# Networking For reference:

# Requirements:

* The player shall be able to choose between four classes of characters
* The player shall be able to control his character’s movement
* The player shall be able to activate his character’s abilities
* Each character shall have distinct attributes (ex: speed, damage, health)
* A player shall not be able to play as a character that has been defeated (no more health)
* A player shall be able to make a new character of a class that has been defeated
* The game shall have at least two distinct environments (ex: desert, tundra…)
* The player shall be able to save his progress and resume it at a later date
* The game shall have at least three enemy types for the player to defeat
* The player must see growth in his character through experience and/or stat boosts
* The game shall be made using unity
* The game shall be orthographic
* Enemy strength shall scale (way to scale is too be determined)
* The game may allow for multiple players to be in the same environment at the same time\*
* The game may allow for players to play together over the network\*
* Levels shall be generated pseudo-randomly\*
* The player shall have an inventory\*
* The game shall have items for the player to store in his inventory\*

\*Requirements with asterisks are not required but would be nice

# Networking Requirements:

* Players can connect to a server (client)
* Players can host a server (host/server)
* Clients can send commands to their character through the server
* Hosts can safely disconnect clients, including voluntary disconnects and when clients are not responding
* Hosts should not leak memory from network calls
* Hosts update clients through information trackers
* Hosts have authority, clients have none

Scripts:

Network Menu – Hosting/Connecting server

Network Connector – Connects players

Network Controls – Sending/Receiving player commands

Network Trackers – Updates clients with information

**Design:**

Network Menu

-Consists of a menu with a “Host Server” button and “Connect to Server” button. Clicking one of the buttons activates one of the functions below.

|  |  |  |  |
| --- | --- | --- | --- |
| Host Server   |  | | --- | | Start Server  Get myself an ID  Initialize World (Not related to networking)  Send buffered (not to be removed) networking calls that contain the world’s static data (such as the map, physical objects, buildings, NPCs, rand seeds)  Initialize the host’s character | | Connect to Server   |  | | --- | | Poll Server List  Player Selects Server  Connects Player  Assign Player an ID  Sends Player’s Character’s Information  Begin Server’s **Network Connector**  (Server) Initialize new character | |

\*This script will be developed from a network manager template, therefore the detail design of this component is unnecessary.

Network Connector

-A script run by the host that safely connects new players.

|  |  |  |
| --- | --- | --- |
| New Player Connected   |  |  | | --- | --- | | Get list of current Network Objects  (Loop for each Network Object)   |  | | --- | | Re-initialize 1 object, preserving its state and values.  \*This script will have a separate case for every type of Network Object. |   Instantiate the newly connected player  Return “Player can now connect/initialize” | |

Network Controls

-Sends/Receives all inputs from clients.

|  |  |  |
| --- | --- | --- |
| Game Loop   |  |  | | --- | --- | | (Host) On input receive:  Find player object with input’s ID  Set object’s input variables to received variables  [Player objects accept these variables as if they were normal inputs. The inputs will not expire, in order to prevent problems with inconsistent latency and improve movement prediction.] | (Client) On input change:  Remove buffered package  Send package of inputs with my ID | |

Network Trackers

-A host component that tracks and updates clients and movement prediction. There will be a different tracker for each network object.

|  |  |  |  |
| --- | --- | --- | --- |
| Instantiation   |  | | --- | | Get a clean RPC ID from the network menu/manager. |   Game Loop   |  |  | | --- | --- | | (Host) Send tracker data:  Remove buffered package  Retrieve data and package in an argument array  Send array through RPC call | (Client) Receive tracker data:  Update variables based on received arguments.  [If applicable, update variables based on movement prediction data] | |

**Network Trackers:**

(Not an exhaustive list)

Transform Arguments:

-Vector3 Position

Movement Prediction:

-Vector3 Velocity

Animator Arguments:

-Float frame

Movement Prediction:

-Float time

**Exceptions/Other Notes:**

-Trackers for projectiles technically only need to update the movement prediction, once, on instantiation.

-Trackers and Network Controls don’t send out a new network call every frame, that would remove the buffered call before it even made it to the clients. They will need to update based on the worst ping (or on a per-player basis based on their average ping).

-Instantiations/Destroy calls will be buffered and will not be removed manually. They will be removed on destroy or when the host updates for a newly connected player.

-RPC group 0 is for