Juego de Tiroteo

Prepared by

Priyal Ravat (16IT108)

Under the supervision of

Prof. Jalpesh Vasa

A Report Submitted to
Charotar University of Science and Technology
for Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology
in Information Technology

IT345 Software Group Project-II (B.Tech IT 5th sem)

Submitted at

DEPARTMENT OF INFORMATION TECHNOLOGY

Chandubhai S. Patel Institute of Technology

At: Changa, Dist: Anand – 388421

October 2018

CERTIFICATE

This is to certify that the report entitled "**Entertainment with Game**" is a bonafied work carried out by **Ms. Priyal Ravat (16IT108)** under the guidance and supervision of **Prof. Jalpesh Vasa** for the subject **Software Group Project-II(IT345)** of **5th** Semester of Bachelor of Technology in **Information Technology** at Faculty of Technology & Engineering – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate **herself**, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under supervision of,

Prof. Jalpesh Vasa
Assistant Professor
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Parth Shah
Head & Associate Professor
Department of Information Technology
CSPIT, Changa, Gujarat.

Chandubhai S Patel Institute of Technology

At: Changa, Ta. Petlad, Dist. Anand, PIN: 388 421. Gujarat

TABLE OF CONTENTS

- **Abstract 5**
- **Acknowledgement 6**
- **Chapter 1 Introduction 7**

1 Project Overview 7

1.1 How to do 7

2 Scope 8

- **Chapter 2 System Analysis 9**

1 User Characteristics 9

2 Tools & Technology

.... 10

2.2.1 How to do 11

- **Chapter 3 System Design 13**

3.2 GUI Forms 13

- **Chapter 4 Implementation 15**

4.3 Coding Standards 15

4.4 Snapshots of project 19

- **Chapter 5 Constraints and Future Enhancement 22**
- **Chapter 6 Conclusion 23**
- **References 24**

LIST OF FILES

- **Fig 1.1 Character.h file 15**
- **Fig 1.2 Character.cpp file 16**
- **Fig 2.1 ProjectGameModeBase.cpp file 18**
- **Fig 2.2 ProjectGameModeBase.h file 18**

ABSTRACT

In this report, I present the detailed development and implementation of simple Shooting game. The Shooting game consists of graphical user interface, moving character ; implemented using c++. The moving character is implemented using efficient algorithm. This project gives an insight in to different aspects of c++ programing.

Acknowledgment

I would like to express my deepest appreciation to all those without whom this project would not have been possible without the support, guidance and help of many individuals. First, I would like to thank the Department of Information Technology (CHARUSAT) for making me aware about the need of exposure in the era of severe competition. I would like to thank Mr. Praveen Jain, Founder and CEO of Indian Wealth Management (IWM) for giving me an opportunity to use my knowledge and skills in real-time development. I would also like to express special gratitude to Prof. Nirav Bhatt, Department of Information Technology, for guiding and imparting necessary information whenever needed. I would like to thank my team member Jay Nankani for helping and working with me throughout the project. I would also like to thank the Finance Team of IWM for helping us bridge the gap between the finance and technical related theories.

- Introduction

1. **Project Overview:**

In this project I am making a fun Game, which is really interesting for play and quit difficult to make. I make a first person shooting game which moves forward, it also moves backward and jumps when we press the keys. It also has a main menu from where you can select to play, you can select to quit the game and also change the resolution of the screen using the main menu options tag.

1.1 How to do

For making any game is not an easy task. First you have to choose which type of game you have to make, then you have to choose the editor for making the game. There are multiple types of game and multiple types of editor for making the game. I choose a first person shooting game, and I choose the unreal engine editor for making this game.
I am using unreal engine 4.20 version which is supported by visual studio 2017. I choose the unreal engine because in this engine you can write the code in programing c++ language. Which is only little bit of easy to code. But making this game is going to be difficult a bit.

2. Scope:

In the entertainment making field it is not an easy task to make an amazing game. You have to put your all efforts for the game. I know a little bit about the programing c++ language. I done the main menu for the game which is contained of play, options and quit game. Using options you can choose the screen resolution in your screen. I have done some movements in the character. Like he can move forward, backward, left and right using c++ code. And I also add a jump action in the character so he can jump during the game. I also made a website for the game. I don't have enough time to do all the things so I have done the mentioned things in the game.

