FATEH SHEHRYAR
**01-134131-025**
ASFANDYAR SHAKEEL
**01-134131-014**

# Maverick

**Bachelor of Science in Computer Science**

Supervisor: Dr. Sabina Akhtar

Department of Computer Science
Bahria University, Islamabad

May 2017

# Certificate

We accept the work contained in the report titled “Maverick”, written by Fateh Shehryar AND Asfandyar Shakeel as a confirmation to the required standard for the partial fulfillment of the degree of Bachelors of Science in Computer Science.

Approved by . . . :

Supervisor:

________________________________________

Internal Examiner:

________________________________________

External Examiner:

________________________________________

Project Coordinator:

________________________________________

Head of the Department:

________________________________________

# Abstract

The report discusses the use of different tools such as blender, Unity, C etc. and to how the tools will manage to provide the required output (game) for instance blender was used as it has a constructive workflow that is any mesh you use can be altered easily (you can change the number of vertices by using modifiers) which other tools like maya don't provide. Photoshop is used when simple textures were needed (without the mapping) and a tool named crazybump helped generating the different maps for the textures used. Furthermore the game focuses on design and efficient programming making the game user friendly. The objective for choosing a game project was to gain experience in computer graphics so in future the team can contribute to the computer graphics society of Pakistan by having a portal where people will be allowed to submit similar projects and ideas; also to use our experience to motivate others to pursue their passion in the computer graphics discipline.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Project Background/Overview

This project is a first person shooter game built on the gaming engine (Unity). It is a windows based application (game). The game focuses on design and efficient programming making the game user friendly, with realistic and interactive environment. The two members mentioned above have worked on the project with the help and support of their supervisor. The team faced risk (calculated and uncalculated), the risks included the reliance on external code (found on the net). Management was another risk but the team handled it well by creating a work breakdown structure and defining the ultimate scope of the project in the initial stages. Also extra work was efficiently removed by hand drawing the interfaces and critically analyzing the interfaces for improved version.

## 1.2 Problem Description

Levels in today games are static designed by a level designer, due to this these some games fail to keep the players interested to the end. We want to tackle this issue by generating the game levels procedurally (in future). Procedural level generation allows the coder to generate a random level at the end of every level. Each level is different and you can generate infinite many levels and still make them look different. Moreover as the storyline will revolve around a Pakistani family, which will promote our culture and help improve Pakistan'simage globally. By making this game we further want to strengthen the Pakistani game industry and to encourage others to do the same. The game has the following main assets (subfolders) that can be manipulated (have C-Sharp functions which are performed according to the situation in the game) while in runtime, these are the player characters and are as following:

- Camera
- Weapons
- Player
- Effects
- HUD
- Items
- AI
- Objects

These objects function on the terrain through scripts and manipulation of Unity's [1] physics components. These assets fall under the Unity's behavior manager. The NPC's involve mathematics and physics are focus on how much damage does the target take based on where the target is hit etc. Some of these character are however interactive while the others do not impact the player what so ever.

## 1.3 Objectives

To design a highly interactive 3D shooter game, with graphics good enough to compete in the market and to take Pakistani game industry to its height. Moreover it is to add in the experience for further development of better game in the near future and to have a better understanding of different tools and techniques.

## 1.4 Project Scope

The game is made using the popular Unity 3D gaming engine. Unity allows you to write code either in JavaScript or C-sharp [2]. The project uses C-sharp scripts due to its robustness and also as we have experience in it. While Unity will handle all the physical aspects of game from bringing everything together to the end product, blender [3] and photoshop were used for 3D modeling and texturing of the game. 3D models will be made using Blender. Photoshop was used for texturing these assets and for UI design.

# Chapter 2

# Literature Review

There are a few basics step to making a first person shooter game that you need to follow (may it be in unity or any other gaming engine). First and foremost (if needed) you have to get user requirements for additional information and as this was a game, hence it asked for not just one user's information rather a survey. The survey helped as the users, from the past experience, explained what they would like to see in a first person shooter game and what bugged them in the previous first person shooter games. The second phase is where you hand draw (just to have a vague idea) the terrain/terrains of the project and things like how to main characters/items will look like. This will provide you with the idea of what you need to focus on the most and when to implement what phase of the project. Render time for an object is most important in the sense that if it takes too long for objects to render then game would not appeal to the user due to the laggy circumstances. Having the idea of how to want the game to be played, these blue prints/hand drawings will help in the rendering process as you will be able to know what parts of the objects need the rendering process and what parts of the objects won't need rendering as they won't be visible to the user in the game time hence rendering them will only result in heavier requirements for the game to be played. Rendering can also be reduced by the help of different techniques one of them used in the project is handling of the polygon count. This depends on what object and what part of the object is important for its recognition, the parts which aren't that important don't require many polygons hence improving the render time. Other techniques involve using of textures [4] instead of models and looping a certain texture. The work schedules start after the hand drawing the interfaces etc. where you need to implement the scripts and work on the designs (interfaces). The interfaces comprise of assets. An important thing while creating objects is that you need to focus on the hierarchy of the objects and which objects falls under which parent object. A common mistake made while creating items in the hierarchy is creating multiple items with the same name, this results in error when working on the scripts as the items are referred in the scripts, and if such errors are not detected in the earlier stages then it might cause major problems in later stages as coordination is affected. As working with Unity you also have to be aware of the glitches in the Unity environment, that are, bugs related to the physical aspects of the game. Unity works in a perfect environment initially where there isn't much physics implemented, for instance to add gravitational affects you need to add it's physics to the object and even after adding gravity there is no friction or any other force applied to the object other than gravity, in other words if you rotate the object to a 45 degree angle and click the play button the object will simple fall at the 45 degree angle and will stay in that angle even after hitting the ground. Hence being aware of the glitches in the gaming engine helps you process the game a lot better as you are aware of what physical aspects needs to added where and when. Another thing with Unity is that the developers needs to know that Unity is not a modeling tool, that is you need to model everything yourself in order to make them look realistic etc. As for the modeling phase the model you are to design should usually be of a higher quality as you might need to change the scale of the original model. If the model isn't of a higher quality and you scale the model to a larger size as that of the original model then its pixels might break and it may look unprofessional and unclear, this applies usually when you're designing the main plane for the game. Another way to avoid unclear environment is to use the texture in a loop as it will look clear even on a huge plane. A way used to remove modeling is using textures with mapping, this makes them look like realistic models but these can't be applied to the main objects (the one the user focuses on the most). Texturing is an important phase of the game as you take a 2D picture (diffused map) and introduce different map on to that diffused map in order to make it look 3D [5]. Different maps focus of different aspect, for instance if you want to change how the shine affects the parts of the image you manipulate the specular map of that particular image. Multiple maps merge and make a 3D like image. Problems related to integrating occur too as how to integrate textures built on other applications in Unity for Unity does not support every texture (file) format. Online tutorials and help from individuals help with the integration processes of other applications with Unity and how Unity saves their information. Integration problems usually occur when textures are to be integrated in the Unity environment (world) as they support just certain file formats. After the modeling, texturing, integrating and whole designing of the items/terrains is done, comes the testing part; either you can test the system all at once or you can perform the testing module by module. Different people use different approach to testing while in this project it was essential to perform testing module by module as fixing bugs in later stages would require more time and cost. After the testing by the project team it is necessary to launch the beta version of the game as a whole for further testing by the stakeholders/users which is to be done at the end of the semester.

# Chapter 3

# Requirement Specifications

## 3.1 Existing System

The existing systems (in Pakistan) lack quite a few things, for instance the existing systems in Pakistan the work on gaming graphics isn't on point. The system either just focuses on graphics while increasing the render time of the game. Another issue is the physical aspect of the games where the games lack physical sense. The existing systems also don't focus much on the interfaces of the game, the game needs to have an interface easy to understand and it shouldn't just have too many colors which don't have a soothing effect on the human eye.

