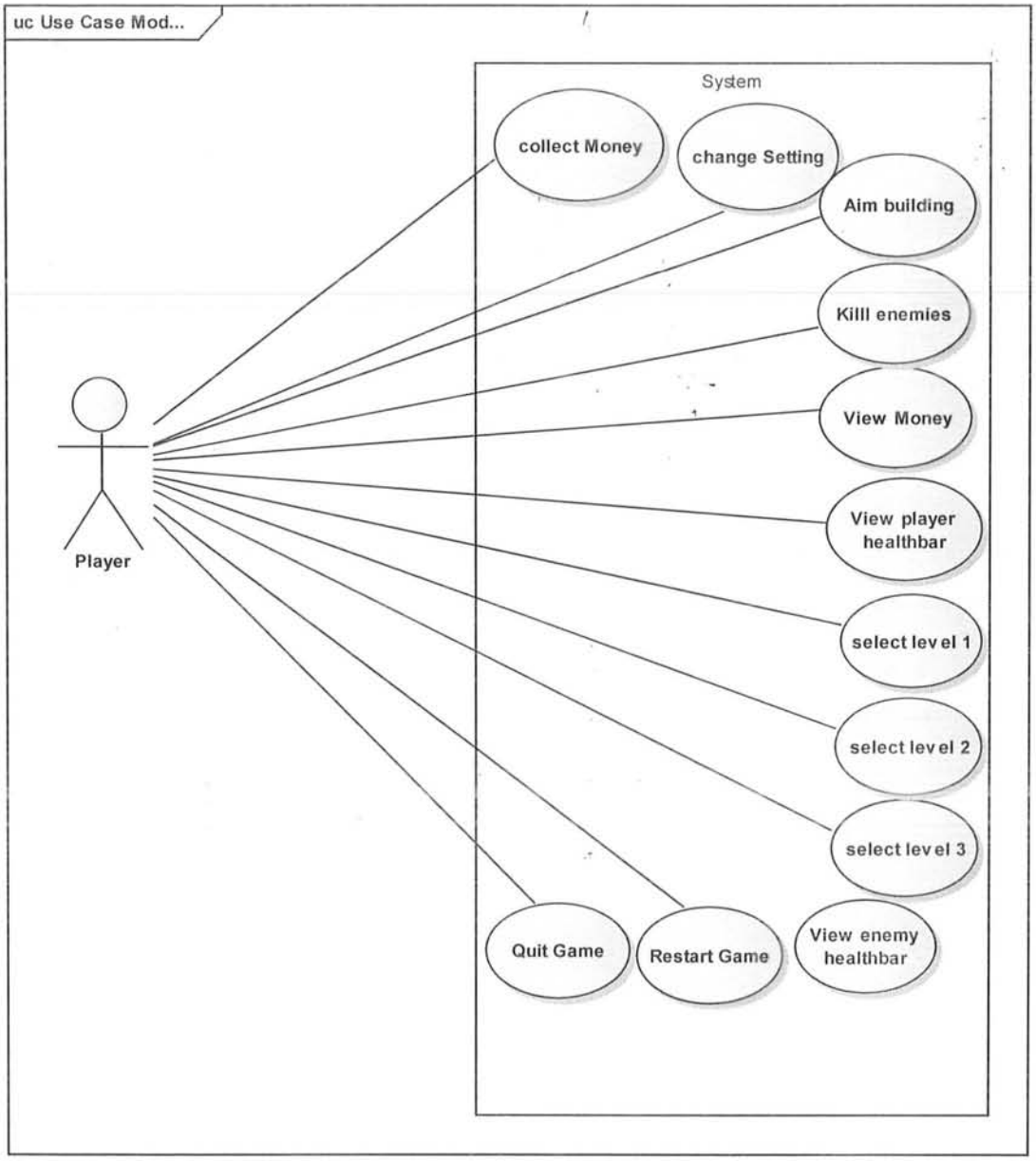


Figure 2.1(b): Sample Use case Diagram

2.5 Use/Case Description:

The use case description for the above use case model is provided in the following.

Table 2.4: Start Game

Use Case ID:	01		
Use Case Name:	Start game		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player select play option from main menu to start the game		
Trigger:	Play Button		
Preconditions:	The game must be installed on the android platform		
Post conditions:	The game will be in running state		
Normal Flow:	1: the player installs the apk file of the game on device 3:the player choose the play game option to play the game	2: the system will load the game and show the main menu to the player 4:System start the game for player	
Alternative Flows:	Reinstall the game		
Exceptions:	If any error occur during loading, system display error message and request the player to reinstall or restart the game		

Table 2.5: left move

Use Case ID:	02		
Use Case Name:	Left		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	User move the player toward left.		
Trigger:	Swipe Joystick Left.		
Preconditions:	The game must be in running state.		
Post conditions:	Player will move toward left.		
Normal Flow:	1:user move the player toward left.	2:system load component require for movement and move the player leftward.	
Alternative Flows:	Move player in another direction.		
Exceptions:	There is no way to move left side.		

Table 2.6: Right Move

Use Case ID:	03		
Use Case Name:	Right		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	User move the player toward right.		
Trigger:	Swipe Joystick Right		
Preconditions:	The game must be in running state.		
Post conditions:	Player will move toward right.		
Normal Flow:	1:user move the player toward right.	2:system load component require for movement and move the player rightward.	
Alternative Flows:	Move player in another direction.		
Exceptions:	There is no way to move right side.		

Table 2.7: Forward

Use Case ID:	04		
Use Case Name:	Forward		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	User move the player forward.		
Trigger:	Swipe Joystick Upward		
Preconditions:	The game must be in running state.		
Post conditions:	Player will move forward.		
Normal Flow:	1:user move the player forward.	2:system load component require for movement and move the player forward.	
Alternative Flows:	Move player in another direction.		
Exceptions:	On forward side there is no way.		

Table 2.8: Backward

Use Case ID:	05		
Use Case Name:	Backward		
Created By:	Mehwish	Last Updated By:	Mehwish
Date Created:	15-10-2018	Last Revision Date:	2-1-2019
Actors:	Player		
Description:	Swipe Joystick Backward		
Trigger:	Player will swipe the joystick in downward direction.		
Preconditions:	The game must be in running state.		
Post conditions:	Player will move forward.		
Normal Flow:	1:user move the player Backward.	2:system load component require for movement and move the player Backward.	
Alternative Flows:	Move player in another direction.		
Exceptions:	On backward there is no path.		

Table 2.9: Jump

Use Case ID:	05		
Use Case Name:	Jump		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	25-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	User move the player upward.		
Trigger:	Jump Button		
Preconditions:	The game must be in running state.		
Post conditions:	Player will jump.		
Normal Flow:	1:User press the jump button.	2:system load the required component for jump.	
Alternative Flows:	No Alternative.		
Exceptions:	On upward side there is not enough space to jump.		

Table 2.10: Pause Game

Use Case ID:	06		
Use Case Name:	Pause game		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	25-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player clicks Pause button from screen to pause the game		
Trigger:	Pause Button		
Preconditions:	The game must be in running state.		
Post conditions:	The game will pause.		
Normal Flow:	1:Player select the pause button to pause the game.	2:System pause the game	
Alternative Flows:	No alternative		
Exceptions:	No exception		

Table 2.11 Resume Game

Use Case ID:	07		
Use Case Name:	Resume game		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	25-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player clicks resume button from menu to resume the game		
Trigger:	Resume Button		
Preconditions:	The game must be in resume state.		
Post conditions:	The game will resume.		
Normal Flow:	1:Player select the resume option to pause the game.	2:System resume the game.	
Alternative Flows:	Start game again.		
Exceptions:	Game would be stuck.		

Table 2.12: Freeze Time

Use Case ID:	08		
Use Case Name:	Freeze Time		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	25-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can freeze time by pressing button.		
Trigger:	Freeze Time Button		
Preconditions:	The game must be in running state.		
Post conditions:	The time will slowdown.		
Normal Flow:	1:Player select the time freeze button to freeze the time.	2:System freeze the time.	
Alternative Flows:	No alternative.		
Exceptions:	Freeze ability already been used.		

Table 2.13: Aim Building

Use Case ID:	09		
Use Case Name:	Aim Building		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	25-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can aim buildings around.		
Trigger:	Move Joy Stick		
Preconditions:	The game must be in running state.		
Post conditions:	Target will set on the building.		
Normal Flow:	1:Player moves the joy stick to aim buildings around.	2:system sets the target.	
Alternative Flows:	No alternative		
Exceptions:	No exception		

Table 2.14: Quit Game

Use Case ID:	10		
Use Case Name:	Quit game		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	25-10-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player clicks Quit button to Quit the game		
Trigger:	Quit Button		
Preconditions:	The game must be in running state.		
Post conditions:	The game will stop.		
Normal Flow:	1:Player select the Quit button to quit the game.	2:System quit the game.	
Alternative Flows:	No alternative		
Exceptions:	No exception		

Table 2.15: Change Setting

Use Case ID:	11		
Use Case Name:	Change Setting		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	20-11-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can use menu to change the setting		
Trigger:	Change Setting Button		
Preconditions:	The game must be in running state.		
Post conditions:	The game settings will change.		
Normal Flow:	1:Player clicks the change setting button on the main menu. 3:Player changes the required setting. 4:Player press the save button to save setting.	2:Systemshows setting menu to the player. 5:System save the settings for the player.	
Alternative Flows:	Game stay at default settings.		
Exceptions:	No exceptions.		

Table 2.16: Restart Game

Use Case ID:	12		
Use Case Name:	Restart Game		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	20-11-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player clicks the restart button to restart the game.		
Trigger:	Restart Game Button		
Preconditions:	The game must be in running state.		
Post conditions:	The game will restart.		
Normal Flow:	1:Player clicks the restart button to restart the game.	2:System loads component and restart the game for the player.	
Alternative Flows:	Reinstall the game.		
Exceptions:	The game stuck.		

Table 2.17: View Money

Use Case ID:	13		
Use Case Name:	View Money		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	20-11-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can view collected money on the phone screen while playing.		
Trigger:	View screen		
Preconditions:	The game must be running and player is playing.		
Post conditions:	Money will updates in parallel.		
Normal Flow:	1:Player use different buttons available to play the game.	2:System shows the result or collected money on the top of the screen.	
Alternative Flows:	Kill enemies to collect money.		
Exceptions:	There is no collected money.		

Table 2.18: Collect Money

Use Case ID:	14		
Use Case Name:	Collect Money		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	20-11-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can view collected money on the shop screen after the game ends.		
Trigger:	Shop Item button		
Preconditions:	The game must be in running state.		
Post conditions:	Collected money will be visible to the player.		
Normal Flow:	1:Player clicks the shop item buttons to view the collected money.	2:System shows the result or collected money on the screen.	
Alternative Flows:	Play the game and collect some money first.		
Exceptions:	You don't have any collected money yet.		

Table 2.19: kill Enemies

Use Case ID:	15		
Use Case Name:	kill Enemies		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	20-11-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can aim the enemies and kill them.		
Trigger:	Mouse left click		
Preconditions:	The game must be in running state.		
Post conditions:	Target will set on the enemies.		
Normal Flow:	1:Player press mouse left click to kill enemies around.	2:system .will destroy the targeted enemy.	
Alternative Flows:	No alternative		
Exceptions:	No exception		

Table 2.21: Buy Grenade

Use Case ID:	16		
Use Case Name:	Buy Grenade		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can buy grenade from shop by coins.		
Trigger:	Buy button		
Preconditions:	The player played the game and have some coins.		
Post conditions:	Grenade will be available to the player during game.		
Normal Flow:	1:Player go to the shop menu. 3:Player select the Grenade from the available items. 4:Player press the buy button to buy grenade.	2: system will show the available shop items. 5: System check the player coins, if coins are enough then grenade will be added to the player shooting items otherwise show the message "Not enough coins".	
Alternative Flows:	Play the game and collect some coins.		
Exceptions:	No exception		

Table 2.22: Buy Jetpack

Use Case ID:	17		
Use Case Name:	Buy Jetpack		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can buy jetpack from shop by coins.		
Trigger:	Buy button		
Preconditions:	The player played the game and have some coins.		
Post conditions:	Jetpack will be available to the player during game.		
Normal Flow:	1:Player go to the shop menu. 3:Player select the jetpack from the available items. 4:Player press the buy button to buy jetpack.	2:system .will show the available shop items. 5:System check the player coins, if coins are enough then jetpack will be added to the player shooting items otherwise show the message "Not enough coins" .	
Alternative Flows:	Play the game and collect some coins.		
Exceptions:	No exception		

Table 2.23: Select level 1

Use Case ID:	15		
Use Case Name:	Select level 1		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can select the level 1 to play level 1.		
Trigger:	Level 1		
Preconditions:	The game should be installed.		
Post conditions:	Level 1 is loaded and ready for playing.		
Normal Flow:	1:Player load the game. 3:Player select the level 1. 5:Player start playing level 1.	2:system .will show the available levels to the player. 4:System load the components and start the level 1 for playing. .	
Alternative Flows:	Restart the game.		
Exceptions:	Level 1 loading failed.		

