FMETP STREAM (Guide)

Version: 1.327

Forum: https://forum.unity.com/threads/670270/
Website: http://www.frozenmist.com/
Support: thelghome@gmail.com

FMETP STREAM (Introduction)

FMETP STREAM is a plugin for Unity3D, which aims for sharing your game view, remote assistance and VR Interactive application. In local area network, it can achieve low-latency live streaming between Unity3D apps.

The encode module and network module are totally separated. In the encode module, it includes Visual and Audio encoders. Meanwhile, our package provides few customized networking systems, which are **FM Network UDP**, **FM WebSocket** and **FM Network Action(TCP & UDP)**. Besides those networking systems, our encode module is compatible with 3rd party networking system.

As the source code are developed based on Unity3D C#, it is compatible with most of Unity3D build platforms. For example, it supports iOS, Android, Mac, Windows and Linux. Since FM WebSocket can communicate via the Internet, our package supports Unity3D WebGL Build and HTML website.

In this version, native encoding solution is included in our encode module, which does hugely improve the performance on mobile devices. It solves the bottleneck on low-end devices. For instance, 720P(24fps) Live streaming content runs smoothly on iPhone 5 in our previous testing.

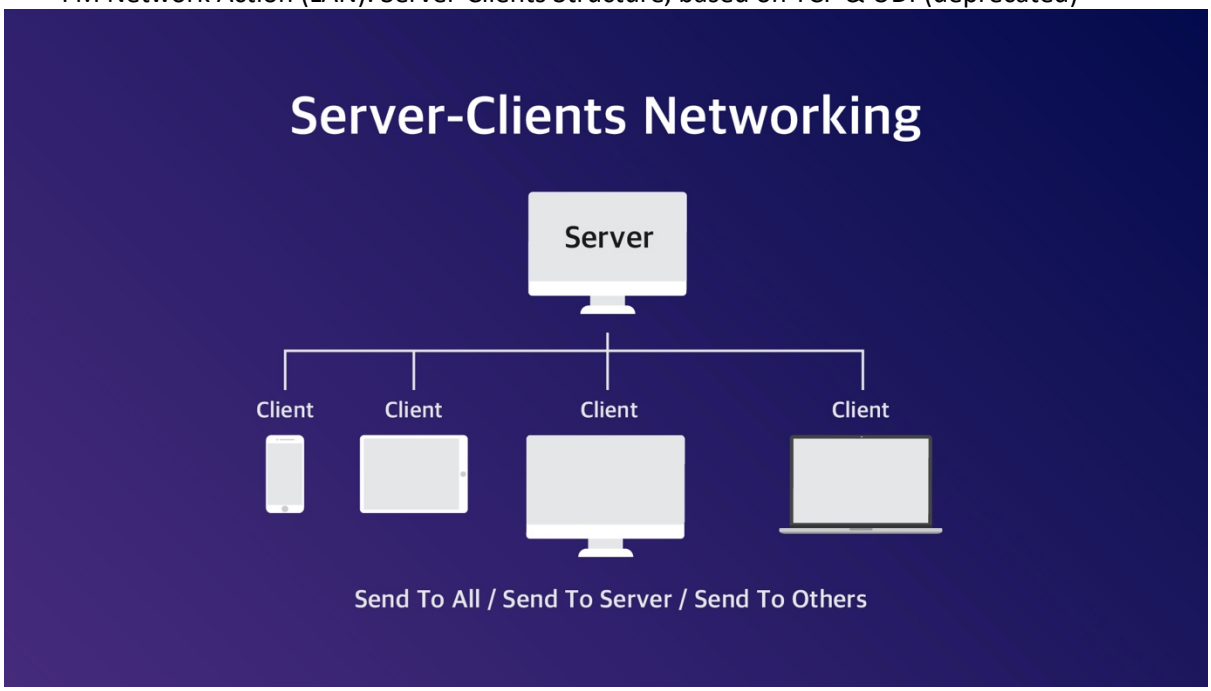
Technically, our plugin can be separated into two parts:
1) Encode Module
2) Network Module

In order to keep the flexibility for users, below is the logic of our core components.

Game View Encoder -> Network Manager -> Game View Decoder
Audio / Mic Encoder -> Network Manager -> Audio Decoder

In our package, we provided 3 different types of networking system.

- FM Network UDP (LAN): Server-Clients Structure, based on UDP
- FM WebSocket (Internet): Server-Clients Structure, based on WebSocket
- FM Network Action (LAN): Server-Clients Structure, based on TCP & UDP(deprecated)



Since the network manager is replaceable, our streaming solution is also compatible with other networking system, for example: UNet, Photon, Mirror…etc

Use Cases:
Share Game View, Video Chat, Remote Assist, Teaching & Learning, Research, VR Monitoring, Streaming for Immersive projection system

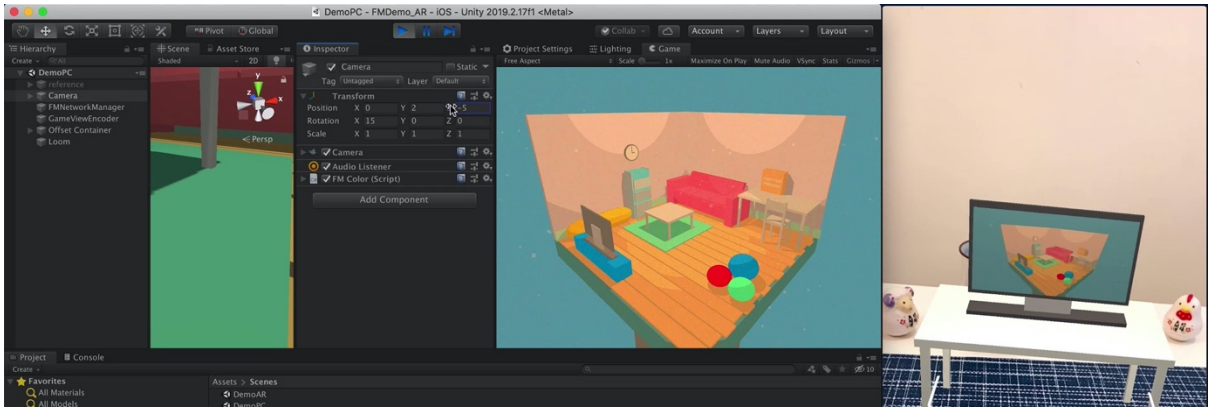
Hardware Requirement(Minimum):
Processor: 1.1 GHz Processor
Memory: 1 GB RAM

Supported OS: iOS, Android, Mac, Windows, Linux

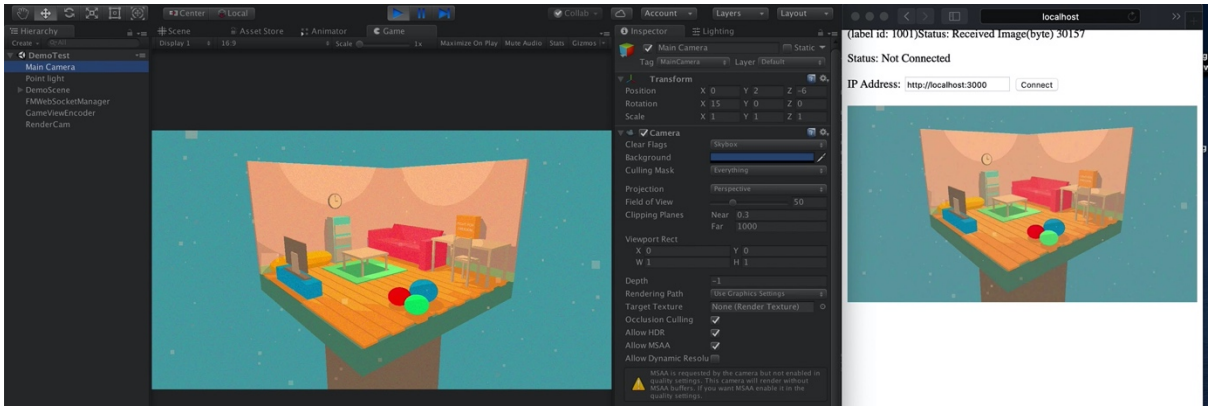
Software Requirement: Unity3D Version 2017.4.32f1 (Older version may also work)

Tested Platform: iOS / Android / Mac / PC / Linux / WebGL / HTML / AR / VR

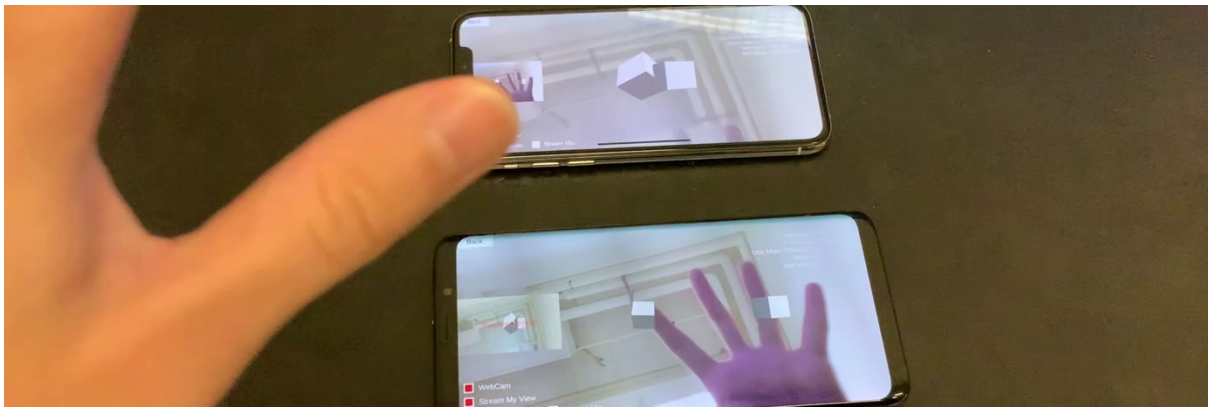
Example: Streaming Game View from Unity3D Editor(Left) to iPhone XS AR world(Right)



