PROJECT REPORT
ON
UNREAL TOURNAMENT

SUBMITTED BY

Ashwin Ramesh Muthukumar

SEAT NO: 716

PROJECT GUIDE

Ms. Pooja Salvi

BSC. (COMPUTER SCIENCE) SEM -V

2021 – 2022

CONDUCTED AT

CHIKITSAK SAMUHA'S

S. S. & L.S. PATKAR COLLEGE OF ARTS & SCIENCE

AND

V. P. VARDE COLLEGE OF COMMERCE & ECONOMICS

GOREGAON (W). MUMBAI -400062

CHIKITSAK SAMUHA'S
S.S. & L.S. PATKAR COLLEGE OF ARTS & SCIENCE
AND
V. P. VARDE COLLEGE OF COMMERCE & ECONOMICS
Accredited A Grade by NAAC
S.V.Road, Goregaon (West), Mumbai - 400 062. Tel.: 91-022-28723731/28781188 Fax : 91-022-2874 4755
Website : www.patkarvardecollege.edu.in
E-mail : principal@patkarvardecollege.edu.in info@patkarvardecollege.edu.in
Self-Financed Courses

Project Certificate

This is to certify that Mr. /Ms. _____Ashwin Ramesh Muthukumar_____of T.Y.B.Sc. Computer Science with University Seat no _Roll No_ has completed his/~~her~~ project titled _____Unreal_Tournament_____ under the guidance of Project Guide ~~Mr~~./Ms._____**Pooja Salvi**_____ as laid by University of Mumbai in the college during the year 2020-21.

Project Guide **B.Sc. Computer science**

Co-ordinator

External Examiner

Date: _____

ACKNOWLEDGEMENT

First and foremost , I take this opportunity of submitting this report to express my profound gratitude to

the management of “S.S. & L.S. Patkar College of Science and Commerce” for giving me the opportunity to accomplish this project work.

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to our final year project Guide, Ms Pooja Salvi, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report.

Furthermore I would also like to acknowledge with much appreciation the crucial role of all the teachers who put so much efforts for us student. I have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

Thanking you,

Ashwin.R.Muthukumar

INDEX

	6.4 Graphs	
	6.5 Screen shots	
	6.6 Reports	
7	**Conclusion and Future Scope** (Specify the Final conclusion and future scope)	
8	**References** (Books, web links, research articles, etc.)	

1. Introduction

1.1 History of Gaming:

Since its commercial birth in the 1950s as a technological oddity at a science fair, gaming has blossomed into one of the most profitable entertainment industries in the world.

The mobile technology boom in recent years has revolutionized the industry and opened the doors to a new generation of gamers. Indeed, gaming has become so integrated with modern popular culture that now even grandmas know what Angry Birds is, and more than 42 percent of Americans are gamers and four out of five U.S. households have a console.

The first recognized example of a game machine was unveiled by Dr. Edward Uhler Condon at the New York World's Fair in 1940. The game, based on the ancient mathematical game of Nim, was played by about 50,000 people during the six months it was on display, with the computer reportedly winning more than 90 percent of the games.

However, the first game system designed for commercial home use did not emerge until nearly three decades later, when Ralph Baer and his team released his prototype, the "Brown Box," in 1967.

Sega and Taito were the first companies to pique the public's interest in arcade gaming when they released the electro-mechanical games Periscope and Crown Special Soccer in 1966 and 1967. In 1972, Atari (founded by Nolan Bushnell, the godfather of gaming) became the first gaming company to really set the benchmark for a large-scale gaming community.

During the late 1970s, a number of chain restaurants around the U.S. started to install video games to capitalize on the hot new craze. The nature of the games sparked competition among players, who could record their high scores with their initials and were determined to mark their space at the top of the list. At this point, multiplayer gaming was limited to players competing on the same screen.

In addition to gaming consoles becoming popular in commercial centers and chain restaurants in the U.S., the early 1970s also saw the advent of personal

computers and mass-produced gaming consoles become a reality. Technological advancements, such as Intel's invention of the world's first microprocessor, led to the creation of games such as Gunfight in 1975, the first example of a multiplayer human-to-human combat shooter.

Multiplayer gaming over networks really took off with the release of Pathway to Darkness in 1993, and the LAN Party" was born. LAN gaming grew more popular with the release of Marathon on the Macintosh in 1994 and especially after first-person multiplayer shooter Quake hit stores in 1996. By this point, the release of Windows 95 and affordable Ethernet cards brought networking to the Windows PC, further expanding the popularity of multiplayer LAN games.

The real revolution in gaming came when LAN networks, and later the Internet, opened up multiplayer gaming. Multiplayer gaming took
the gaming community to a new level because it allowed fans to compete and interact from different computers, which improved the social aspect of gaming. This key step set the stage for the large-scale interactive gaming that modern gamers currently enjoy. On April 30, 1993, CERN put the World Wide Web software in the public domain, but it would be years before the Internet was powerful enough to accommodate gaming as we know it today.

1.2 Theoretical background and History

Unreal Tournament(Game) is inspired by Unreal Engine 2005 which was developed by Epic Games and Quake 3 developed by Bullfrog Productions.The Title of my game itself is taken from the Unreal Tournament game.

Unreal Engine was originally built to create FPS games so it only makes sense to create one using it.

The Game is both FPS (first person shooter) and TPS(Third person shooter) game developed using unreal engine with many features and one can complete the game in short amount of time.

Unreal is open-source which is a great advantage because it makes development easier and more efficient. Rendering technology is one of the biggest benefits of this game engine. Post processing is really fast and there is support for many features. Concerning virtual reality, both engines have good VR integration.

Unlike most FPS games which requires reflex and experience to complete it, Unreal Tournament is very easy to play.

The game is developed specifically for Computer consoles and requires a decent specification of computer.

Controls are easy to use , The universal FPS gaming keys eg W,S,A,D and other keys.

1.3 The main objectives of the project:

Objective of the game:

- There are 3 Levels and player has to kill the enemies and enter the portal for travelling between levels.
- Player has default guns with limited ammo and limited health .
- The ammo for default gun and health can be found exploring the map.The player has to use intellect and some reflexes to kill enemy without taking damage and explore the map to find hidden portal.
- The main objective is to find portals and travel through them.

Objective of the project:

- To Create a FPS/TPS game with simple controls and requires only short amount of time to complete

- To entertain players with satisfactory and simple shooter gameplay .

The scope of the project:

This Report describes all the requirements for the project. The purpose of this research is to

provide a virtual image for the combination of both structured and unstructured

information of our project “Unreal Engine” in the gaming environment. The player will progress

through various challenges in the gaming environment. The number of hours of gameplay,

graphics quality, quality of the AI, and other aspects that would be

considered to analyze how interesting the game is.

2. Requirement Specification

2.1 Functional Requirements:

The functional requirements specify what the product must do. They relate to the actions that the

product must carry out in order to satisfy the fundamental reasons for its existence. The functional requirements must fully describe the actions that the intended product can perform in this game.

There are 3 levels in the game and player has to reach the portal to complete one level.

The game takes place in a Sci-Fi Themed world , where player has to kill enemies and reach the portal in each level .

The enemies will try attacking player but the player has to kill them and explore the areas for recovering health and ammo by collecting spawnables items.

2.2 Non-Functional Requirement:

The non-functional requirement add functionally to the product‟s Existence, but are needed to make

the product perform in the desired manner. In Space Escape, these requirements are met with utmost

care. The design is made so that even the naivest of users can understand the navigation and walk

through of the game. This program requires above average specification of hardware but most users

can run it on their system with included options.

Platform constraints: This game run on PC. It require 4GB RAM and minimum 8GB space to run smoothly.

