Assignment 2  
CIS002-3 Advanced Computer Games Technologies  
Artur Rudzinski

Introduction

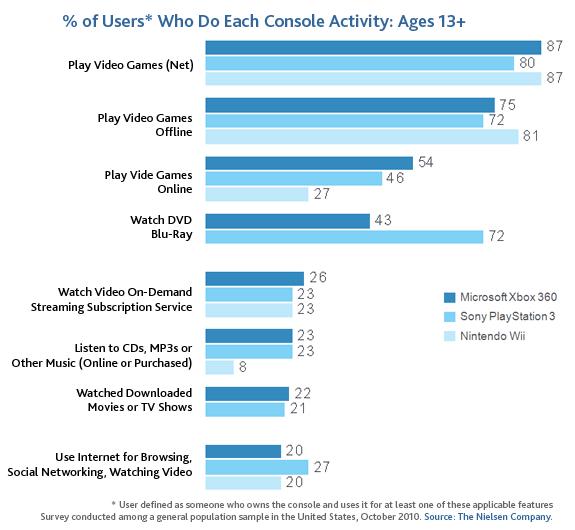
For this assignment I was tasked with contributing towards a polished game project that I can use towards my portfolio. In order to achieve this task I have chosen to polish an area of the game by myself. My chosen area of polish was the server and client interaction so that I can achieve multiplayer connection in my game that used the Steam servers as a connection host. To focus on this area of game design I created a simple team arena shooter in which I can construct my server and client interaction.

Experience from previous work

This is my second attempt at using the Steam Subsystem within Unreal Engine as I have worked with it in my second year of study, however every asset that will be made within this project will be remade from scratch. At any point I have not used my old assets however implementation of ideas and techniques will remain similar as my other first person shooter game project from second year.

Market Research

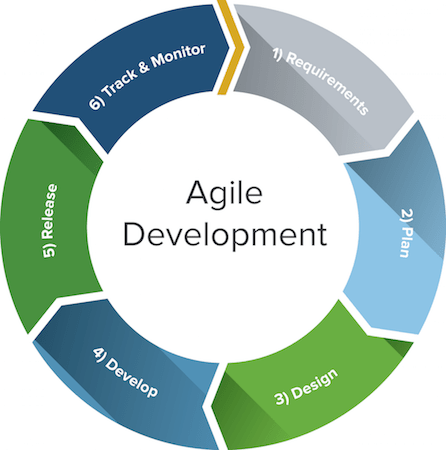
In the current age multiplayer experience is one of the most important aspect of action genre games such as shooters and MOBA type games. Most people who play games in the current generation use it for the online aspect of the given game. This was proven by the Nielsen Company in October 2010 (See Image below)

[1]

Seeing that many users who play online allowed me to come up with an idea to create a team shooter game, arena style as these are fairly popular today due to franchises like Call Of Duty and Battlefield making the popular on console and on PC.

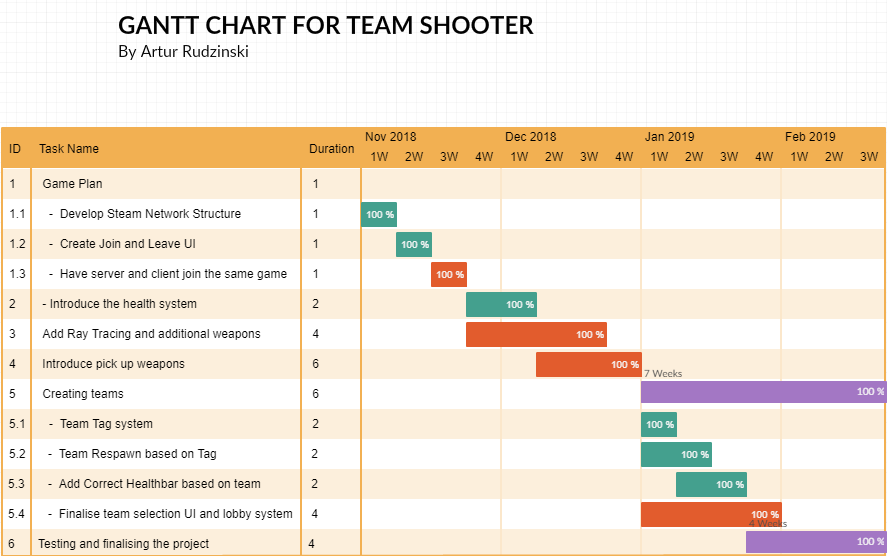
Methodology & Design

For this development process I decided to use the Agile approach. I have chosen the Agile development approach because it enables me to release patches and bug fixes into my project, as it is an online game constant bugs and content updates are required to keep players interested in my game when the first official alpha is released. It also gives me a chance to experiment with user feedback based on their experience with the game to see if they enjoy the content and the way the game works. I see this as a fit strategy for my online game as it will help me constantly update the game and fix it if issues arise. An example game that uses this approach is the game Overwatch By Blizzard Activision where they release bug fixes and balance patches every week or 2 into the game so players have a better gameplay experience. I think this approach will suit my game best and deliver the best results for my users and their experience.

