**Advanced Networking Topics**

Zach, Wilson, Florin

After completion of the basic tutorial we are able to connect to a server and move our own player object, however it will be apparent that there are some very serious lag issues. Any other players will appear to jump around the screen; this tutorial will focused on fixing these issues.

The first step we will take to resolve this is by interpolating the network characters movement to prevent the jump from data packet loss.

1. To do this correctly, we will be switching our player movement code to a C# script for the ease of use.
2. Create a new C# script with the name Playerscript and attach it to the prefab we had before. Make sure to remove the old javascript version of Playerscript and drag the new one into the “observed” section of the network view component.
3. Copy the following code into the new Playerscript file.

using UnityEngine;

using System.Collections;

public class Playerscript : MonoBehaviour {

public float speed = 10f;

private float lastSynchronizationTime = 0f;

private float syncDelay = 0f;

private float syncTime = 0f;

private Vector3 syncStartPosition = Vector3.zero;

private Vector3 syncEndPosition = Vector3.zero;

void Update()

{

if (networkView.isMine)

{

InputMovement(); //Handle our movement

}

else

{

SyncedMovement(); //Handle other players movement

}

}

void InputMovement()

{

//wasd response

if (Input.GetKey(KeyCode.W))

rigidbody.MovePosition(rigidbody.position + Vector3.forward \* speed \* Time.deltaTime);

if (Input.GetKey(KeyCode.S))

rigidbody.MovePosition(rigidbody.position - Vector3.forward \* speed \* Time.deltaTime);

if (Input.GetKey(KeyCode.D))

rigidbody.MovePosition(rigidbody.position + Vector3.right \* speed \* Time.deltaTime);

if (Input.GetKey(KeyCode.A))

rigidbody.MovePosition(rigidbody.position - Vector3.right \* speed \* Time.deltaTime);

}

private void SyncedMovement()

{

syncTime += Time.deltaTime; //Time since last call

rigidbody.position = Vector3.Lerp(syncStartPosition, syncEndPosition, syncTime / syncDelay); //Lerp at the correct speed

}

void OnSerializeNetworkView(BitStream stream, NetworkMessageInfo info)

{

Vector3 syncPosition = Vector3.zero;

Vector3 syncVelocity = Vector3.zero; //We will be adding velocity into the consideration

if (stream.isWriting)

{

syncPosition = rigidbody.position;

stream.Serialize(ref syncPosition); //Send Pos

syncVelocity = rigidbody.velocity;

stream.Serialize(ref syncVelocity); //Send Velocity

}

else

{

stream.Serialize(ref syncPosition); //Recieve Pos

stream.Serialize(ref syncVelocity); //Recieve Velocity

syncTime = 0f;

syncDelay = Time.time - lastSynchronizationTime;

lastSynchronizationTime = Time.time;

syncEndPosition = syncPosition + syncVelocity \* syncDelay; //We multiply the velocity by the delay to create

//the effect of movement prediction through times of lag.

syncStartPosition = rigidbody.position;

}

}

}

1. Create a new build of the game and test it. You should notice that the movement is much smoother, but there is still some input delay. What we can to do combat this delay is involve movement prediction into the code.
2. Copy this new version of the NetworkView serialization into the Playerscript.

void OnSerializeNetworkView(BitStream stream, NetworkMessageInfo info)

{

Vector3 syncPosition = Vector3.zero;

Vector3 syncVelocity = Vector3.zero; //We will be adding velocity into the consideration

if (stream.isWriting)

{

syncPosition = rigidbody.position;

stream.Serialize(ref syncPosition);

syncVelocity = rigidbody.velocity;

stream.Serialize(ref syncVelocity);

}

else

{

stream.Serialize(ref syncPosition);

stream.Serialize(ref syncVelocity);

syncTime = 0f;

syncDelay = Time.time - lastSynchronizationTime;

lastSynchronizationTime = Time.time;

syncEndPosition = syncPosition + syncVelocity \* syncDelay; //We multiply the velocity by the delay to create

//the effect of movement prediction through times of lag.

syncStartPosition = rigidbody.position;

}

}

1. This code is a good building block, but it can be improved upon to make it better, more sophisticated movement prediction and only interpolating after the first sequence are easy modifications that will work wonders.