Usability: It is run to play the game It has very simple controls. This game has exceptional graphics

And requires a decent computer with GPU.

Performance: The Game runs smooth in minimum 4GB RAM & with a decent GPU(Recommended). Simple controls with no game input lag.

Maintainability: The Game is C++ based , codes and classes are correctly arranged so it is very easy to maintain.

Availability: The project can be deployed on a public shared service or it will be available all the time and will be accessible anywhere of the world.

Software Requirements:

Name of Component	Specifications
Operating System	Windows 7 or Higher Version
Programming language	C++
Software Application	Unreal Engine (For Developing)
Other support applications and softwares	DirectX(SDK),Visual Studio Code(for C++)

Hardware Requirements:

Name of Component	Specifications
Processor	Minimum 2.5 GHz; Recommended 3.5GHz or higher.
Ram	Minimum 4 GB; Recommended 8 GB or above
Hard disk	Minimum 8 GB; Recommended 10 GB or above
Monitor	-

3. System Planning:

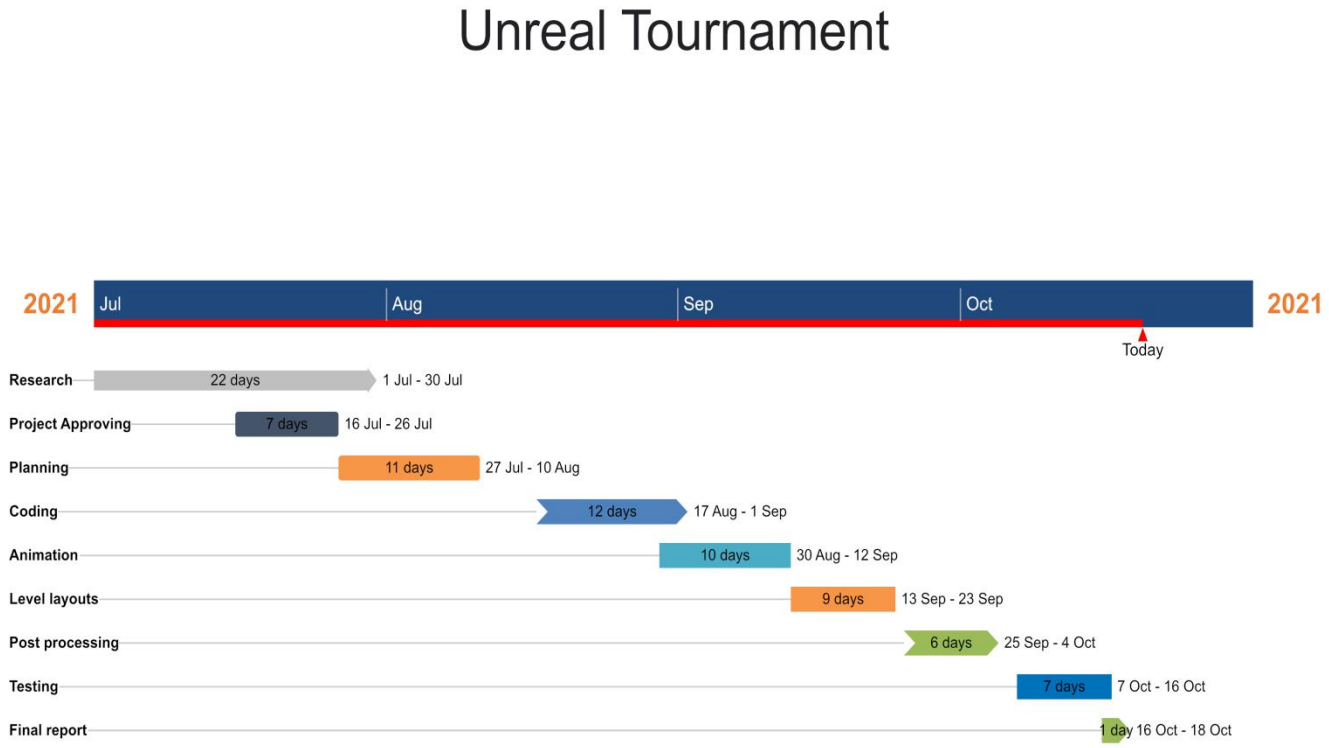
3.1 GANTT CHART:

Gantt chart, commonly used in project management, is one of the most popular and useful ways of

showing activities (tasks or events) displayed against time. On the left of the chart is a list of the

activities and along the top is a suitable time scale. Each activity is represented by a bar, the position

and length of reflects the start date, duration and end date of the activity



4. System Designing Details:

4.1 PROCESS MODEL:

Spiral model:

Is a combination of sequential and prototype model. This model is best used for large projects which involves continuous enhancements. There are specific activities which are done in one iteration (spiral) where the output is a small prototype of the large software. The same activities are then repeated for all the spirals till the entire software is build.

Functions of Spiral Model:

Requirements: Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

Planning: All the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.

Modeling: In this Section we will Analysis and requirements, according to needs.
We will create a design of our projects and analysis of project. Internal testing and Deployments are done in Modeling.

Construction: The team delivered high-quality working software in priority order, which was created in accordance with the changing needs of our potential users. What‟s more important, the team could deploy this solution into a pre-production testing/QA sandbox for system integration testing.

Deployment: Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

4.2 METHODOLOGY ADOPTED:

OBJECT ORIENTED PROGRAMMING:

Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define not only the datatype of a data structure, but also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions.

In addition, programmers can create relationships between one object and another .For example, objects can inherit characteristics from other objects.

THE BASICS CONCEPTS:

If you are new to object-oriented programming languages, you will need to know a few basics before you can get started with code.:

ABSTRACTION: The process of picking out (abstracting) common features of objects and procedures.

CLASS: A category of objects. The class defines all the common properties of the different objects that belong to it.

ENCAPSULATION: The process of combining elements to create a new entity. A procedure is a type of encapsulation because it combines a series of computer instructions. INFORMATION HIDING: The process of hiding details of an object or function. Information hiding is a powerful programming technique because it reduces complexity.

INHERITANCE: a feature that represents the "is a" relationship between different classes.

INTERFACE: the languages and codes that the applications use to communicate with each other and with the hardware.

MESSAGE: Message passing is a form of communication used in parallel programming and object oriented programming.

OBJECT: a self-contained entity that consists of both data and procedures to manipulate the data.

POLYMORPHISM: A programming language's ability to process objects differently depending on their data type or class.

PROCEDURE: a section of a program that performs a specific task

OBJECT ORIENTED PROGRAMMING LANGUAGES:

An object-oriented programming language (OOPL) is a high-level programming language based on the object-oriented model. To perform object-oriented programming, one needs an object-oriented programming language. Many modern programming languages are object- oriented, however some older programming languages, such as Pascal, do offer object-oriented versions. Examples of object oriented programming languages include Java, C++, etc.

C++ and it's key feature:

C++ Introduces Object-Oriented Programming

Although it was lacking in C, object-oriented programming was introduced in C++. Among other things, C++ supports the four primary features of OOP: abstraction, inheritance, polymorphism and encapsulation. With that said, C++ is unique in the sense that it supports deterministic destructors for classes — a feature that's not found in other OOP languages.

C++ has More than 35 Operators

C++ currently has more than 35 different operators, ranging from arithmetic and bit manipulation to logical operations, comparisons and more. Virtually all of these operators can be overloaded for specific types, although there are a few exceptions, one of which is the conditional operator. This vast array of operators makes C++ user definitions more like built-in types.

C++ has Two Main Concepts

C++ has two primary concepts on which the language was built: direct mapping for hardware features and zero-overhead abstractions for mapping. Perhaps this is why the language is often touted as a lightweight abstraction programming language used for creating efficient abstractions while also offering hardware access.

