6CCGD003W

Network Games and Security

Tutorial 05

Applying Networking to Tanks

Paul Rosson

UNIVERSITY OF WESTMINSTER

Introduction

- Last week you should have finished the Unity network tutorial. If you haven't you should finish it before you continue.

- Now you need to apply these new network principles to your own tanks game. Unfortunately this is not trivial and is the biggest challenge in your coursework 1.

Network Manager

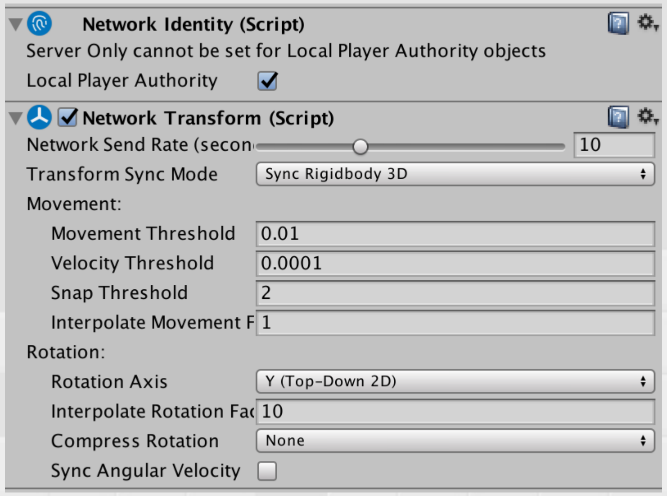
1. Create a new scene and delete everything in it. Call it 'Bootstrap'. Then create an empty GameObject in the scene called NetworkManager.

2. Give it a NetworkManager and NetworkManagerHUD component. This NetworkManager will control the spawning of the Player prefab.

3. Try to add the Player prefab into the 'Spawn Info' section of the NetworkManager. Check the error message that appeared…

Player Prefab

1. In order for the Player prefab to be spawned by the NetworkManager it needs a NetworkIdentity component, so add this. Tick 'Local Player Authority' so the local player will control the instance of this prefab.



Network Identity (Script)

Server Only cannot be set for Local Player Authority objects

Local Player Authority	✓

✓ **Network Transform (Script)**

Network Send Rate (secon	10
Transform Sync Mode	Sync Rigidbody 3D
Movement:	
Movement Threshold	0.01
Velocity Threshold	0.0001
Snap Threshold	2
Interpolate Movement F	1
Rotation:	
Rotation Axis	Y (Top-Down 2D)
Interpolate Rotation Fac	10
Compress Rotation	None
Sync Angular Velocity	☐

2. Also add a NetworkTransform component to this player prefab which will synchronise the transform across the network. Give it these values:

Network Manager

1. Now you can add the Player prefab to the Network Manager under the Spawn Info under both the Player Prefab and Registered Spawnable Prefab sections.

2. Also add your ‘Game’ scene as the online scene in the Network Manager. This means that the Game scene will launch when a player connects to the server.

Game Manager

1. Disable the Bot Spawner now since we're moving over to a multiplayer game.

2. Finally we want to take the following lines of code out of the Game Manager since the network manager now controls the spawning of the player:

```
//get the team value for this player
int teamIndex = GameManager.GetInstance ().GetTeamFill ();
//get spawn position for this team and instantiate the player there
Vector3 startPos = GameManager.GetInstance ().GetSpawnPosition (teamIndex);
playerPrefab = (GameObject)Instantiate (playerPrefab, startPos, Quaternion.identity);

//assign name and team to Player component
Player p = playerPrefab.GetComponent<Player> ();
p.myName = playerName;
p.teamIndex = teamIndex;
```

Build Settings

1. Don’t forget to add the new Bootstrap scene to the build settings, it should be the first scene in the list.

2. Now build and test your game. The two clients will connect together but there will be issues involving both server and client controlling the same tank, with the client unable to fully move it properly.

3. Try to work out why this is happening. It involves both client and server being confused about which tank they should be controlling and following (isLocalPlayer).

Debugging

1. Debugging issues like this is a big part of network programming. You may want to add some debug logs to find out what is happening.

2. Hint: Currently our start function is called on both player objects by both the server and the client. Instead make Player a NetworkBehaviour and move some of the Start logic into OnStartLocalPlayer instead (check the documentation). This is only called by the machine that creates the object and is generally where you will want to set up any camera and input.