Table 2.23: Select level 2

Use Case ID:	15		
Use Case Name:	Select level 2		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can select the level 1 to play level 2.		
Trigger:	Level 2		
Preconditions:	The game should be installed.		
Post conditions:	Level 2 is loaded and ready for playing.		
Normal Flow:	1:Player load the game. 3:Player select the level 2. 5:Player start playing level 2.	2:system .will show the available levels to the player. 4:System load the components and start the level 2 for playing. .	
Alternative Flows:	Restart the game.		
Exceptions:	Level 2 loading failed.		

Table 2.23: Select level 3

Use Case ID:	15		
Use Case Name:	Select level 3		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can select the level 1 to play level 3.		
Trigger:	Level 3		
Preconditions:	The game should be installed.		
Post conditions:	Level 3 is loaded and ready for playing.		
Normal Flow:	1:Player load the game. 3:Player select the level 3. 5:Player start playing level 3.	2:system .will show the available levels to the player. 4:System load the components and start the level 3 for playing. .	
Alternative Flows:	Restart the game.		
Exceptions:	Level 3 loading failed.		

Table 2.23: Drive vehicle

Use Case ID:	15		
Use Case Name:	Drive vehicle		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can drive the vehicle on the road.		
Trigger:	Joystick		
Preconditions:	Player should be inside the vehicle.		
Post conditions:	Vehicle start moving according to player input.		
Normal Flow:	1: Player press the button to sit in the vehicle. 3: Player move vehicle using joystick.	2: System adjust the player and vehicle. 4: System move the vehicle accordingly.	
Alternative Flows:	Player walk.		
Exceptions:	No exception		

Table 2.23: View building Health bar

Use Case ID:	15		
Use Case Name:	View building health bar		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	15-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can view the building health status shown on the top of every building.		
Trigger:	View screen		
Preconditions:	The game must be in running state.		
Post conditions:	Building health bar will be visible to the player.		
Normal Flow:	1:Player start playing the game.	2:system .will show the building health bar on the top of each bilding.	
Alternative Flows:	No Alternative		
Exceptions:	No exception		

Table 2.24: Volume setting

Use Case ID:	15		
Use Case Name:	Volume setting		
Created By:	Mehwish	**Last Updated By:**	Mehwish
Date Created:	25-12-2018	**Last Revision Date:**	2-1-2019
Actors:	Player		
Description:	Player can control the volume of the game.		
Trigger:	Volume Button		
Preconditions:	The game must be in running state.		
Post conditions:	Volume changed according.		
Normal Flow:	1:Player clicks on the change setting option. 3:player select the change volume option. 5:player set the volume.	2:systemshows the setting menu. 4:system display the volume setting menu of game. 6:system set the volume accordingly.	
Alternative Flows:	No Alternative		
Exceptions:	No exception		

2.6 System Sequence diagrams

System sequence diagram (SSD) is a sequence diagram that shows, for a particular Scenario of a use case, the events that external actors generate their order, and possible inter-system events.

Start Game: User starts the game by pressing play button from main menu System loads the game and display a mission List. User selects the mission and system start the mission for player.

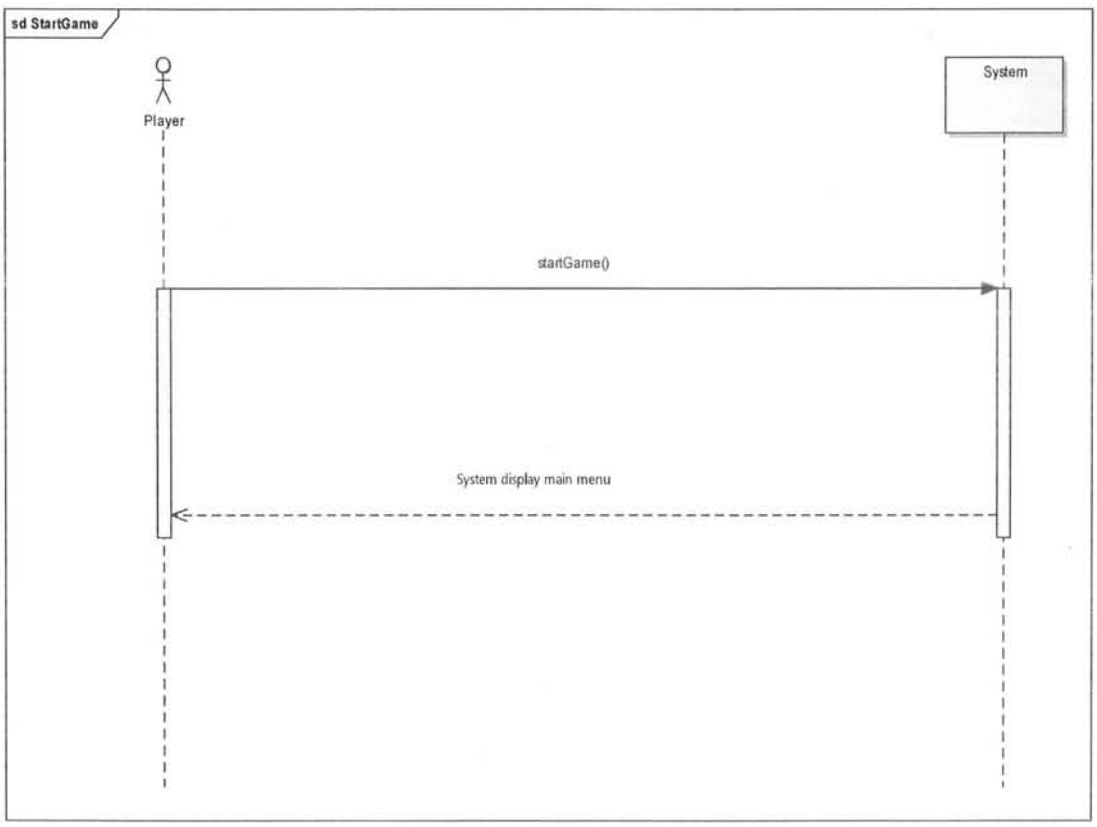


Figure 2.2: Start Game

Move Left: User swipe the joystick in left to move the player in leftward.

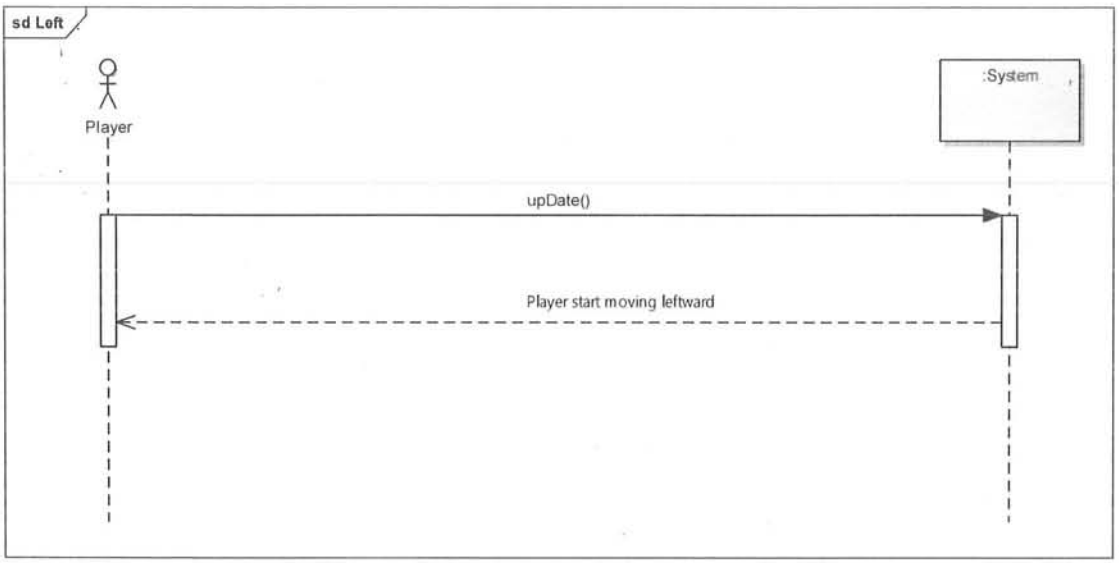


Figure 2.3: move left

Move Forward: User swipe the joystick in upward To move the player forward.

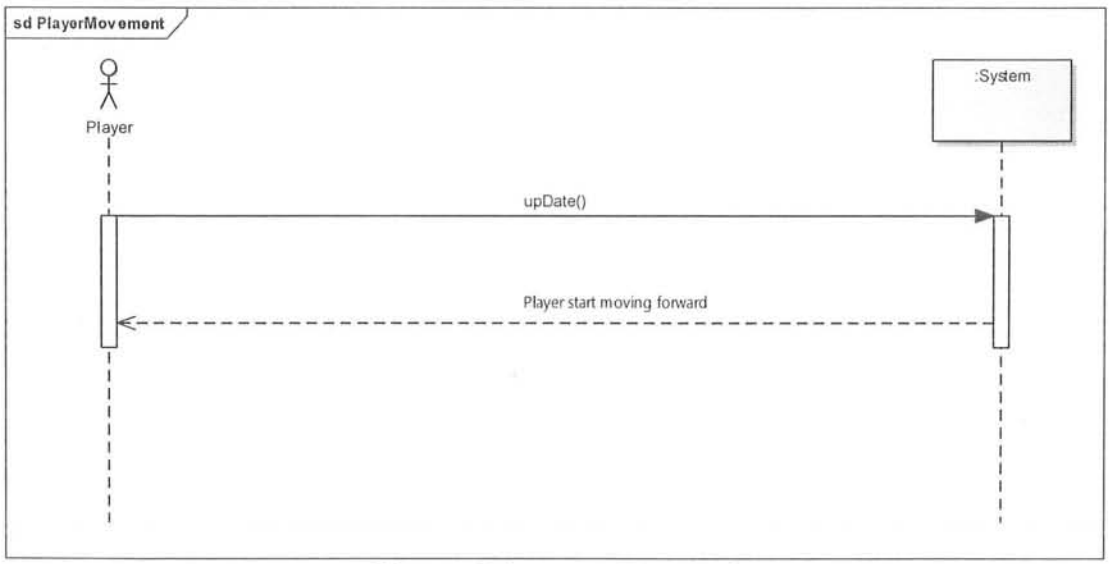


Figure 2.4: move Forward

Move Backward: User swipe the joystick toward down to move the player in backward

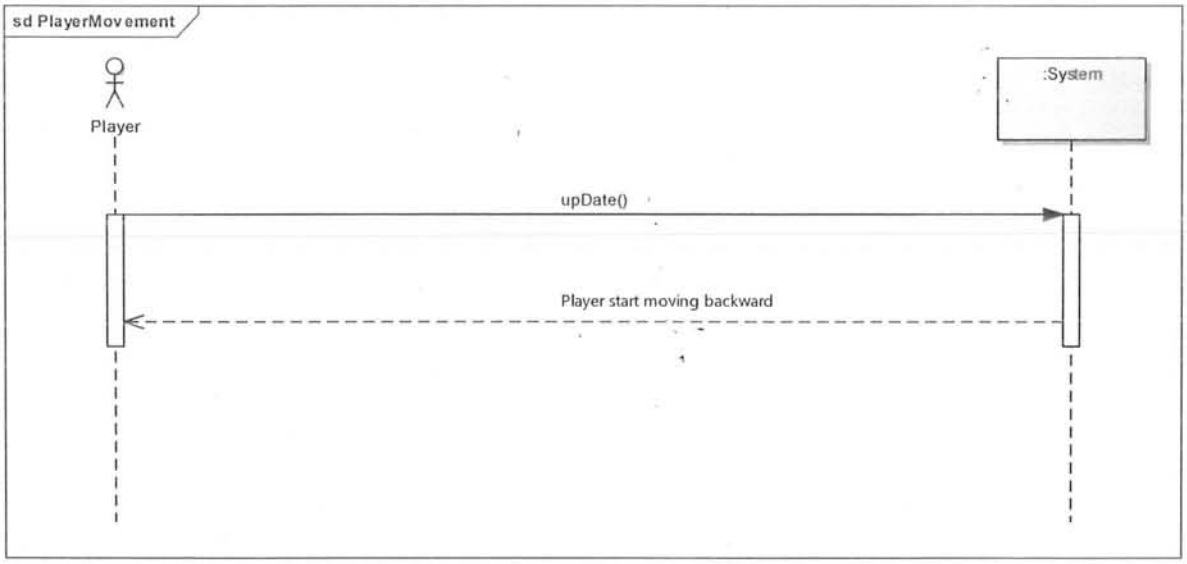


Figure 2.5: move Backward

Move Right: User swipe the joystick right to move the player in right side

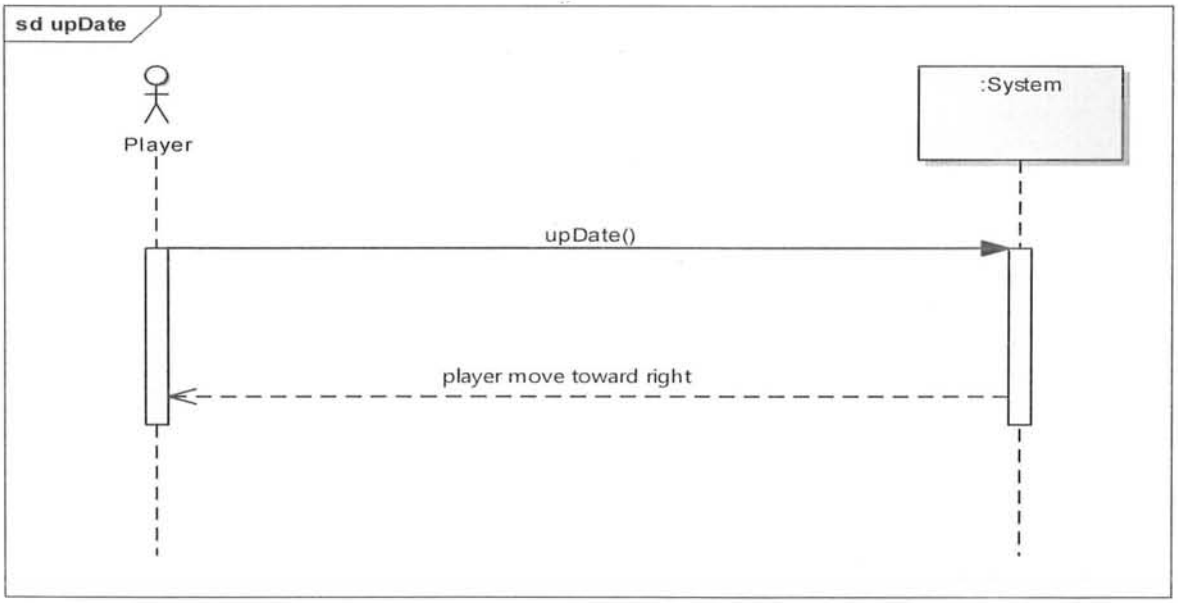


Figure 2.6: move Right

Jump: user press the jump button and player will jump.