## 3.2 Proposed System

The project has focused on human computer interaction firstly by improving the visual effect of the game. Secondly the project manages to blend the color combinations in perfectly giving the user a great experience. A usual gaming experience most users don't like is that most games focus too much on the graphical details and extra effects which can result in nausea and dizziness, this project has managed to reduce these effects by altering setting minimizing the screen shakes so that the user gets to have a great experience. Another thing the project manages to deliver is a good render time because of the graphical techniques such as texturing and giving a realistic effect by using multiple maps on a single image and giving an effect of that of a real object. The interfaces make use of usually the light color found appealing to the human eye. The glitches in the game however need some work and are likely to be improved in the near future. Every module of the game has been tested separately and also with modules it depends on.

## 3.3 Requirement Specification

The non-functional requirements focused in the project were to maintain a balance between the graphics of the game and the performance that is minimizing the render time of the game (in other words faking high quality graphics by using the mapping technique instead of rendering every object in the world, even the ones the user won't focus on much). Maintaining game human computer interaction standards were the top priority of the project which was achieved through research on interaces (what color combination would suit what interface).

## 3.4 Use Case

The use case diagram basically illustrates most of the functional requirements of the application and an overview of how the users will interact with the application once it was built and put into use. Every use case represents a different task or event that our application will do. The main use case diagram of our application is shown in figure 3.1. The actors in this case are the persons who will be using the system which are our general users.

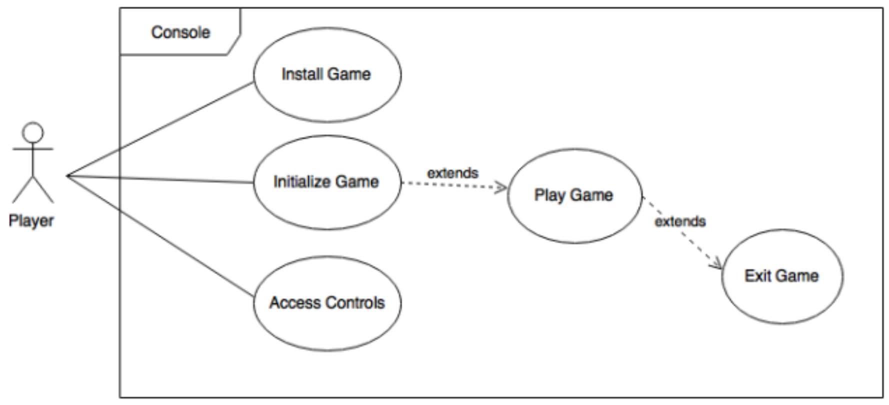

Figure 3.1: Use Case: Console

| | |
|---|---|
| **Actor** | Player/User |
| **Brief Description** | Describes how player can access controls and use them while playing the game. |
| **Pre-Condition** | The machine must meet the system requirements. |
| **Post-Condition** | (Beta version) The player provides feedback to if the game needed improvement. |

Table 3.1: Use Case: Console Specificaion Table

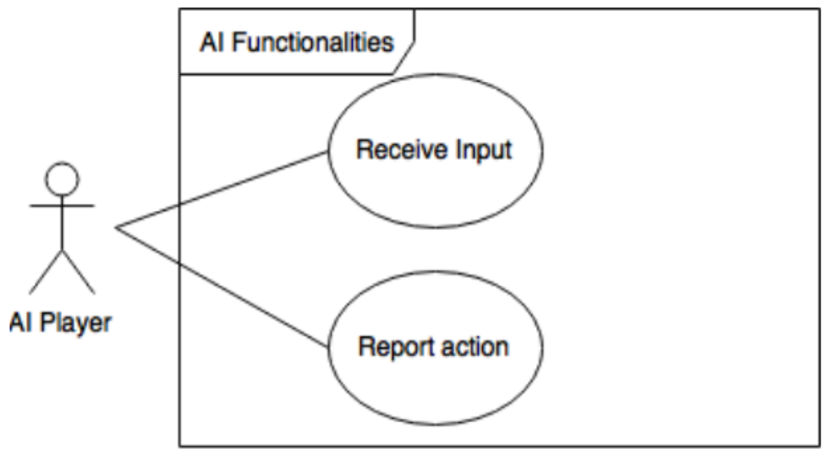

Figure 3.2: Use Case: AI Functionalities

| | |
|---|---|
| **Actor** | Artificial Intelligence |
| **Brief Description** | Describes the artificial intelligence's response to different input. |
| **Pre-Condition** | The player must be in the AI's territory |
| **Post-Condition** | Elimination of either the player or the AI (in case the player dies the game ends) |

Table 3.2: Use Case: AI Functionalities Specificaion Table

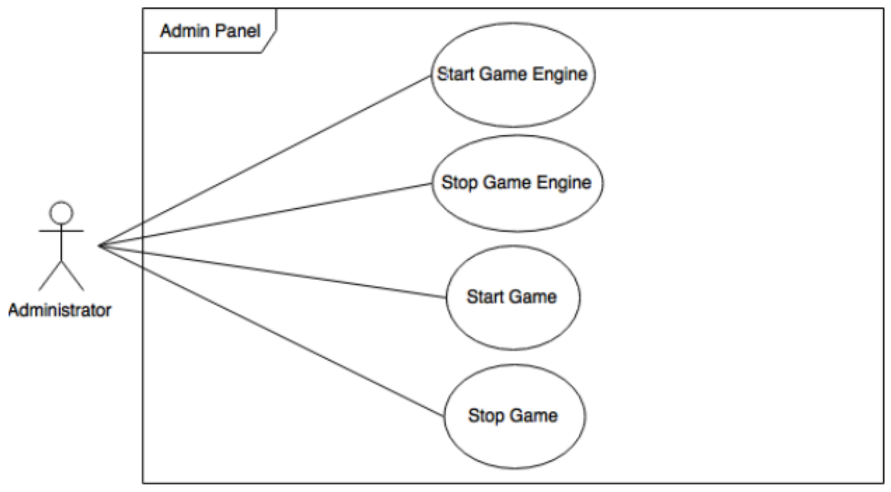

Figure 3.3: Use Case: Admin Panel

| | |
|---|---|
| **Actor** | Production team/Admin |
| **Brief Description** | Enable/Disable user's access to the game. |
| **Pre-Condition** | Reports submit (for a certain user) |
| **Post-Condition** | Re-access after a certain period of time (for the player) |