- ## System Analysis:

1. User characteristic:

Newbie Gamer
The Newbie Gamer doesn't play many games and is not savy on the popular subculture surrounding modern games. This has several consequences: He is doesn't know the definition of common gaming terms, he is unfamiliar with conventional user interface practices (such as hot keys and control groups, for instance), he hopes to find adequate tutorials and help when learning to play the game, he is likely to stop playing the game if the controls are complex or non-intuitive, and finally he's looking for a game that will be fun to play even if he's not good at it.

Power Gamer
The Power Gamer has played every game and is a master at all of them. He is aware of modern terminology and conventional interface practices. He is looking for a challenge, a game which will allow him to demonstrate that his skill and ability are superior to all other gamers. He plays to win and will spend hours playing the game to hone their play for even the slightest advantage. However, he must feel that he has control over the outcome of the game for the game to be meaningful. He must feel that if he hadn't played at peak performance and used all of his resources to win then he could have lost or would have lost. He is likely to enjoy games that require skill rather than luck.

Mr Smarty Pants
This gamer wants a game that forces him to think and use his brain to win. The game must push him intellectually for him to find it engaging. He is likely to know the terminology and conventional interface practices surrounding modern strategy games. He delights in outwitting an opponent, even if he loses in the end. He is likely to spend hours analyzing the game to find optimal strategies and best practices. He is more likely to prefer an interesting match or an even match to a landslide victory. He likes a game which can surprise him with novel situations each time he plays it.

2. Tools and Technology:

Unreal Engine

UNREAL ENGINE

Unreal Engine 4.20 screenshot

Developer(s)	Epic Games
Initial release	May 1998; 20 years ago
Stable release	4.20.3 / September 19, 2018; 32 days ago
Preview release	4.21 / October 10, 2018; 11 days ago
Written in	C++
Type	Game engine
License	Source-available commercial software with royalty model for commercial use[1]
Alexa rank	▲ 1,296 (As of May 22, 2018)[2]

Website	unrealengine.com

The **Unreal Engine** is a source-available game engine developed by Epic Games, first showcased in the 1998 first-person shooter game *Unreal*. Although primarily developed for first-person shooters, it has been successfully used in a variety of other genres, including stealth, fighting games, MMORPGs, and other RPGs. With its code written in C++, the Unreal Engine features a high degree of portability and is a tool used by many game developers today.

The current release is Unreal Engine 4, designed for Microsoft Windows, macOS, Linux, SteamOS, HTML5, iOS, Android, Nintendo Switch, PlayStation 4, Xbox One, Magic Leap One, and virtual reality.

UnrealScript:

UnrealScript	
Paradigm	Object-oriented, generic
Developer	Tim Sweeney
First appeared	May 1998; 20 years ago
Typing discipline	Static, strong, safe
OS	Cross-platform (multi-platform)
Filename extensions	.uc .uci .upkg
Website	api.unrealengine.com
Influenced by	
C++, Java	

UnrealScript (often abbreviated to UScript) was Unreal Engine's native scripting language used for authoring game code and gameplay events before the release of Unreal Engine 4. The language was designed for simple, high-level game programming. The UnrealScript interpreter was programmed by Sweeney, who also created an earlier game scripting language, ZZT-oop.

Similar to Java, UnrealScript is object-oriented without multiple inheritance (classes all inherit from a common Object class), and classes are defined in individual files named for the class they define. Unlike Java, UnrealScript does not have object wrappers for primitive types. Interfaces are only supported in Unreal Engine generation 3 and a few Unreal Engine 2 games. UnrealScript supports operator overloading, but not method overloading, except for optional parameters.

At the 2012 Game Developers Conference, Epic announced that UnrealScript was being removed from Unreal Engine 4 in favor of C++. Visual scripting would be supported by the Blueprints Visual Scripting system, a replacement for the earlier Kismet visual scripting system.

Visual Studio 2017:

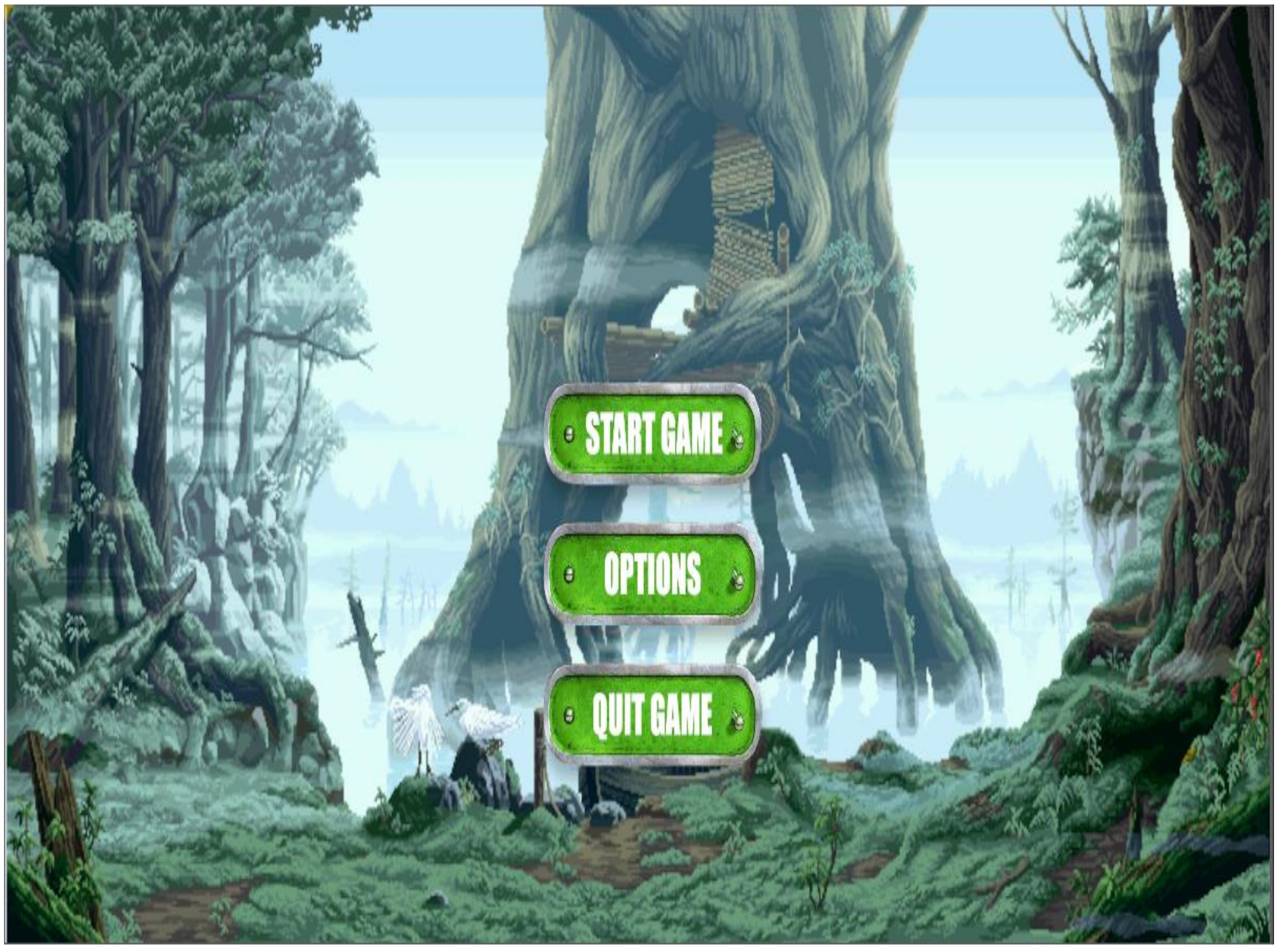
Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

