RS Overcharged

Programmer Report

Authors:

Micael Silva

15.07.2022

Introduction

In the context of the curricular unit 3D Game Project belonging to the Digital Games and Multimedia degree of the Escola Superior de Tecnologia e Gestão of the Polytechnic Institute of Leiria, this report is intended to present the project Ruined Sector: Overcharged, essentially focused on the programming developed.

In the development of the report, I present specifically the content already completed, and a breakdown of how everything works together.

This project taught me a lot, not only because I had never touched c++ in my life, but also because multiplayer was something that always scared me.

We started the project with a completely empty project without any assets. To make it as lightweight as possible

However, I did the best I could from the beginning, within what I knew in terms of knowledge and what I could do. The project expanded in a tremendous way once I managed to replicate the first things, and here we are now, I hope you enjoy the project as much as we do.

Table of Contents

Implemented Features

Systems

- Online Subsystem Steam API
- Hosting
- Joining
- Server List
- If the server is full remove him from the Server List
- Handle Client and Host Disconnection

World

- Random World Generation
- SkyDrops
- SkyDrops Travelling

Match

- Waiting Room
- Start the Match
- Match Timers
- End the Match
- Choose and Display the Winner
- Match statistics

Main Menu

- Server list
- Create Server
- Graphics
- Sound
- Training Mode

Core Gameplay

- Player Overcharge
- Shooting
- Reloading
- HealthSystem
- Damage
- Respawn
- Pickups
- Charges
- FOV
- Reload Speed

Sound

- Player
- Announcer
- Weapon
- Environment

In Game UI

- Crosshair
- HP
- Ammo
- Amount of Boosts
- Match Timer
- Kill Counters
- Reload Hint
- Hit Feedback
- Kill Feedback

Systems

Online Subsystem Steam API

The Unreal engine provides all the basics we need to host and join a game. However, once we exit the editor, the simple action of joining a server no longer works. Within the editor, the Unreal engine provides an online sub-system that is no longer present once we are in a standalone game instance.

What we need to do is, implement this on our own, or use one of the provided, for development purposes the OnlineSubSystemNull is suggested to work for LAN gameplay only. And we have used it. So no port-forwarding is required since it only works in a LAN environment

And now we know that there are two major Interfaces involved: IOnlineSubsystem and IOnlineSession each handles crucial parts of integrating our game with the subsystem.

In the end everything is basically driven by sessions, we notify a service about our servers presence, the service is then queried by clients who want to find active game sessions. A client then requests to join a given session, and then if everything is well, the client is allowed to travel to that server.

However the idea is to expand it and we must be able to connect in a non LAN environment

To do so, we have the Online Subsystem Steam API that enables us to ship Unreal Engine 4 (UE4) applications to Valve's Steam platform. The main purpose of the Steam module is to help us distribute our application with a set of features (such as matchmaking and leaderboards) to Steam users.

Additionally, the Steam module implements several of the interfaces being exposed by the Online Subsystem, supporting most of what is offered by the Steamworks Software Development Kit (SSDK).

Some of the available Steam Interfaces include:

- Matchmaking (Lobbies and GameServer APIs)
- Leaderboards
- Achievements

Steam supports peer-to-peer matchmaking (for both dedicated and listen server games) through lobbies, and provides the ability to run dedicated servers.

Lobbies give users the ability to learn information about game servers, and are usually used to convey game-specific information about listen server games, such as what map or mode is being played.

Users can also see the number of other users connected to a lobby without joining, through the matchmaking system.

Hosting

When hosting a game we first need to create a session for the online sub system. When a session has been created a delegate is fired which we can opt in to handle.

This is done by assigning a delegate like so. In general we should signal the subsystem that a match has started, in case we don’t want to list games in progress or maybe only list games in progress. This will tell the subsystem that this session and gameplay has started.

Joining

The online subsystem provides a way to find ongoing game sessions.

This is a pretty powerful feature that allows us to query the subsystem to retrieve a set of hosts that is currently running a session of our game.

The Online Subsystem Interface declares methods to do this, however it's up to the subsystem itself to allow querying.

Once we have a set of sessions we can either create a UI widget that allows players to pick a session and join it. Or we can choose for them by creating a matchmaking system where we put players against each other based on a set of parameters for instance, like for example, skill-level.

