# Network programming for Games

Written Report, Moa Jansson GP23

### Introduction

The game is mainly designed for communication with a focus on accessibility. The players will use emotes / animations of the character to express emotions, more so than words. This eliminates language barriers and allows players of a wide age range to explore a connection.

Throughout the development process I’ve been wanting to use basic functionalities offered by Unity’s Netcode for GameObjects and learn as much as possible about network programming. This include Remote Procedural Calls, Server Authority and the Network Manager together with Network Objects.

### Development Process

As a client in the game you are able to change emotes and have your character look sad, angry, happy or just idle. This is made possible by custom GameObjects within each player prefab to adjust the expression of the character.

En bild som visar skärmbild, Animering, text, Multimedieprogram

Automatiskt genererad beskrivning

All the GameObjects are hidden at the start for the Host and the Client and when a key is pressed, the relevant GameObject will be made active through the enum EmoteType. Initially I tried to use GameObjects with the Rpc in the script, but learned soon enough that GameObjects are considered an invalid type. Luckily, enums assisted me to set the right EmoteType to the right object through the SwitchEmote function.

En bild som visar text, skärmbild, programvara

Automatiskt genererad beskrivning

An issue with hiding the objects with the ClientsAndHost Rpc is that the state isn’t updated and synchronized fully when a second client joins the server. The first character’s emote at the time of second player connecting is what is transferred across the server and not until the first character alters the emotes to match the one that the second player sees, it will not be synchronized. After that though, the hiding of GameObjects works as desired. If I were to spend more time on fixing this issue I’d look into having the server confirm and update the current emote that both player’s are using at any given time. I’d also consider adding code to get the current emote of player one to load when player two is connecting to the server, so it is synchronized from the start.

For the rain system in place, I made sure to make a prefab of the objects that are required for rain and to add a Network Object component onto the object itself, but also adding the prefab to the Default Network Prefab list, so that it is recognized by the Network Manager. In the Game Manager script, the rain prefab is spawned through the OnServerStarted function that is subscribed to at the start of the application. Having the server initialize this prefab made the most sense as weather is more realistic if there only is one instance of the same rain, thunder etc. at a time. I did try spawning the rain locally and then sending it to the server as an Rpc but that resulted in multiple rain clones, which was a bit too crazy.

The actual player character’s are spawned as player objects with their own clientId to be able to assign the right prefab to the right client. I wanted player one to always spawn as Kitty and player two to spawn as Squirelly in the back. The problem with predefining what prefab will be used for which player was that I had to use an integer to keep track of the Client Count and hardcode values when Clients disconnect, to make sure that the next client connecting won’t spawn as the same character as player one.

Another feature is that both characters jump when spacebar is pressed and that happens though the InputAction where I check if it is the owner, meaning the client that initialized the player object, and if so the Jump function will be called on the Client and Host side. Not sure if they need to be sent to the Server as well.

Additional Objects that are spawned within the game, are cakes. There is a button that both players can press and depending on which character you possess, you initialize either strawberry or chocolate cake. The idea was to have them compete with their different flavours so I implemented a Cake Count and wanted to have results being shown for both players. Due to time constraints, I resorted to displaying it in the console as a Debug Log though and it is only shown on the Host’s side, as the cakes and the cake counter are spawned on the server.

En bild som visar Multimedieprogram, skärmbild, Grafikprogramvara, Pc-spel

Automatiskt genererad beskrivning

There were many challenges with achieving the initialization of different cakes and I pursued another solution before having the Clients send an Rpc to the server about spawning a cake based on their clientId. The first solution was to have the cake spawn the same way as I performed jumping, but this resulted in exceptions regarding ownership, as the server is the only entity that can spawn Network Objects so player one, as a host, could spawn cakes and have it sent to the client, but the client wasn’t able to do the reverse.

### En bild som visar text, skärmbild, programvara, Operativsystem Automatiskt genererad beskrivning

### Conclusion

Using Unity’s learning page proved to be important to be able to set up the Network Manager and I kept the StartHost, StartClient and StartServer prompts to be able to troubleshoot and explore the topic further. I believe that sever is best used for synchronization of states of the game and keeping track of rules / data / scores and that Clients can in extension to that take part of any game logic and then add inputs from their local system as well as request to change certain things in the game for all other clients. For example, adding another object or changing their position. They should also be able to adjust data that the server keeps track of, like the score, but the server should always check if the adjustment can be valid. I still think I’m very green when it comes to network programming but I’m starting to get the bigger picture, much thanks to the console in Unity and, of course, having the Log Level set to “Developer”.

### References

[Dedicated game server sample | Unity Multiplayer Networking (unity3d.com)](https://docs-multiplayer.unity3d.com/netcode/current/learn/sample-dedicated-server/)

[Client-server topologies | Unity Multiplayer Networking (unity3d.com)](https://docs-multiplayer.unity3d.com/netcode/current/terms-concepts/client-server/)

[About Netcode for GameObjects | Unity Multiplayer Networking (unity3d.com)](https://docs-multiplayer.unity3d.com/netcode/current/about/)