C++ Supports Four Types of Memory Management

C++ supports four different types of memory management: static storage duration objects, thread storage duration objects, automatic storage duration objects, and dynamic storage duration objects.

MAIN FEATURES OF C++

1. Object Oriented Programming

Unlike C, a procedural language, C++ is an object-oriented programming (OOP) language. OOP helps modularize and maintain a program efficiently. It improves code clarity, code readability, troubleshooting and makes it easier to incorporate modifications without any significant restructuring.

2. MODERN

1.

C++ has been based according to the current trend and is very powerful and simple for

building interoperable, scalable, robust applications.

2.

C++ includes built in support to turn any component into a web service that can be invoked

over the internet from any application running on any platform.

3. OBJECT ORIENTED

1.

C++ supports Data Encapsulation, inheritance, polymorphism, interfaces.

2.

(int,float, double) are not objects in java but C++ has introduces structures(structs) which

enable the primitive types to become objects.

4. TYPE SAFE

1. In C++ we cannot perform unsafe casts like convert double to a Boolean.

2. Value types (primitive types) are initialized to zeros and reference types (objects and classes) are initialized to null by the compiler automatically.

3. Arrays are zero base indexed and are bound checked.

4. Overflow of types can be checked.

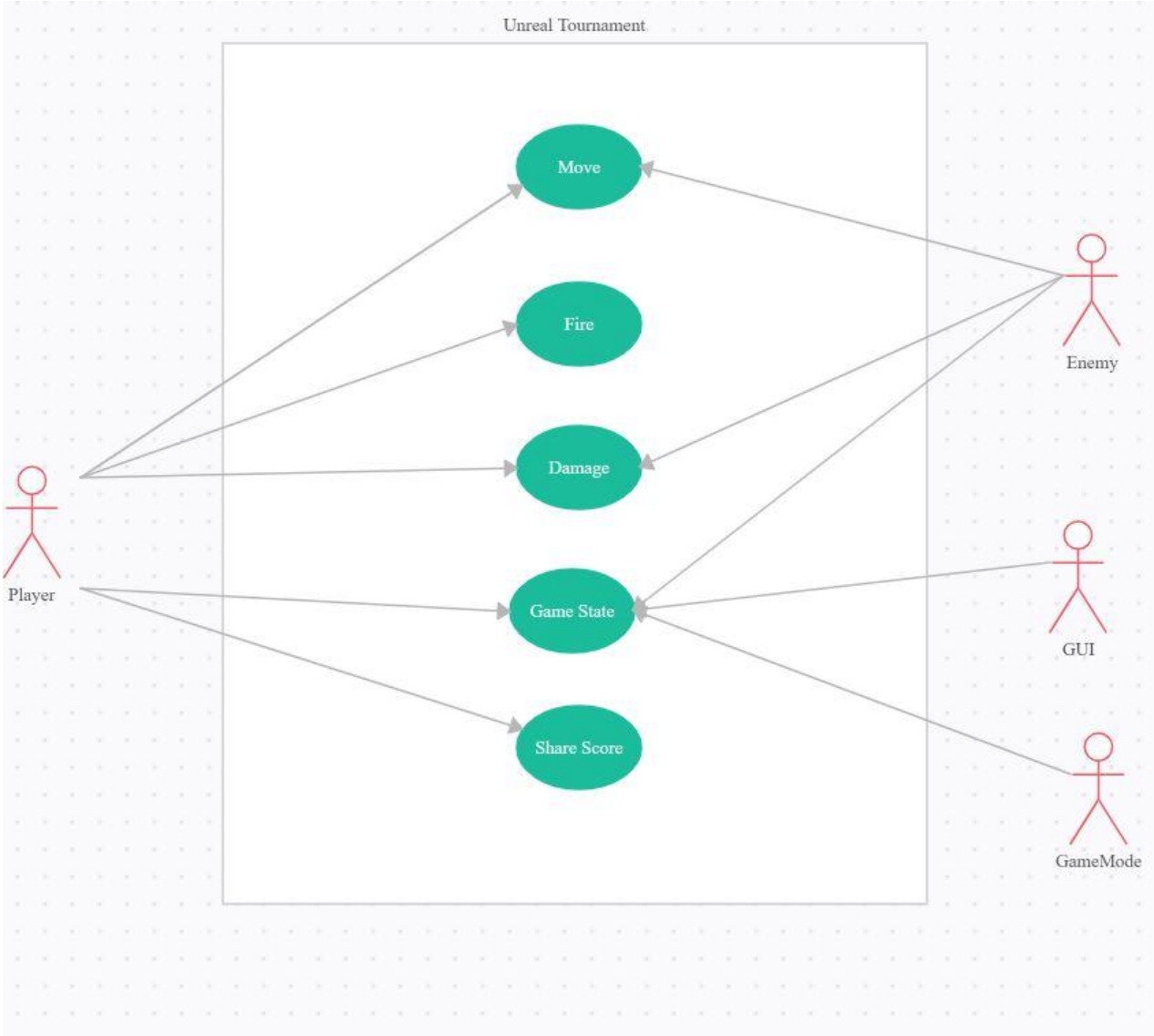
4.3 UML DIAGRAMS:

4.3.1 Use-Case Diagram:

Use case diagrams are the diagrammatic representation depicting user‟s interaction with the system.

a) Use-Case diagram represents graphical model that summarizes information about actor and use case.

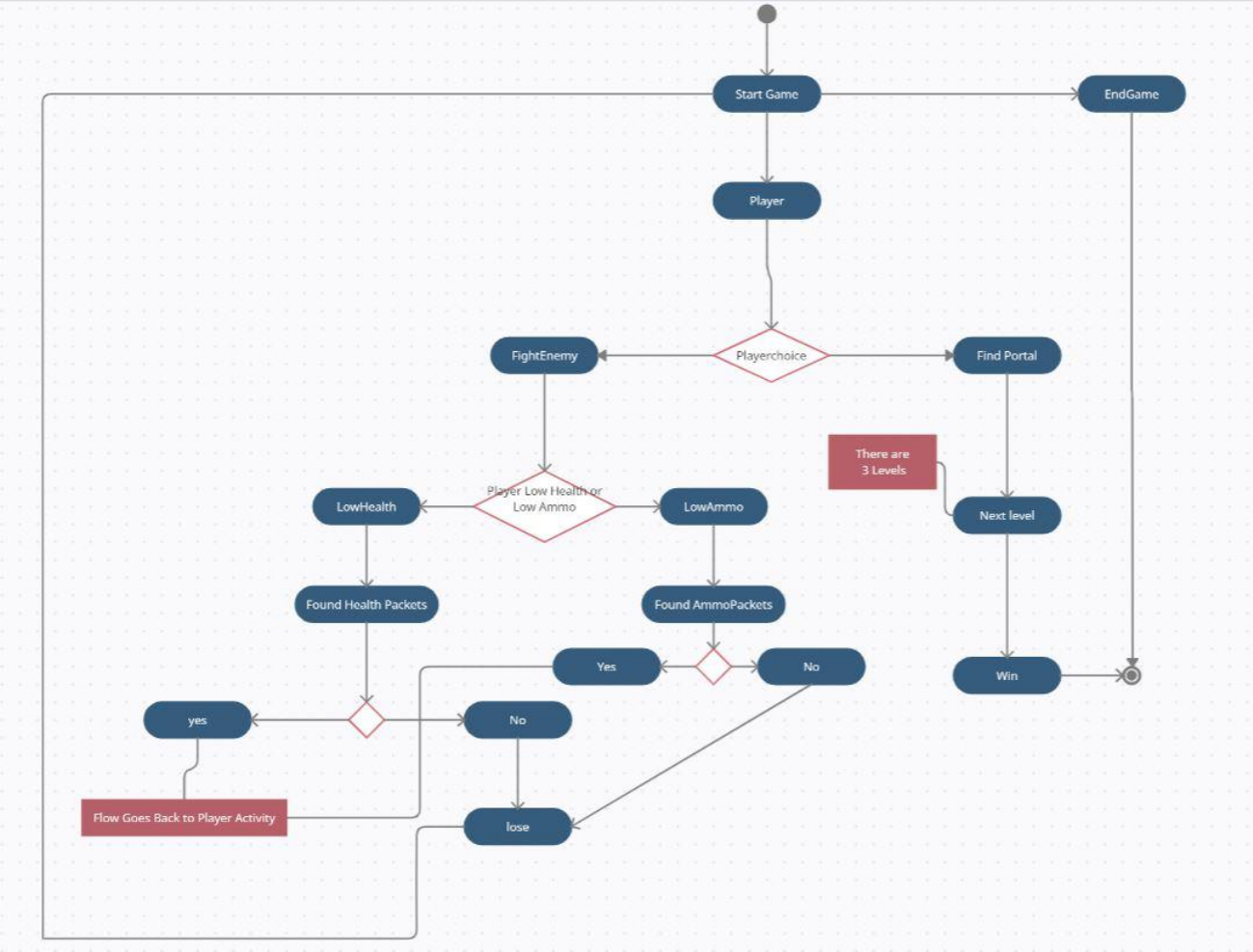
b) Boundary denotes the boundary between the environment, where the actors reside and the internal functions of the automated system.