Example: Streaming Game View from Unity3D Editor(Left) to Web browser(Right)

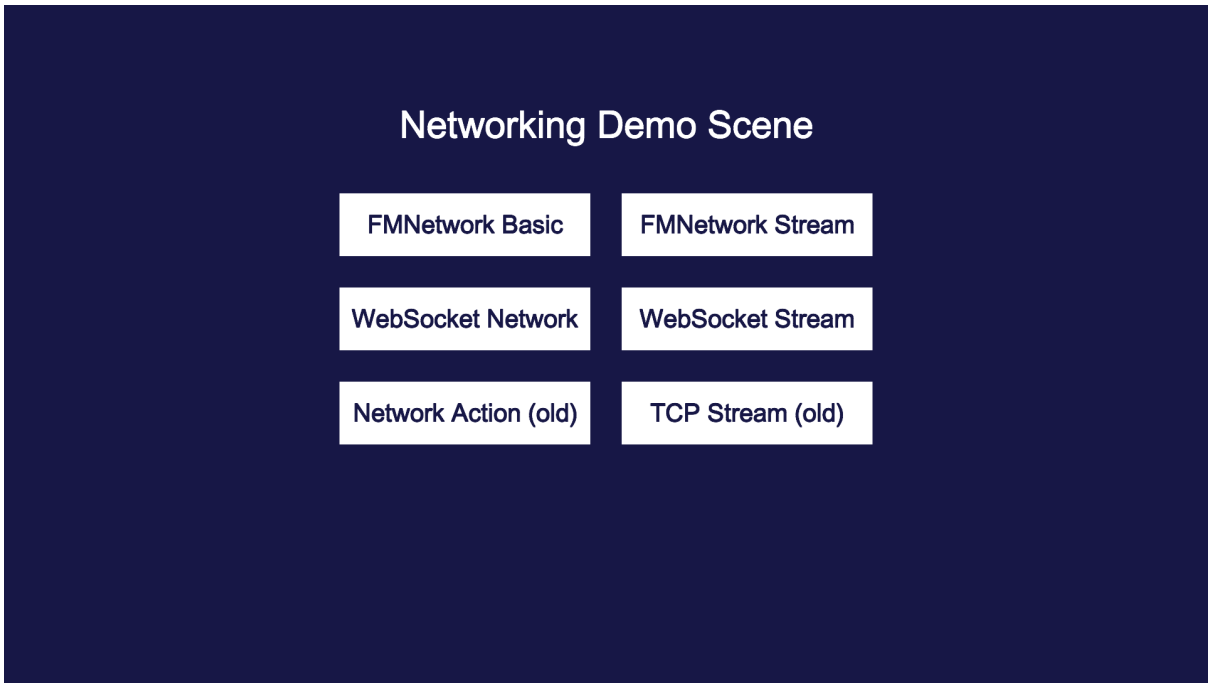


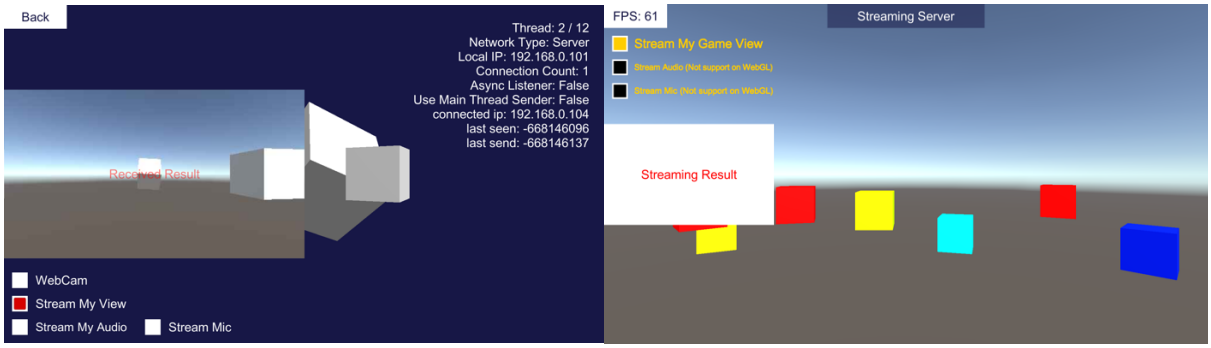
Example: Webcam Streaming between iOS & Android Devices



Demo Scenes are included in our package:

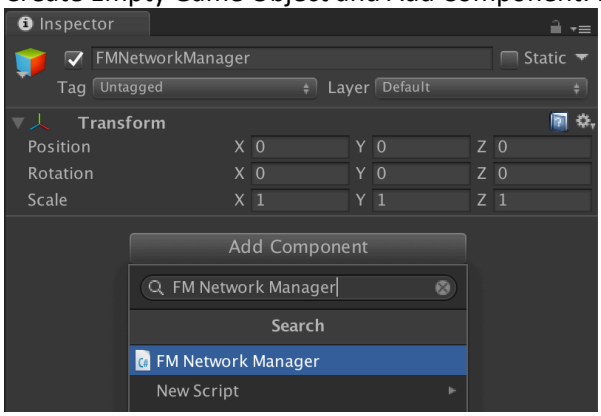
You could open and run demo scene “Demo_NetworkingMain” for your first test.



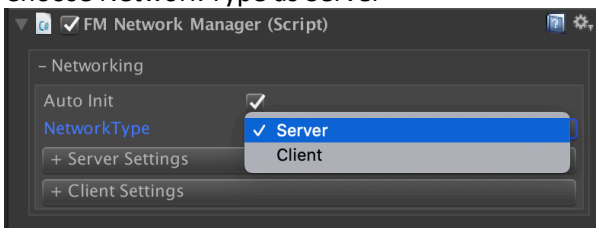


FMETP STREAM (Basic Setup with FM Network UDP: Server Scene)

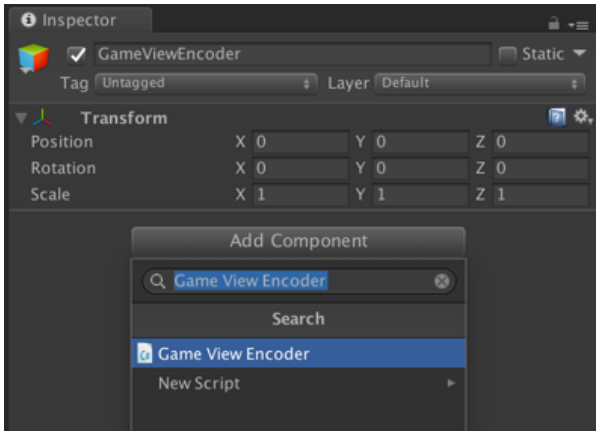
- Create Empty Game Object and Add Component: FM Network Manager



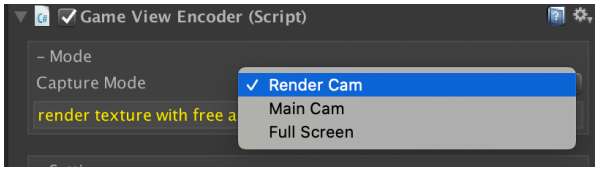
- Choose Network Type as Server



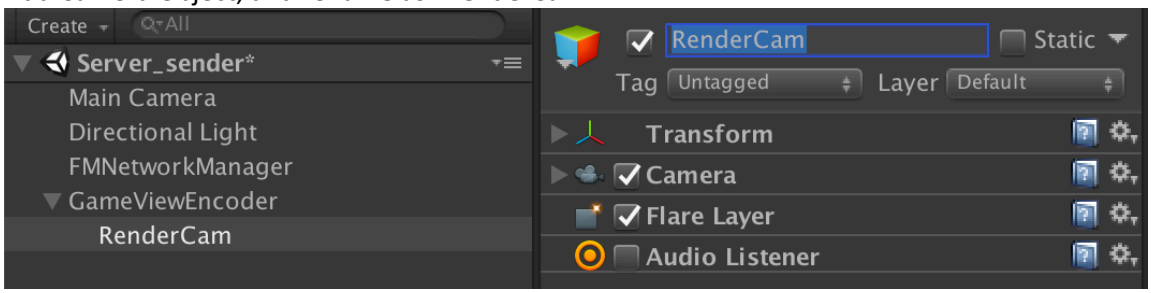
- Create Game View Encoder



- Choose Render Camera Mode

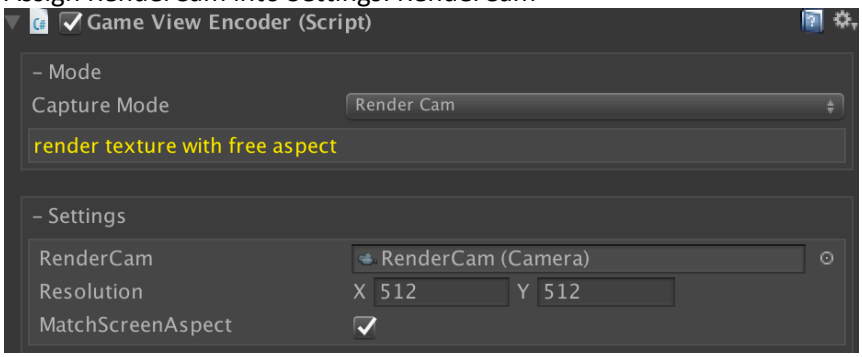


- Add Camera Object, and rename as “RenderCam”