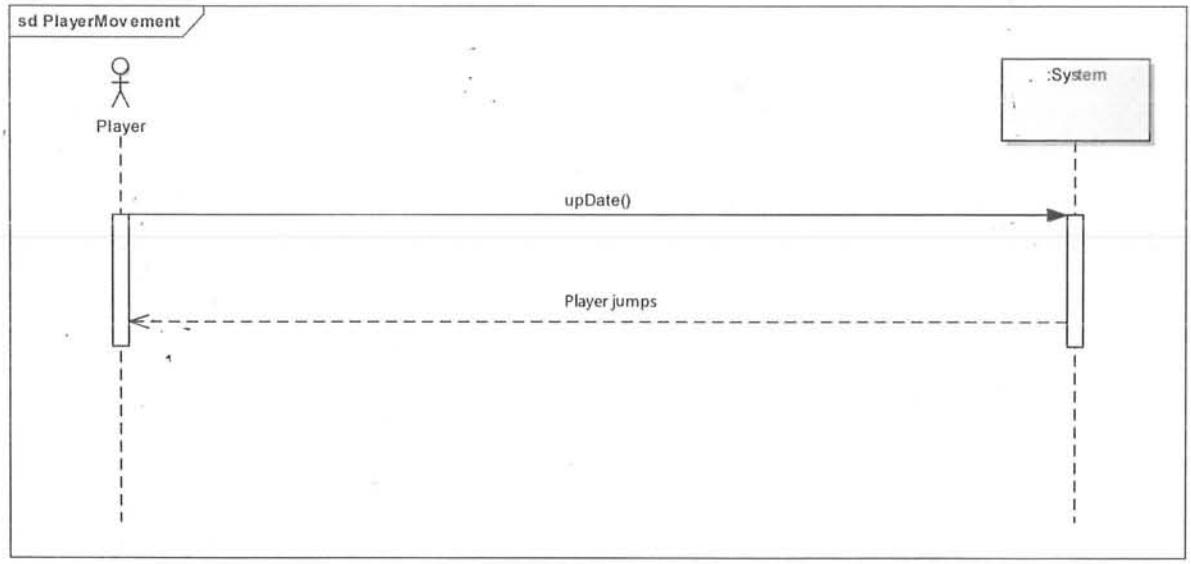


Figure 2.7: Jump

Pause Game: user press the esc key to pause the game.

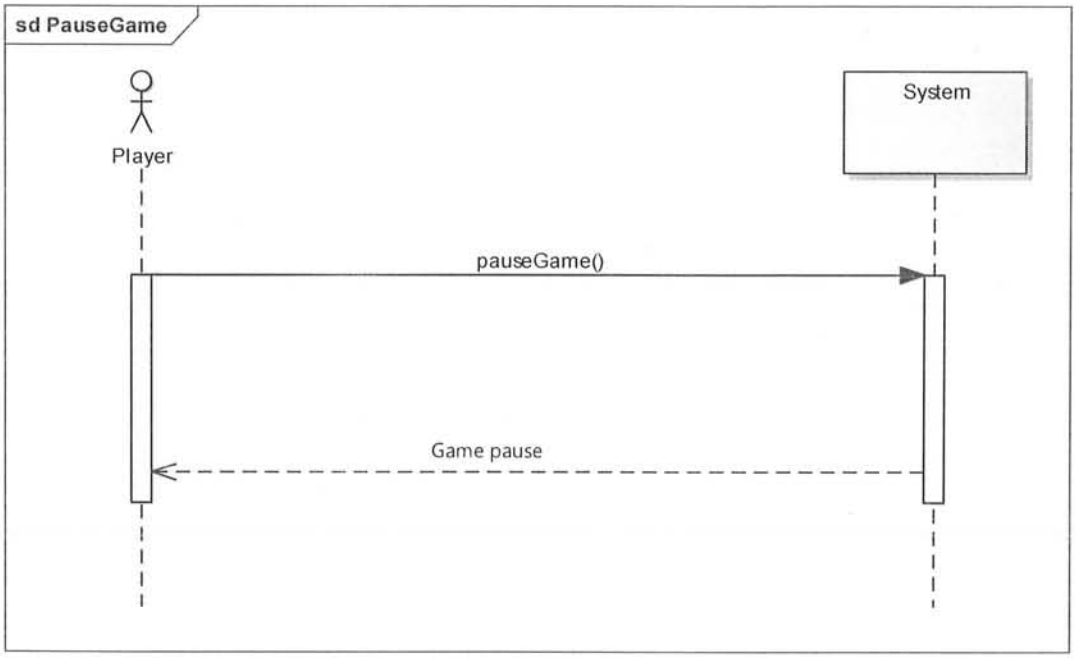


Figure 2.8: Pause Game

Resume Game: user press the Resume button to resume the game.

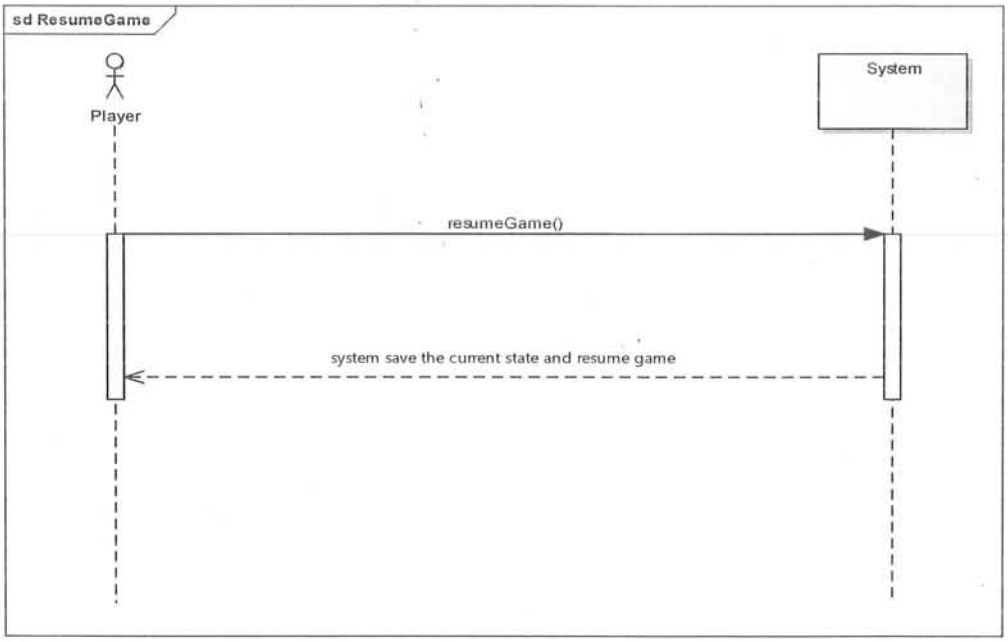


Figure 2.9: Resume Game

Freeze Time: user press the freeze button to slow down the time.

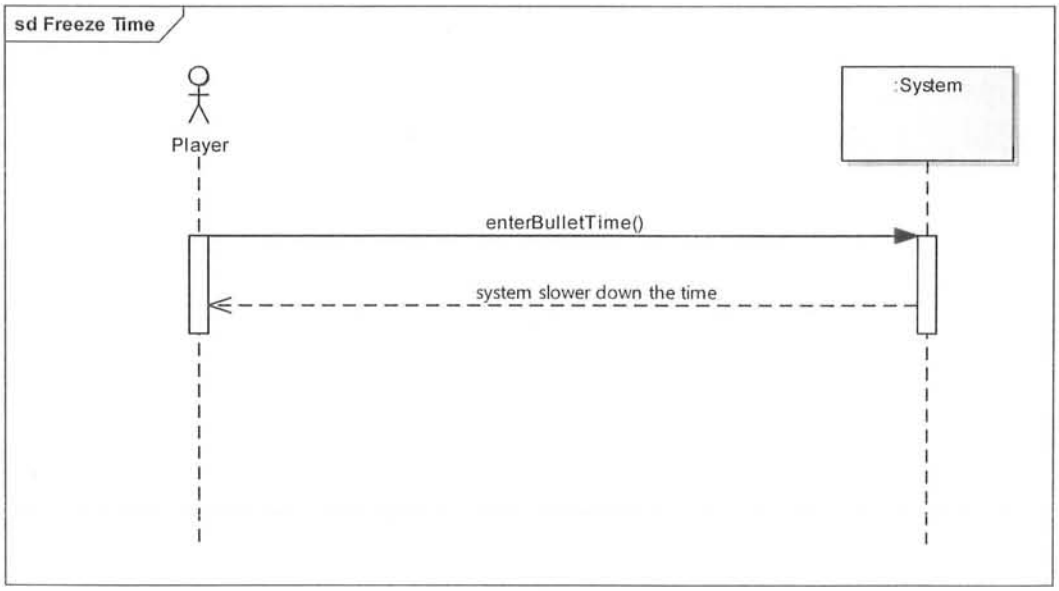


Figure 2.10: Freeze Time

Aim: user will control the aim by controlling camera using joystick.

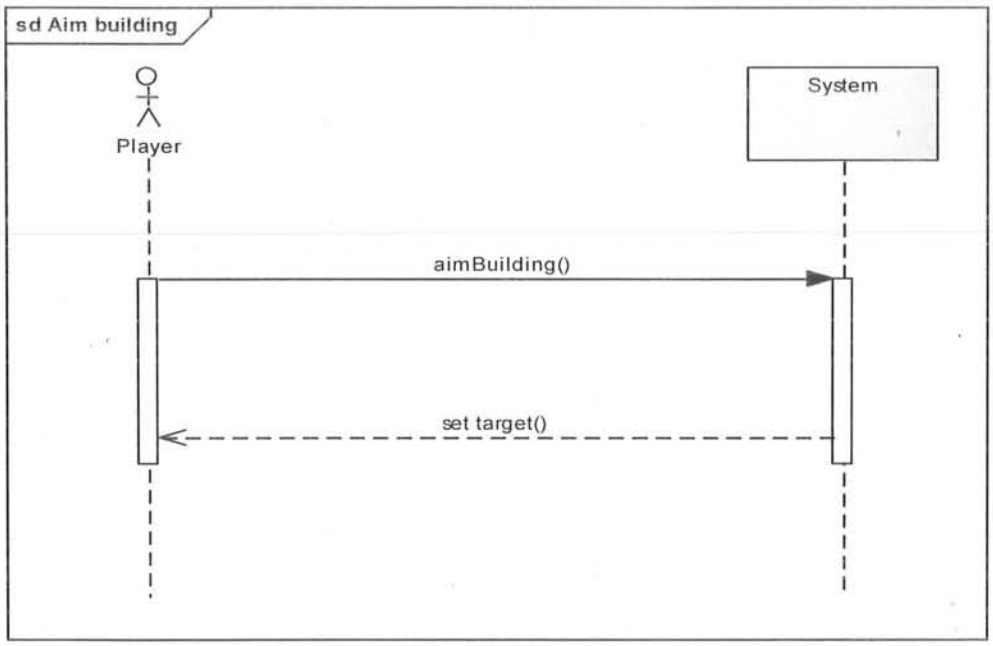


Figure 2.11: Aim building

Quit: user press the quit button to quit the game.