Table 3.3: Use Case: Console Specificaion Table

# Chapter 4

# Design

## 4.1 Class Diagrams

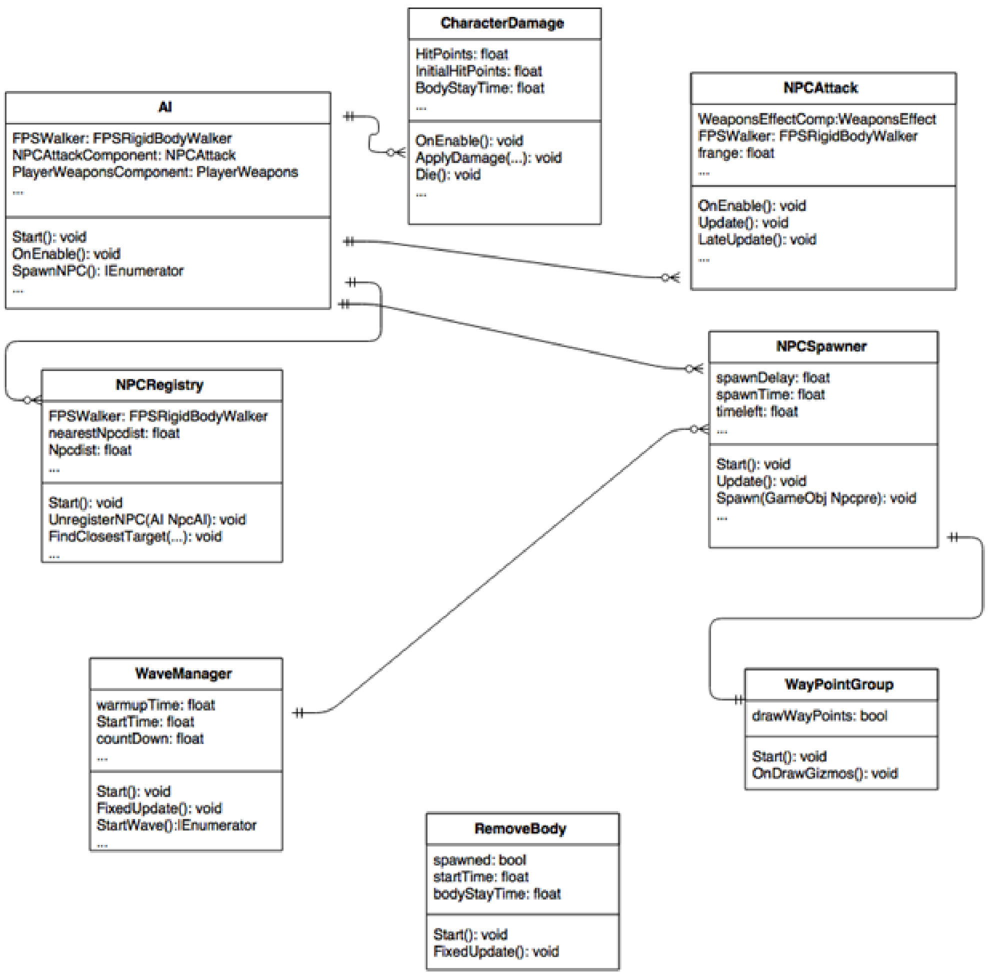

Figure 4.1: AI Asset

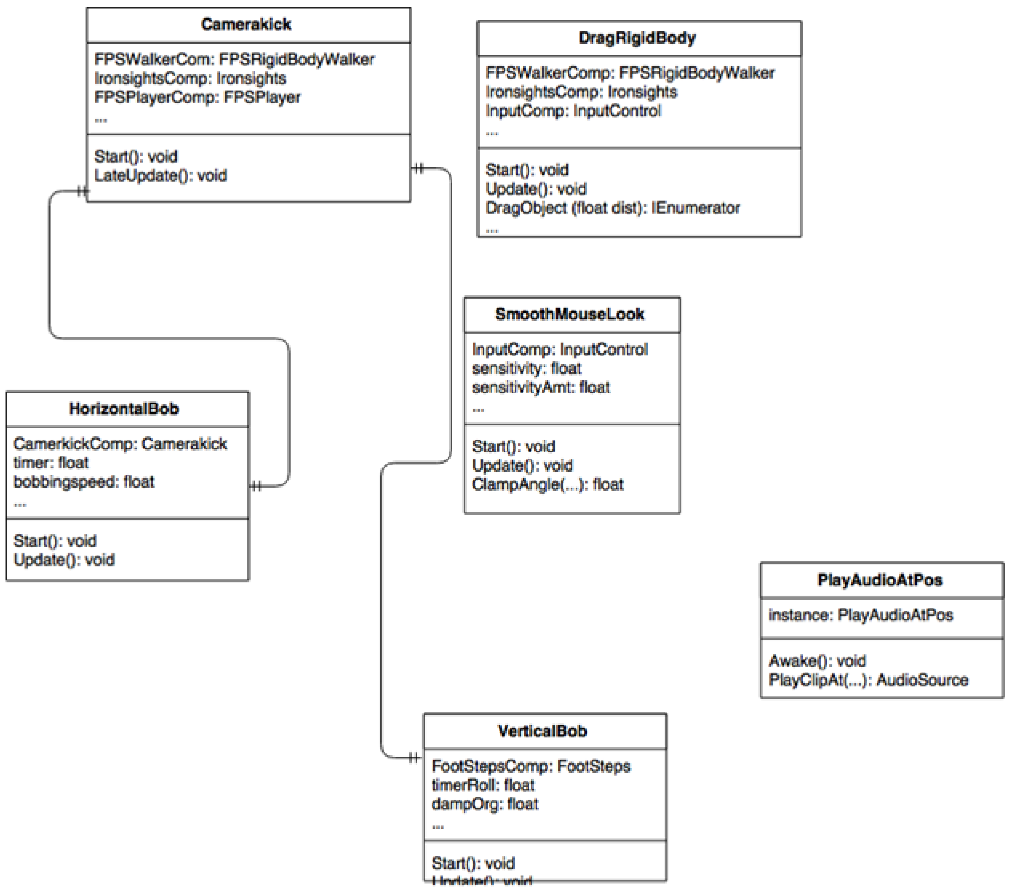

Figure 4.2: Camera Asset

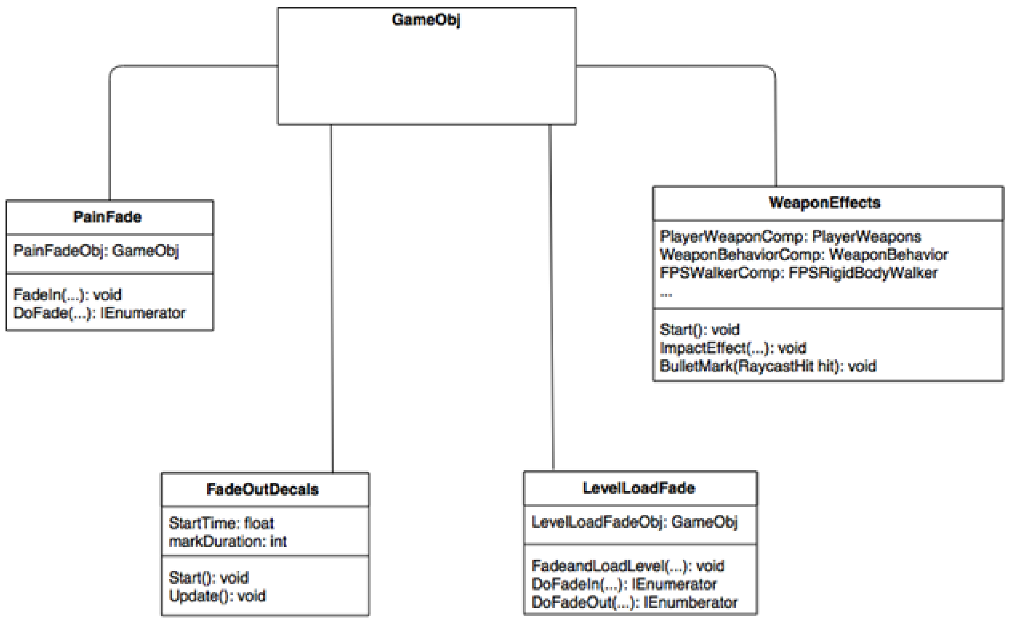
GameObj
PainFade
PainFadeObj: GameObj
FadeIn(...): void
DoFade(...): IEnumerator
WeaponEffects
PlayerWeaponComp: PlayerWeapons
WeaponBehaviorComp: WeaponBehavior
FPSWalkerComp: FPSRigidBodyWalker
...
Start(): void
ImpactEffect(...): void
BulletMark(RaycastHit hit): void
FadeOutDecals
StartTime: float
markDuration: int
Start(): void
Update(): void
LevelLoadFade
LevelLoadFadeObj: GameObj
FadeandLoadLevel(...): void
DoFadeIn(...): IEnumerator
DoFadeOut(...): IEnumberator