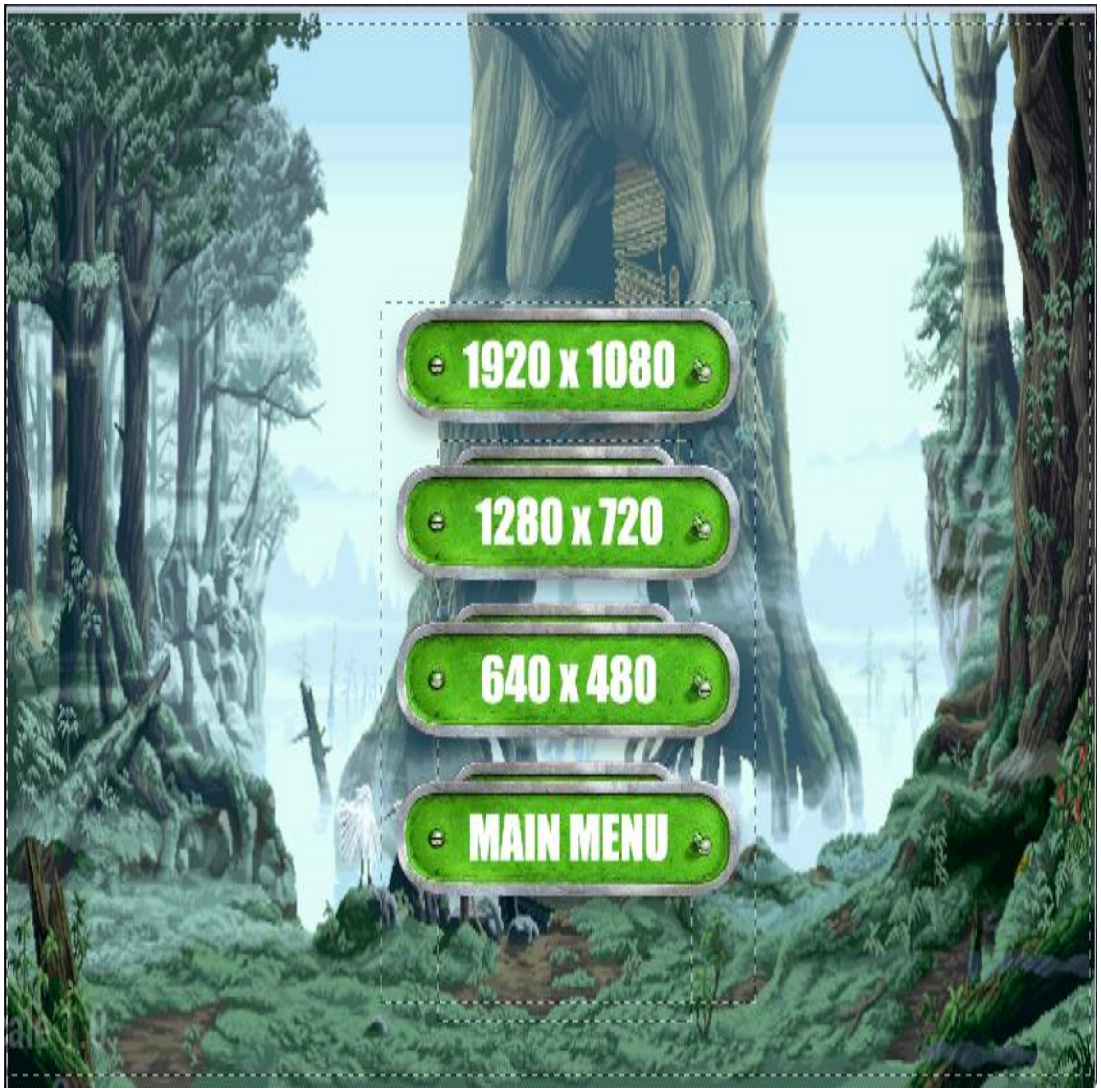
Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source control systems (like Subversion and Git) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS. Support for other languages such as Python, Ruby, Node.js, and M among others is available via plug-ins. Java(and J#) were supported in the past.

- **System Design:**

1. GUI Design:




1920 x 1080
1280 x 720
640 x 480
MAIN MENU

- **Implementation:**

1. Coding Standard:

Character.h file:

```
// Fill out your copyright notice in the Description page of Project Settings.

#pragma once

#include "GameFramework/Character.h"
#include "FPSCharacter.generated.h"

UCLASS()
class FPSPROJECT_API AFPSCharacter : public ACharacter
{
	GENERATED_BODY()

public:
	// Sets default values for this character's properties
	AFPSCharacter();

protected:
	// Called when the game starts or when spawned
	virtual void BeginPlay() override;

public:
	// Called every frame
	virtual void Tick(float DeltaSeconds) override;

	// Called to bind functionality to input
	virtual void SetupPlayerInputComponent(class UInputComponent*
PlayerInputComponent) override;

	// Handles input for moving forward and backward.
	UFUNCTION()
		void MoveForward(float Value);

	// Handles input for moving right and left.
	UFUNCTION()
		void MoveRight(float Value);

	// Sets jump flag when key is pressed.
	UFUNCTION()
		void StartJump();

	// Clears jump flag when key is released.
	UFUNCTION()
```

```
		void StopJump();

	// Function that handles firing projectiles.
	UFUNCTION()
		void Fire();



	// Gun muzzle's offset from the camera location.
	UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Gameplay)
		FVector MuzzleOffset;

	// Projectile class to spawn.
	UPROPERTY(EditDefaultsOnly, Category = Projectile)
		TSubclassOf<class AFPSProjectile> ProjectileClass;
	};
```