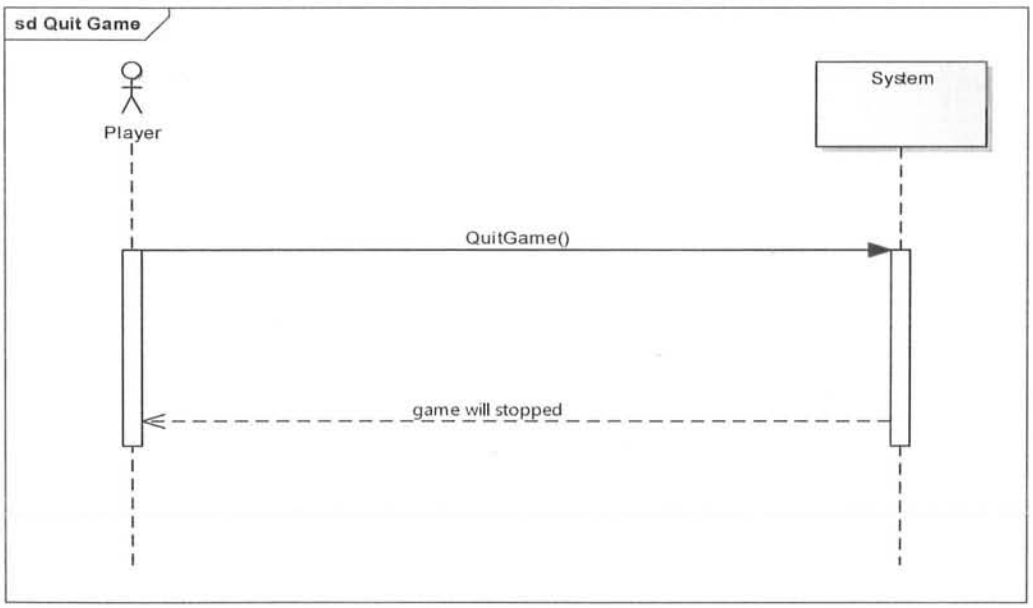


Figure 2.12: Quit Game

Change Setting: user press the setting button from menu, system shows the setting fields to the user. User change the desired fields and press save button to save the changes, system save the changing fields for the player

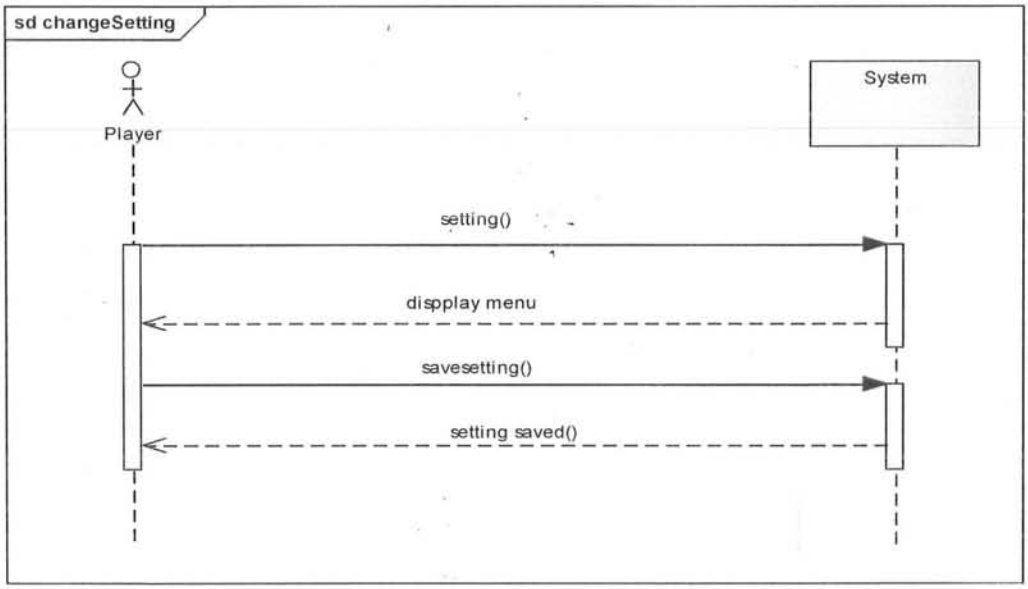


Figure 2.13: Change Setting

Restart Game: user press the restart game button to restart the game. System set the default values in the fields and restart the game.

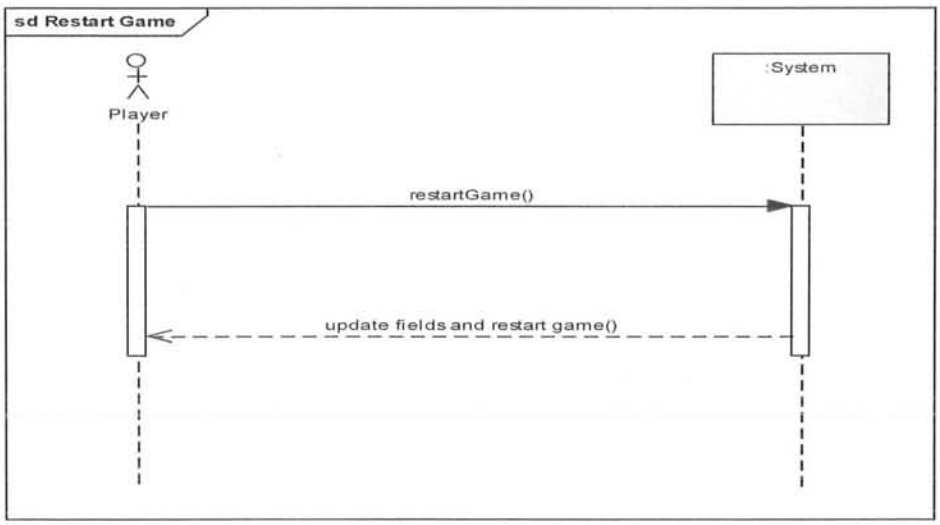


Figure 2.14: Restart Game

View collected Money: user press the shop item button from menu, system shows different shop related options to the user. User select the view collected money buttons, system shows the collected money to the player.

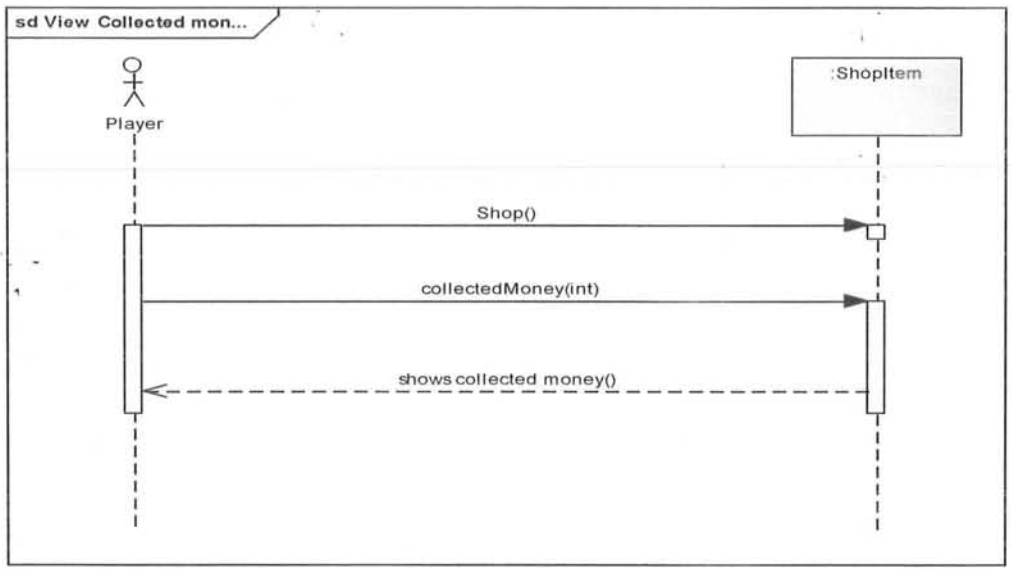


Figure 2.15: View Collected Money

Kill Enemies: Player use mouse left button to kill the targeted enemy and system destroy the enemy.

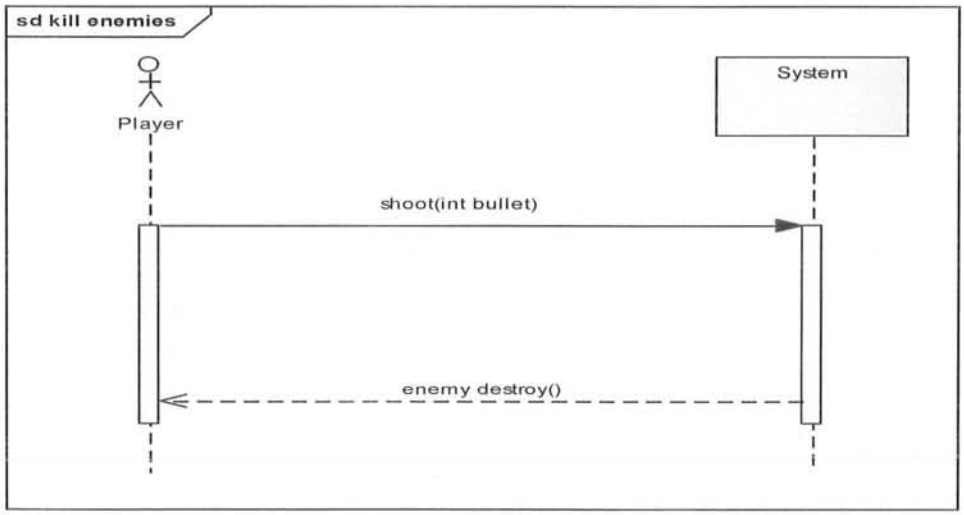


Figure 2.16: Kill enemies

Buy Grenade: Player select the shop button from main menu. System shows the available items in the shop .User select the grenade from the available items and press buy button to buy the grenade. System check the money and give confirmation message to the user.

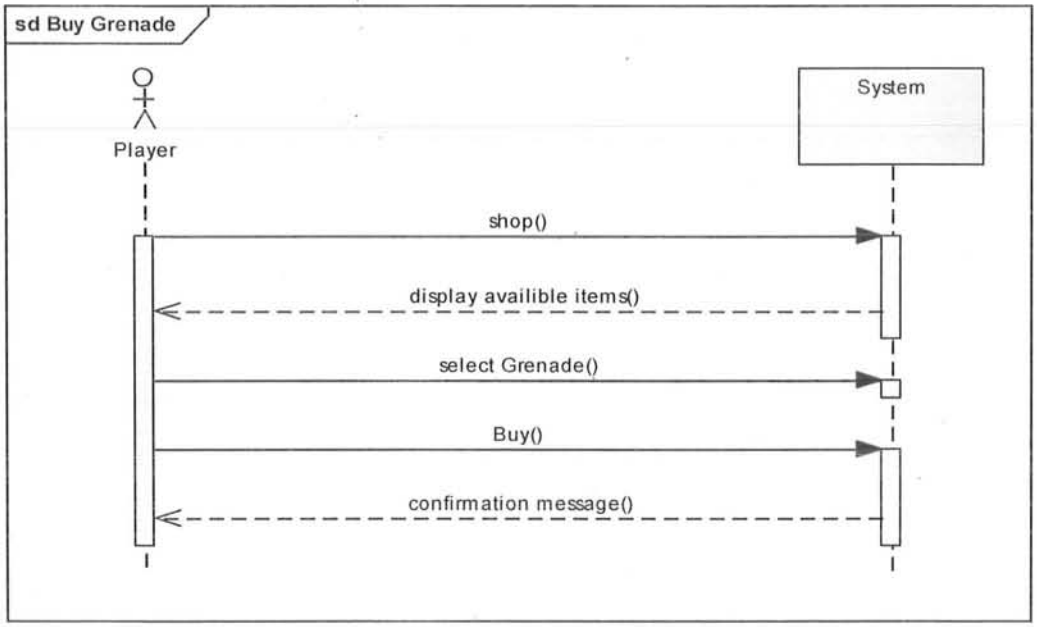


Figure 2.17: Buy grenade

Buy JetPack: Player select the shop button from main menu. System shows the available items in the shop.User select the jetpack from the available items and press buy button to buy the grenade. System check the money and give confirmation message to the user.

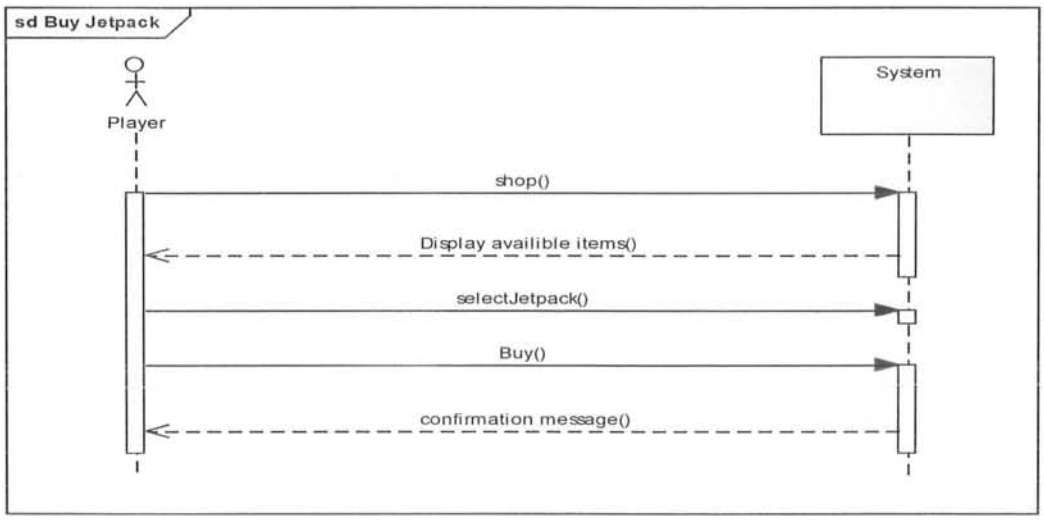


Figure 2.18: Buy Jetpack

Select level 1: Player select the level 1 from main menu to play level 1. System start level 1 for the player.