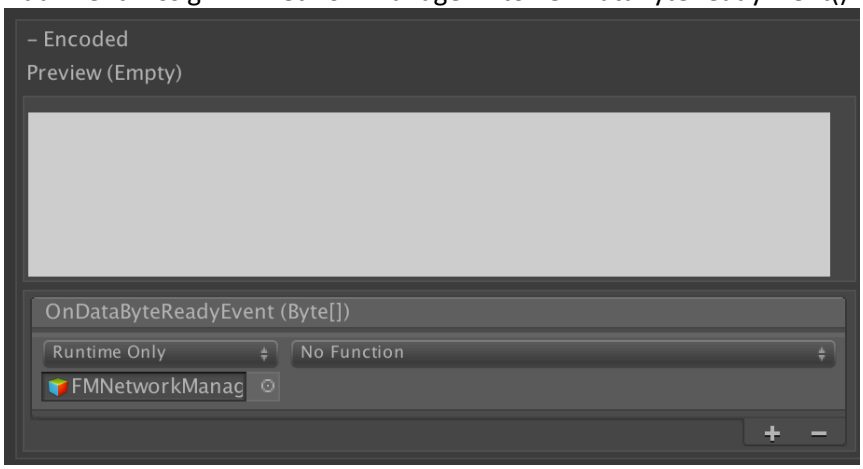


FMETP STREAM (Basic Setup with FM Network UDP: Server Scene)

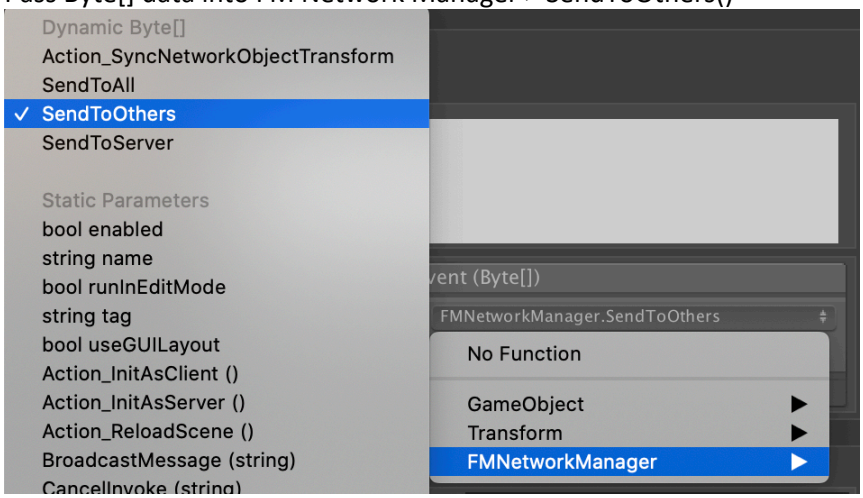
- Assign RenderCam into Settings: RenderCam



- Add Event: Assign FMNetworkManager into “OnDataByteReadyEvent()”

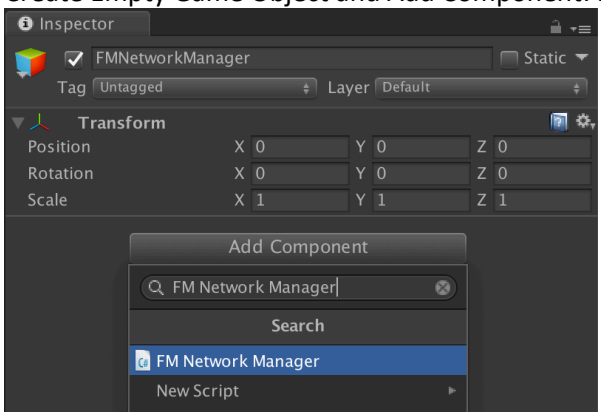


- Pass Byte[] data into FM Network Manager > SendToOthers()

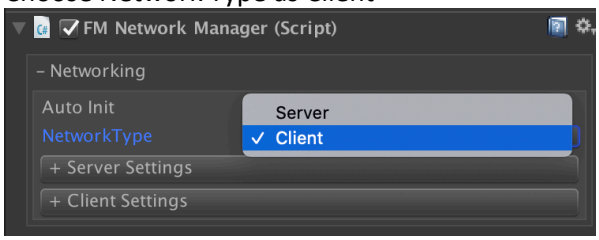


FMETP STREAM (Basic Setup with FMNetworkUDP: Client Scene)

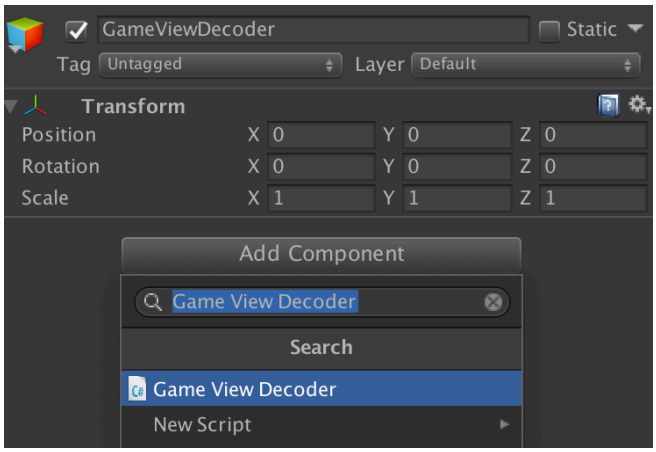
- Create Empty Game Object and Add Component: FM Network Manager