Figure 4.3: Effect Asset

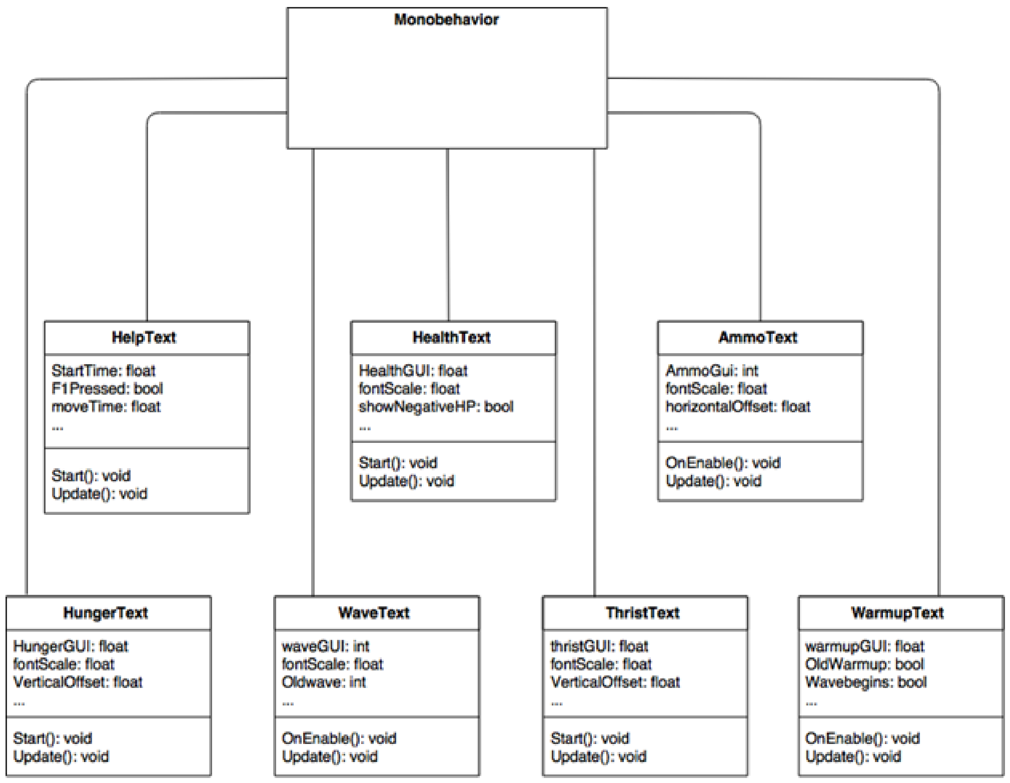

Figure 4.4: HUD Asset

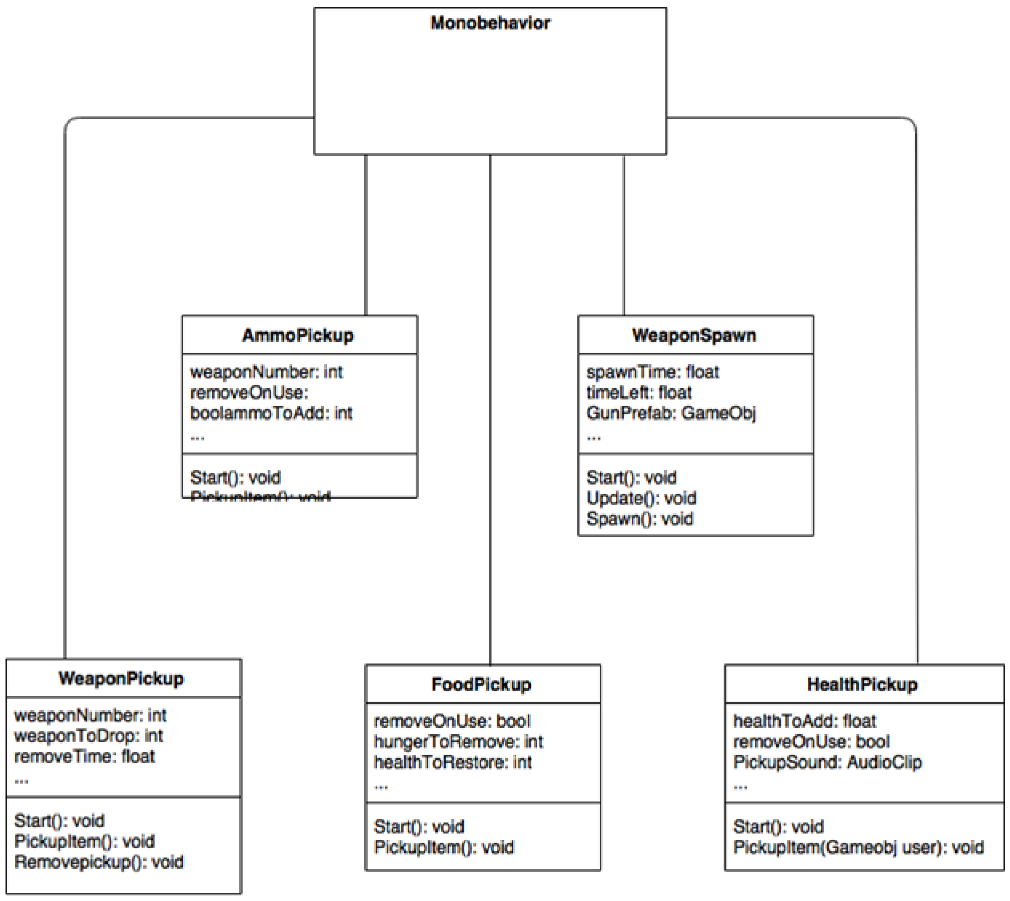

Figure 4.5: Items Asset

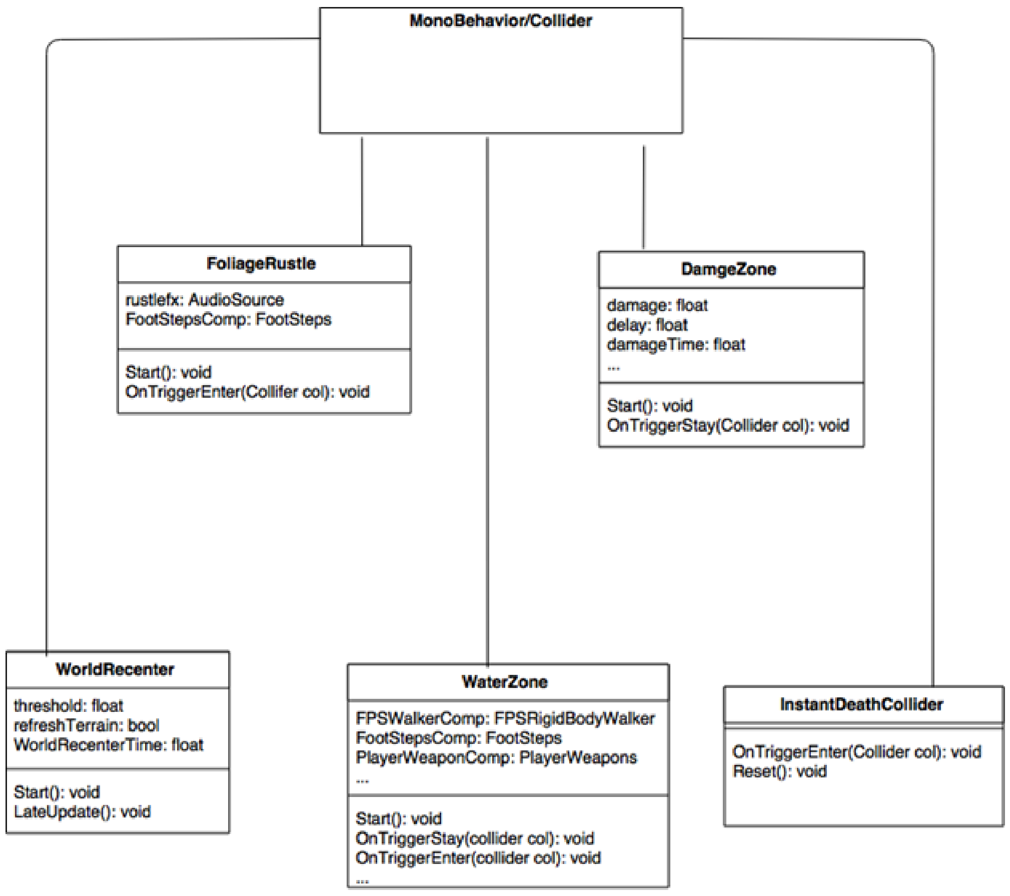

Figure 4.6: Objects Asset

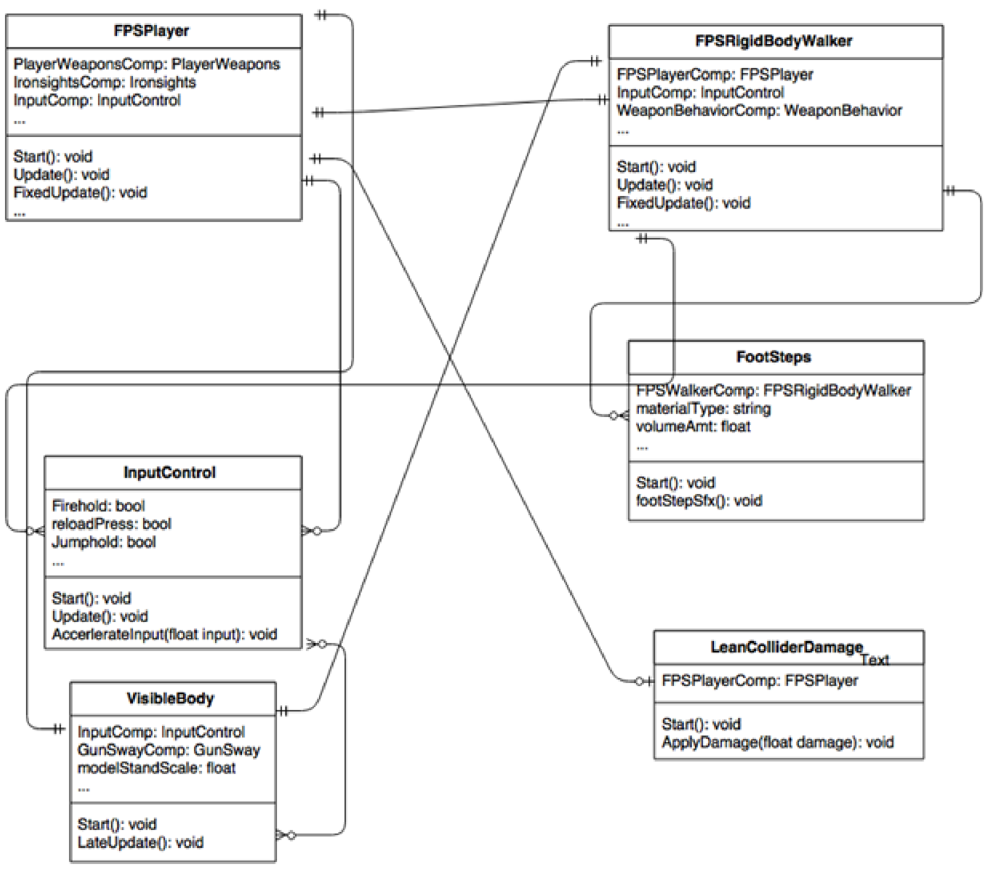

Figure 4.7: Player Asset

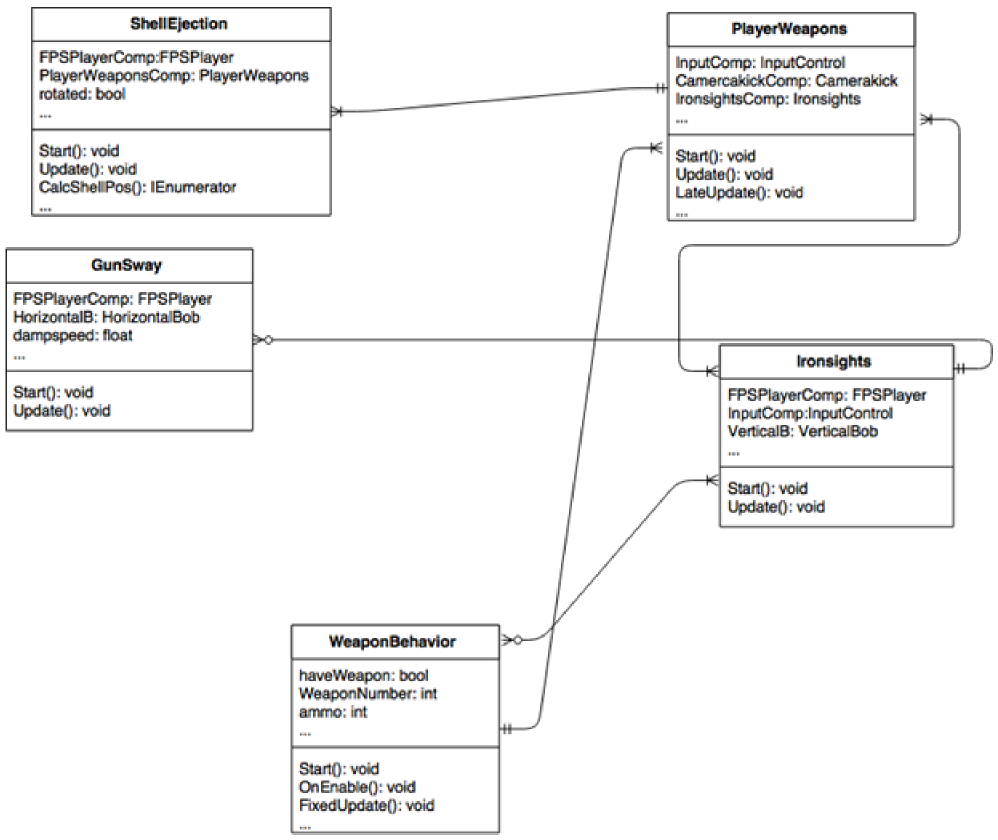

Figure 4.8: Weapons Asset

## 4.2 System Architecture

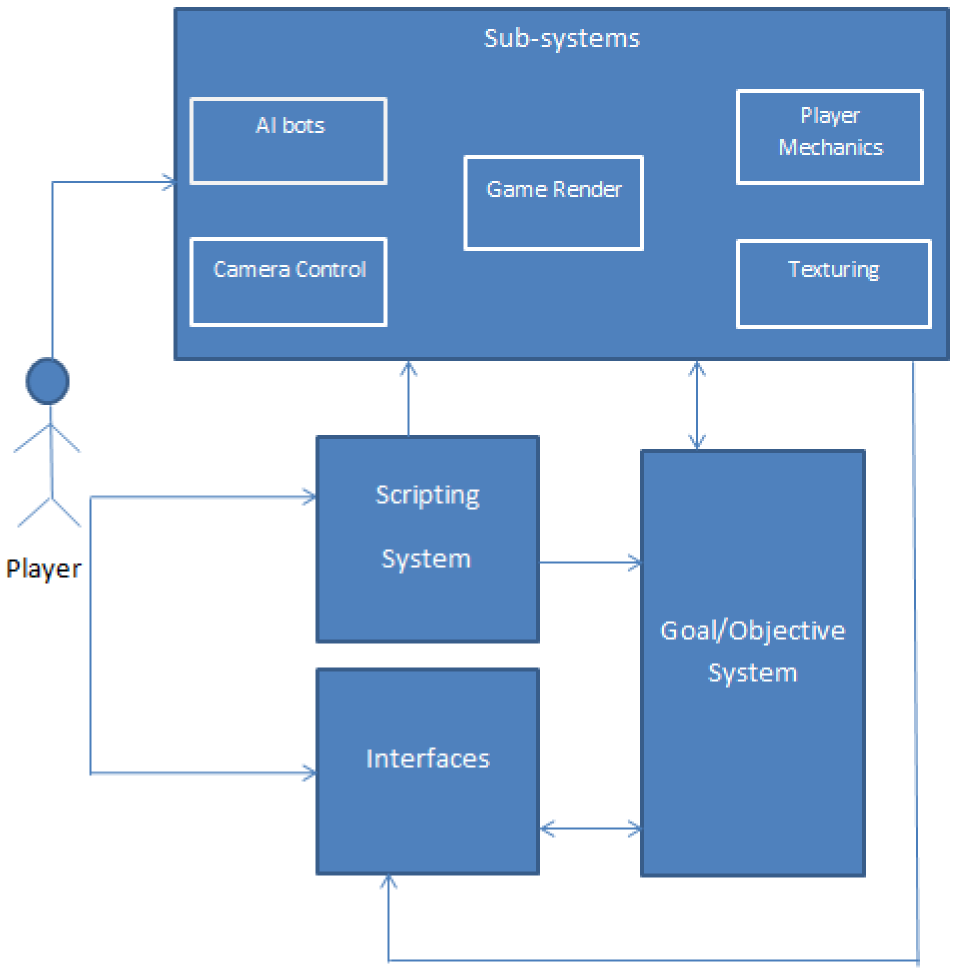

Figure 4.9: System Architecture

## 4.3 Basic Flow of the Class Diagram between assets

The diagram shows the basic interdependencies among classes of different assets (the boxes in the diagram represent the assets while the edges show the use of objects of different classes between the assets):