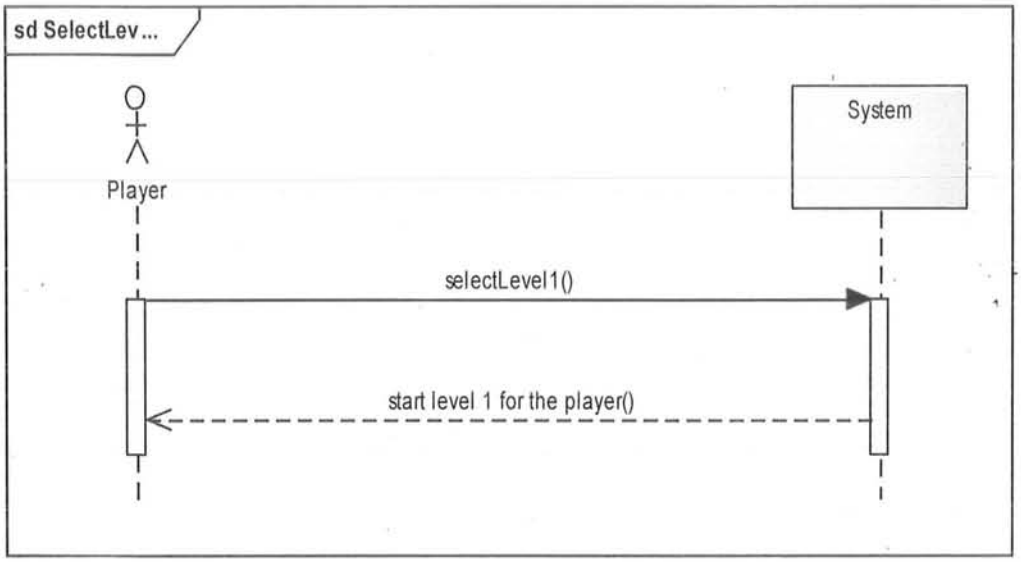


Figure 2.19: select level 1

Select level 2: Player select the level 1 from main menu to play level 2. System start level 2 for the player.

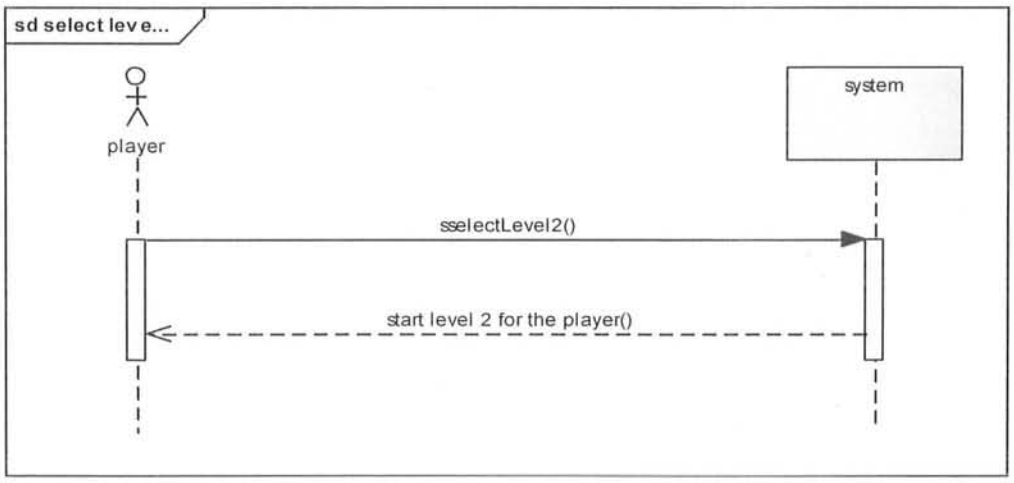


Figure 2.20: select level 2

Select level 3: Player select the level 3 from main menu to play level 3. System start level 3 for the player.

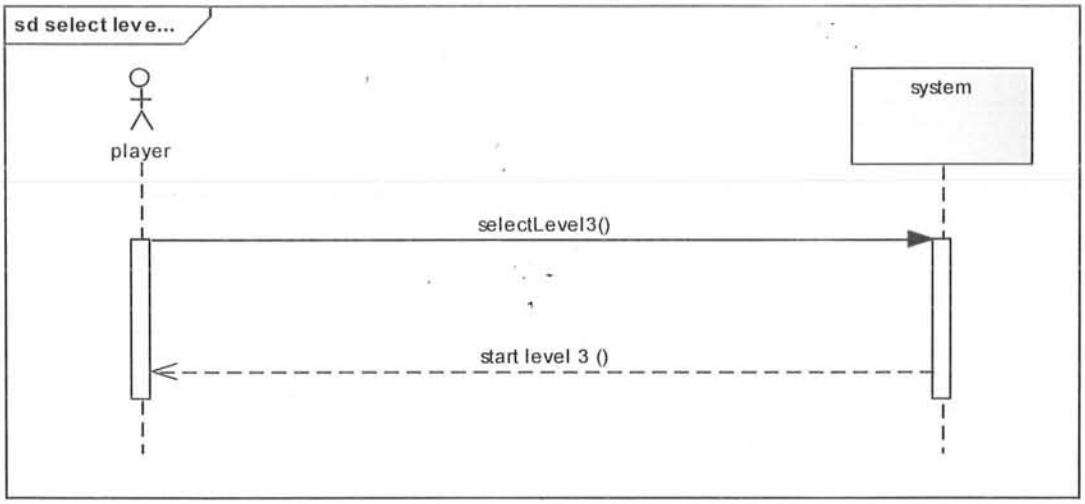


Figure 2.21: select level 3

Drive vehicle:.

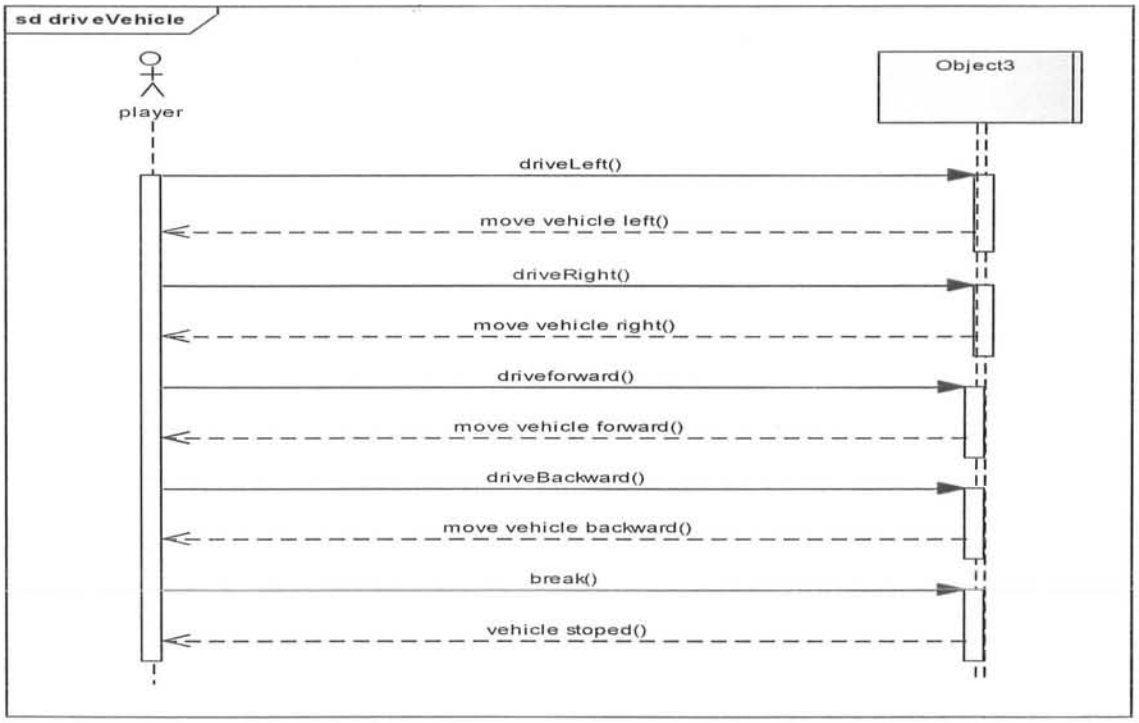


Figure 2.22: Drive Vehicle

Volume setting

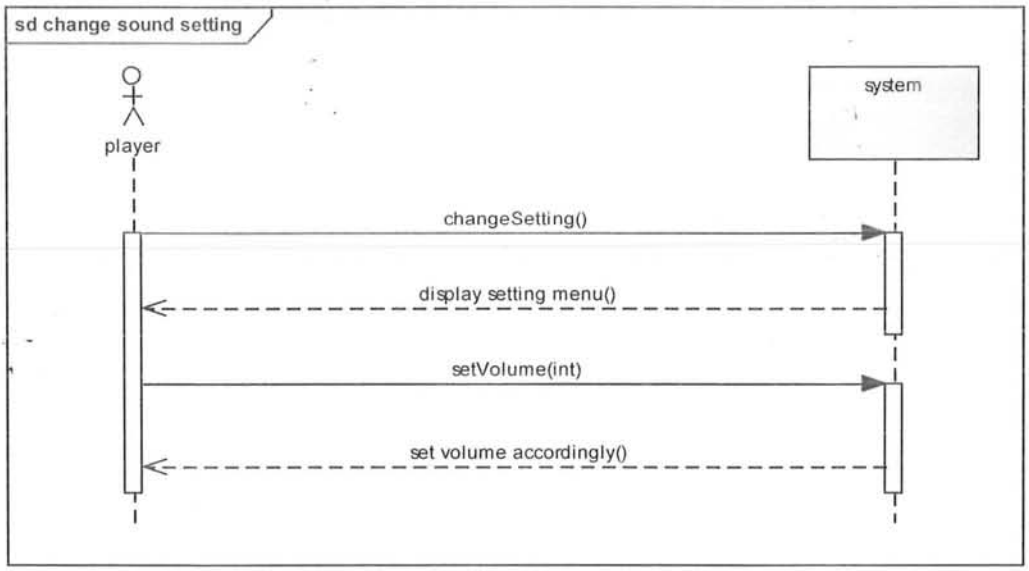


Figure 2.23: volume setting

2.6 Domain Model

Part of your initial architectural modeling efforts, particularly for a business application, will likely include the development of high-level domain model as you see in Fig. 2.7.

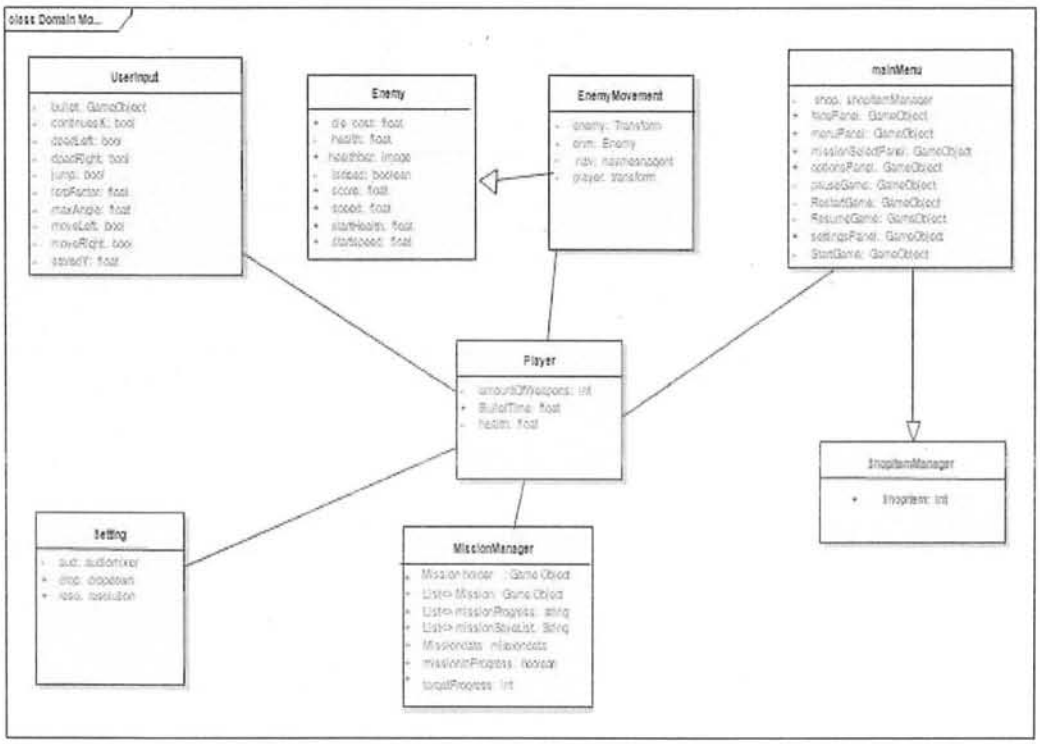


Figure 2.17: Domain Model

CHAPTER 3

System Design

3.1 Software Architecture

Software architecture is described as the organization or structure of a system, where The system represents a collection of components that accomplish a specific function or set of functions.