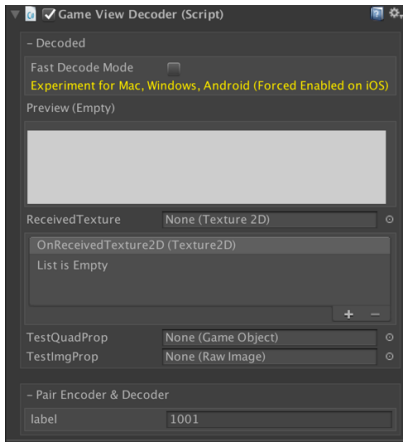


- Choose Network Type as Client

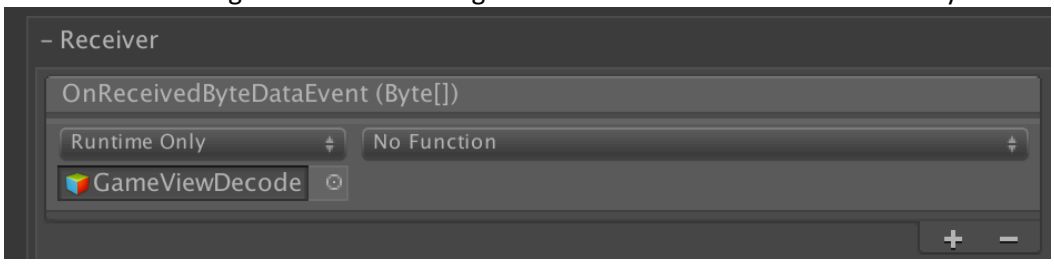


- Create Game View Decoder

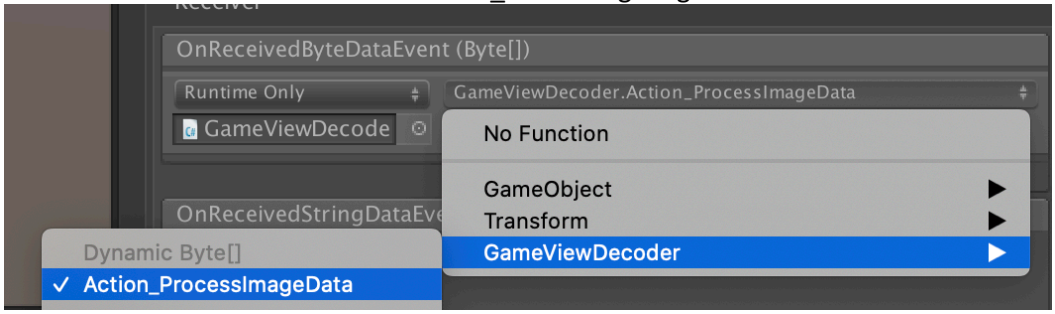




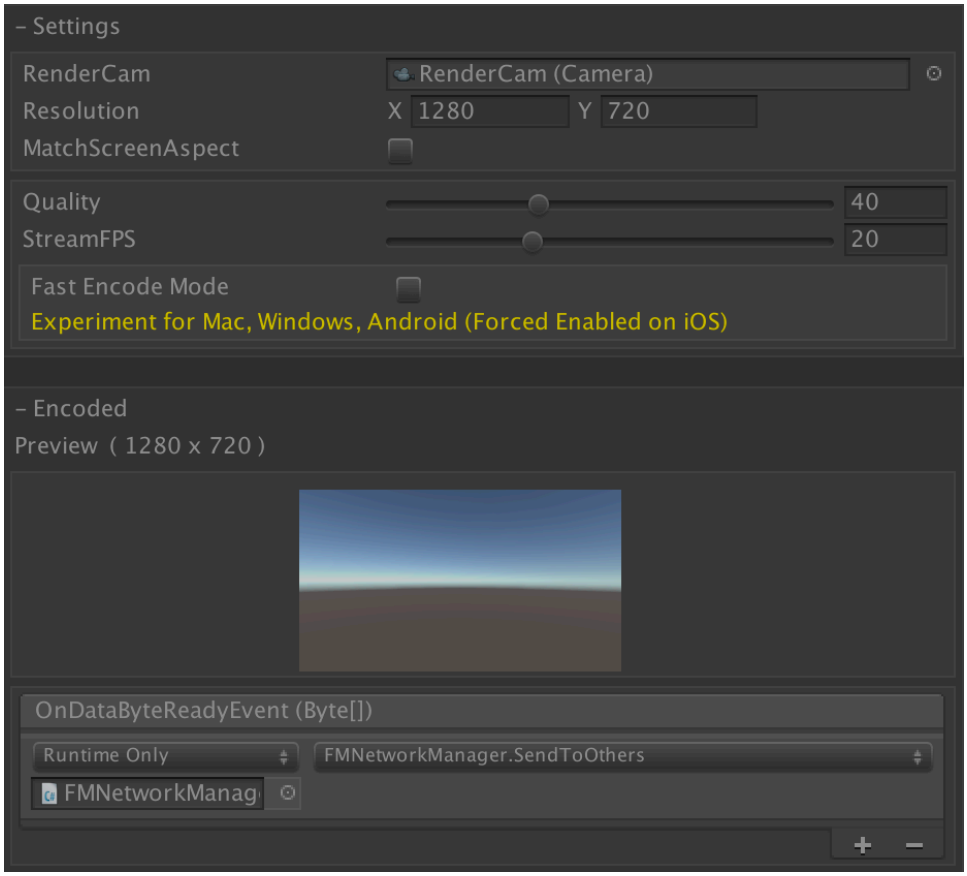
- FM Network Manager > Receiver > Assign Game View Decoder in OnReceivedByteDataEvent()



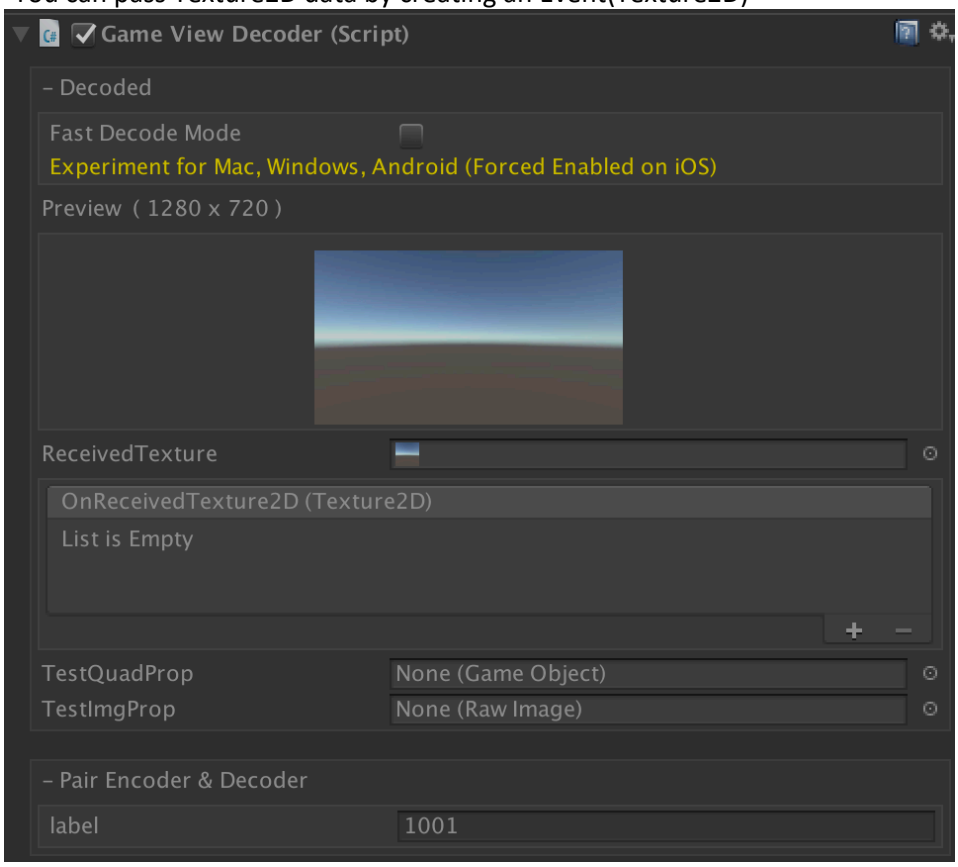
- Pass data to GameViewDecoder > Action_ProcessingImageData



- Result on Game View Encoder

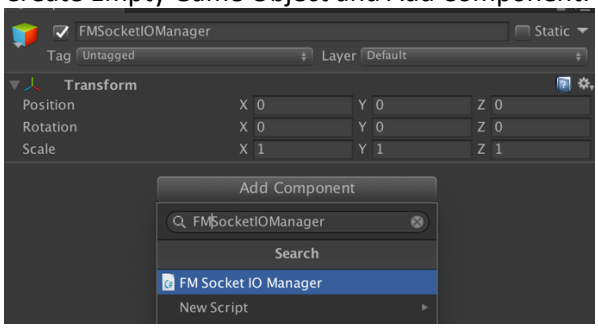


- Result on GameViewDecoder Inspector
-You can pass Texture2D data by creating an Event(Texture2D)

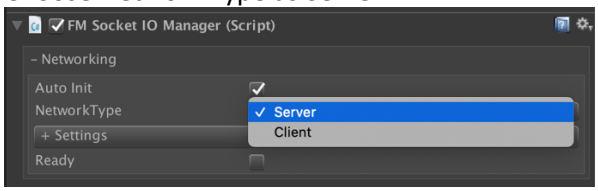


FMETP STREAM (Basic Setup with FM WebSocket: Server Scene)

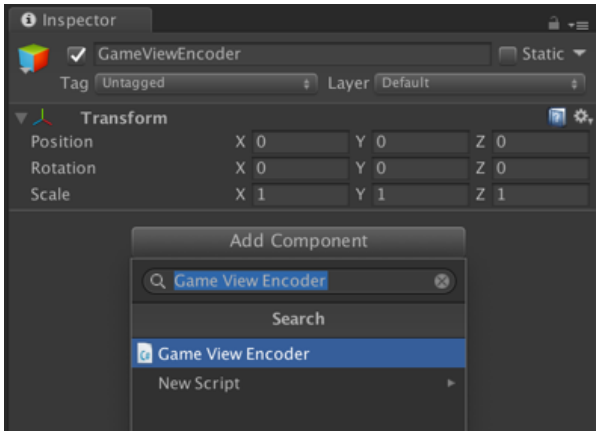
- Create Empty Game Object and Add Component: FMSocketIOManager