Character.cpp file:

```
// Fill out your copyright notice in the Description page of Project Settings.

//#include "FPSProject.h"
#include "FPSCharacter.h"
#include "FPSProject.h"
// Sets default values
AFPSCharacter::AFPSCharacter()
{
	// Set this character to call Tick() every frame.  You can turn this off to
improve performance if you don't need it.
	PrimaryActorTick.bCanEverTick = true;

}

// Called when the game starts or when spawned
void AFPSCharacter::BeginPlay()
{
	Super::BeginPlay();

	if (GEngine)
	{
		// Put up a debug message for five seconds. The -1 "Key" value (first
argument) indicates that we will never need to update or refresh this message.
		GEngine->AddOnScreenDebugMessage(-1, 5.0f, FColor::Red, TEXT("We are using
FPSCharacter."));
	}
}

// Called every frame
void AFPSCharacter::Tick(float DeltaTime)
{
	Super::Tick(DeltaTime);

}
```

```
// Called to bind functionality to input
void AFPSCharacter::SetupPlayerInputComponent(class UInputComponent*
PlayerInputComponent)
{
	Super::SetupPlayerInputComponent(PlayerInputComponent);

	// Set up "movement" bindings.
	PlayerInputComponent->BindAxis("MoveForward", this, &AFPSCharacter::MoveForward);
	PlayerInputComponent->BindAxis("MoveRight", this, &AFPSCharacter::MoveRight);

	// Set up "action" bindings.
	PlayerInputComponent->BindAction("Jump", IE_Pressed, this,
&AFPSCharacter::StartJump);
	PlayerInputComponent->BindAction("Jump", IE_Released, this,
&AFPSCharacter::StopJump);
	PlayerInputComponent->BindAction("Fire", IE_Pressed, this, &AFPSCharacter::Fire);
}

void AFPSCharacter::MoveForward(float Value)
{
	// Find out which way is "forward" and record that the player wants to move that
way.
	FVector Direction = FRotationMatrix(Controller-
>GetControlRotation()).GetScaledAxis(EAxis::X);
	AddMovementInput(Direction, Value);
}

void AFPSCharacter::MoveRight(float Value)
{
	// Find out which way is "right" and record that the player wants to move that
way.
	FVector Direction = FRotationMatrix(Controller-
>GetControlRotation()).GetScaledAxis(EAxis::Y);
	AddMovementInput(Direction, Value);
}

void AFPSCharacter::StartJump()
{
	bPressedJump = true;
}

void AFPSCharacter::StopJump()
{
	bPressedJump = false;
}

void AFPSCharacter::Fire()
{
}
```

ProjectGameModeBase.cpp file:

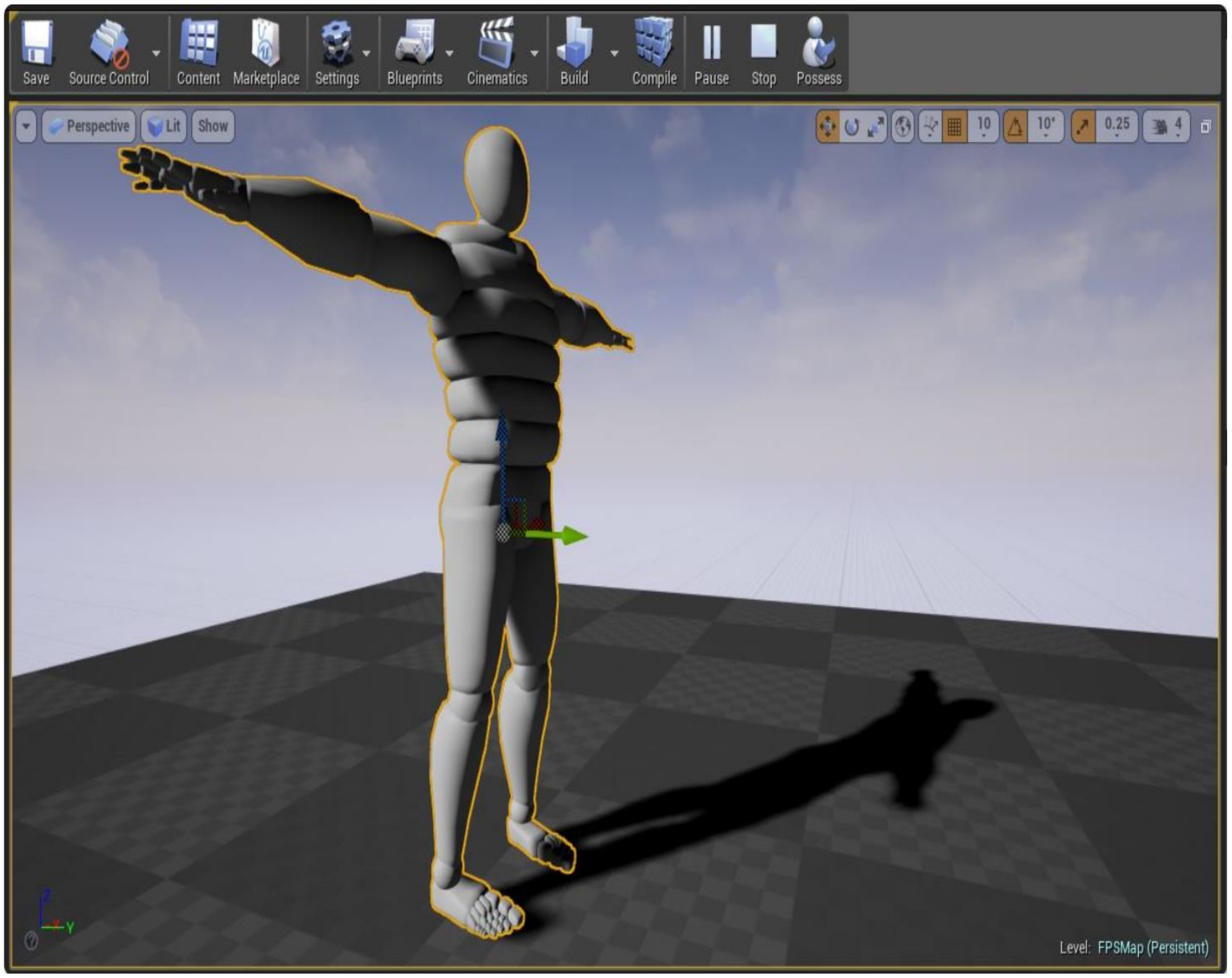
```
// Fill out your copyright notice in the Description page of Project Settings.

#include "FPSProjectGameModeBase.h"
#include "FPSProject.h"

void AFPSProjectGameModeBase::StartPlay()
{
	Super::StartPlay();

	if (GEngine)
	{
		// Display a debug message for five seconds.
		// The -1 "Key" value (first argument) indicates that we will never need to
update or refresh this message.
		GEngine->AddOnScreenDebugMessage(-1, 5.0f, FColor::Yellow, TEXT("Hello
World, this is FPSGameMode!"));
	}
}
```

ProjectGameModeBase.h file:

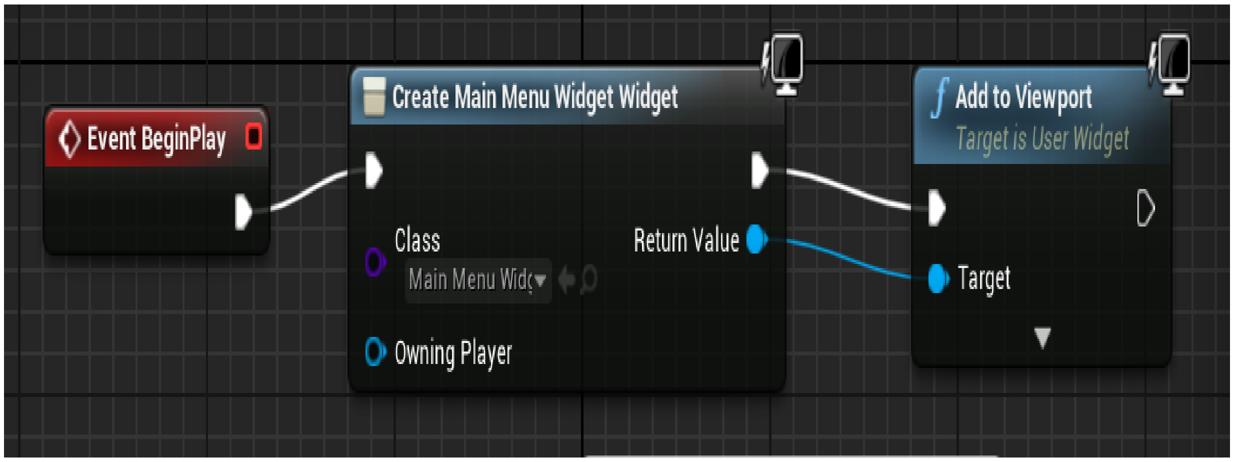
```
// Fill out your copyright notice in the Description page of Project Settings.

#pragma once

//#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "FPSProjectGameModeBase.generated.h"

/**
 *
 */
UCLASS()
class FPSPROJECT_API AFPSProjectGameModeBase : public AGameModeBase
{
	GENERATED_BODY()

		virtual void StartPlay() override;


};
```