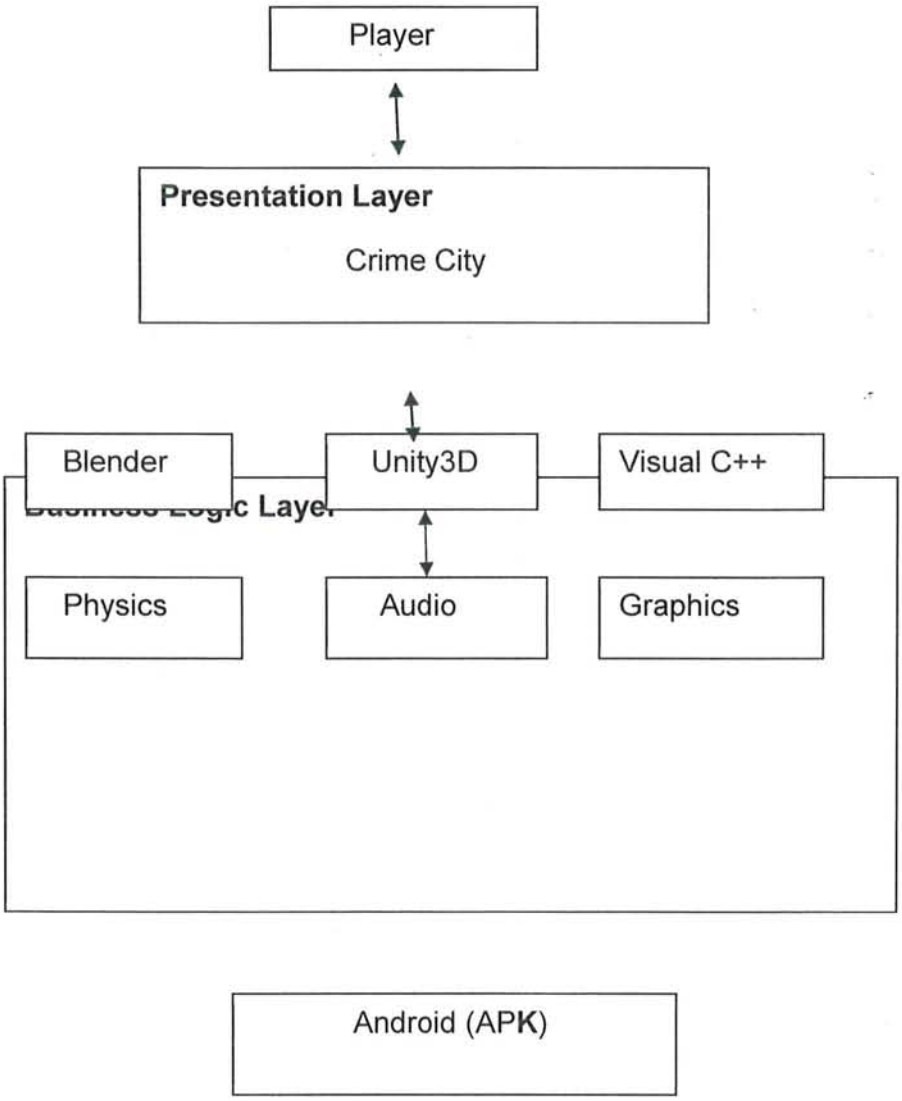


Figure 3.1: Software Architecture Diagram

3.2 Class Diagram

Class Diagram as shown in Fig. 3.2 provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from modeling the domain-specific data structure to detailed design of the target system.

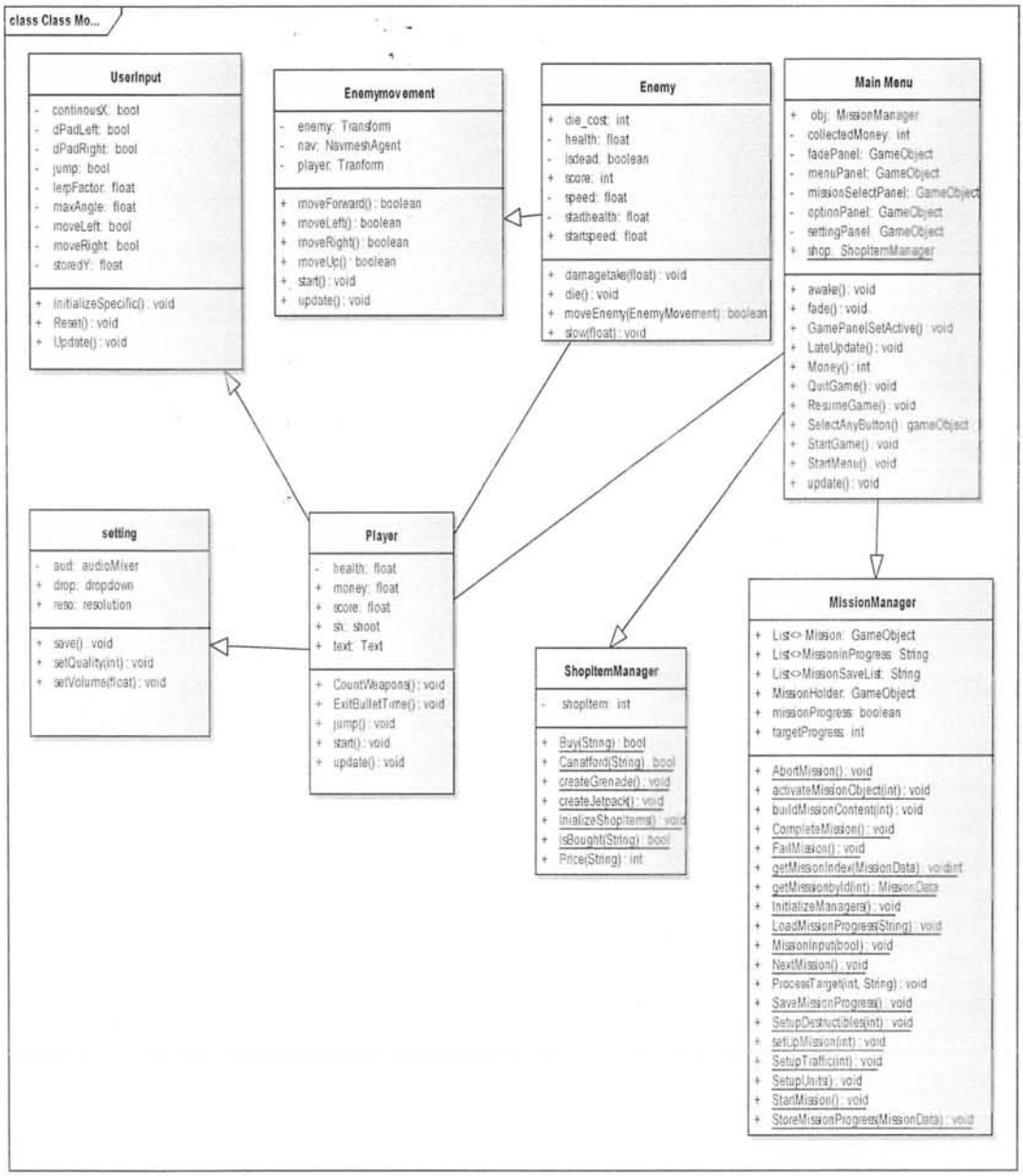


Figure 3.2: Class Diagram

3.3 Sequence Diagram

Sequence diagrams, when used in conjunction with class diagrams; provide an extremely effective communication mechanism. UML sequence diagrams as shown in Fig. 3.3 are used to show how objects interact in a given situation.