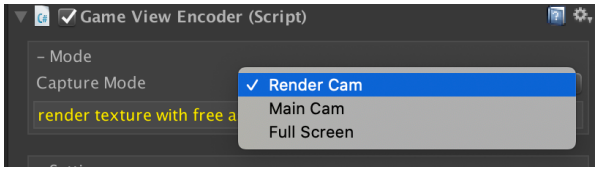
- Choose Network Type as Server



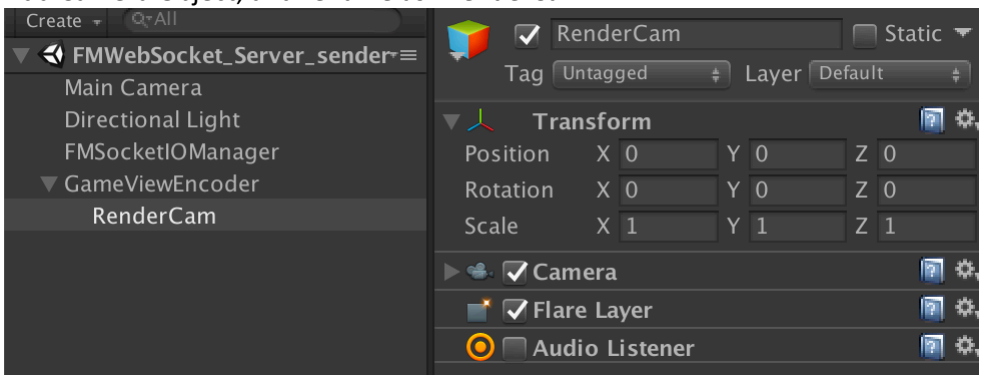
- Create Game View Encoder



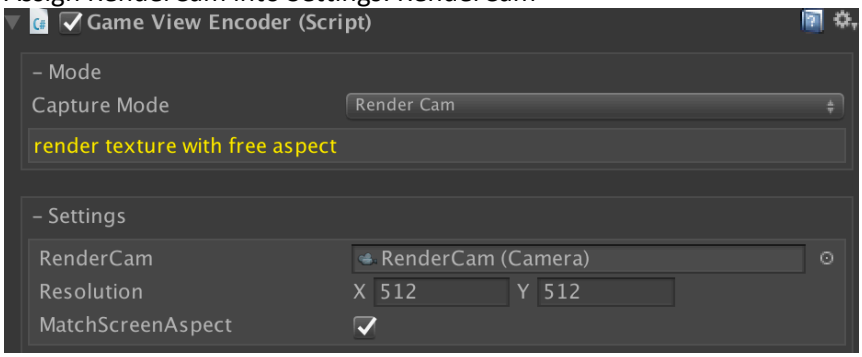
- Choose Render Camera Mode



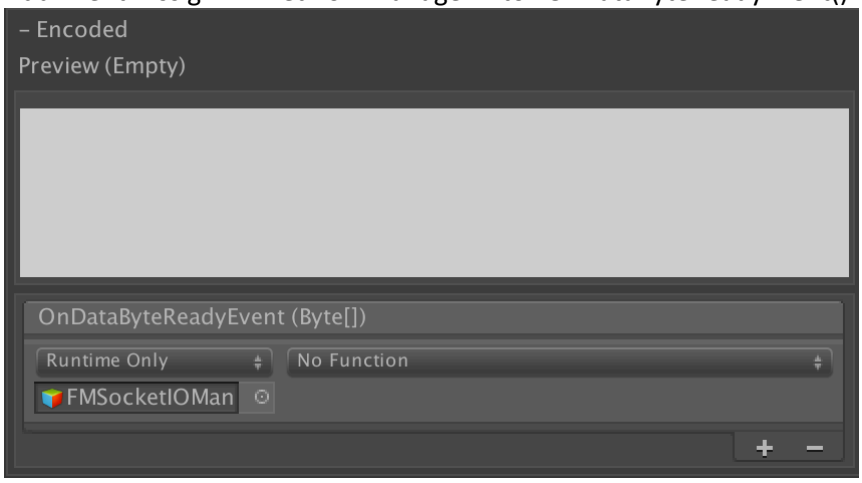
- Add Camera Object, and rename as “RenderCam”



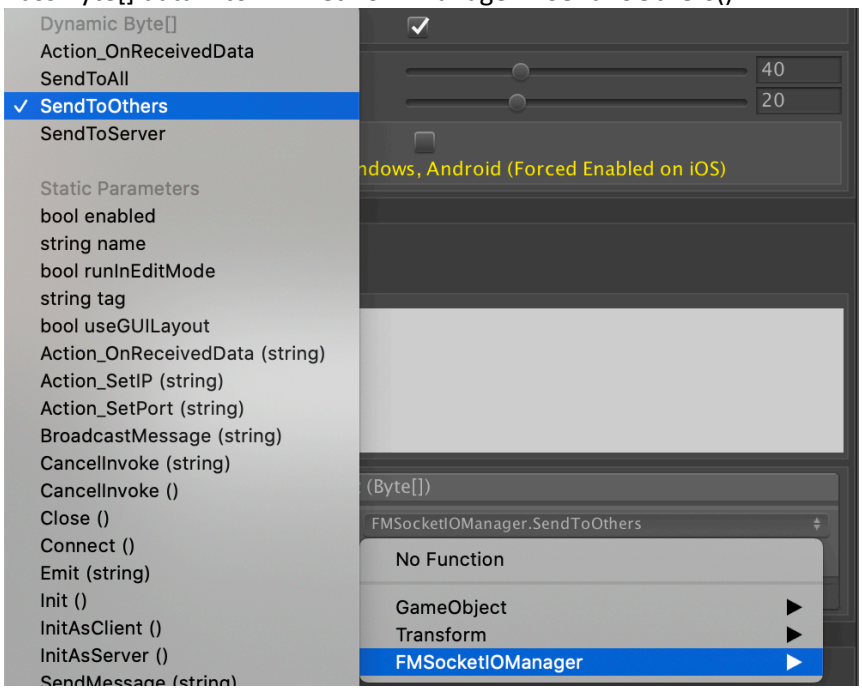
- Assign RenderCam into Settings: RenderCam



- Add Event: Assign FMNetworkManager into “OnDataByteReadyEvent()”

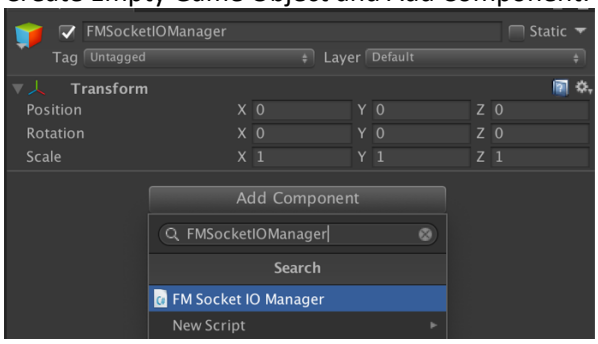


- Pass Byte[] data into FM Network Manager > SendToOthers()

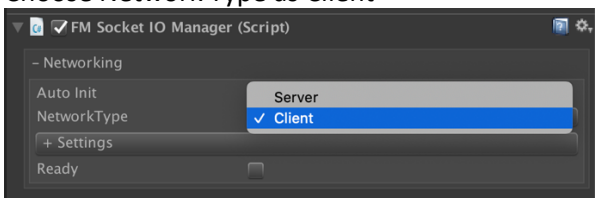


FMETP STREAM (Basic Setup with FM WebSocket: Client Scene)

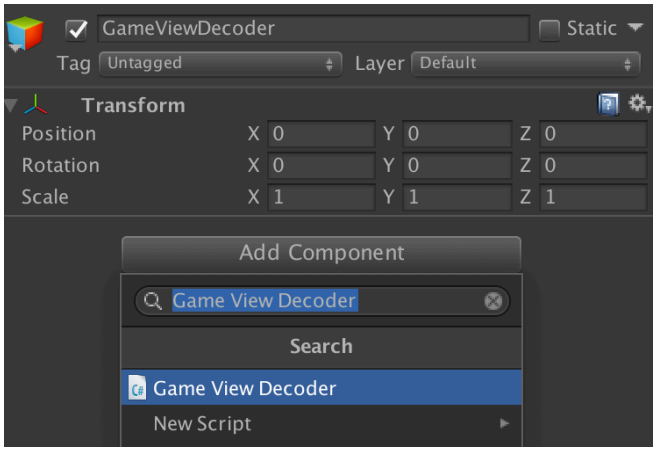
- Create Empty Game Object and Add Component: FMNetworkManager

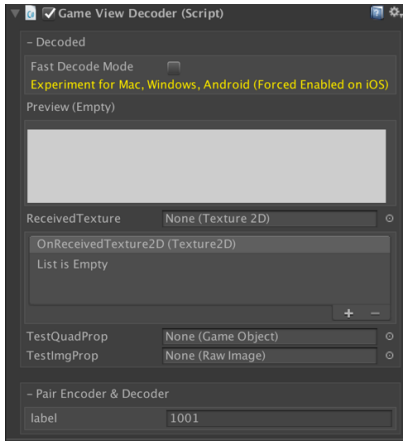


- Choose Network Type as Client

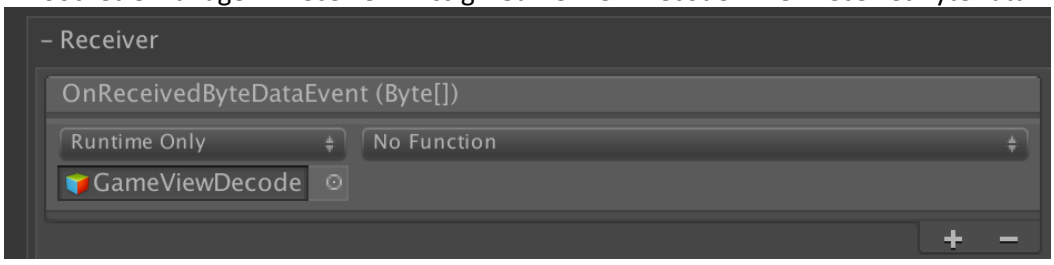


- Create Game View Decoder

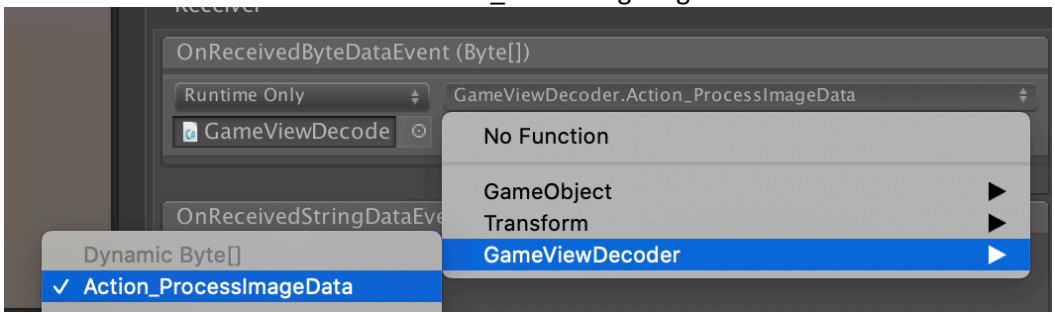




- FMSocketIOManager > Receiver > Assign Game View Decoder in OnReceivedByteDataEvent()



