

2019-05-17

Unity同士で別のPCにセンサーデータや入力データを送りたい時のMirrorを使うサンプル

Tips SHOWROOM xR Lab Unityエンジニア

こんにちは。SHOWROOM xR Lab エンジニアの @izm です。

Unityを使ってハードウェア絡みのプロジェクトを作っていると、例えば端末を2台用意して

- PC1: メインPC、表示を行う
- PC2: サブPC、ViveTrackerをぶら下げてPC1にデータを送る

というようなことがしたくなります。

特に OculusTouchは1PCに2個までしかペアリング出来ないとか、Noitom Hi5は1PCに1個しかペアリング出来ないとかの制限があったりするため、こういう需要は結構あります。他にはOculusGoに外部機器を無理やり外付けするかのような処理、などでも使い道がありそうです。

一番定番の方法としてはUdpClientを使ってSendとReceiveをするのですが、双方向性を担保したり、GUIを挟めたりするのが結構面倒くさいです。


このエントリは、Unityアプリ同士でこういった通信を(比較的)簡単に実現する手法を解説します。
普通のUNETやPhotonを使ったネットワークゲームを作る場合は、同一のシーンであることを前提にした各種チュートリアルがありますが、今回の用途だと送り手側と受け手側で、別のシーンを使う事が多いと思います。
なので今回のサンプルも送り手と受け手が別シーンという前提です。(もちろん同一シーンでも動きます！)

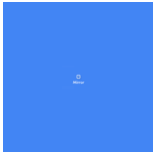
ネットワークライブラリにMirrorを使う

通信するためのネットワークライブラリにはUNET後継のMirrorを使います。

Mirror - Asset Store

Mirror is a high level Networking API for Unity, built on top of the low level Telepathy library. (Forum) (Discord) (Docs) Mirror is built and tested for MMO Scale Networking by the developers of uMMORPG , uSurvival and Cubica . Mirror is optimized for ease of use and...

 assetstore.unity.com



assetstore.unity.com

MirrorはほとんどUNETと同じような使い勝手で、別途NobleConnectなどのRelay&NAT PunchThroughソリューションと組み合わせることで、インターネット越しに処理を送ることも出来ます。

つまり、LTE回線のiPadをコントローラとして使ったVJアプリ、とかお客さんが手元のスマホから情報を送る、などの使い道もあります。

(NobleConnectやMirrorの説明については拙blogエントリにまとめてあります)

カテゴリー

SHOWROOM Tech Studio (20)

SHOWROOM xR Lab (7)

サーバーサイドエンジニア (11)

フロントエンドエンジニア (2)

iOSエンジニア (4)

Unityエンジニア (2)

【イベント】参加しました (8)

【イベント】開催しました (1)

【イベント】登壇しました (3)

やってみた (14)

制度・文化 (4)

Tips (1)

月別アーカイブ

- ▼ 2019 (15)
- 2019 / 12 (1)

2019 / 9 (1)

2019 / 7 (2)

2019 / 6 (1)

2019 / 5 (1)

2019 / 3 (6)

2019 / 2 (1)

2019 / 1 (2)

▶ 2018 (14)

▶ 2017 (1)

リンク

SHOWROOM

【採用】SHOWROOM Tech Studio

【採用】SHOWROOM xR Lab

SHOWROOM株式会社HP

izm_11's blog
id:izm_11

Hatena Blog

2019年における個人開発あるいは小規模開発のUnityリアルタイムネットワークの技術選定

概要 Unityネットワーク完全に理解した勉強会でしゃべったことのフォローアップエントリです。 基本的にはPUN2(あるいはPUN1)を推奨していますが、PUNがマッチしないときの技術選定についての話をします。 リアルタイムネットワークゲームをUnityで作

2019-05-03 20:48 ★★★★★ 30 users

izm-11.hatenablog.com

単純なセンサー情報を送るためのMirrorの使い方

最初に注意点

大事な点を一個先に書きます。シーン上で **NetworkIdentity**と**NetworkBehaviour**を使わない ようにしましょう。

なぜかというはUNETやMirrorのNetworkIdentityというのはネットゲームを想定した作りになっていて、

PC1とPC2が 同一シーンに置いた ゾンビ発生器 の発生確率パラメータを同期する、などの時に有用なのですが、同一シーン同士でないと意味がありません。

今回のように送り手と受け手が別シーンになると、NetworkIdentityの同一性チェックが失敗します。(シーンID+GUIDが完全一致していないと動かないので)