4.3.2 Activity Diagram:

An Activity Diagram visually presents a series of actions of control is system similar to a flowchart or a dataflow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram. Activity modelled can be sequential and concurrent.

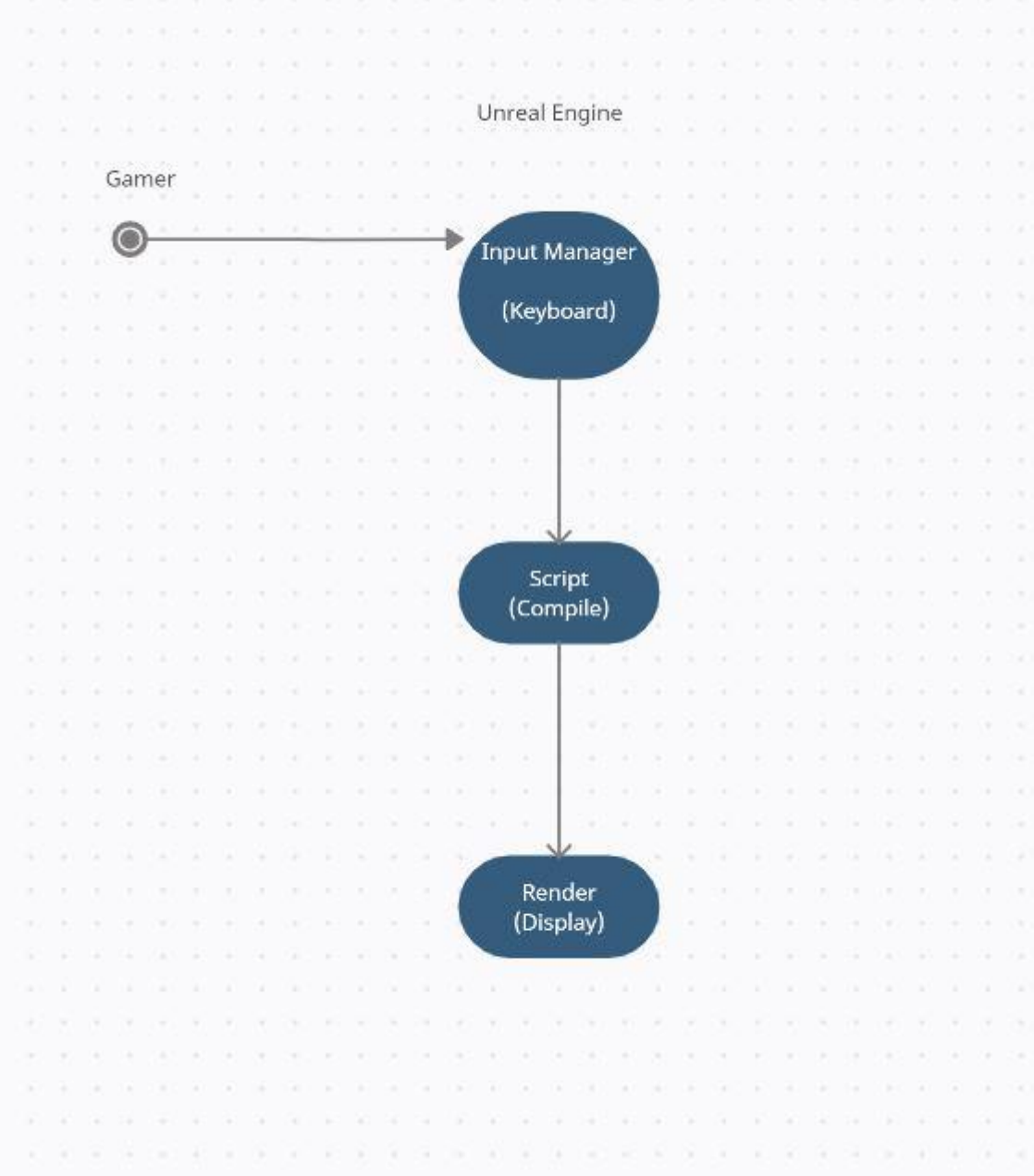
ACTIVITY DIAGRAM:



4. 3.5 System environment diagram

In engineering System environment diagram is a **diagram** that defines the boundary between the **system**, or part of a **system**, and its **environment**, showing the entities that interact with it. This **diagram** is a high level view of a **system**. It is similar to a block **diagram**.

SYSTEM ENVIRONMENT DIAGRAM:

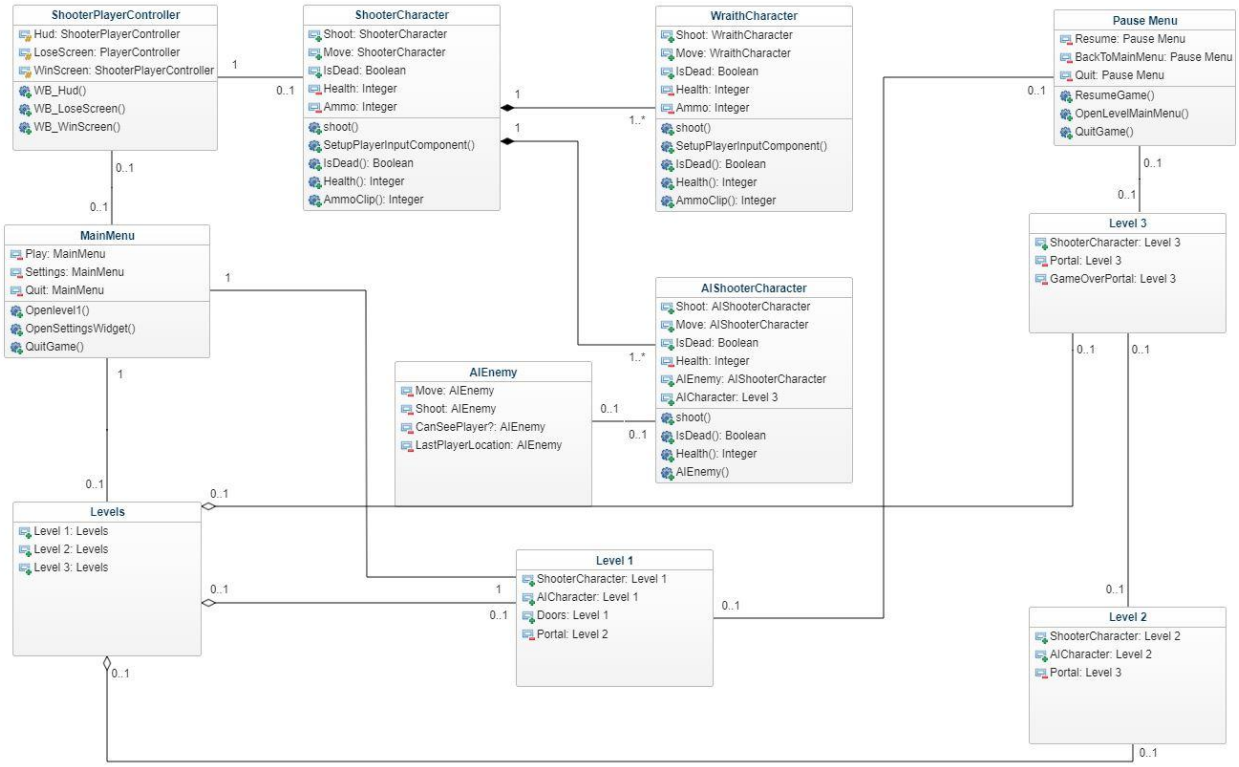


4. 3.4 Class Diagram:

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class.

In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

CLASS DIAGRAM:



5. System Implementation:

Sample Code:

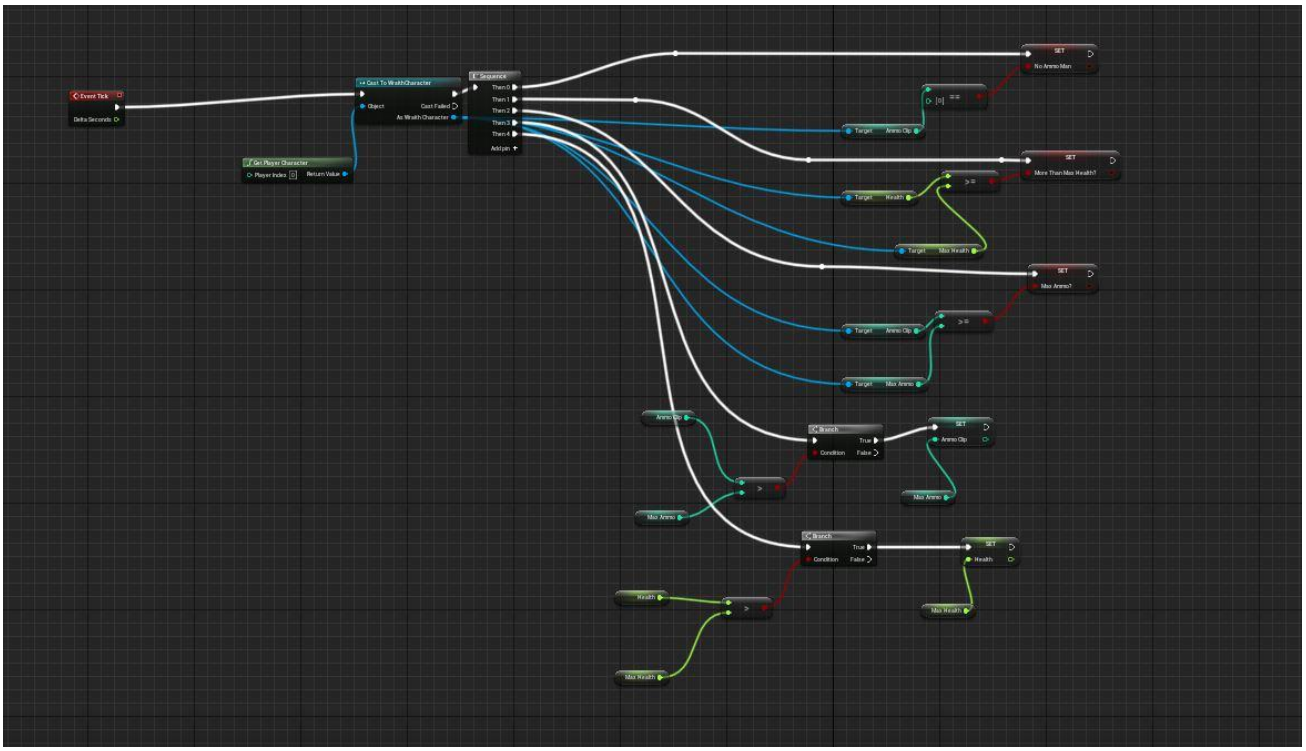
```
// Fill out your copyright notice in the Description  of Project Settings.


#include "ShooterCharacter.h"

#include "Gun.h"

#include "Components/CapsuleComponent.h"

#include "TournamentGameModeBase.h"

#include "Blueprint/UserWidget.h"

#include "ShooterAIController.h"

#include "KillEmAllGameMode.h"

#include "EngineUtils.h"


#include "GameFramework/Controller.h"


// Sets default values

AShooterCharacter::AShooterCharacter()

{

   // 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 AShooterCharacter::BeginPlay()

{

   Super::BeginPlay();

   Health = MaxHealth;
```

```
   AmmoClip = MaxAmmo;
   Gun = GetWorld()->SpawnActor<AGun>(GunClass);
   GetMesh()->UnHideBoneByName(FName("weapon_r"));
   Gun->AttachToComponent(GetMesh(), FAttachmentTransformRules::KeepRelativeTransform,
TEXT("WeaponSocket"));
   Gun->SetOwner(this);

}
bool AShooterCharacter::IsDead() const
{
   return Health <= 0;
}
float AShooterCharacter::GetHealthPercent() const
{
   return Health / MaxHealth;
}
float AShooterCharacter::GetAmmoPercent() const
{
   return AmmoClip;
}
// Called every frame
void AShooterCharacter::Tick(float DeltaTime)
{
   Super::Tick(DeltaTime);
}
// Called to bind functionality to input
void AShooterCharacter::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
{
   Super::SetupPlayerInputComponent(PlayerInputComponent);
    PlayerInputComponent -> BindAxis(TEXT("MoveForward"),this,&AShooterCharacter::MoveForward);
```

```
    PlayerInputComponent -> BindAxis(TEXT("LookUp"),this,&AShooterCharacter::LookUp);

    PlayerInputComponent -> BindAxis(TEXT("MoveRight"),this,&AShooterCharacter::MoveRight);

    PlayerInputComponent -> BindAxis(TEXT("LookRight"),this,&AShooterCharacter::LookRight);

    PlayerInputComponent -> BindAction(TEXT("Jump"),EInputEvent::IE_Pressed , this , &ACharacter::Jump);

}


float AShooterCharacter::TakeDamage(float DamageAmount, struct FDamageEvent const& DamageEvent,
class AController* EventInstigator, AActor* DamageCauser)

{

   float DamageToApply = Super::TakeDamage(DamageAmount,DamageEvent ,
EventInstigator ,DamageCauser);

   DamageToApply = FMath::Min(Health,DamageToApply);

   Health -= DamageToApply;



   if (Health==0)

   {

      SetLifeSpan(3);

   }

   int32 NumPawns = GetWorld()->GetNumPawns();

   if(GEngine)

   {

      GEngine->AddOnScreenDebugMessage(-1,15.f , FColor::Green , FString::Printf(TEXT("Number of Pawns
Left: %d"),NumPawns));

   }

   UE_LOG(LogTemp, Warning , TEXT("Health left %f"),Health);

   UE_LOG(LogTemp, Warning , TEXT("Pawns left %d"),NumPawns);

   if (IsDead())

   {

      ATournamentGameModeBase *GameMode = GetWorld()-
>GetAuthGameMode<ATournamentGameModeBase>();

      if (GameMode != nullptr)
```

```
        {
            GameMode->PawnKilled(this);
        }
        DetachFromControllerPendingDestroy();
        GetCapsuleComponent()->SetCollisionEnabled(ECollisionEnabled::NoCollision);
    }
    return DamageToApply;
}
void AShooterCharacter::MoveForward(float AxisValue)
{
    AddMovementInput(GetActorForwardVector() * AxisValue);
}
void AShooterCharacter::LookUp(float AxisValue)
{
    AddControllerPitchInput(AxisValue);
}
void AShooterCharacter::MoveRight(float AxisValue)
{
    AddMovementInput(GetActorRightVector() * AxisValue);
}
void AShooterCharacter::LookRight(float AxisValue)
{
    AddControllerYawInput(AxisValue);
}
void AShooterCharacter::Shoot()
{
    AmmoClip-=1;
    Gun ->PullTrigger();
}
```