In the end we displayed each server through a server slot that would populate a scroll box widget, and players will be able to choose which server they want to join, they also can just click on a button, and they will join one of the servers that are available from the list.

Handle Client and Host Disconnection

Destroying a Session has to happen on both Server and Clients when they leave. We might find ourselves in a situation where we left a Server, but we forgot to clean up the Session and trying to Create or Join a new one won't work anymore until restarting the Game/Editor.

At that point we should think about Destroying the Session (if it exists already) before Creating or Joining.

That way we can ensure that our Players aren't getting stuck.

Or just come up with a secure way of Destroying the Session whenever the Player is not supposed to be in one.

For example when entering the Main Menu. Since when the match ends we are kicked out of the match and we return to the main menu, we destroy it there.

However if the hosts disconnects the game would crash because the client lost the connection to the host, as so we had to Handle some network errors, this error is handled by telling to the clients to go back to the main menu so they can destroy the session that they were in, by doing so the players are no longer locked to the previous session.

World

Random Arena Generation

Basically to generate the world, we have the arena divided by 9 tiles, these tiles are then stored in GameMode where at the beginning of the match take a shuffle to be spawned in specific areas of the world, thus constituting the arena.

These tiles interchange only in the position, and only by changing the position we have 362880 unique arenas

SkyDrops

There are two types of drops, Escape Pods, which contain the shards, that during the game serve as pickup, and Altars, which during the game support the use of shards, thus giving the player more charges to use every 3 shards delivered

As soon as we reach 4 minutes of the game we start spawning Escape, these drops spawn every X seconds, in order for them to spawn we have the information of which tile they will spawn on, since each tile has an X amount of spawnable places if that tile still has free slots we spawn a drop on it, if not, we skip it.

In the end, once we go around the list of tiles, we repeat the same process again and again until there are no more available slots.

The altars spawn at 3 minutes and 2 minutes, on specific tiles, these tiles are the Roundabout and the Temple

SkyDrops Travelling

The traveling of them is done with an interpolation from the initial position to the final position that is given by the tile.

Since vectors use floats, sometimes running logic at the exact moment they land is annoying because the interpolation is easing in and out, and will adjust his values slowly in the end of the interpolation, so, in order to run logic and stop the interpolation from being done, we create a tolerance and compare the two floats of the current and final position, and if it is within this same tolerance then the movement is done completely.

Core Gameplay

Overcharge

The overcharge works by increasing the speed of the player, whenever the player presses the F key and has charges available for use, it gradually increases the speed of the player, when he reaches the maximum number of speed, he cannot use any more charges.

The speed of the jump is also influenced by speed, so to control the speed of the player in the air we also have a list of speeds that are applied at the moment the player jumps

This overcharge also affects the music the player hears, the reload time, the character size, and the field of view.

Shooting

The firing of the weapon is done with the mouse click and as soon as it fires it checks the amount of ammo available, since it has infinite ammo, we just need to worry about the amount of ammo available in the magazine, as soon as it fires we create 5 traces.

Since the shotgun is not completely random, and its spread is only one shot per quadrant, for these same reasons, the center shot is always straight and deals a ton of damage, and the other spread shots have their own minimum randomization and maximum for each quadrant, these shots also deal less damage than the center shot that is always straight.

Reloading & Reload Speeds

The reload is very simple, we press R and start reloading the weapon, as soon as it ends we receive the ammo in the magazine, but depending on the overcharge stage this time is reduced, as we don't want the player to have a very low reload time either , in higher overcharge states the value is locked

HealthSystem & Damage

The health system is pretty simple, the health starts at 100, if the player is hit and his life drops more than 0 the player enters ragdoll, and adds a kill to the statistics of the player who killed him and to the counter next to the match timer, triggering the respawn.

Respawning

The respawn is based on where the enemy is in the world in the moment that he killed the player that is about to respawn, we start by getting the killer position and comparing this distance to each spawn point, the longer distance preserves, after checking the distance for each spawn point we create a transform so we can reset the player at that transform. Now the player is fully restarted and ready to start fighting again.

Pickups

The pickups can be found in the Escape Pods that fall from the sky, when entering in the zone the player auto pickups the shard and a message will appear to the player screen to deliver that pickup to the altar.