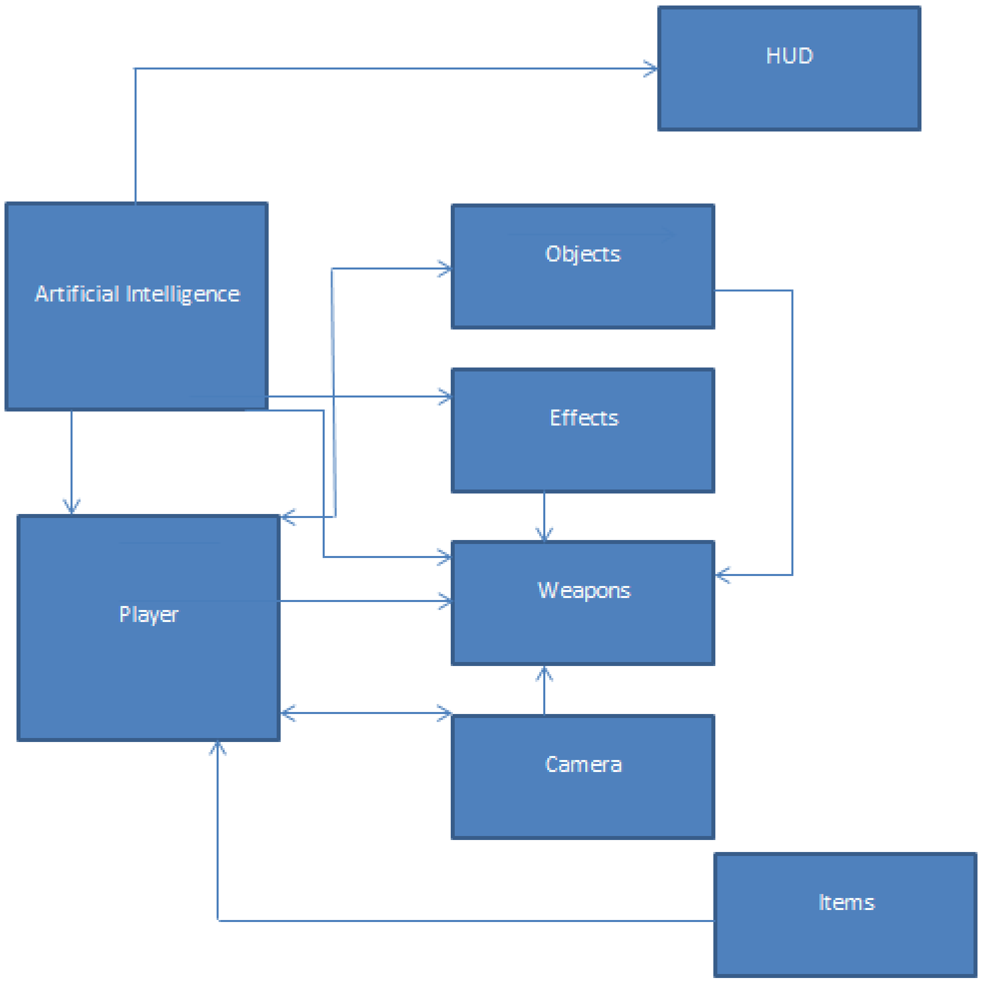

Figure 4.10: Basic Asset Diagram

## 4.4 Design Constraint

There were a few design constraints as their cost (time) exceeded the expected completion date, the constraints included the inter communication between softwares used in the project. The texturing done had to be in a specific format so that Unity could support it. This was achieved using the software known as blender as it also functions in the format supported by the Unity gaming engine. Another constraint was that of the programming language for scripting as Unity only supported C-Sharp or javascripting. The project didn't involve any funding, unfortunately the team had work on the free version of the softwares which resulting in them not having the access to many extensive software features. There were only two maps used on the objects while preparing them; one being the diffused map and the other being the normal map. A group of mixers, adders etc. were used to make shaders which supported both metal and non-metal objects which were used in the node mode to add final touches to the objects (manipulating the strength, color/non-color data etc. according the requirement for the object). Following are a few screen shots of the objects while being prepared; the screenshots showing three views (3D view, node editor, rendered view NOTE: cycle render is used as blender render doesn't support node editor ):

