Multiplayer FPS: 3D Game Development using Unreal Engine

Abhishek Jain
School of Engineering
Jawaharlal Nehru University
New Delhi, India
abhish34_soe@jnu.ac.in

Kartik Ohri
School of Engineering
Jawaharlal Nehru University
New Delhi, India
kartik80_soe@jnu.ac.in

Shubham Singh
School of Engineering
Jawaharlal Nehru University
New Delhi, India
shubha65_soe@jnu.ac.in

Sidharth Kumar
School of Engineering
Jawaharlal Nehru University
New Delhi, India
sidhar39_soe@jnu.ac.in

Tanuj Raghav
School of Engineering
Jawaharlal Nehru University
New Delhi, India
tanuj81_soe@jnu.ac.in

Vishal Chaudhary
School of Engineering
Jawaharlal Nehru University
New Delhi, India
vishal90_soe@jnu.ac.in

***Abstract*—Many games based on first person [1] graphical perspective [2] have been made after the advent of 3D accelerated graphics [3]. Here, in this paper we discuss how to build a 3D Multiplayer First-Person Shooting (FPS) game. Built with the powerful Unreal [4] game engine, the game code is written using Unreal's own gameplay scripting system called Blueprints [5]. Different game modes like Conquest, Rush, Deathmatch have been implemented. The final game comes out in two packages, one for server installation, and the other as a binary for Linux based operating systems.**

I. Introduction

3D Multiplayer FPS games like Player Unknown's Battlegrounds (PUBG), Call of Duty and Halo are hugely popular these days. At the same time, making FPS games (as learning projects) is popular among beginners in the game making field (this can be observed from the fact that the Unreal Marketplace is mostly filled with assets related to FPS games[6]). Owing to this and our own familiarity with such games, we decided to make a 3D FPS game. The game is made using the powerful Unreal gaming engine and it's Blueprint Visual Scripting system. The game features various game modes (like Deathmatch, Capture the Flag, Conquest, etc.), AI players (bots), different weapons (like SMG, shotgun, sniper rifles, grenades, etc.) with respective fire modes and audio-visual effects. The game building process can be divided into two main parts - 1. Designing blueprints for gameplay elements, world rendering and game modes (using Level Blueprints) and 2. Interconnecting each component according to a predefined game logic.

II. Discussion

A lot of planning is required before designing and then building a game. For making this FPS game, we had to plan and research about many things. First was how to design the complicated game elements like *guns and knives*. We decided to import the blueprints for these elements from various free projects available in the Unreal Engine marketplace. Most game sounds were obtained from the website, The Sound Resource[7]. Player and bot characters were taken from the default *Character component* of the engine. However new movement and actions had to be separately designed for these characters since these were not available by default.

A very difficult problem was to create a blueprint for the projectile movement of the grenade. At first we decided to handle the movement using C++ code by adding a *projectileMovementComponent* class to the Grenade blueprint. However owing to implementation errors, we then moved on to create the projectile movement math using blueprints only.

Level blueprints were used to created different game modes. Game events were created for each of these depending on the nature of the game mode. Each level blueprint has its own world map class related to it.

Most of the time in this project went in designing and creating the blueprints for the game elements. Another major challenge was implementing multiplayer functionality. Since the project was to be submitted within a deadline, we decided to first make the single player game mode and then implement the multiplayer functionality later. This involved many different tasks including setting up actor connection ownership, designing function replication across the network and configuring actor relevancy and prioritization.

III. Methodology

A. Designing Blueprints

Each game element including the game modes, weapons, game menu, control interface, textures and animations are individually created blueprints. Blueprints of weapons have been imported from the EPIC games marketplace and extended further depending on the need. We explain the desigining of some essential blueprints below.

1) Player Character Blueprint: We made a `BaseCharacterPC` parent blueprint class from an inbuilt Character component (which comes with the

CharacterMovement component but had to be extended with additional functionality for crouching, melee attack and vertical jump). This component automatically handles movement such as walking and jumping. We call the appropriate function and it will move the Character. Movement keys were mapped to cursor keys and different axis mappings - *ForwardMove, RightMove, VerticalMove* were created to handle movement in different directions (forward/backward, left/right, jump respectively). Scale values of the axis are multiplied with the Character's Axis vector to create a new movement vector which is added to the current position vector of the character resulting in its movement.

2) Weapons (Gun type) blueprint: A base Blueprint Class of type Actor, `BP_BaseGun`, was created. Some float variables that define different properties of the gun are:

- `MxBltDistance`: Maximum distance travelled by a bullet
- `Damage`: Damage caused by a single bullet
- `FireRate`: Bullet shots per second

A static mesh component on it, called GunMesh was also added. Guns are child blueprint classes created using `BaseGunPC` as the base class. Some of them are listed below:

1) `BP_Rifle`
2) `BP_AssaultRifle`
3) `BP_SMG`
4) `BP_SniperRifle`
5) `BP_MachineGun`
6) `BP_Shotgun`

Each of these gun child classes had corresponding Static Mesh blueprint classes set up for them.

3) Weapons (Non-gun type) blueprint: Blueprints classes of type Actor were created to implement other weapons -

1) BP_Knive - Blueprint class for Knives.
2) BP_Grenade - Blueprint class for grenade. Projectile movement and related math of the grenade was implemented using Blueprints. We followed suggestions given in a community forum post[8] to create this specific blueprint.

4) Interconnecting Blueprints: All the blueprints were finally connected to create the whole game. This was first done without adding the multiplayer function. Figures 1-3 show the interconnected blueprints.