 [2]

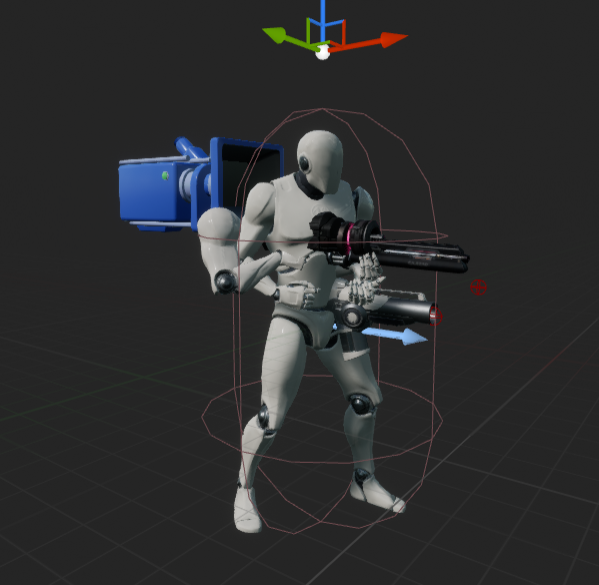
Project Plan

For this project I have focused on the multiplayer integration. I have planned for my game to provide users with a multiplayer experience. When I initially thought about this project I wanted to keep it simple and focus mainly on the multiplayer aspect of the game. This means that I was ready to use visual assets and particle effects from the Epic Games asset store. At this point I have decided to create a simple team based arena shooter with 3 weapons you can pick up around the map. 2 of those will use line tracing to deal damage to other player and 1 rocket launcher which explodes on hit and deals damage to all in blast radius. There were many factors to consider when I created my game, for instance I decided to use Steam servers to host and join games. I also decided to get player names from the Steam server using their Steam profiles instead of letting users set a name for themselves. This made implementation of servers easier as I wouldn’t have to setup my own server manually.



Implementation

When it came to creating my game I used blueprints within Unreal engine. This is a way of scripting a game that uses visual C# code to generate and edit events. There are many events that run within my game to make it work. These events are run within the character the player uses and UI that is given to the player when the game runs.

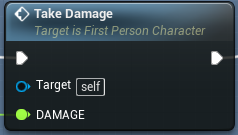
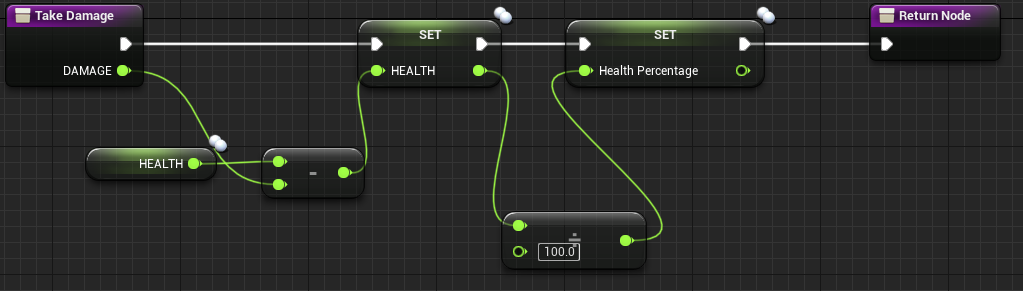
First person character in viewport.

First Person Character

Starting off the, first person character itself is responsible for setting up player controls and inputs such as mouse camera movement and character movement itself. It is also responsible for dealing damage to players and receiving damage from players. It is also responsible for holding the health bar of the player above its head and holding onto the correct team tag when the game is running.  
Some inputs within the first person character send custom inputs to the server so that they can be executed. Actions such as shooting and the direction of the bullets are required to be run on the server so that the clients can recognise that this action has been performed and therefore can be shown on the screen. This is done through a simple use of booleans to check if the shoot button, in this case the left mouse button , is being pressed and if it is then send this information to the server so other clients can access it(Source4, appendix).  
  
The code within this character is responsible for calling on specific functions when there are inputs given by the player. Those are then replicated on the server and executed on all clients. Features such as LineTracingForObjects use this to deal damage to other players when they are hit and also receive damage from other players when they are touched by another players’ line trace(source 1, appendix).   
The first person character also uses a weapon switching mechanic. It works by using a weapon array called “Guns” to verify if we have a second weapon available to switch to and uses a function called “Weapon Index” to give a number to a weapon and cycle through it in a number sequence based on how many items are in the “Guns” array. When cycling through the weapon index there is a small animation that will play which will cause the hands to pop off the screen to get an effect of the character reaching for his second weapon (source 2 & 3, appendix).