Sample Unreal Blueprint nodes:



6.1 TEST CASES:

Test case:

Sr no.	Test name	Expected Result	Status	Output	Test Result
1	Gameplay Test	All player input should work properly.	Performed	Working Fine.	Success
2	Sound Test	If all Audio are working properly.	Performed	It Works.	Success
3	Animation Test	Check if character animations and level animations are working properly.	Performed	Working fine.	Success
4	Gamemode Test	Check if game logics are working properly.	Performed	Working Perfectly.	Success
5	UI test	Check if UI widgets are working properly.	Performed	Good.	Success

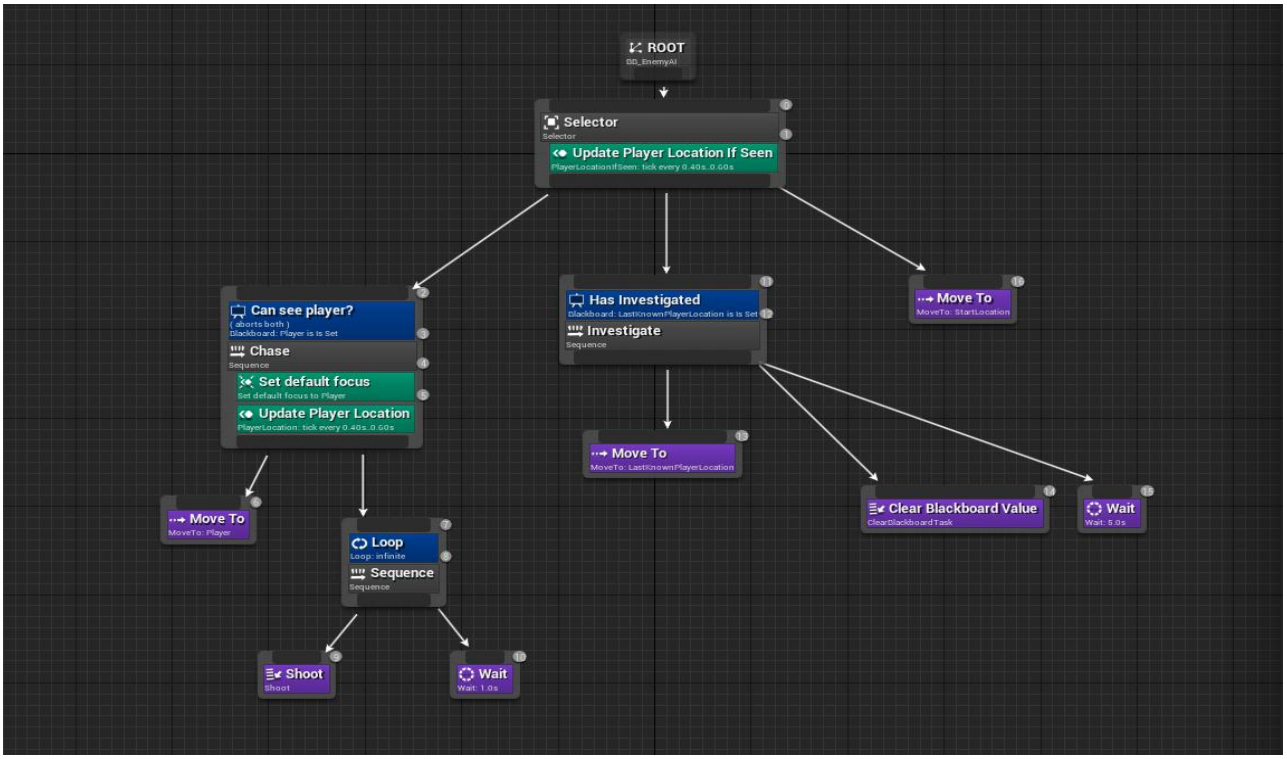
6.2 SCREENSHOTS:

Main Menu:

Settings Menu:

Pause Menu:

EnemyAI:



Player Control features :

There are 2 weapons in the Game , Rifle and Launcher

Rifle has Long Range Barrel But low Damage Rate.

Launcher has high damage but less range.

Rifle:

Launcher:

Player can toggle flashlight:

Player can zoom in TPP as well as FPP:

Player can toggle between fpp and tpp:

The game begins with player spawning in Level 1.

Player must find the portal and travel to next level

There are total 3 levels in the game

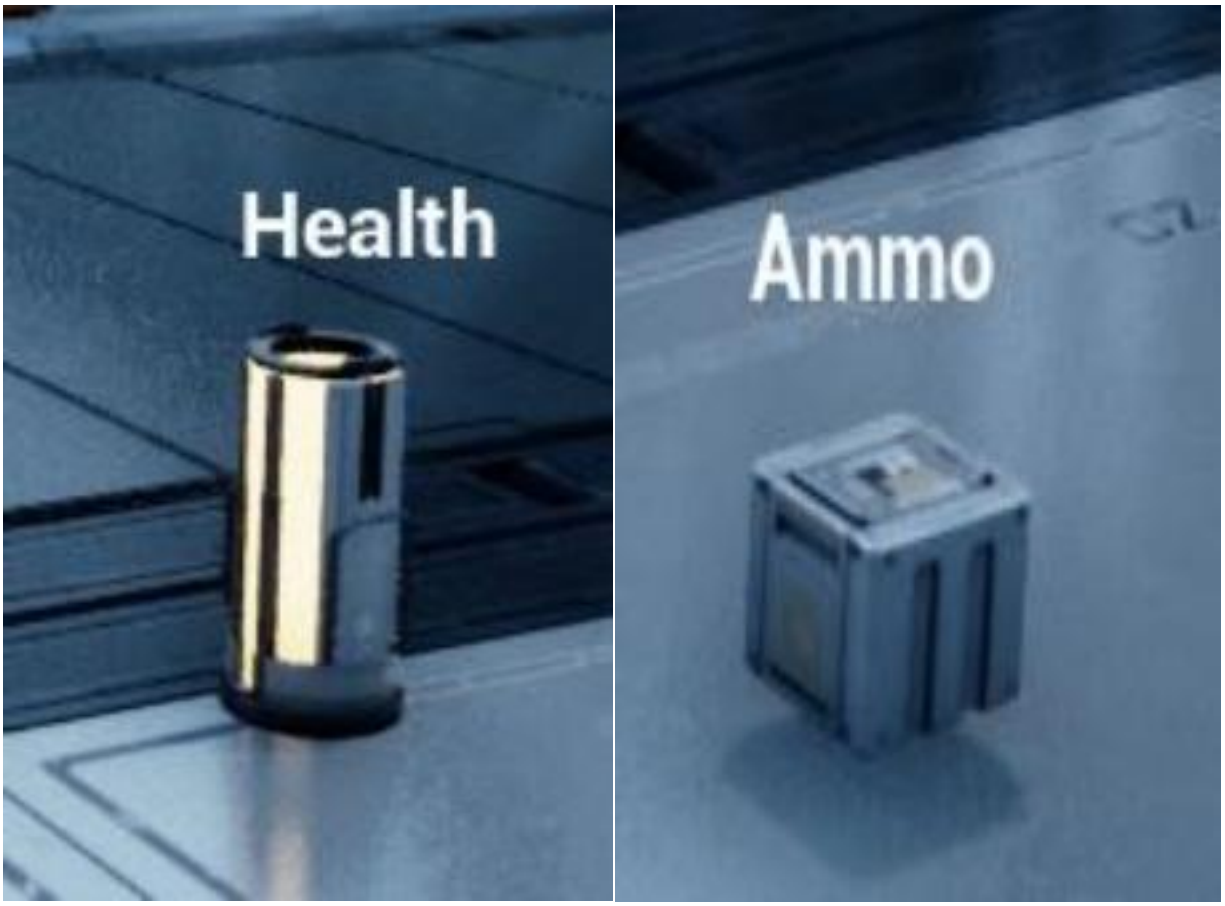
Level 1 :