2. Snapshot of Project:

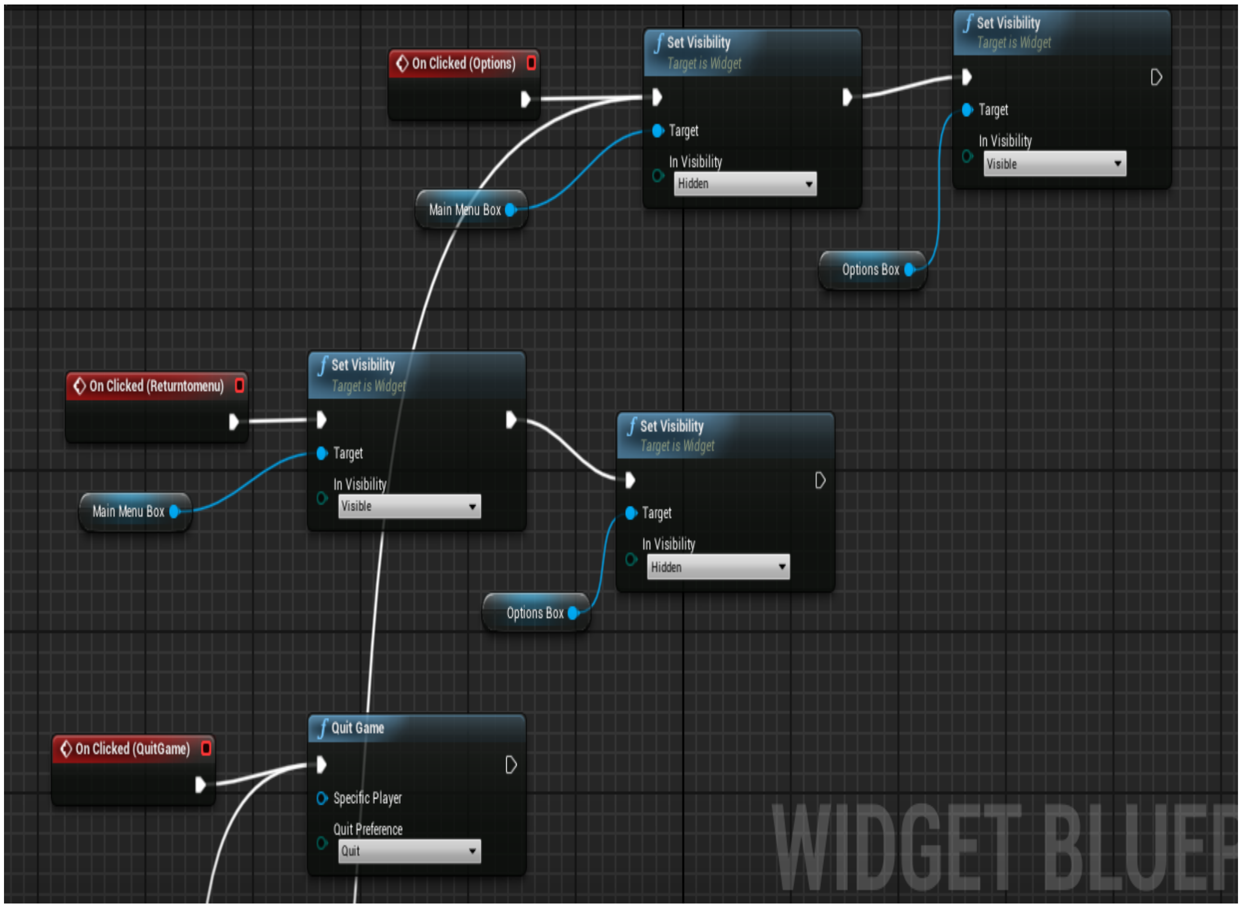
How do the character look like :