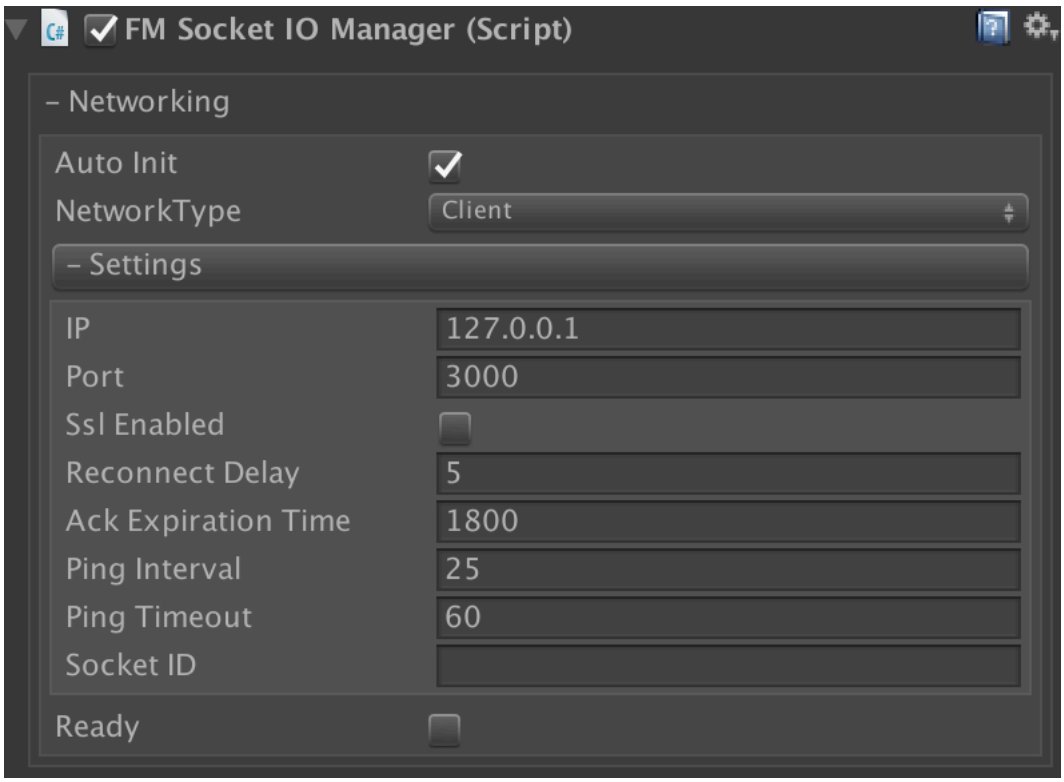
- Pass data to GameViewDecoder > Action_ProcessingImageData



FMETP STREAM (Basic Setup with FM WebSocket: Node.js Server)

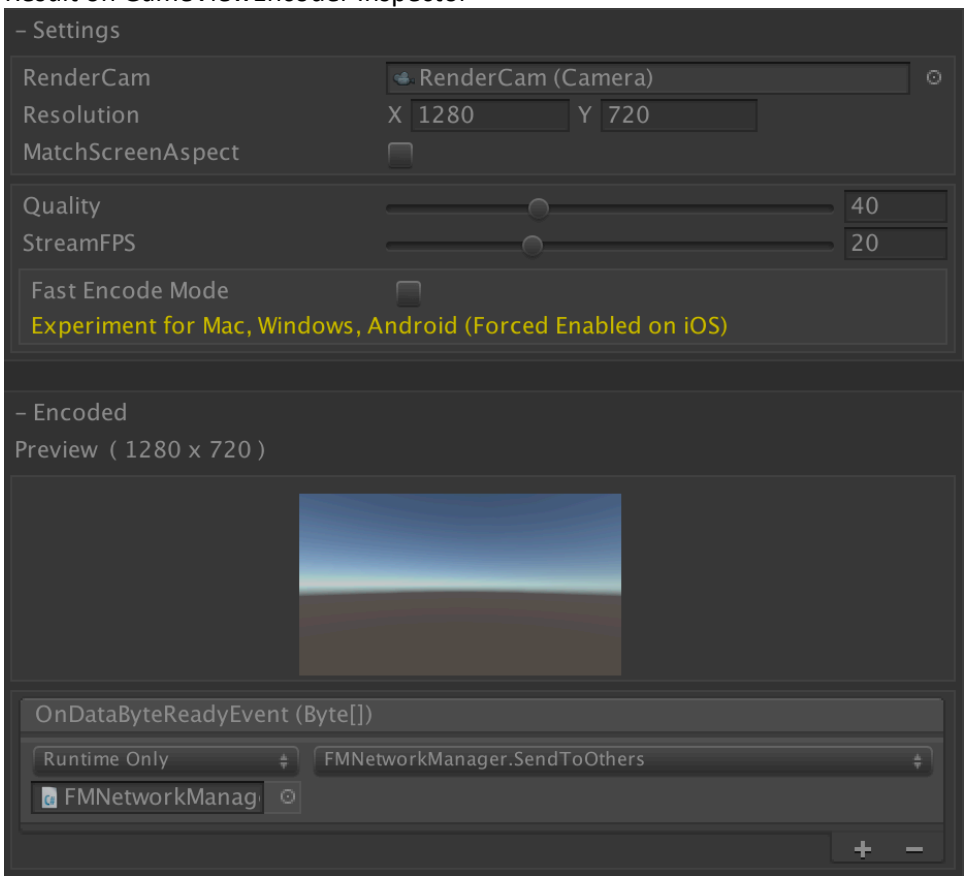
- 1) Install npm + node.js
  - download and install all necessary components: https://nodejs.org/en/download/
- 2) install socket io
  - open terminal/cmd and type:
  npm install socket.io
- 3) Install express
  - open terminal/cmd and type:
  npm init
  //press Enter…
  npm install express --save
- 4) Finish & Test on localhost: demo server is in FMWebSocket/"TestServer.zip"
  - Copy & Unzip the demo server into other location, cannot be in Asset folder
  - open terminal/cmd and type:
  node /[path]/index.js
- 5) IP & Port of node.js server should match the settings in FMSocketIOManager.

- Step-by-step Video Tutorial: https://youtu.be/Zjm5KGHyceU

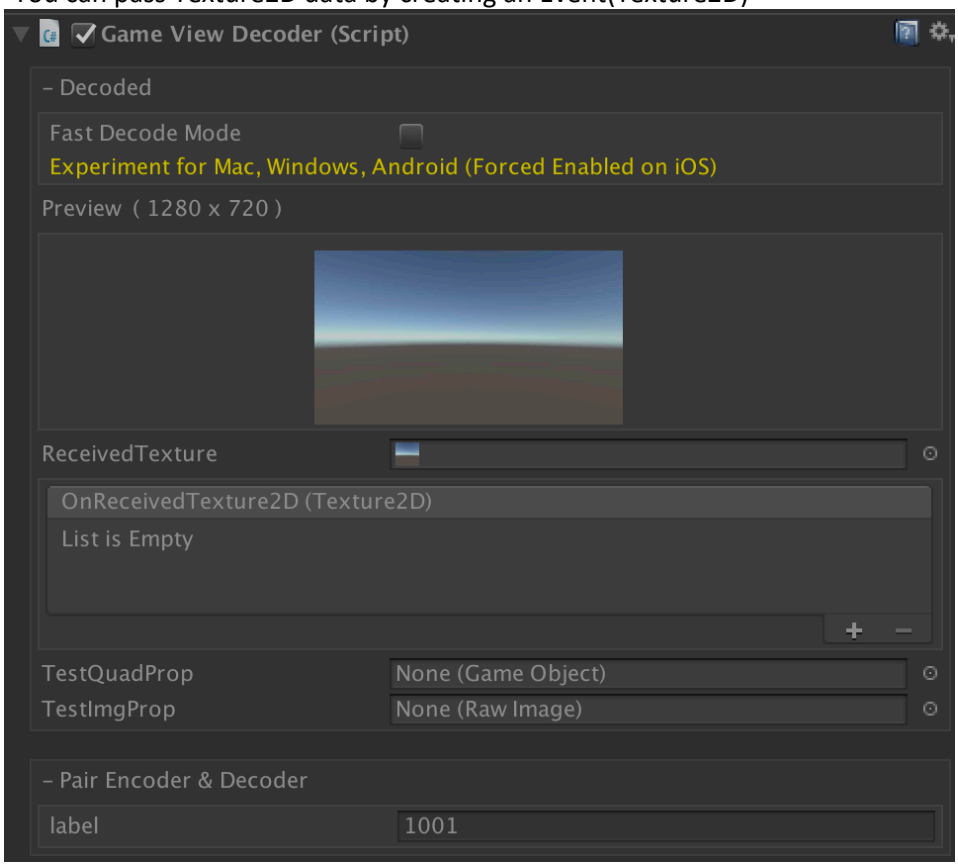


FMETP STREAM (Basic Setup with FM WebSocket: Result)