代わりに **MonoBehaviour**内で**NetworkClient.Send**とか**NetworkServer.isConnected** みたいに直接staticなNetworkClientとかNetworkServerクラスを叩いてください。

//PhotonのPhotonViewは別シーンでもPhotonViewIDを揃えれば動いた気もします！

具体的な作り方

前置きが長くなりましたが作り方です。

送るデータはこんな感じでクラス定義します。MessageBaseクラスを継承するのを忘れないようにしましょう。

SendData.cs

```
[System.Serializable]
public class SendData:MessageBase{
    public string name;
    public Vector2 axisData;
}
```

送り手.unity

送り手側はクライアントとして振る舞います。

NetworkManagerを置いてNetworkManagerHUDを付けておきます。

送り手スクリプトは別GameObject(Sender)を作って SendTest.csを以下のように書きます。簡単！！！！

```
public class SendTest : MonoBehaviour
{
    void Start()
    {
        //ここで、センサーの初期化をする
        //DoSomethingInitSensors();
    }
}
```

```
    }

    void Update()
    {
        //接続してない時はセンサーデータを送らない
        if (NetworkClient.isConnected)
        {
            //ここでSendDataを作る、センサー情報を差し込む
            SendData data = new SendData();
            data.name = "Hoge";
            data.axisData= new Vector2(Random.Range(-3f,3f),0);

            //NetworkIdentityとか無視して、直接送信する!!!
            NetworkClient.Send(data);
        }
    }
}
```

受け手側.unity

受け手側はサーバとして振舞います。NetworkManagerを置いてNetworkManagerHUDを付けておきます。

受け手スクリプトは別GameObject(Receiver)を作って ReceiveTest.csを以下のように書きます。

```
public class ReceiveTest : MonoBehaviour
{
    public static Action<SendData> OnReceivedSensorData;

    void Start()
    {
        //サーバ側でイベントデータを登録しておく
        NetworkServer.RegisterHandler<SendData>(ReceivedInfo);
    }

    private void ReceivedInfo(NetworkConnection conn, SendData data)
    {
        Debug.Log(JsonUtility.ToJson(data));

        //ここでActionを生やす
        OnReceivedSensorData?.Invoke(data);
    }
}
```

受け手シーン側では

```
ReceiveTest.OnReceivedSensorData+= 何かAction
```

みたいな感じで使いましょう。

注意点

Playerは念のためEmptyPlayerみたいなprefabを作る ようにしてください。NetworkManagerの AutoSpawnPlayerをオフにするのは、要らぬバグを踏むリスクがあります。

また、送るカスタムメッセージ定義はGenericsを含められません。JsonUtilityでシリアライズ可能でも、Mirrorの カスタムメッセージ定義では失敗します。つまりListはダメで string[] はオッケーです。このあたりは最初にテス トを書いてください。

最後に、C++アプリとUnityの通信とか、Unity同士ではない組み合わせだと破綻するので注意してね！！

サンプルプロジェクト

コントローラの入力を送る最小サンプルプロジェクトを以下に公開しています。
GitHub - neon-izm/MirrorOtherSceneServerClientSample



まとめ

毎回UdpClientやNetworkReaderをスレッド分けて書くのも大変なので、こういったお手軽通信方法を覚えておくと、モックを作ったりデモを作る時に便利です。

今回使用したMirrorは情報自体はそれほど多くないものの、UNETと基本的な書き方は変わらないです。そのためUNET解説記事を参考にすることが出来るので、安心して使えると思いました。

SHOWROOM xR Labはリアルタイムネットワークで苦しんだ経験があるエンジニアも募集、歓迎しています。よろしくお願いします。お問い合わせはお気軽に！

【xR事業部】Unityエンジニア | SHOWROOM株式会社

209日前

3

1

ツイート

シェア

« 初海外＋英語が喋れないけど、WWDC2019に...

try! Swift Tokyo 2019 に行ってきたお話... »