Level 2 :

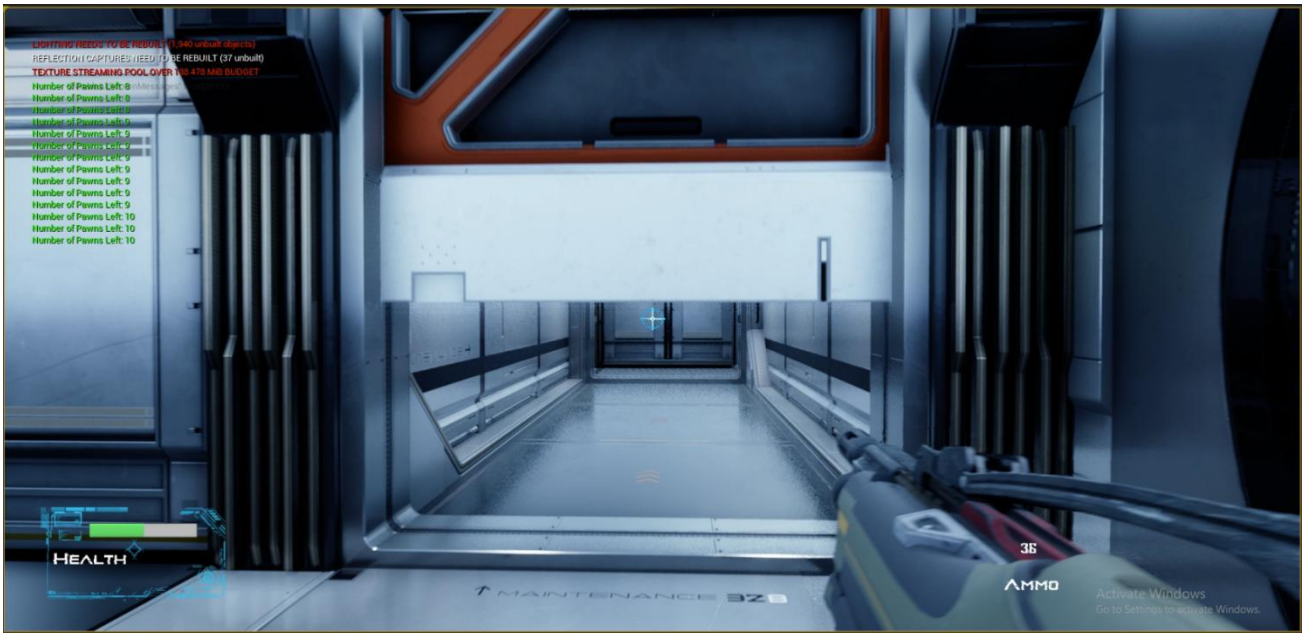
Level 3 :

The player spawns at level 1:

When the player has low ammo or low health …player can find the spawnables items while exploring the level.



Player has to clear all enemies in ground floor and go upper floor.
A door will open when player clears ground floor

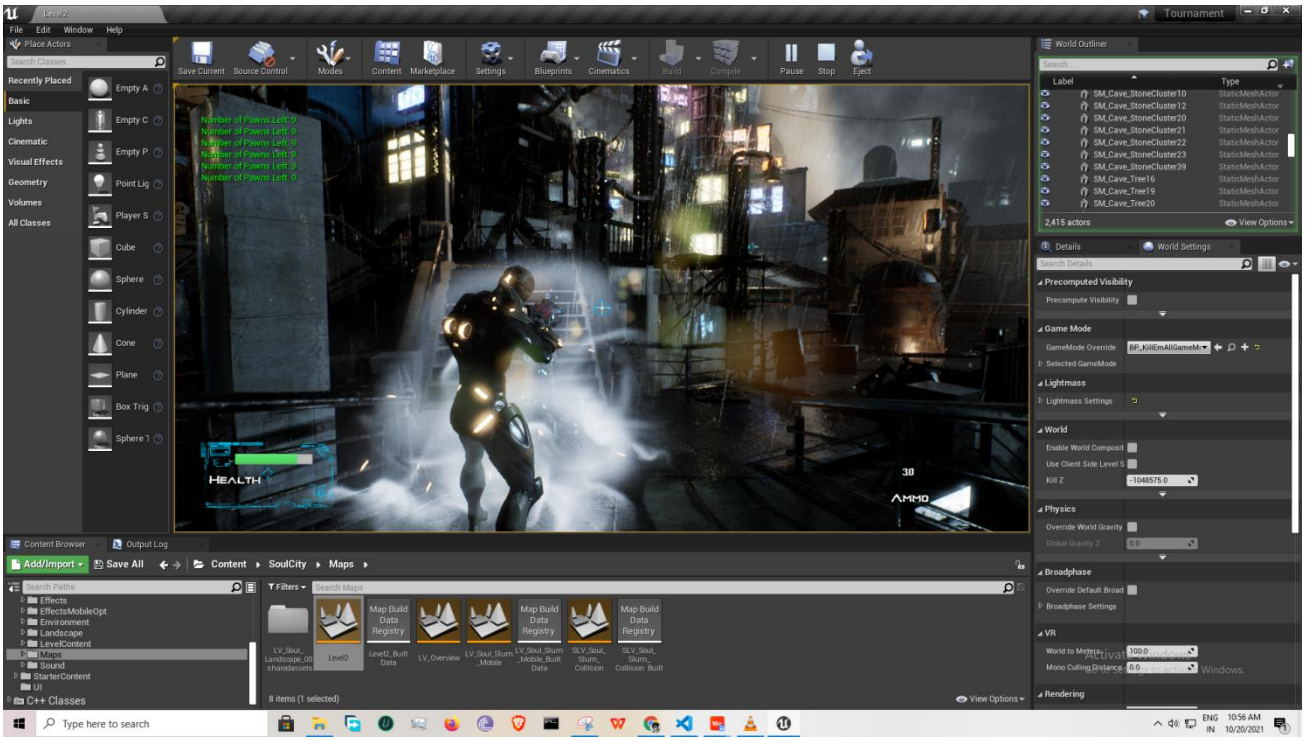


When the player kills all the enemies in the level , a secret hidden door will open .

Player has to find it and enter the portal.