- Result on GameViewEncoder Inspector

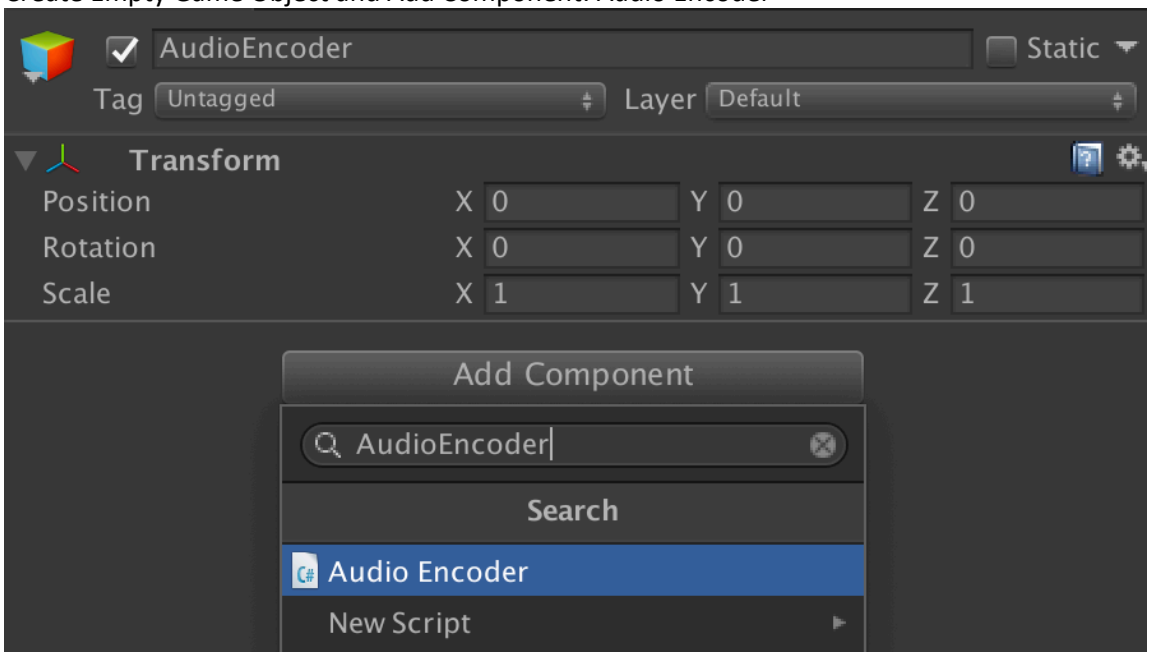


- Result on GameViewDecoder Inspector
  -You can pass Texture2D data by creating an Event(Texture2D)

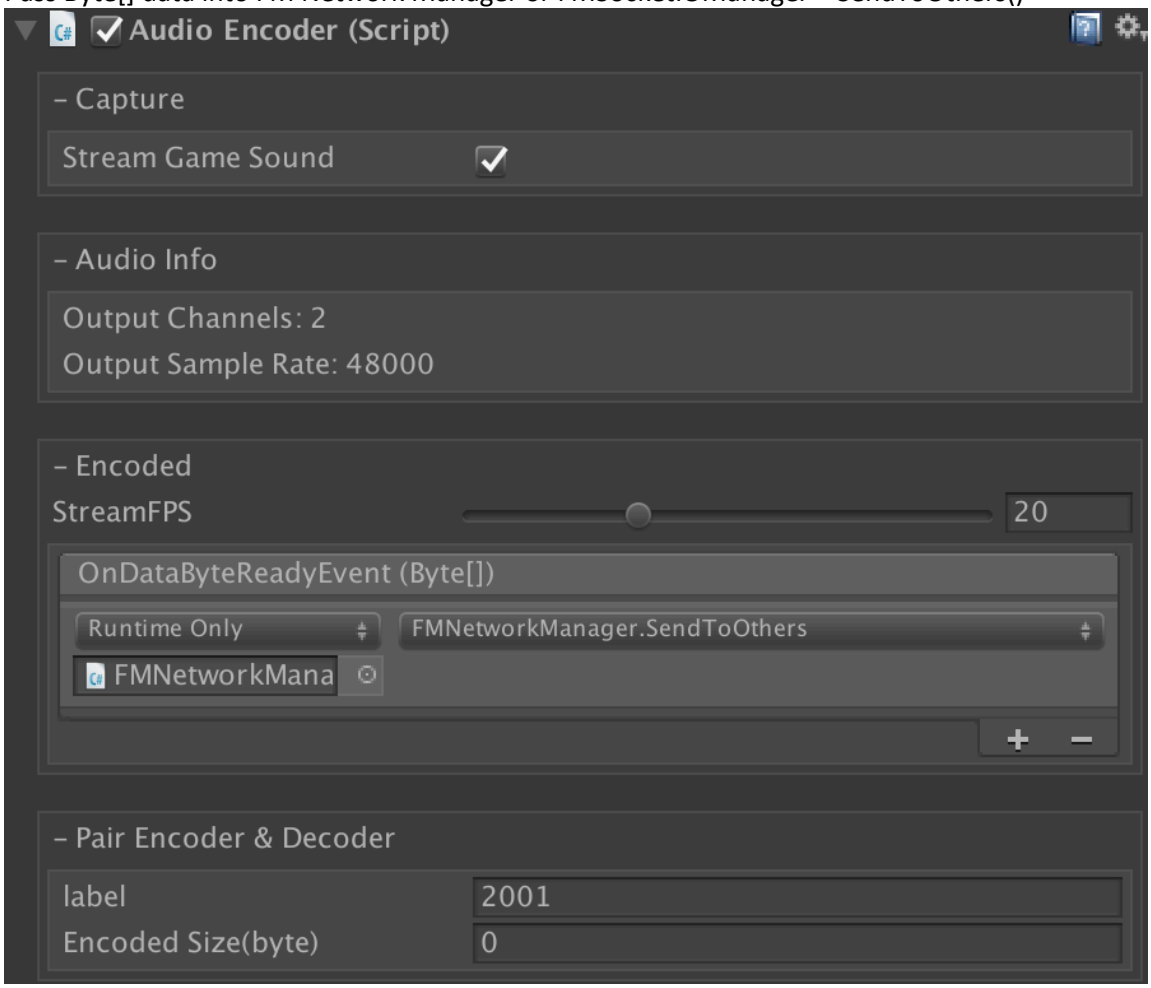


FMETP STREAM (Audio Streaming)

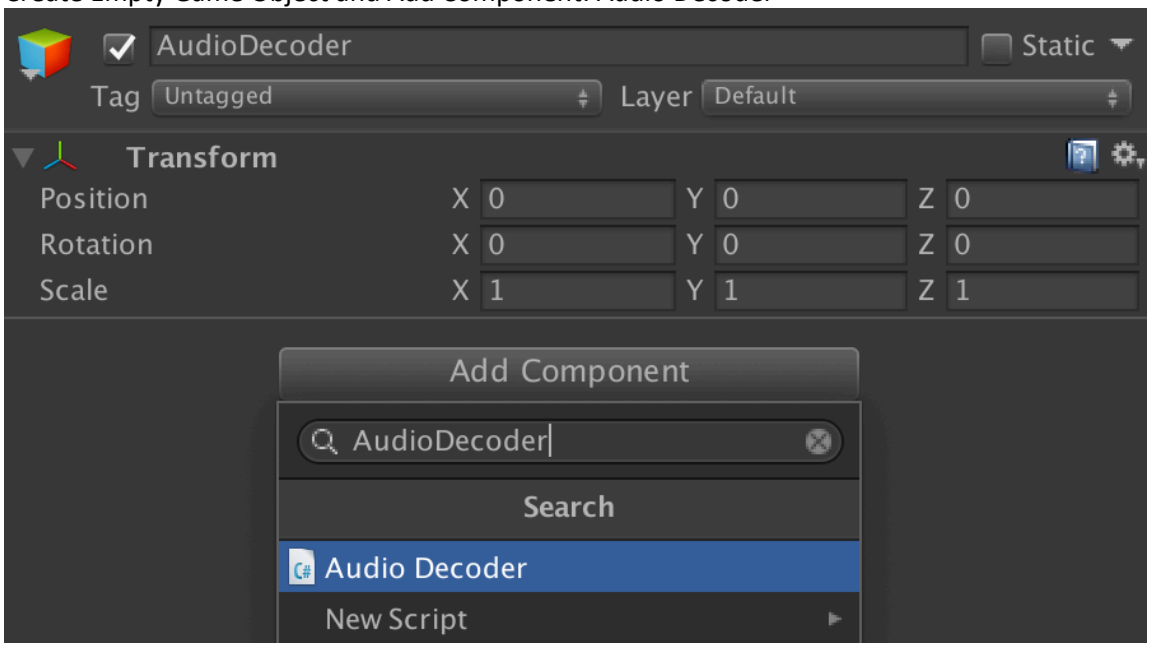
- Create Empty Game Object and Add Component: Audio Encoder