When a gun shoots it needs to notify all other clients on the server that a bullet has been shot and display that bullet to everyone on the server. For this event I have created simple logic which spawn the appropriate bullet based on what gun you are currently holding. That bullet will then be shown to clients so that projectiles like rockets from the rocket launcher can be seen by other players(source 5 & 6, appendix)

When the character receives any damage from a bullet trace or from a rocket explosion there is a function in place which deals damage to the character. This function called “Take Damage” gets the damage set by the weapon we’ve equipped and applies it to the character health when we receive damage(Source 7, appendix).

”Take Damage” function

After the “Take Damage” function is initialised there is a Boolean check in place to check whether or not the health of the player is less than one. If it is false we proceed to check if the health is equal or less than 0 to determine whether the character is alive or not. If the character is alive then nothing needs to happen apart from the “Take Damage” function, however if the Boolean checks the health of the player and it returns 0 then the player will be considered dead and based on their team Tag they will be respawned with 100 health and it will remove any weapons they have picked up(source 8, appendix).

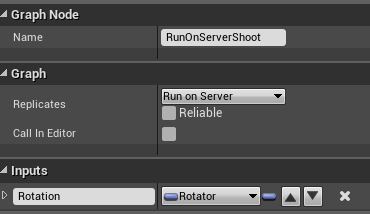
When the player health reaches 0 a function is called to teleport the player to a respawn location around the map. This function is called “Find Respawn Location”. What the function does is it calls a node called “Switch on Name”. This node then checks the Tag of the player, whether it is Red or Blue, and then picks a random number between 0-4 to pick 1 of the 5 respawn locations for that team around the map.

The Networking



Example of a “Multicast” node

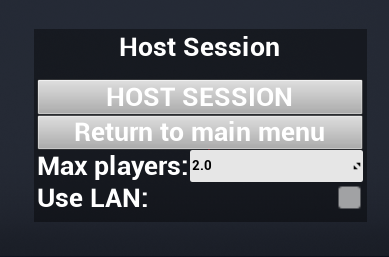
When it came to running my game on the server I encountered many problems. I lacked the knowledge as to how networking works in Unreal Engine and it was a major obstacle to replicate functions between the server and the clients. I wanted to keep the code relatively simple so I opted to learn the basics of how networking works within the engine. When I was designing the game I had to make sure most of my values were replicated so that both the server and the clients can access each other’s values such as health and damage dealt values. Clients needed to see each other’s actions, this included bullets rockets and anything else spawned by players, even particle effects. In order to display particle effects and bullets I had to use “Multicast” so that it executes on all clients and the server at the same time. Multicast is also used to display animation data both on server and between clients so that it does not interrupt gameplay when some packages are lost(source 11, appendix). The Multicast node is usually called on the server so that the server validates that this action has taken place. Every other function such as calculations and events are done on the server so that they replicate without issues because it is better to do actions on the server than doing it between the clients as it can prevent cheating in the future.



Example of “RunOnServer” Node

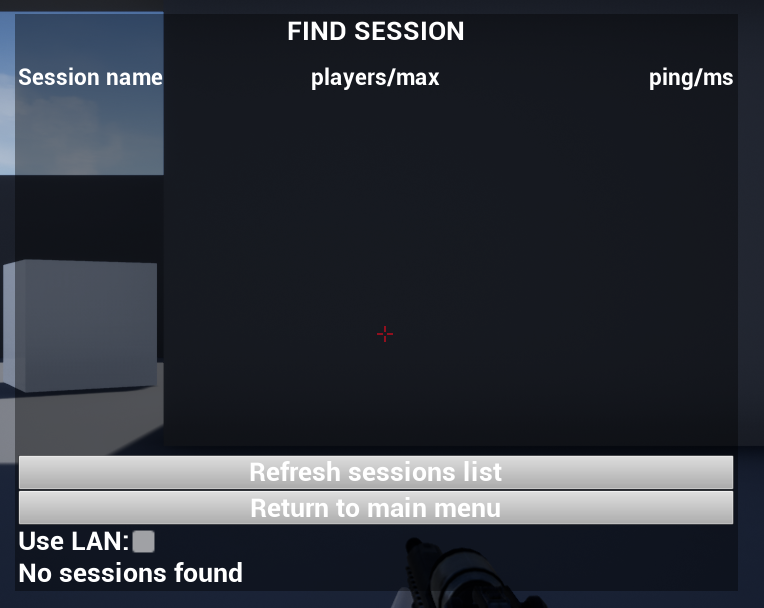
UI

The UI is what allows players to create and host lobbies via the Steam Subsystem. It allows users to create lobbies, set player limits and choose whether the player wants to use a LAN connection or Steam connection which used by default.



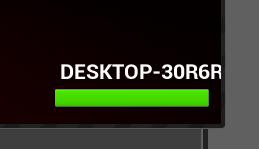
Host Lobby UI

When the player join the lobby that is being hosted they are sent to the waiting lobby(source 12, appendix). In the waiting lobby the players are introduced to the games’ control scheme and they are given options within the UI to start the lobby, return to lobby finding or to just close the widget and mess around in the waiting room.