Fig. 1. Character Blueprint and its related actions

B. Implementing Multiplayer Functionality

Multiplayer functionality was implemented after completing all other parts in the game. Replications (RPC functions) were added to each Actor and Event type object and variables. Network roles were also setup to filter function calls required to trigger various in game actions and events. To enhance game

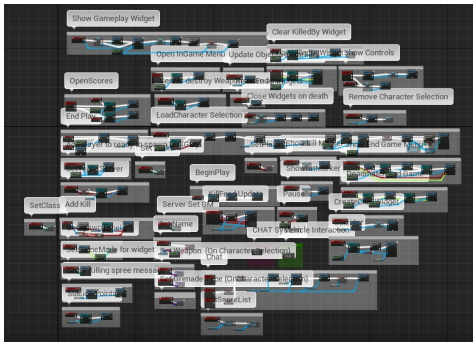

Fig. 2. Interconnected blueprints showing the main game logic

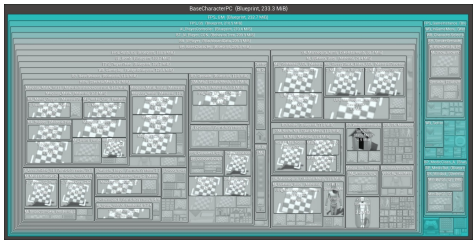

Fig. 3. BaseCharacterPC Blueprint

performance, `RepNotify` was used to synchronize all essential variable updates without relying on RPCs.

Figures 4-7 show some blueprints related to the multiplayer configurations.

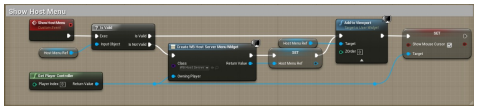

Fig. 4. Show Host Menu Event Object Blueprint

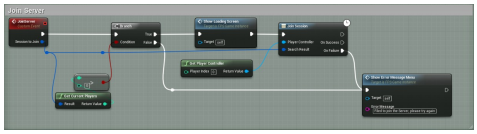

Fig. 5. Join Server Event Object Blueprint

IV. Results

The final release of the game has three packages - one for Linux and one which can be directly installed on a server and accessed from a browser. File size of the source code and final packages are given in Table X.

V. Conclusion

The final game has been packaged and released for different platforms - server installation and Linux based operating systems. The Linux binary can be directly launched without requiring any previous setup. On the other hand, the HTML package, made to be installed on web servers requires an advanced web server administrator to set up. The game is not

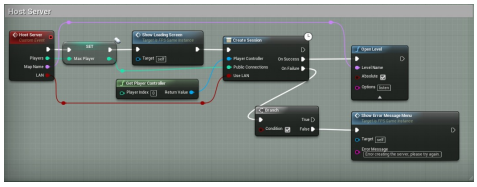

Fig. 6. Host Server Event Object Blueprint

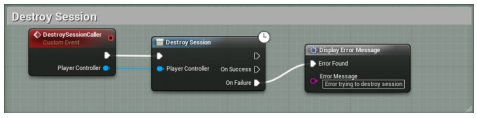

Fig. 7. Destroy Session Event Object Blueprint

demanding in terms of hardware capacity and easy to setup and play.

VI. Further Work

The final game release has certain bugs and glitches like character freezing in internet multiplayer mode, unregulated jump boost (causing player death due to falling from great heights), damp lightning issues in the web server version and game window freeze on changing audio quality to *Ultra* in settings menu. These errors need to be fixed.

Multiplayer playing experience can be further improved by creating custom RPC functions (with the target of reducing communication latency and bandwidth required for communication) overhauling the replicated multicast, run-on-server and client functions. Also, reliability of most event replication functions are currently set to ”reliable”. Changing some of these to ”unreliable” can reduce bandwidth requirement and ensure safe function calls (reducing player disconnect instances).

The source code of the game discussed in this paper can be used as a boilerplate template to design more complex FPS games. In other cases, specific blueprints can be cherry picked to be used in other projects.

Acknowledgment

We express our heartfelt thanks and sincere gratitude to our Professor - Dr. Prerana Mukherjee for her constant support and fruitful guidance during the course of this project.

References

[1] en.wikipedia.org. 2022. First-person (video games) - Wikipedia. [online] Available at: ¡https://en.wikipedia.org/wiki/First-person_(video_games)¿ [Accessed 31 May 2022].

[2] en.wikipedia.org. 2022. Perspective (graphical) - Wikipedia. [online] Available at: ¡https://en.wikipedia.org/wiki/Perspective_(graphical)¿ [Accessed 31 May 2022].

[3] en.wikipedia.org. 2022. 3D computer graphics - Wikipedia. [online] Available at: ¡https://en.wikipedia.org/wiki/3D_computer_graphics¿ [Accessed 31 May 2022].

[4] Unreal Engine. 2022. Unreal Engine — The most powerful real-time 3D creation tool. [online] Available at: ¡https://www.unrealengine.com/¿ [Accessed 31 May 2022].

[5] docs.unrealengine.com. 2022. Introduction to Blueprints. [online] Available at: ¡https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/¿ [Accessed 31 May 2022].

[6] Unreal Engine. 2022. Marketplace - UE Marketplace. [online] Available at: ¡https://www.unrealengine.com/marketplace/en-US/store¿ [Accessed 31 May 2022].

[7] Sounds-resource.com. 2022. The Sounds Resource. [online] Available at: ¡https://www.sounds-resource.com/¿ [Accessed 31 May 2022].

[8] Engine, U. and Creation, W., 2022. How to create Grenade arc using Blueprint?. [online] Unreal Engine Forums. Available at: ¡https://forums.unrealengine.com/t/how-to-create-grenade-arc-using-blueprint/300186/4¿ [Accessed 31 May 2022].

TABLE I
File Size of Source Code and Final Packages

File Title	File Size
Source Code	≈ 9 GB
Linux Archive (for Linux based OS)	210 MB
HTML Archive (for web server)	145 MB