Charges

As said before the player starts with 5 charges, when pressing F he consumes one of them, since the player will run out of charges when playing, he will need more, that's why we have to deliver the shards to the altars, if 3 are delivered without diying the player receives another charge and is ready to get is overcharge stage that is the last stage of movement.

Match

Waiting Room

When starting the game the players will have their input disabled, until the match is full, with the lobby is full a countdown will start, and the players will be teleporting to the map, where the Scouting Phase will enter in place, this Scouting phase will disable Combat, since the players will have some seconds to look to the map and plan how they will play that match.

Starting the Match

The match starts after the scouting phase is over and both players are teleported to both their positions, when so, the combat is now on and everyone can start fighting each other. After one minute the drops will start falling from the sky, and 5 minutes later, the match will end.

End Match / Choose and Display the Winner

When the match is over the GameMode will get all the stats from the players and he will display them. The winner is chosen by the the game mode based on the total kills that each player has.

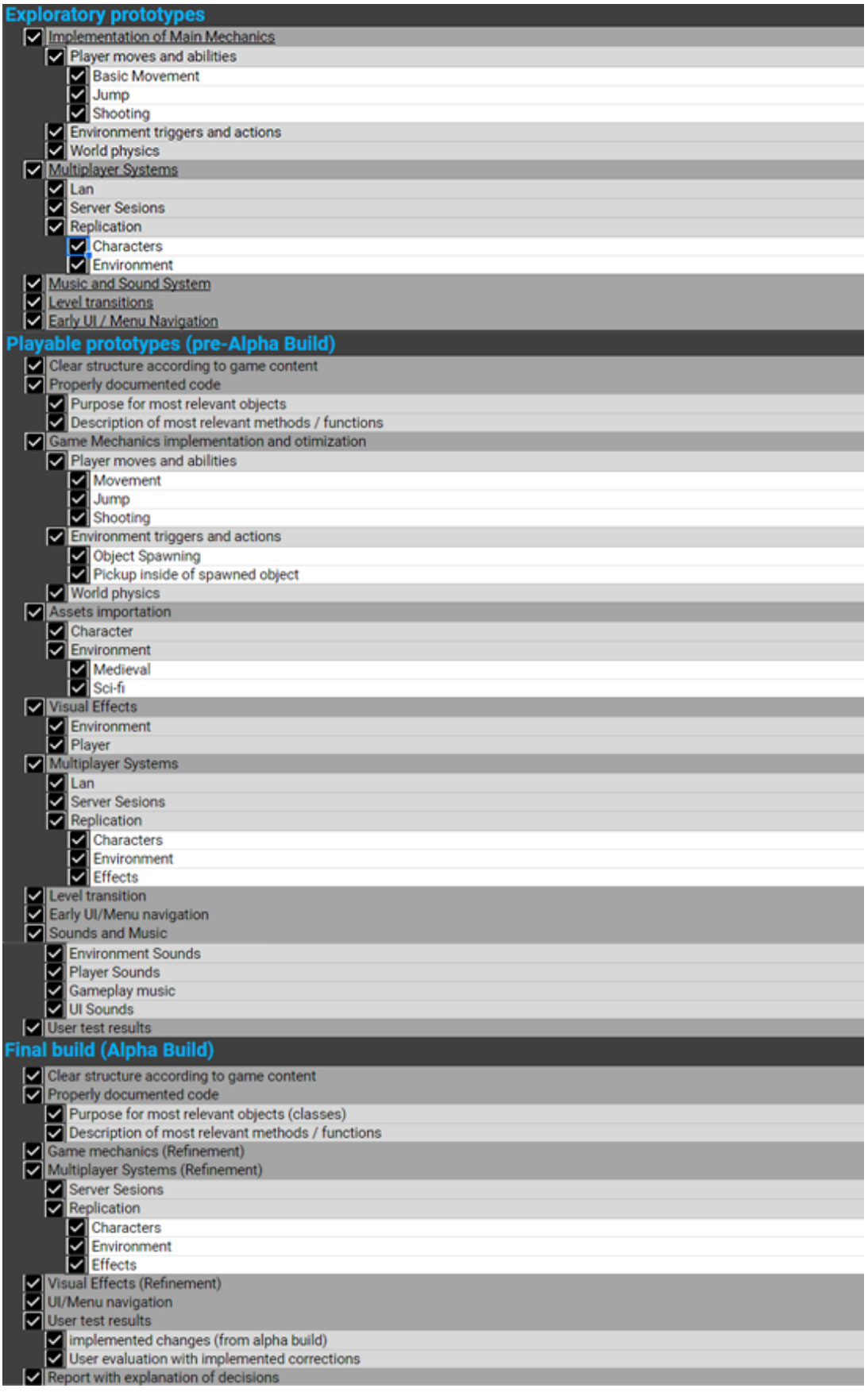
Match Statistics

Kills are increased everytime we kill a player, everytime that the player shoots he increments the shots hit by 5 that is the total number of shots, if the player hits a player, the shots hit increment, if he misses the missed shots increment, everytime we deliver a shard to an altar it increments too, just like the overcharge times, if the player reaches the last stage of movement we increment the number of times that we Overcharged.

Workflow

In this section I will leave the documents I've used to follow to know when things should have been done.

Pre Production



Production

What was left to do this last month during production, most of them were already implemented, just needed some refinement and implement the art and animations.

Micael	
Overall level	
stuff that needs to be added to the level in order to begin with play tests	
☑	**[Drop Pods]** when and where they spawn (drop from the sky)
☑	**[Altars]** when and where they spawn (drop from the sky)
☑	**[Pickups]** Grab a single one and deliver it to altar
☑	**[Altar]** upon having 3 pickups delivered a consumable is given
Player Speed	
Everything that involves the movement	
☑	**[Consumable]** Use it to increase speed
☑	**[Loosing Speed]** When player stops movement stacks decrease