Join Lobby UI

Upon entering the arena the user will be prompted to select a team they prefer to be in and when the team is selected the user can enter the arena and begin the match against others (source 13, appendix). When the match begins players can see their user interface which is simplistic as it includes their name and their HP in the bottom right of the screen, and on the bottom left of the screen it will open a chat window the user can use to communicate with other players.(source 14, appendix)



User interface.

Polished aspect of my game

In my game I have polished the multiplayer experience of the game. As I was working alone on this project I kept everything else relatively simple, things like map layout the look of UI and also the character and weapon animations. Because I have focused on the way clients can interact with each other I didn’t have as much content as I wanted to but I did make sure everything replicates between clients and that the gameplay experience on the server is as smooth as possible.

In this project I have achieved many of my aims that I wanted to provide players with. This includes:

* Players being able to host, join and leave lobbies.
* Set lobby player limit
* Decide between LAN and Steam connection
* Let the player decide what team they want to join which tags them red or blue.
* A lobby wait and launch system that checks if the number of users is even so teams are balanced
* A notification telling players that teams are not balanced if teams don’t have equal values.
* Line trace shoot system which is replicated between players, also deals damage and sends players back to the correct spawn based on
* Health bar above the player’s head showing their health
* Lobby chat window which let player communicate messages between each other.
* Multicast of particle and effects between the players.

As this was my area of polished I focused heavily on player to play interaction between server, clients and the players themselves so they can interact with each other within the game. Because I focused on this area specifically I had to use specific assets from the Epic Games asset store so that my game can be presentable and playable to a first person shooter standard. The assets that helped me achieve my game:

* Sci Fi Weapons Dark by Ying Pei Games
* Animation Starter Pack by Epic Games

Evaluation

In conclusion this project was really difficult. It required me to research way of implementing techniques within the Unreal Engine that are unknown to me, however after grasping the concept of some functions and why they are there it made for a relatively enjoyable project. There are many thing I could have improved within my final build that I could have included. I had ideas such as different maps 2 more weapons, a more stylised UI and even improved team features such as no friendly fire and capture the flag mode. The reason I couldn’t achieve those was not only a time constraint but also the lack of knowledge to achieve such a result as it would require me to do more research into the techniques and logic behind such game modes within Unreal Engine, especially that it will require more networking logic between the server and clients.   
If I had more time and more team members I would have implemented those features into my game so that it is more enjoyable to play with friends and so that content doesn’t become stale and repetitive to the player.

REFERENCES

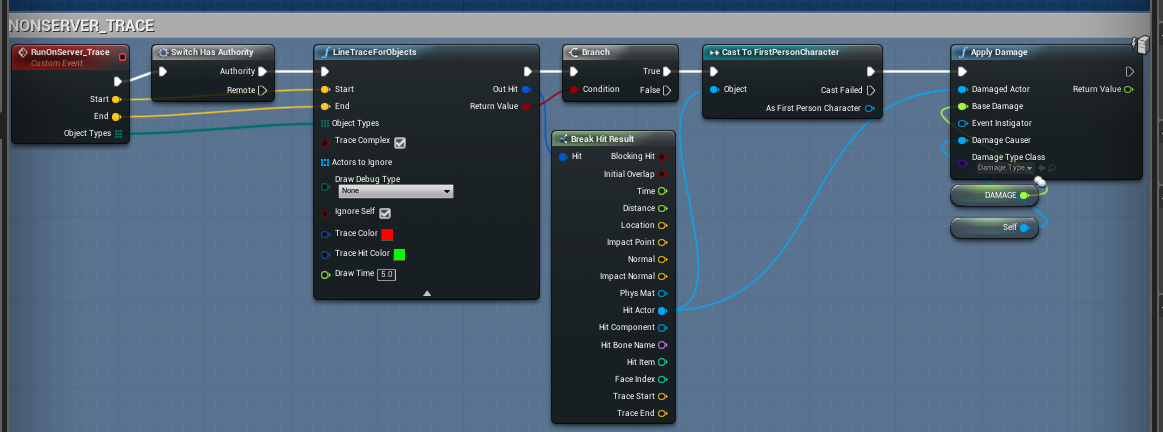
[1] The Nielsen Company (October 2010), *% of users who do each console activity*, Available at:  
<https://www.nielsen.com/us/en/insights/news/2010/game-consoles-edge-closer-to-serving-as-entertainment-hubs.print.html> (Accessed 16/01/2019)

[2]Agile Development diagram, Available At:  
<https://number8.com/kanban-versus-scrum/>   
(Accessed 16/01/2019)