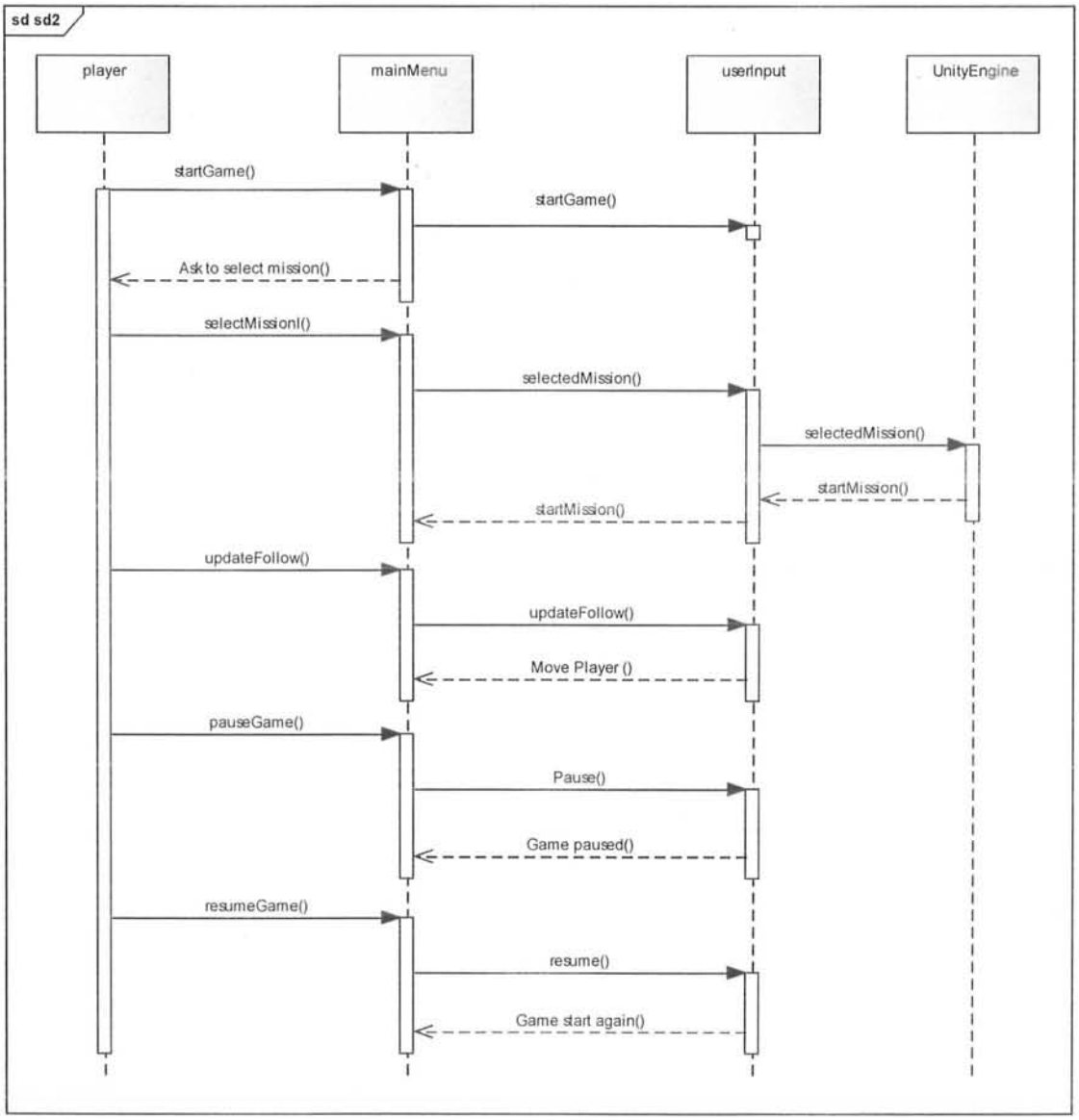


Figure 3.3(a): Sequence Diagram

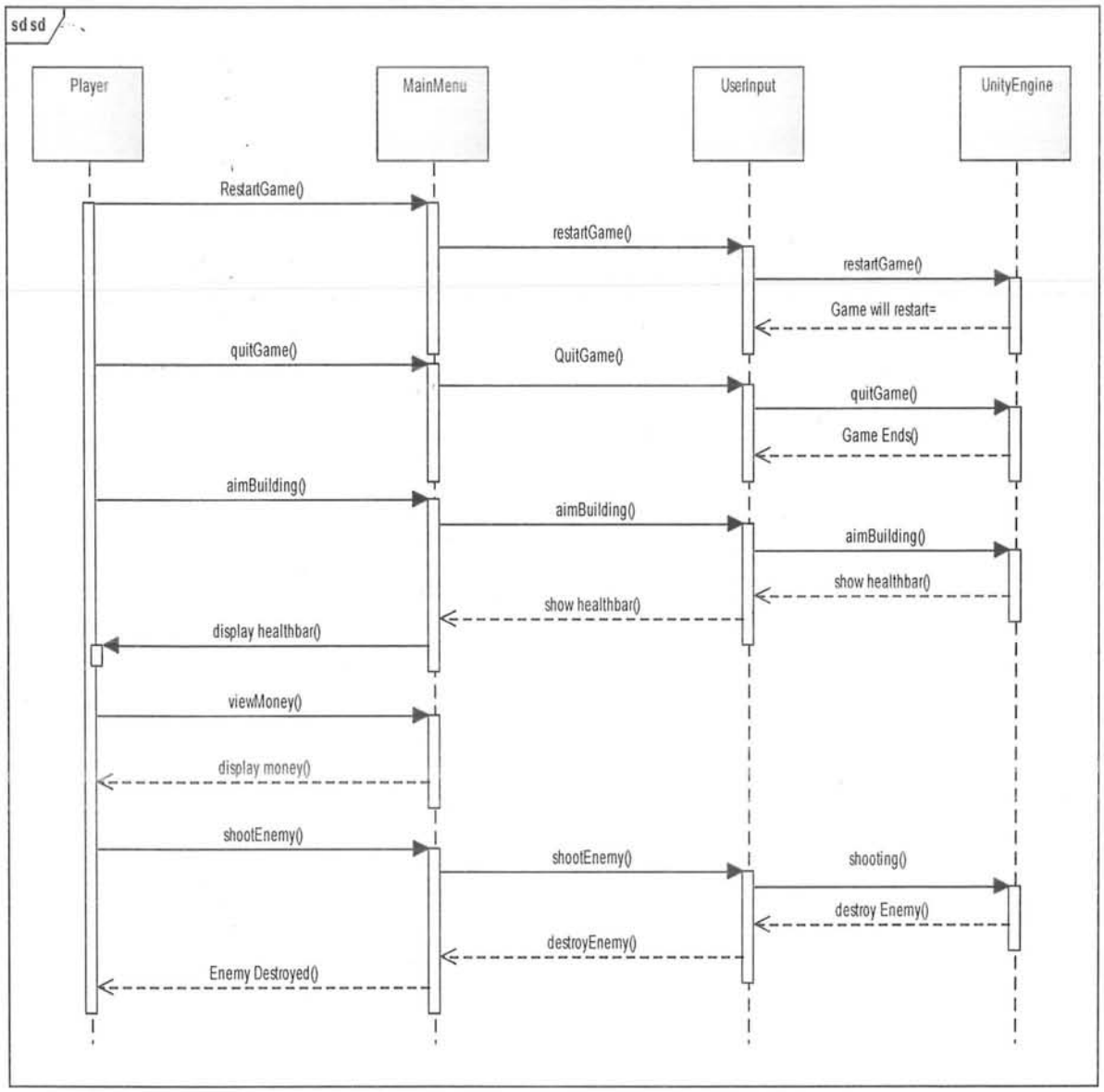


Figure 3.3(b): Sequence Diagram

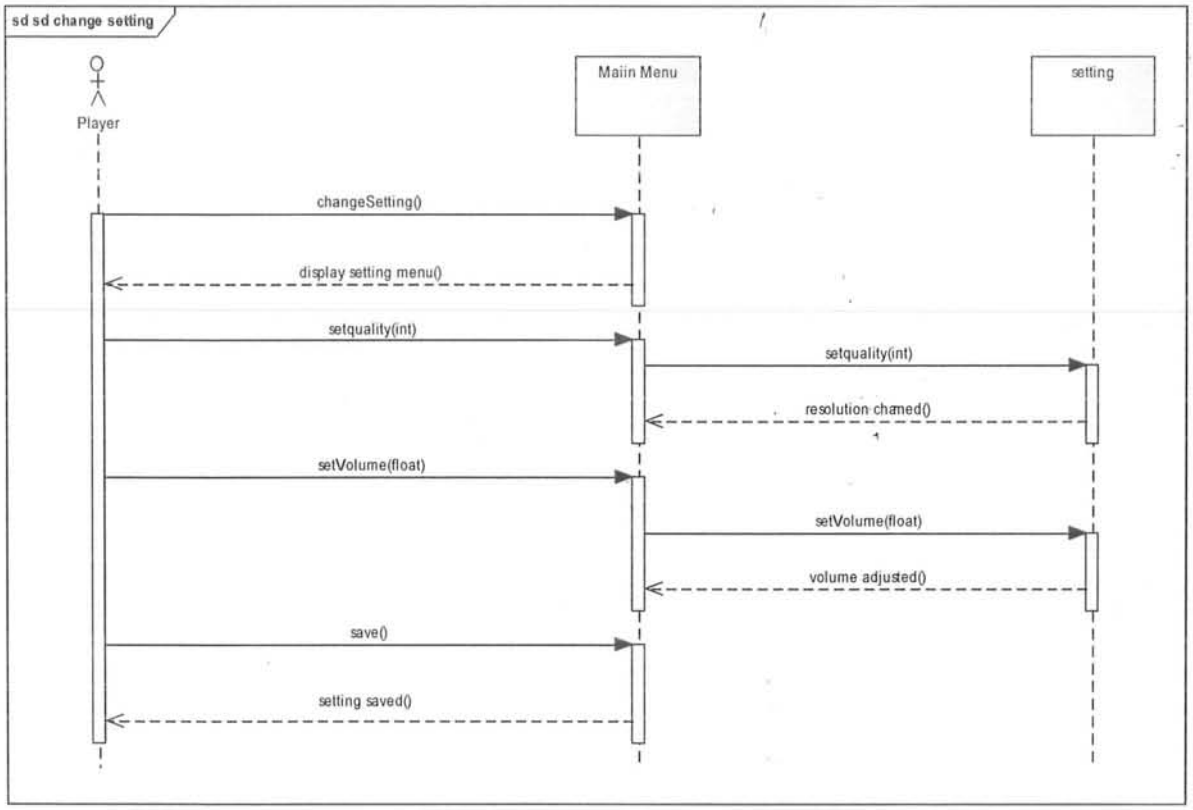


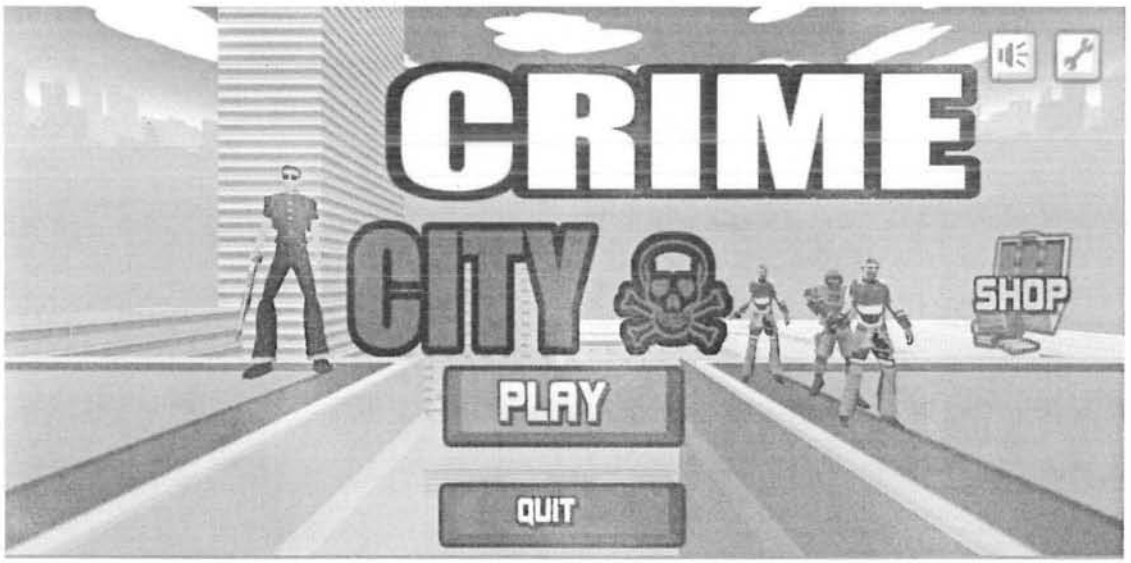
Figure 3.3(c): Sequence Diagram

3.4 User Interface Design

This Section describes the interfaces of game the users are expected to interact with the game. These interfaces are designed in unity3D.Following are some of the game user interfaces with which users will interact.

3.4.1 Main Menu

This is the main menu screen of the game in which different buttons are shown. And on pressing any of the buttons an operation is performed.



3.4.2 Start Game

This is starting scene of a game.

3.4.3 Pause Menu

This is the pause menu screen of the game in which user can Quit and Retry the game.

Chapter 4

Software Development

4.1 Coding Standards

This chapter will provide the details about the coding standard, we adopted during implementation phase.

4.1.1. Indentation

The Line spacing is used during the code. There is only one line gap between the declaration of public, private and local variables and this pattern is used throughout the code.

4.1.2. Declaration

Declaring one variable per line to understand the code easily.

- Class name declare as a camel Case notations.
- The Method or Function name should be start with capital letters.
- There will be one line space between the public/private variables.
- The public variables should be declared first then private variables with one line of gap.
- The local variables should be declared after the private variables or immediate after the start of any function.

4.1.3 Statement Standards

The coding statements will be declared one per line. The nested statement like for loops or if else statements their scope or braces will be closed immediate after opening of the brace.

4.1.4 Naming Convention

The conversion is used to understand the code easily. The variable name should be start with the capital word or with underscore for the same variable if it is declared already. For Example (Bullet Time , _Bullet Time).

4.2. Development Environment

Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop both three-dimensional and two-dimensional video games and simulations for computers, consoles, and mobile devices .As far unity3D is most popular among developers than any other game development software.

Visual studio:

Visual studio is used to develop computer games, as well as mobile apps or game. It use Microsoft software development platforms (windows API).

Blender:

Blender is an industry leading 3D Modeling software application. It enables video developers who work with animated film, video games to create highly quality 3D Models.

Photoshop:

Photoshop is used to develop the user interface with many visual effects and can be used to alter images.

4.3 Software Description

Main modules of our project are

- **Restart Game.**
- **Change Setting.**
- **Collect Money.**
- **Kill Enemies.**

4.3.1. Restart Game

Snippet 1

```
public void Restart Game()
{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}
```

Description:

The **Restart** function restart the current loaded scene again as soon as the player die and or when the user will press the restart button.

4.3.2. Change Setting

Snippet 2

```
public void set volume(float vol)

{
audioMixer.setFloat("volume", vol);
```

```
}

public void set quality(int index)

{

QualitySettings.setQuality(index);

}
```

Description:

The **set volume** Method will mute or un mute the audio of the game where as the **set quality** method will set the graphics setting to (low , medium , high)

4.3.3. Collect Money

Snippet 3

```
publicstaticint score;

Text text;

public void money(int value)

{
text= Get Component <Text> ();

score= 0;
score=value;
text. Text ="Score: "+ score;

}
```

Description:

The **money** Will increase the score count variable and update it when the enemies die.

4.3.4. Kill Enemies

Snippet 4

```
public Money mon;

public int Value = 10;

public void Take Damage (int amount, Vector3 hit Point)
   {
if(is Dead)
return;
enemyAudio.Play ();

current Health -= amount;

mon. Money(value);

      }
```

Description:

This method will reduce the enemies health and update the score count variable in money class as the enemies die .

Chapter 5

Software Testing

This chapter provides a description about the adopted testing procedure. This includes the selected testing methodology, test suite and the test results of the developed software

5.1. Testing Methodology

In testing methodology we use black box testing because it is very efficient .Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested.

NUnit is an open source unit testing framework for .Net. Unit provides a console runner which is used for batch execution of tests. The console runner works through the NUnit Test Engine, which provides it with the ability to load, explore and execute tests.

- Tests can be run from a console runner.
- Tests can be run in parallel.
- Strong support for data driven tests.
- Supports multiple platforms.

5.2. Testing Environment

Test Runner

The Unity Test Runner is a tool that tests your code in both Edit mode and Play mode, and also on target platforms such as Android, or iOS.The Unity Test Runner uses a Unity integration of the NUnit library, which is an open-source unit testing library for .Net languages.

MonoDevelop

Mono Develop is an open source integrated development environment for Linux, macOS, and Windows. Its primary focus is development of projects that use Mono and .NET frameworks. *MonoDevelop* integrates features similar to those of Net Beans and Microsoft Visual Studio.

There are 10 use cases in our project to test all those use cases we have written test cases to test the functionality of the system.

Steps to perform:

- Navigate to windows->select Test Runner.
- Select the play mode option.
- Create new test and rename the file.
- Write the functionality in the test script.
- Select all the test you want to run.
- Run the game and test cases parallel.

5.3.TestCases

Test Case: kill Enemies

Table 5.1: kill Enemies

Date:	07 FEB 2019
Test ID:	1
Objective:	kill Enemies
Version:	1
Test Type :	Unit Testing
Input :	Press the shoot button to kill enemies
Expected Result:	enemies die.
Actual Result:	Passed Successfully

Description:

In this figure test case is written in Mono develop and those test cases are tested on Test Runner which is the built-in automated tool in unity3d. First create the Editor Test c# script and then write the functionality to test enemies die.

While playing mode run your test case through test Runner it will generate the test case result in the console.