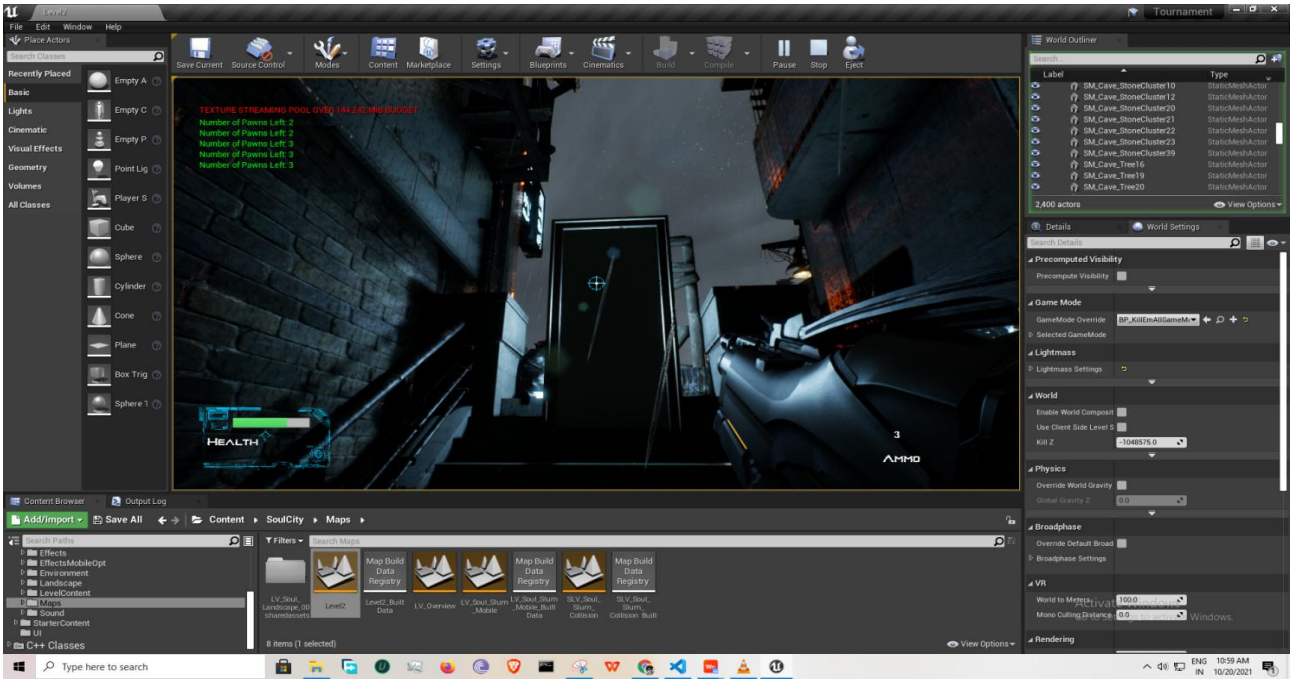
In the below screenshot we can see the portal.

When player enters the portal , he reaches Level 2



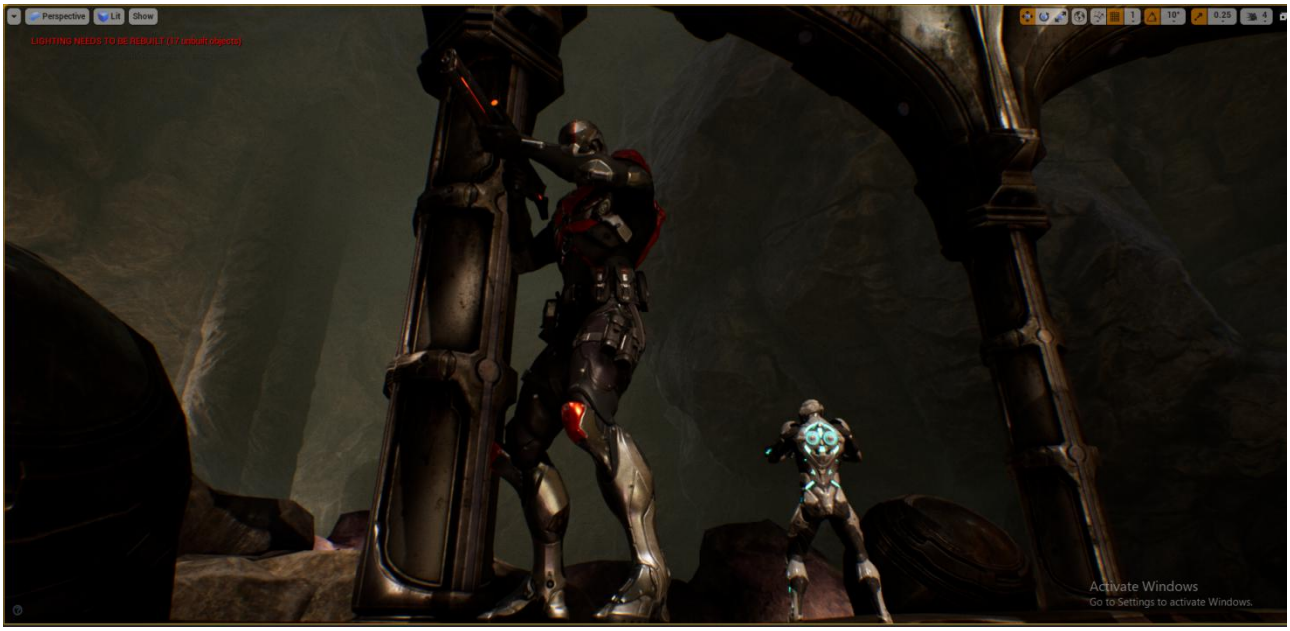
Level 2 is a bit Tougher than Level 1.

Again player has to kill the enemies and travel to level 3 which is the final level where the boss awaits.



Now in level 3 player has to kill the boss who is a lot Tougher than other enemies and deals a lot of damage.

In order to survive the fight, the player has to constantly search for ammoclip and health kit spawnables.



After killing the boss

Player has to find the portal and the game ends.

If Player dies due to low health , Losescreen is displayed and current level restarts.

6.3 SPRITES & 3D models USED

Player character mesh asset are unreal engine store asset.

And Level Assets are from Unreal Market Place.

User Manual

7.1 CONTROLS AND MANUAL FOR NDF ON PC

Check out the gameplay controls and manual for SpaceEscape on PC.

Basic controls for playing NDF, external accessories required: Keyboard

Key bindings:

Action	Key
Move Forward	W
Move Backward	S
Move Left	A
Move Right	D
Jump	Spacebar
Shoot	Left Mouse Button
Aim (zoomed)	Right Mouse Button(Hold)
Toggle Perspective	V
Toggle Flashlight	F
Switch Weapons	Mouse Scroll Wheel Down

7.2 Conclusion:

This Project taught me a lot about game programming and developing

I’ve learned how to program C++ and use Unreal engine

Faced many challenges and learned from it

The project has enlightened in following ways :

- Increased my confidence in developing a project
- Sharpened my game designed skills and game logic skills.
- Improved my technical skills.

7.3 Future Enhancement:

This Project has ‘n’ Number of possibilities and updates that can be added

And user feedback will really help me decide what to update

Some of my future enhancement plan for the projects are as follows :

- Add Advanced Enemy AI
- Set difficulty levels
- Add Campaign mode
- Add online multiplayer mode
- Add new weapons and maps to the game
- Add more different player Characters with different anmimations

References:

http://unrealengine.com

https://www.gamecareerguide.com/

https://en.wikipedia.org/wiki/UnrealTournament

http://www.stackoverflow.com/

https://gamedevacademy.org/

https://youtube.com