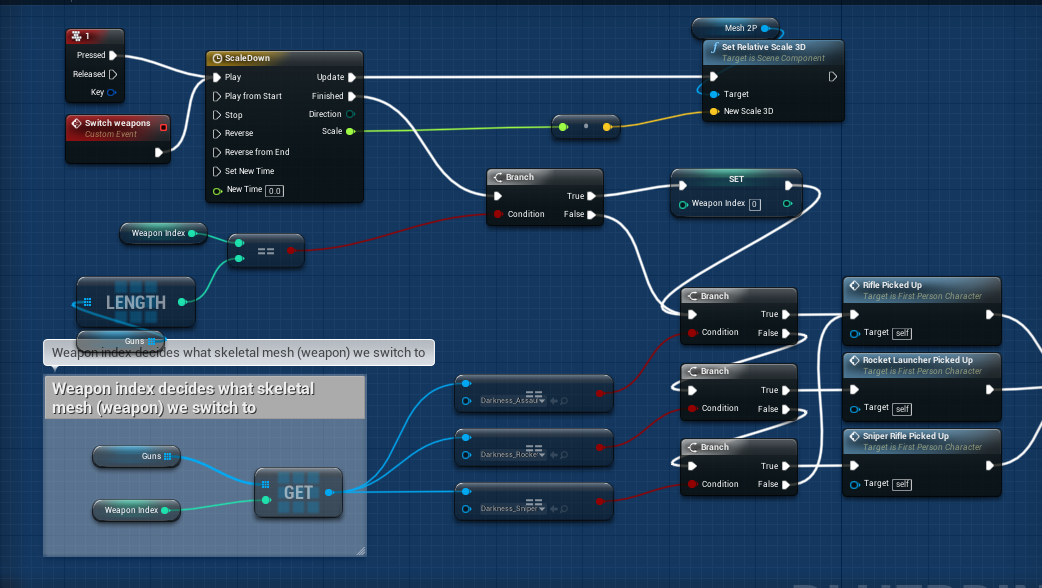
Epic Games 2019, Asset Store, digital download, Windows 10. Epic Games, UK

Appendix

[1] LineTraceForObjects



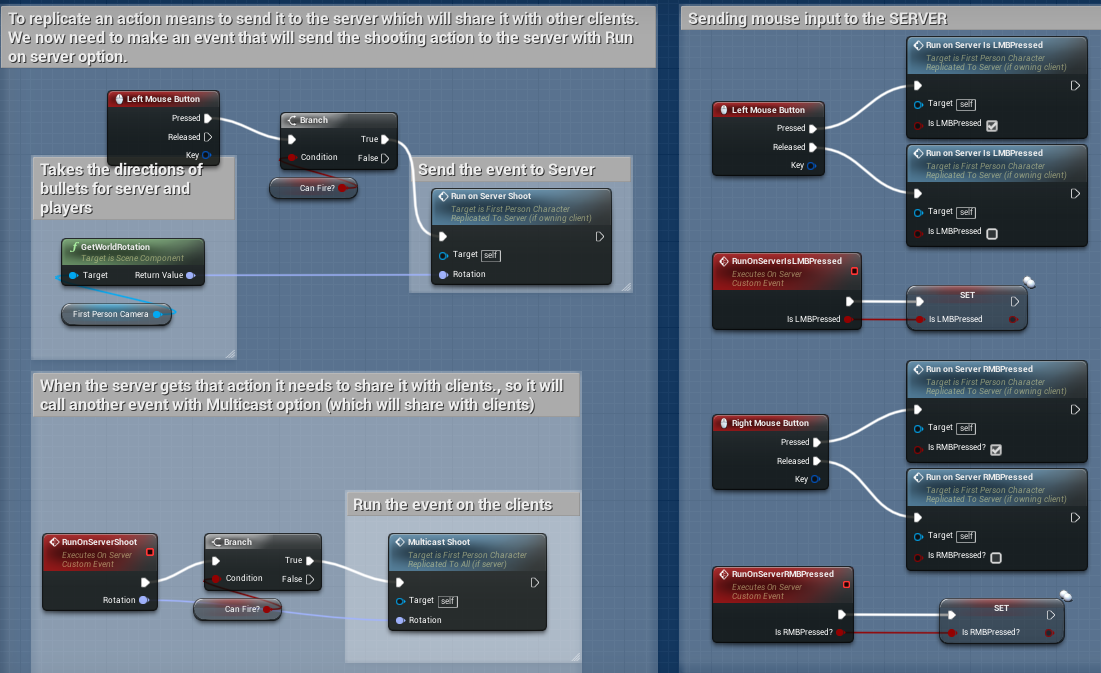
[2] Weapon Switching



[3] Weapon Switching Continued



[4] Mouse server inputs.

