Networking Mirror Unity bez PlayFaba

Synchronizacja eventów – rodzaje skryptów

- NetworkManager – mózg całej operacji, zarządza tym co się dzieje przy połączeniach i tym co się dzieje po aktywacji serwera, zmiany sceny itp.
- NetworkIdentity – kontroluje osobowość obiektu podczas pracy serwera. Sprawia, że serwer wie o obecności danego GameObjecta
- NetworkBehaviour – skrypt przypisany do obiektu który posiada NetworkIdentity opisujący jego zachowanie podczas całego lifetime serwera.

Callbacki NetworkManager'a

- Dla Servera:
  - OnStartServer
  - OnServerSceneChanged
  - OnServerConnect
  - OnServerReady
  - OnServerAddPlayer
  - OnServerDisconnect
  - OnStopServer
- Dla Klienta:
  - OnStartClient
  - OnClientConnect
  - OnClientChangeScene
  - OnClientSceneChanged
  - OnStopClient
  - OnClientDisconnect

Callbacki NetworkManager’a cd.

- Callbacki serwera w poprzednim slajdzie opisują to, co serwer ma wykonać po danej akcji (podobnie dla klienta). Przykładowo, po zmianie z Lobby na Map callback OnServerSceneChanged będzie ważny żeby zespawnić graczy. OnStartClient może posiadać Invoke() eventów albo inne akcje. U nas są to tylko Invoke(), potem dla eventów są przypisane listenery które robią daną rzecz po Invoke() tj. schowanie przycisku „Join”.

NetworkServer i NetworkClient

- NetworkServer i NetworkClient są używane głównie po to, żeby użyć Invoke() danego eventu manualnie lub zarejestrować dany message który serwer obsługuje. RegisterHandler<AuthenticateMessage> opisuje co serwer ma zrobić po otrzymaniu danej wiadomości. To akurat jest wiadomość od playfaba która dodaje klienta do listy aktywnych graczy na serwerze. Głównie użyte są AddPlayerForConnection albo ReplacePlayerForConnection, które przypisują authority danemu obiektowi dla danego połączenia (oraz spawn na scenie). Jeszcze RegisterPrefab od NetworkClienta rejestruje spawnable prefab. Są jeszcze inne metody, jak NetworkClient.AddPlayer(), która jest automatycznie callowana w callbacku OnClientConnect i requestuje dodanie gracza (to, co jest wywoływane w OnServerAddPlayer).

Callbacki NetworkBehaviour

- OnStartServer
- OnStopServer
- OnStartClient
- OnStopClient
- OnStartLocalPlayer
- OnStartAuthority
- OnStopAuthority

Callbacki NetworkBehaviour cd.

Najważniejsze: OnStart[Client/Server/Authority], OnStop[Client/Server/Authority]. OnStartClient wykonuje skrypt po stronie klientów, OnStartServer po stronie serwera i OnStartAuthority w momencie w którym gracz dostaje kontrolę nad obiektem.

Ważne info: NetworkManager.StartHost() startuje serwer oraz klienta, standardowo w P2P. Czyli jak użyje OnStartClient i OnStartServer i w tych skryptach dodam coś do NetworkManagera to podwoją się dane u hosta które zostaną dodane. U nas jest to idealny przykład, bo używamy P2P i Client – Server. Dlatego sprawdzam, czy połączenie jest lokalne żeby nie callować dodawania obiektu do listy NetworkManagera podwójnie – w OnStartClient() jest dodawana instancja obiektu do listy połączonych graczy, w OnServerAddPlayer jest to dodawane przez NetworkConnectionToClient.identity.GetComponent<>, przez co host ma dublowane obiekty w liście.

Eventy w kodzie

- [Client]
- [Command]
- [ClientRpc]
- [Server]
- [TargetRpc]
- [SyncVar]

Eventy w kodzie

- Tutaj akurat każdy event jest ważny, więc każdy wytłumacze
- [Client] – taka metoda jest wykonywana jedynie po stronie klienta. Nie polecam używać (anticheating), ale dla czytelności kodu raczej wymagane. Przy okazji odciąża serwer bo tego typu skryptu są tylko wykonywane u klienta, po stronie serwera od razu jest return;
- [Server] – metoda wykonywana jedynie na serwerze. Ważne, żeby odciążyć klienta i wykonywać metody po stronie serwera bo u klienta metoda od razu przejdzie do return;

Eventy w kodzie cd.

- [Command] – prosta sprawa – metody te są callowane u klientów, ale wykonywane po stronie serwera. Dobre do sprawdzania czy klient nie używa jakiegoś cheata. Najlepiej żeby metody [Command] miały prefix Cmd, tj. CmdMovePlayer();
- [ClientRpc] – dosłownie na odwrót do [Command]. Metoda jest callowana po stronie serwera, ale wykonywana u klientów. Uwaga – nie zawsze jest sens używania tego. Jeżeli coś nie musi być callowane u każdego klienta (np. update ekwipunku gracza), to możemy użyć poniższego eventu.
- [TargetRpc] – podobnie do poprzedniego, tym razem call jest tylko u jednego klienta. Możemy dać NetworkConnection jako argument, wtedy metoda jest wywoływana tylko u klienta z tym połączeniem. W przeciwnym wypadku metoda jest wywoływana u klienta który posiada dany obiekt (authority).

Eventy w kodzie cd.

- [SyncVar] – zmienne synchronizowane przez serwer do klientów. Możemy dodać customową synchronizację przy użyciu metod, tj.

```
[SyncVar(hook = nameof(OnChange)] //nameof ważne, bo musi być string

int x;

void OnChange(int oldX, int newX)

{

//co zrobić z X

}
```