Lab 4

In this lab, we are going to create a networked projectile. When a player shoot a projectile, the projectile will be instantiated on the client. Here we are going to use RPC again to sync to the other clients.

1. Create a Circle prefab and add rigidbody2d, network transform and network object components
2. Add the prefab to the NetworkPrefabsList that attached to the NetworkManager object

A screenshot of a computer

Description automatically generated

1. We will assign the bullet to the script that has the movement of the player (if you continue from the previous lab then the file would be movementPlayerRPC.cs) by adding:

[SerializeField] private Transform spawnObjectTransform;

1. Create a Transform variable for the object to be created later

Transform spawnedObjectTransform;

1. Assign the Circle (which is the bullet) to spawnObjectTransform by drag and drop the Circle object into the variable

Graphical user interface, application

Description automatically generated

1. Next we will make the projectile to be spawned in the network. While object instantiation can be invoked in each client, only the server can set the object to be network-spawned (this would be useful to distinguish each projectile spawned in network). So on detecting a button input (in this case a button “R”) normally in Update or FixedUpdate:

if (Input.GetKeyDown(KeyCode.R))

{

createBulletShotFromClientServerRpc(transform.position.x, transform.position.y, transform.position.z, transform.rotation);

}

1. When the client shoot a projectile, the server will instantiate the projectile and set the network spawn to true on the network object attached to the projectile [ServerRpc]

private void createBulletShotFromClientServerRpc(float positionx, float positiony, float positionz, Quaternion vector3rotation)

{

spawnedObjectTransform = Instantiate(spawnObjectTransform, new Vector3(positionx, positiony, positionz), vector3rotation);

spawnedObjectTransform.GetComponent<NetworkObject>().Spawn(true);

}

1. Then we will make the projectile to move so create a script called projectileMovement to be attached to the projectile prefab
2. projectileMovement class should be derived from NetworkBehaviour (don’t forget using Unity.Netcode)
3. Add the code below to make the projectile to move

private void Start()

{

projectileSpeed = 3.0f;

}

private void FixedUpdate()

{

GetComponent<Rigidbody2D>().AddForce(transform.right \* projectileSpeed);

}

# Destroying Projectile

1. Now we are going create a target. Create a 2d box object, add the Rigidbody2D as static, Box collider 2D and create and assign a new object tag (lets call it *TargetPracticePoint*) to the object
2. Check the projectile prefab and make sure the prefab has Rigidbody2D as dynamic and Circle Collider 2D
3. On the *projectileMovement.cs* which is supposed to be attached to the projectile prefab, we are going to check when the projectile collides with the target

void OnCollisionEnter2D(Collision2D collision)

{

if (collision.gameObject.tag == "TargetPracticePoint")

{

GetComponent<NetworkObject>().Despawn(true);

Destroy(this);

}

}

# Moving platforms

1. Add a square sprite object to the screen and name it *movingPlatform1*
2. Add Rigidbody2D and Collider2D to the gameobject
3. Create a tag called *MovingPlatforms* and assign this tag to the gameobject
4. Add a script called *movingPlatformsScript* and add the code below for the basic movement of the platform to the script

void Update()

{

transform.position = new Vector3(Mathf.PingPong(Time.time \* 2, 10), transform.position.y, transform.position.z);

}

1. Add NetworkObject component to *movingPlatform1* gameobject and make sure to tick *Syncronise Transform*
2. Add NetworkTransform component and set the sync X axis so it will only sync the X axis automatically to all clients

Tasks:

* Create some platforms for your game
* Consider a different way of syncing the position the platforms (bear in mind if we use *Syncronise Transform* and sync through the NetworkTransform then literally the syncronisation will be done in realtime every single time which could be burdensome if you have many moving platforms. This would back to the question on the importance of sync this object simultaneously in realtime)

Notes:

* You should consider if there is any other alternative way to do this
* If you have any other specific things to do with the projectile that require network sync, you need to use RPC