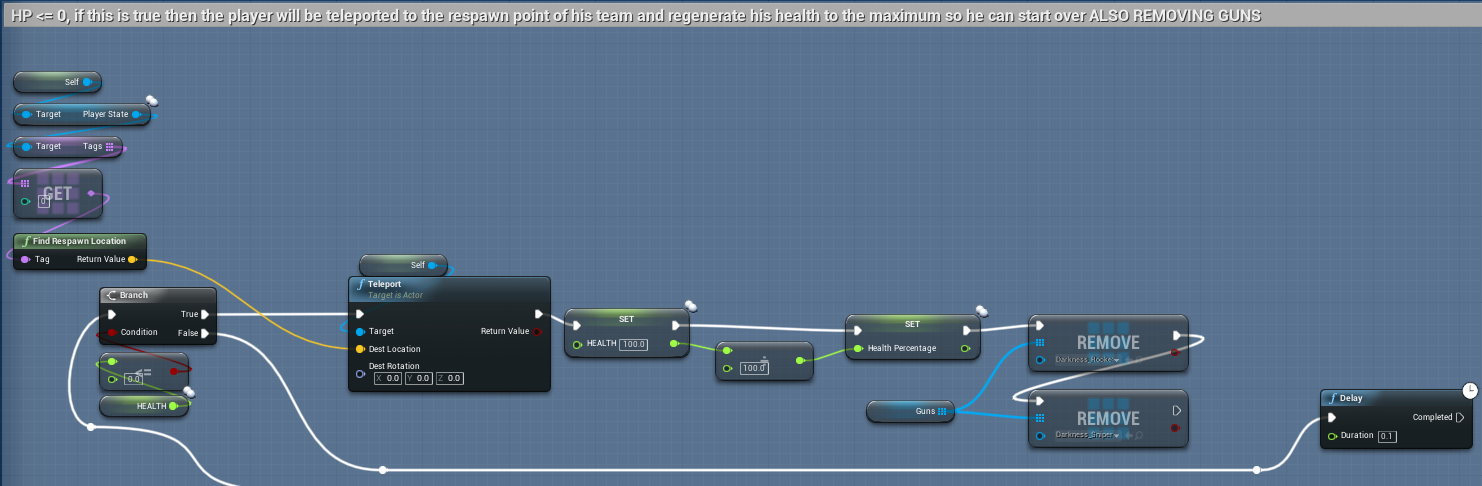


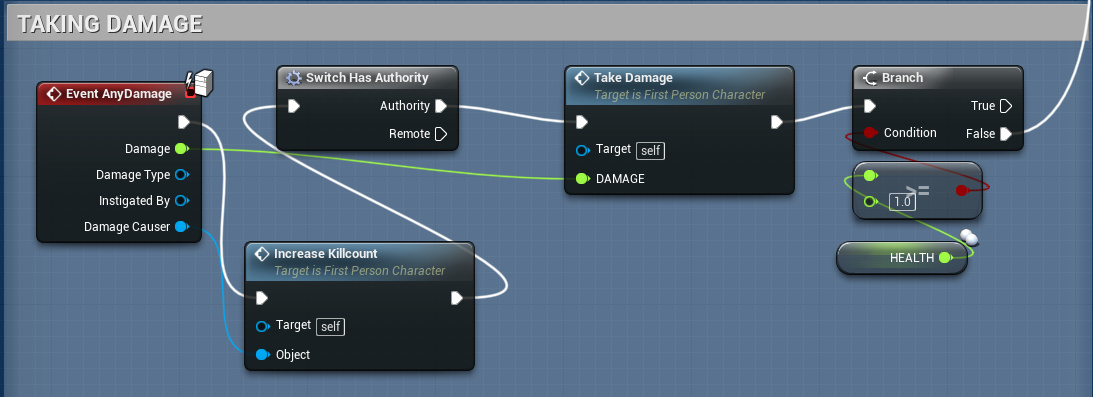
[5] Shooting bullets on the server so it displays to everyone. [6]Shooting bullets on the server so it displays to everyone, continuation.