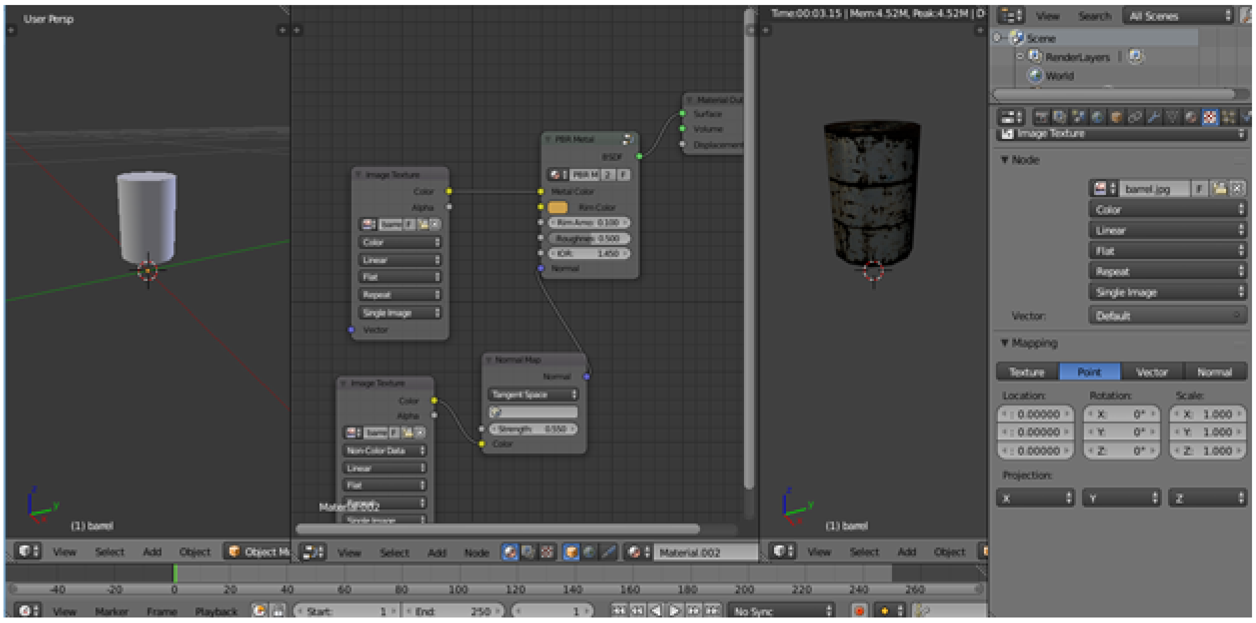

Figure 4.11: Image shows the use of PBR$_m etal(usedformetalobjects)$

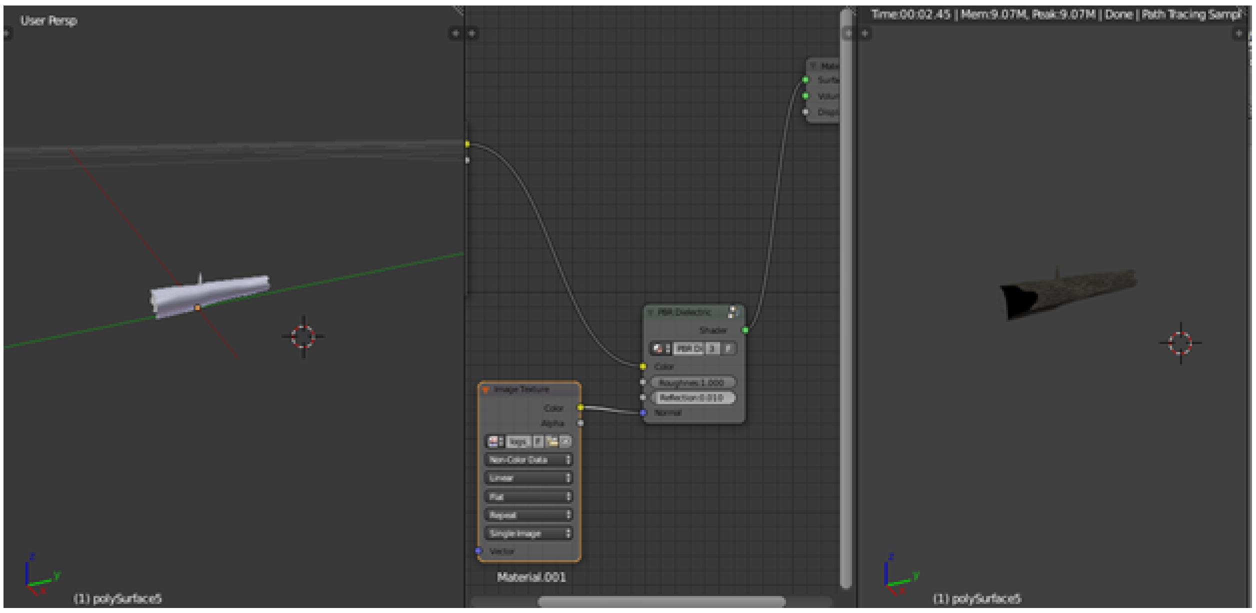

Figure 4.12: Image shows the use of PBR$_dielectric$

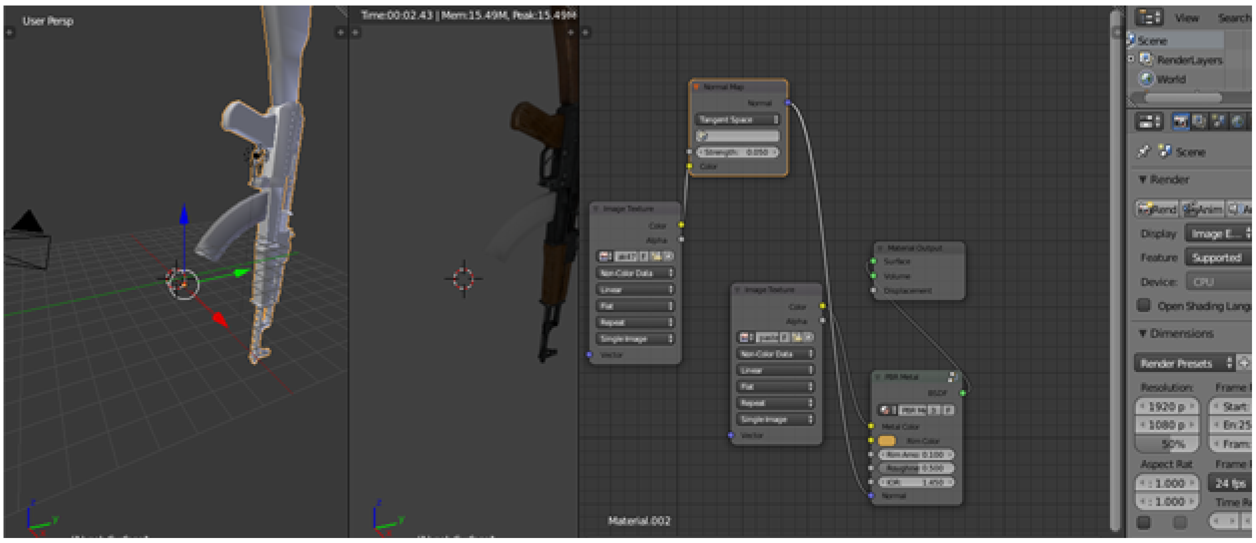

Figure 4.13: Images show the use of both PBR$_metalaswellasPBR_dielectric$

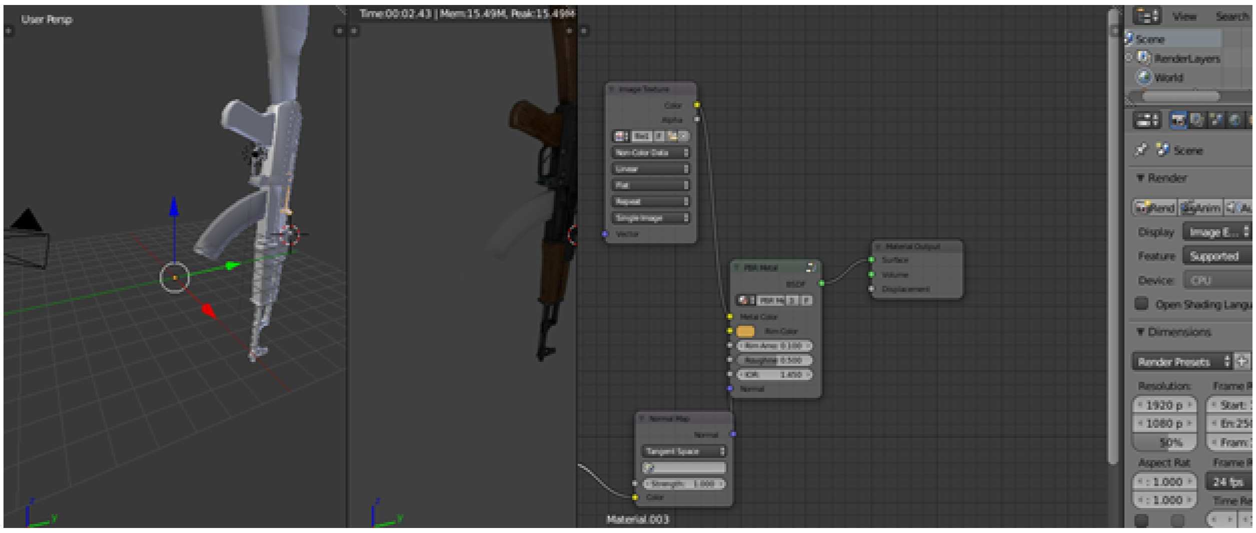

Figure 4.14: Using maps for the main body for the ak 47

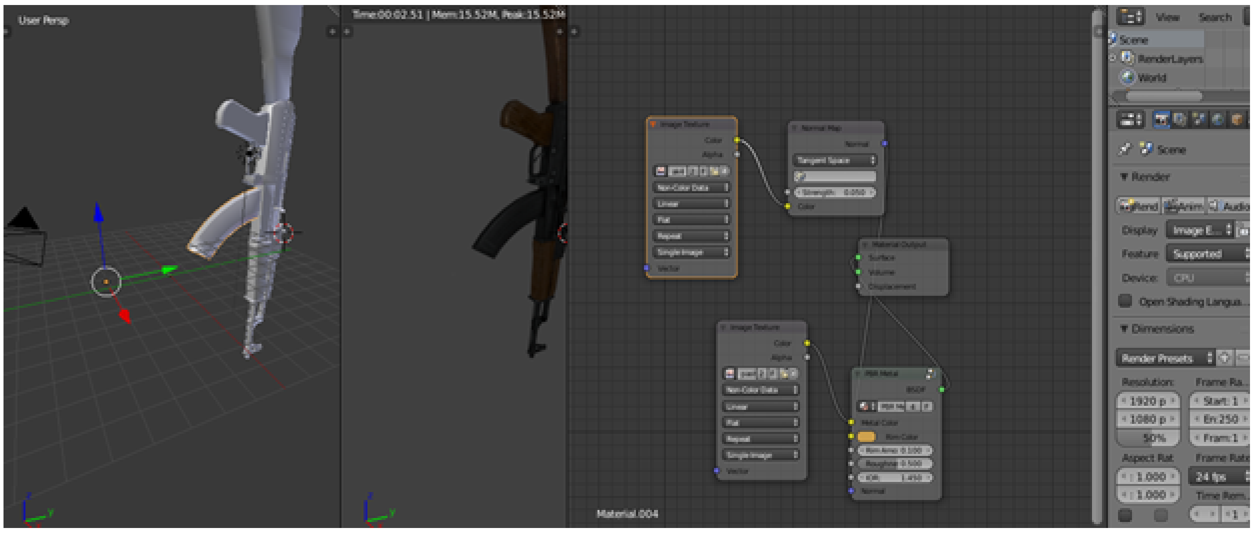

Figure 4.15: Using maps for the recoil of the ak 47 (the metal part)

# Chapter 5

# System Implementation

As mentioned in the previous chapter the project consists of a scripting portion, an interface portion, a sub system portion which touches the modeling and artificial aspect of the game.

## 5.1 System Architecture

The scripting the performed through C sharp, may it be of any object; while the modeling and texturing is done on blender and photoshop. The blender file format is supported in unity hence there weren't any communication (merging) problems. A tool named "crazybump" helped in mapping of different object, it works on the simple principle of creating normal, specular map etc. any picture provided making them give a 3d effect ( used as it improved the cost and time of the project). Collision and rigid body etc. algorithms were used to help with the physical aspect of the game (APIs); artificial intelligence algorithms were studied and manipulated as per the project needs. The project however lacked security however converting the C output to bytes and then encrypting it was the basic technique that was to be applied but due to lack of knowledge about the encryption algorithms and techniques the team left the security part as they resulted in further bugs etc.

# Chapter 6

# System Testing and Evaluation

The testing approach used in this particular project as per requirement was the modular testing; in other words the testing was incrementally testing. The big bang approach wasn't used as this project wasn't of a small scale and there were inter dependencies in modules or assets of the project. The main focus of the testing phase was on the physical aspect of the game (the C code), these testing check if collisions were working fine and what is the impact of an object on other objects present in the game (if contacted with), along with that there were some checks on the bullet projection (whether if it should be influenced by factors like wind etc). All the modules weren't testing due to the time constraints. The testing approach used was white box testing as well as black box testing (in other words we gray box tested the project), for the white box testing we constructed CFGs and then created du path table, definition table and uses table; this helped us make testing cases as we knew about the conditions through the CFGs. The graphics user interface was testing through beta testing of the game, making the user interact (play) and getting their feedbacks to what is to their liking and what module needs improvement (what the module lacks). The user provided us with information like if a module contains too much details ( isn't interactive enough), which resulted in findings like what if the game is interactive, easy to play, fun and other information from the user. As there weren't security measures in the game hence there wasn't any security test.

| | |
|---|---|
| **Test Case ID** | TCAI-1 |
| **Brief Description** | Enemy Radar/Enemy Spawn |