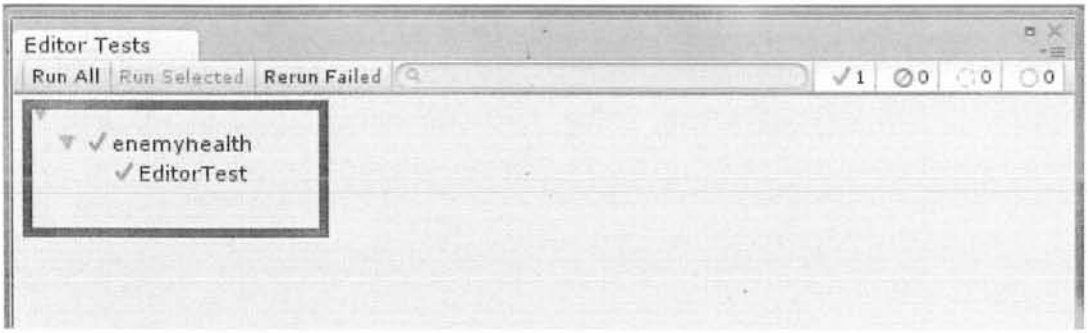


Figure 5.1: Enemies health Test Runner

In this figure, Test cases that are created in Mono Develop are being tested in the test Runner.

```
using UnityEngine;
using UnityEditor;
using NUnit.Framework;

public class eneimes_health
{

    Enemy enm;
    public float currentHealth = 10;
    public float amount = 10;
    [Test]
    public void EditorTest()
    {

        currentHealth -= amount;
        if (Assert.Equals(currentHealth, enm.health)) ;

        Debug.Log("Eniemy Die");

    }
}
```

Figure 5.2: Kill Enemies Test Case

Test Case: Collect Money

Table 5.2: Collect Money

Date:	07 FEB 2019
Test ID:	2
Objective:	Collect Money
Version:	1
Test Type:	Unit Testing
Input:	Press shot button to kill enemies and collect money.
Expected Result:	Player will collect the money.
Actual Result:	Passed Successfully

Description:

In this figure test case is written in Mono develop and those test cases are tested on Test Runner which is the built-in automated tool in unity3d. First create the Editor Test c# script and then write the functionality to test collect money function.

While playing mode run your test case through test Runner it will generate the test case result in the console.

```
using UnityEditor;
using NUnit.Framework;

0references
public class collectMoney
{

    public Enemy enm;
    public float current_health = 10f;
    public float amount = 10f;
    public float score = 10f;
    [Test]
    0references
    public void EditorTest()
    {
        current_health -= amount;
        if (Assert.Equals(current_health, enm.health))
        {
            score = enm.score;
            if (Assert.Equals(amount, enm.score))
            {
                Debug.Log("Money Add");
            }

        }

    }

}
```

Figure5.3: Collect Money Test Case

Figure5.4: Collect Money Test Runner

Test Case: Restart

Table 5.3: Restart Game

Date: 07 FEB 2019	
System:	
Objective: Restart Game	Test iD:3
Vesrsion:1	Test Type: Unit Testing
Input: Press the restart button	
Expected Result: Game will restart.	
Actual Result: Passed	

Description:

In this figure test case is written in Mono develop and those test cases are tested on Test Runner which is the built-in automated tool in unity3d. First create the Editor Test c# script and then write the functionality to test restart game.

While playing mode run your test case through test Runner it will generate the test case result in the console.

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEditor;
using NUnit.Framework;

0references
public class Restart {

    GameObject button;
    [Test]
    0references
    public void EditorTest()
    {

        button = GameObject.Find("RestartButton");
        string name = button.name;
        if (Assert.Equals(name,"RestartButton"));
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
            Debug.Log("Game Restart");
        }

    }
}
```

Figure 5.5: Restart Test Case

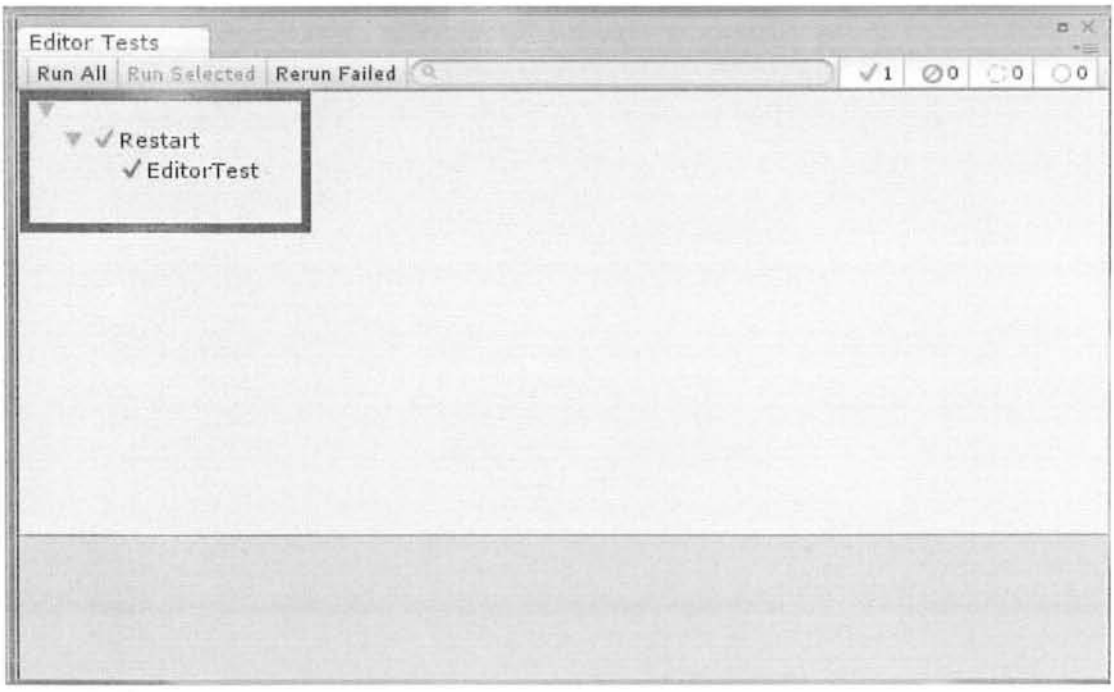


Figure 5.6: Restart Test Runner

Test Case: Change Setting

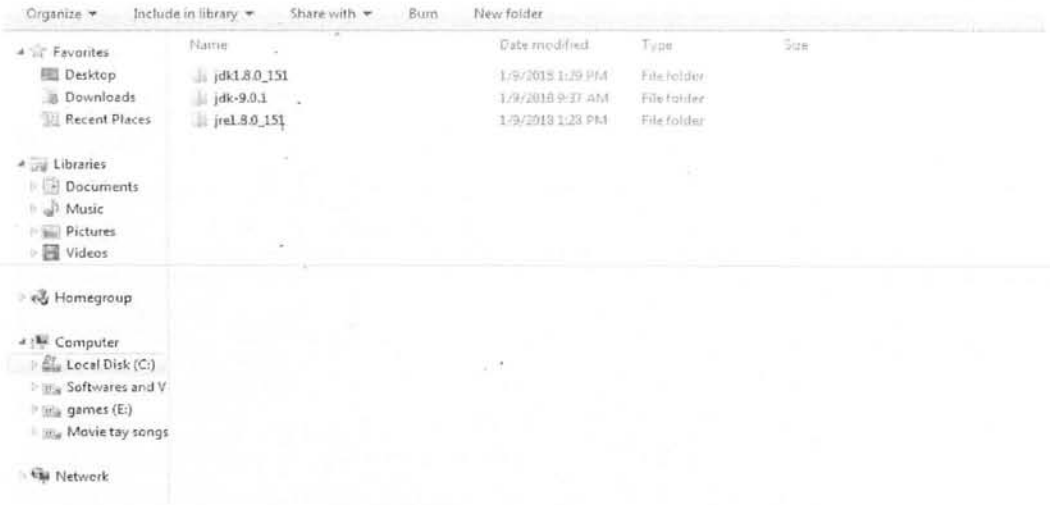
Table 5.4: Setting

Date: 07 FEB 2019	
System:	
Objective: Change Graphics Settings.	Test iD:4
Vesrsion:1	Test Type: Unit Testing
Input: Press the low or high setting button.	
Expected Result: Graphics quality change.	
Actual Result: Passed	

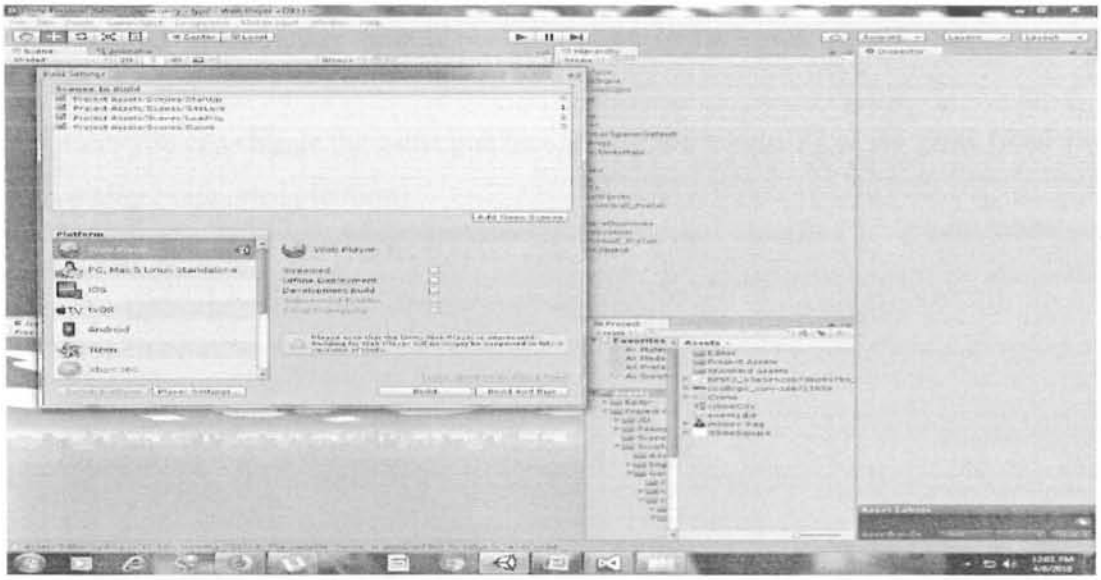
Description:

In this figure test case is written in Mono develop and those test cases are tested on Test Runner which is the built-in automated tool in unity3d. First create the Editor Test c# script and then write the functionality to test graphics settings. While playing mode run your test case through test Runner it will generate the test case result in the console.

- The Jdk version should be 8_151.



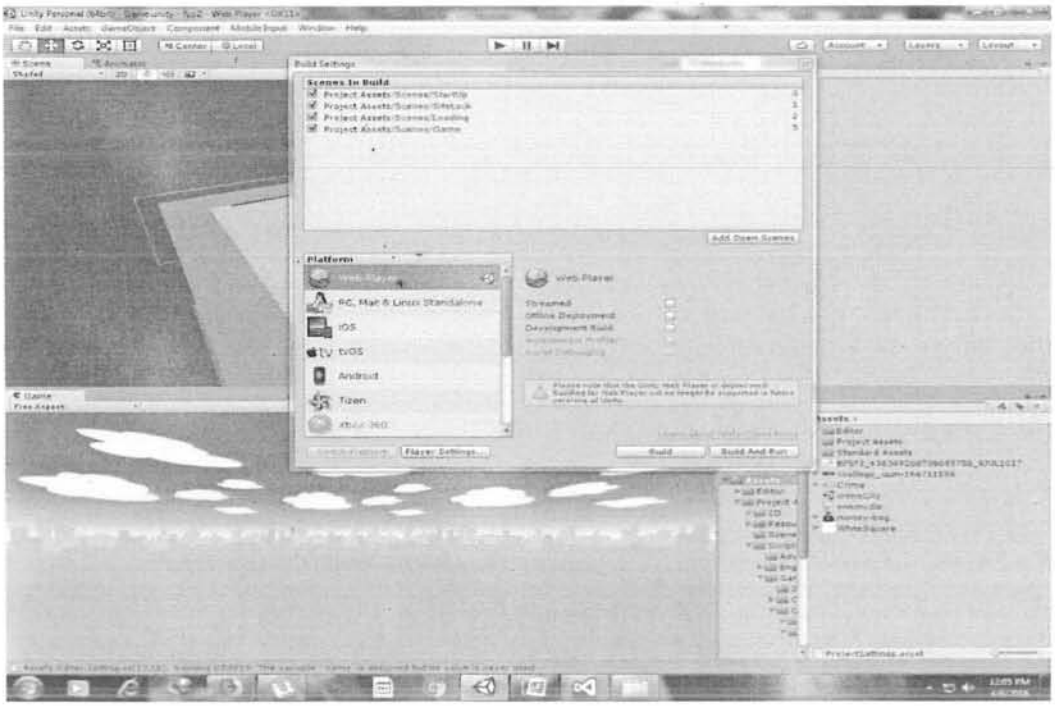
- Then update the android sdk so that it will add the libraries of which are under 25.0.0.
- To build the we need to put all the scenes of our game in build settings so that all the scenes are build for the game.



- Then Navigate to Edit->preference set the android tool path and java jdk path .

- Navigate to File->build settings-> click player setting.

- Click on the build button and give the name of your apk file in our case it would be (Crime City).
- Click build to build the apk.



- Put the Apk (Android Package Kit) on your android device and install it.