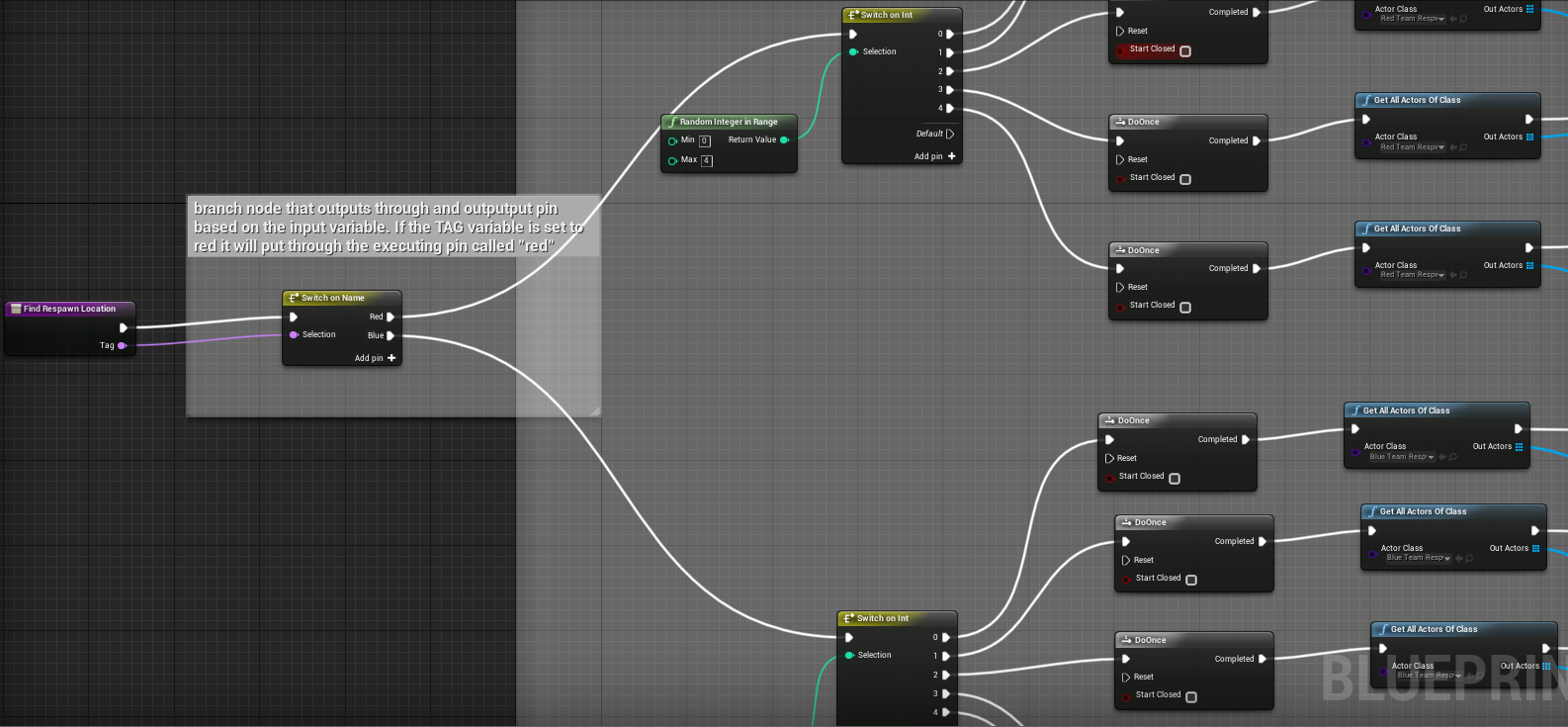
[7] Taking damage function on any damage.

[8] Respawning the player based on their Tag.