☑	**[Loosing Speed]** When the player dies
☑	**[Overcharge]** When player speed is high starts to shine
☑	**[Overcharge]** Camera clarity and FOV changes
Match Start	
As soon as both players join in	
☑	**[Idle]** Match start cooldown
☑	**[No Weapon]** Player can move during 10 secs
☑	**[Combat begin]** Start the game per say
☑	**[1st death]** What happens imediatly after the 1st player death
☑	**[Timer]** Update timer
☑	**[Weapon]** Pickup
Match End	
What happens when the game ends	
☑	**[Results]** Screen with victory/lost followed with some stats
☑	**[Tie]** Game continues until someone dies
☑	**[Results]** Statistics
Sounds	
☑	**[Announcer]**
☑	**[Player]**
☑	**[Weapon]**
☑	**[Environment]**

Conclusion

To sum up, the project was fun and challenging. I know that I still have a lot to learn with C++ but I've been improving everyday.

I had too much fun and too much stress during the development of the project. The lack of knowledge didn't help at all, but practice makes the thing.

I've even noticed the difference in my way of writing code, after working on the Portal 2 project. since sometimes I would hit a wall and for some time I would be stuck without knowing what to do, or what was the problem with the code

But, in the end it was really fun, I've learned a lot.
I would love to keep working on the project and clean it up, so I can use it in my portfolio.

With this being said, I will keep practicing my C++ to become better.

Thanks for your attention!

Micael Filipe Martins Silva
15.07.2022
Undergraduate in Games and Multimedia

Webgraphy

UNetDriver | Unreal Engine Documentation

UEngine::HandleNetworkFailure | Unreal Engine Documentation

Basic Online Subsystem | Unreal Engine Community Wiki

Destroy Session | Unreal Engine Documentation

Online Subsystem Steam | Unreal Engine Documentation

Online Subsystem | Unreal Engine Documentation

How to display OPENING CREDITS in UNREAL ENGINE 4

Player sticking on stair steps - World Creation - Unreal Engine Forums

Audio component only plays on one client?

Unreal Forums | Unable to change Global PostProcess Settings at runtime

ECollisionChannel | Unreal Engine Documentation

UGameplayStatics::PlaySoundAtLocation | Unreal Engine Documentation

Restart Player at Transform | Unreal Engine Documentation

Restart Player | Unreal Engine Documentation

UGameplayStatics::PlayWorldCameraShake | Unreal Engine Documentation

System Settings | Unreal Engine Documentation

UNiagaraComponent | Unreal Engine Documentation

UNiagaraFunctionLibrary | Unreal Engine Documentation

Different animations for lower and upper body - Character & Animation - Unreal Engine Forums