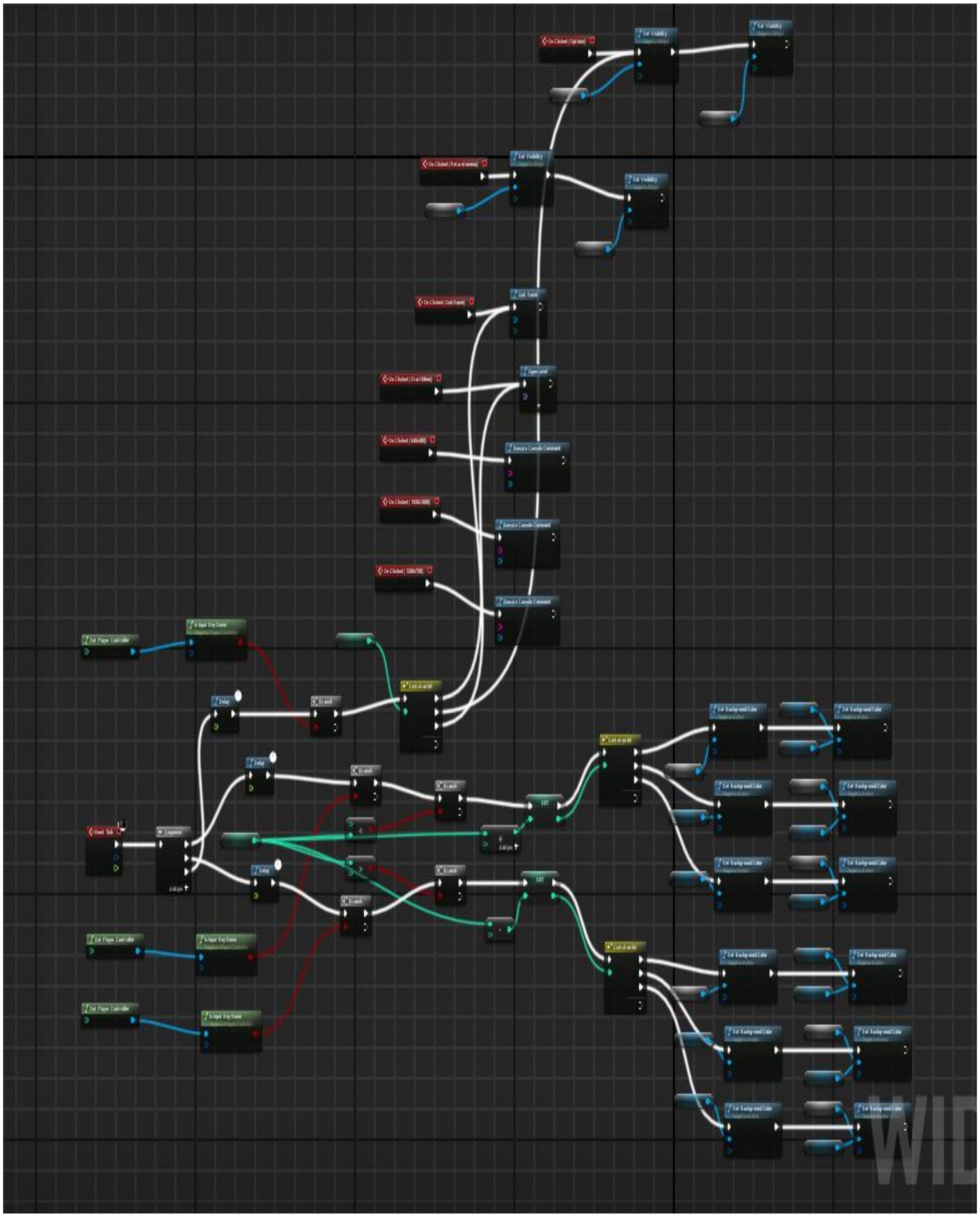
The unreal script for HUD class:



The unreal script for Widget class:



- **The Final Unreal Script:**



• Constraints and Future Enhancement

In future I am going to add visual graphics in the game. Do some animation on the characters in the game. Add some new level for the game which is going to get difficult after some levels. I want to convert this game into a virtual reality game in future. It is an very difficult task to do but I want to do that which make my game an extra ordinary game among the people.

- ## Conclusion:

In the competitive world, it is really necessary for the students to make themselves ready for the challenges one may face in the near future. At the end of the project, there were two benefits, I made a game as a project which is not complete game but for beginner it is good reference and made connections with students, technical and non-technical across India for those who are interested in Game development. Development using unreal engine is really efficient and worthy. Anyone who is interested in the game development should consider unreal engine for development.

- **Reference:**

1. **https://en.wikipedia.org/wiki/Unreal_Engine**
2. **https://wiki.unrealengine.com/First_Person_Shooter_C**
3. **https://docs.unrealengine.com/en-**