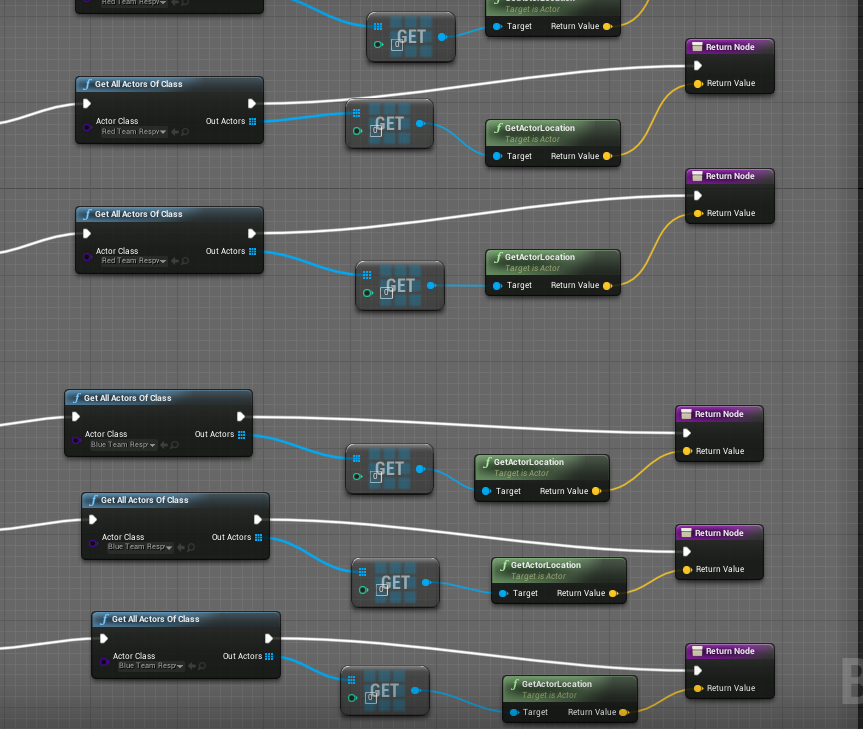




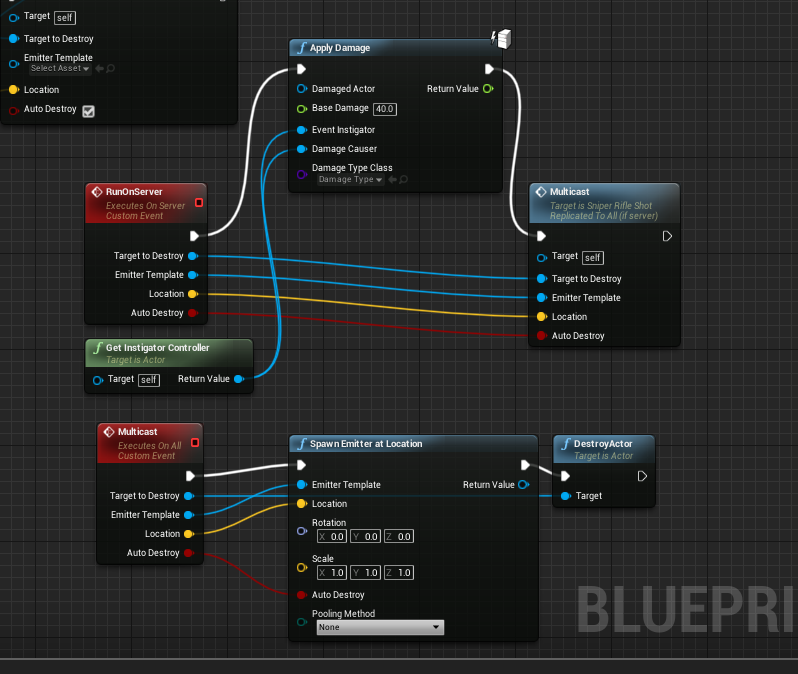
[9] Tag selector and random respawn picker.



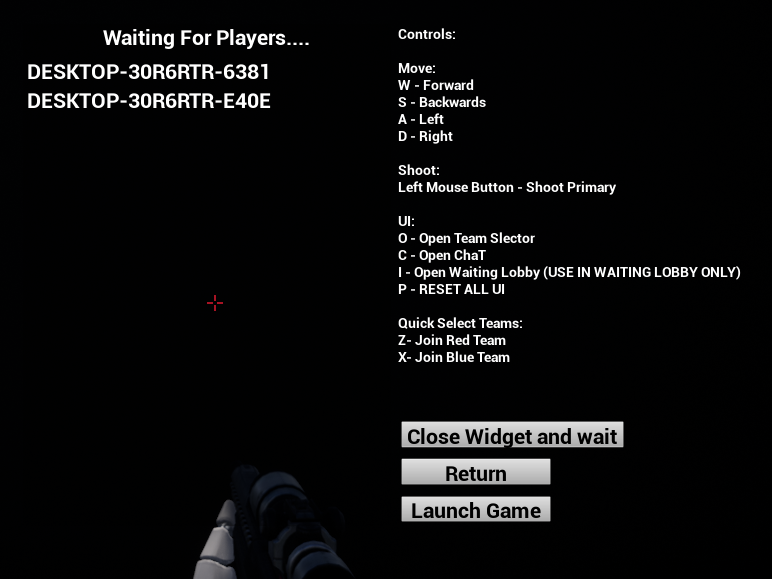
[10] Tag selector and random respawn picker continuation.



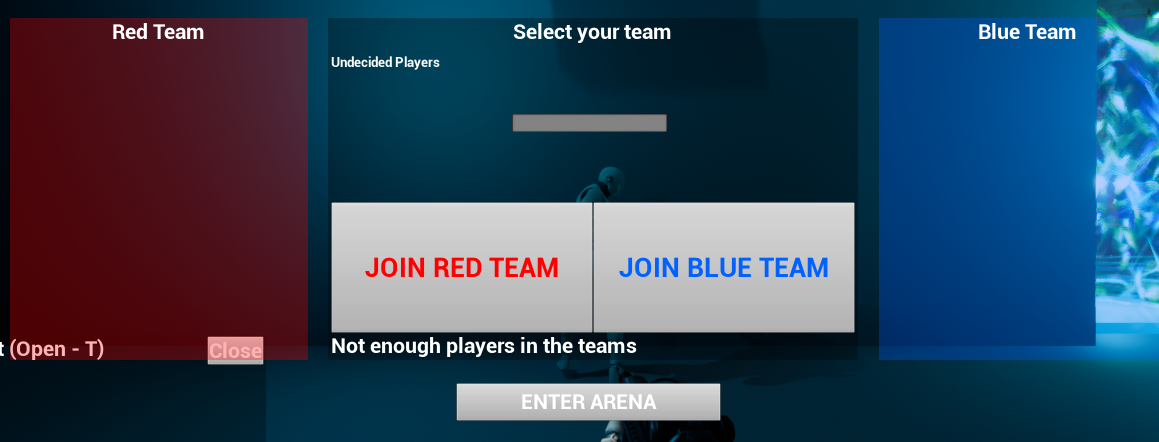
[11] Example of SniperRifle\_Shot being multicast in the engine



[12] Waiting room and controls guide.



[13] Pick your team UI



[14] Lobby chat window