| **Action** | 1) Player reaches in enemy territory<br>2) Gaming Engine spawns enemy |
| **Expected Result** | Player sees enemy. |
| **Status** | True |
| **Remarks** | N/A |

Table 6.1: Test Case: Artificial Intelligence 1

| | |
|---|---|
| **Test Case ID** | TCAI-2 |
| **Brief Description** | AI fire at sight |
| **Action** | AI fires as soon as player comes in range. |
| **Expected Result** | 1) Player/AI loses health if hit<br>2) AI attacks player |
| **Status** | True |
| **Remarks** | N/A |

Table 6.2: Test Case: Artificial Intelligence 2

| | |
|---|---|
| **Test Case ID** | TCWeapon-1 |
| **Brief Description** | Use Weapon |
| **Action** | 1) Fire Weapon (left click mouse)<br>2) Reload (press r) Weapon<br>3) Switch Weapon<br>4) Pickup Weapon<br>5) Hit with Weapon |
| **Expected Result** | 1) Bullet count changes/reloads<br>2) Weapon changes<br>3) Weapon comes in your backpack and disappears from the pickup position<br>4) Player moves arm to swap weapon |
| **Status** | True |
| **Remarks** | N/A |

Table 6.3: Test Case: Weapon 1

| | |
|---|---|
| **Test Case ID** | TCWeapon-2 |
| **Brief Description** | Zoom on special Weapons |
| **Action** | Right click on mouse |
| **Expected Result** | Camera change on basic viewport |
| **Status** | True |
| **Remarks** | N/A |

Table 6.4: Test Case: Weapon 2

| | |
|---|---|
| **Test Case ID** | TCBasics-1 |
| **Brief Description** | Move in 3D world |
| **Action** | Use move keys on keyboard |
| **Expected Result** | Players changes its coordinates in world |
| **Status** | True |
| **Remarks** | N/A |

Table 6.5: Test Case: Basics

# Chapter 7

# Conclusions

TThe project helped us understand about the character skeleton basics, how joints are positioned, material settings and animations of characters. The project furthermore provided how using triangles instead of polygons can change your render mechanics (how it improves it and this also helps users with “not the high end pcs” to be able to play the game). Silhouettes are to be specified before the actual execution of the design; also when you’re making a game the color combination while creating a character is very important as the character should not have to many color or the color contrast on the character’s body shouldn’t vary too much from one body part to another as it is not visually pleasing to the user. Glitches play an important part in the game as they make the game appear to be more enjoyable. Most importantly the project helped in understanding time management and team work.

## 7.1 Future Enhancements

Future enhancements include procedural level generation along with procedural content generation. A multiplayer option is also planned which then will allow more administrative control; for instance the admin will be able to temporarily ban an IP for communication, ability abuse etc., this will also help in establishing a better gaming community. Moreover new weapons and techniques will be introduces to meet better gaming standards.

# References

[1] Unity. (n.d.). Retrieved from Unity 3D: http://docs.unity3d.com/Manual/Preparingacharacterfromscrat
Cited on p. 2.

[2] J. (2014). C Game Programming Cookbook for Unity 3D. W.Murray. Cited on p. 2.

[3] T. (2007). Introducing Character Animation with Blender. Mullen. Cited on p. 2.

[4] Andrew. (n.d.). Retrieved from Blender:https://www.blender.org/support/tutorials. Cited on p. 3.

[5] A. (n.d.). Retrieved from Youtube: https://www.youtube.com/blender$_g$*uruPrice*.Citedonp.4.