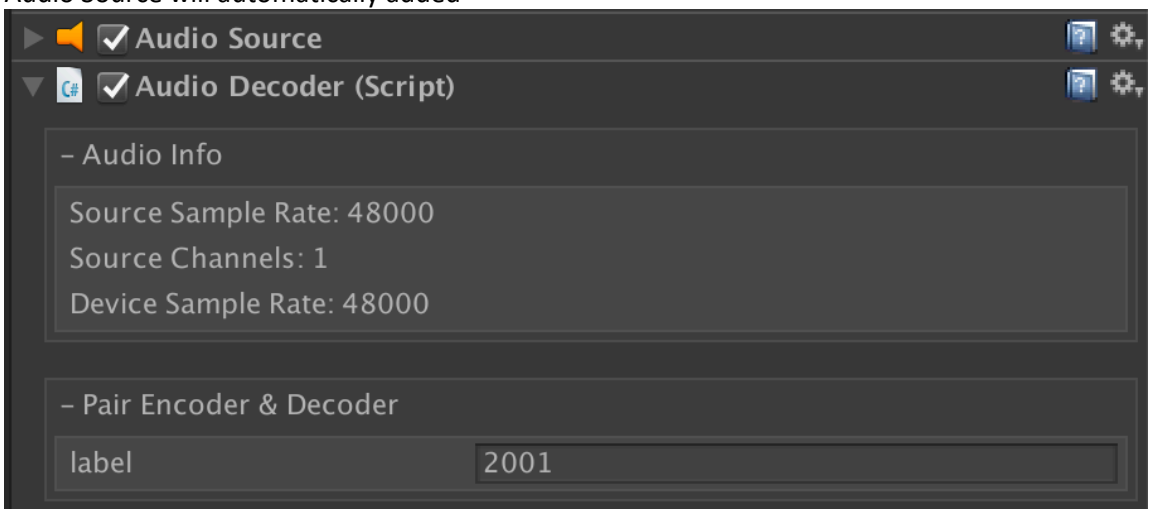
- Assign FMNetworkManager or FMSocketIOManager into “OnDataByteReadyEvent()”
  Pass Byte[] data into FM Network Manager or FMSocketIOManager > SendToOthers()



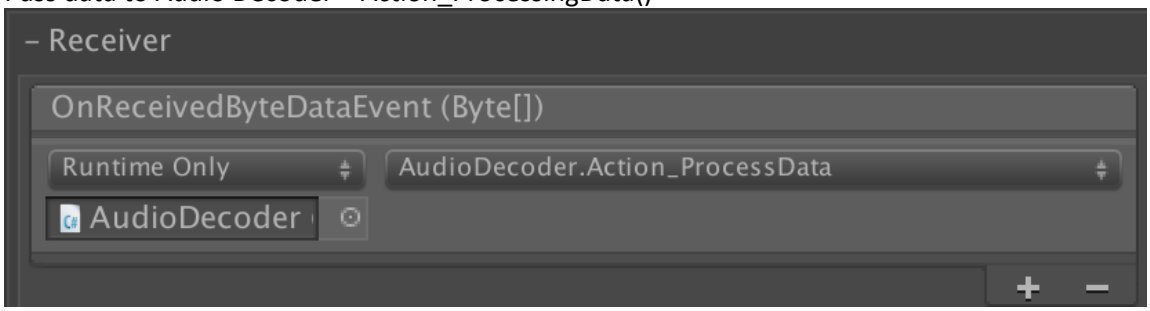
- Create Empty Game Object and Add Component: Audio Decoder



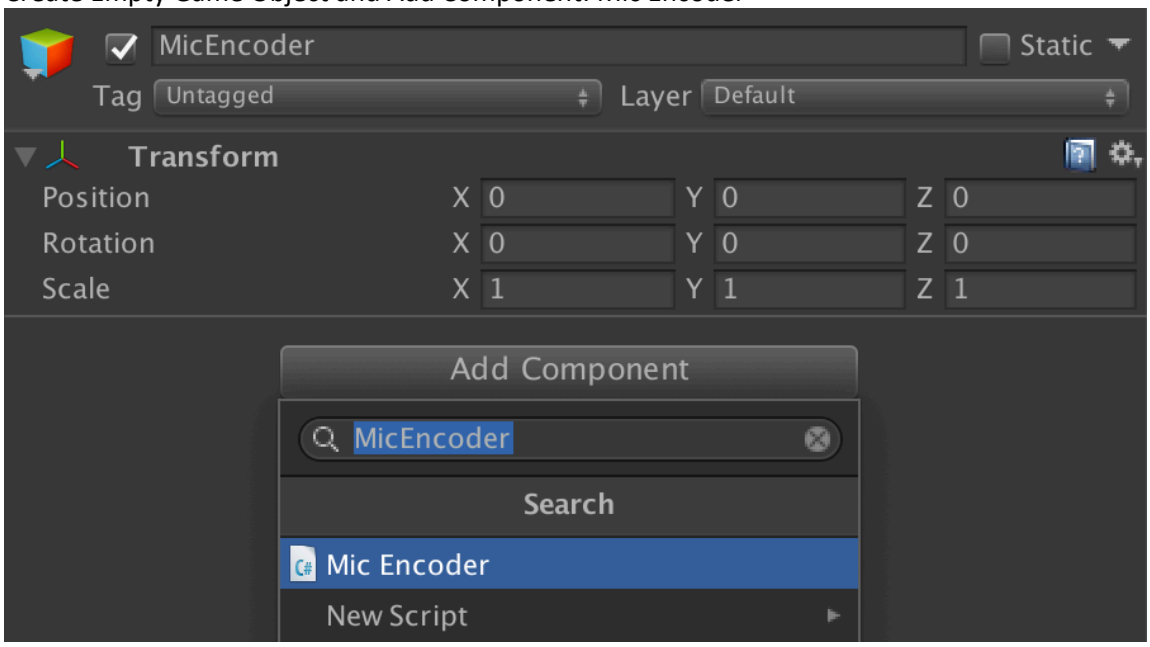
- Audio Source will automatically added



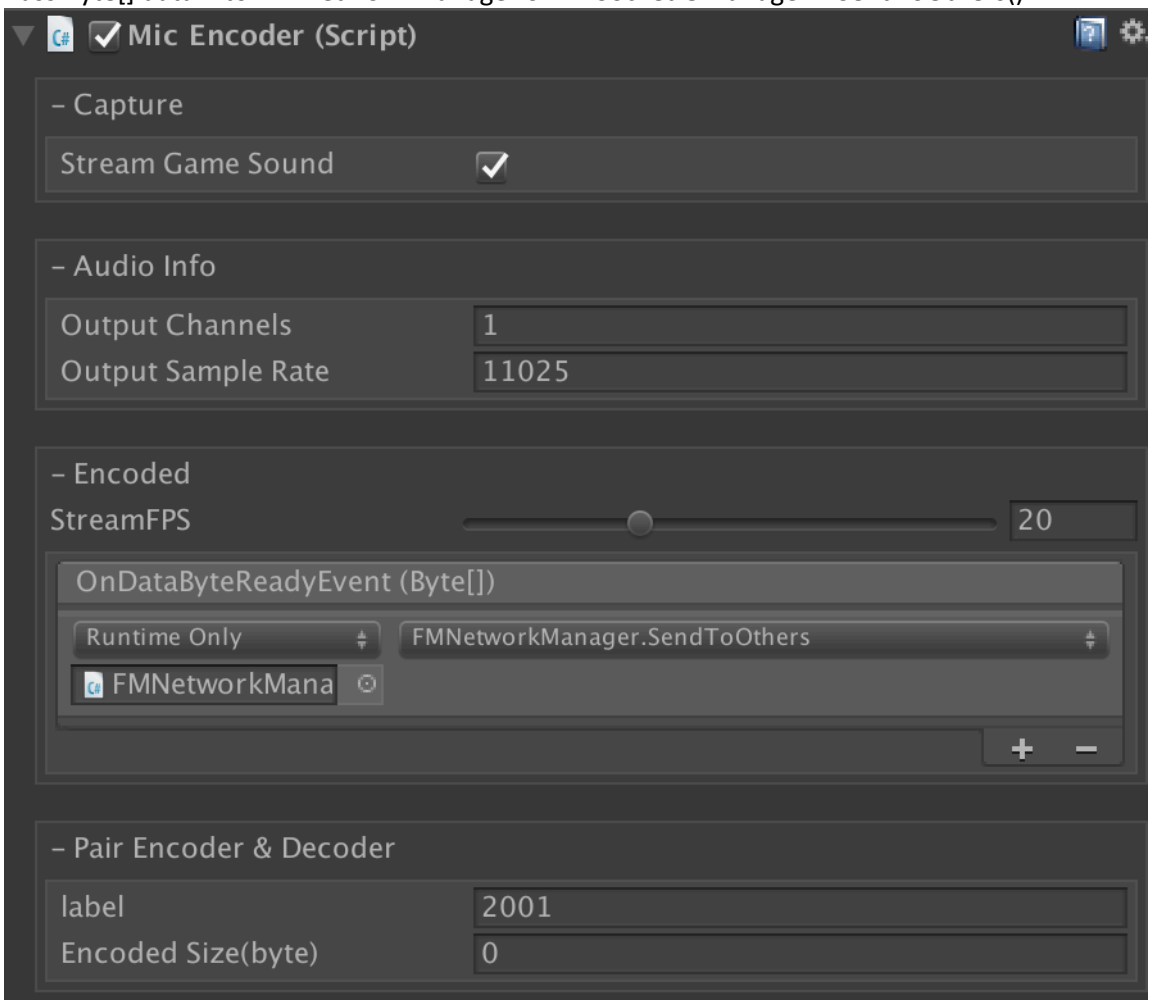
- In FMNetworkManager or FMSocketIOManager:
  Pass data to Audio Decoder > Action_ProcessingData()



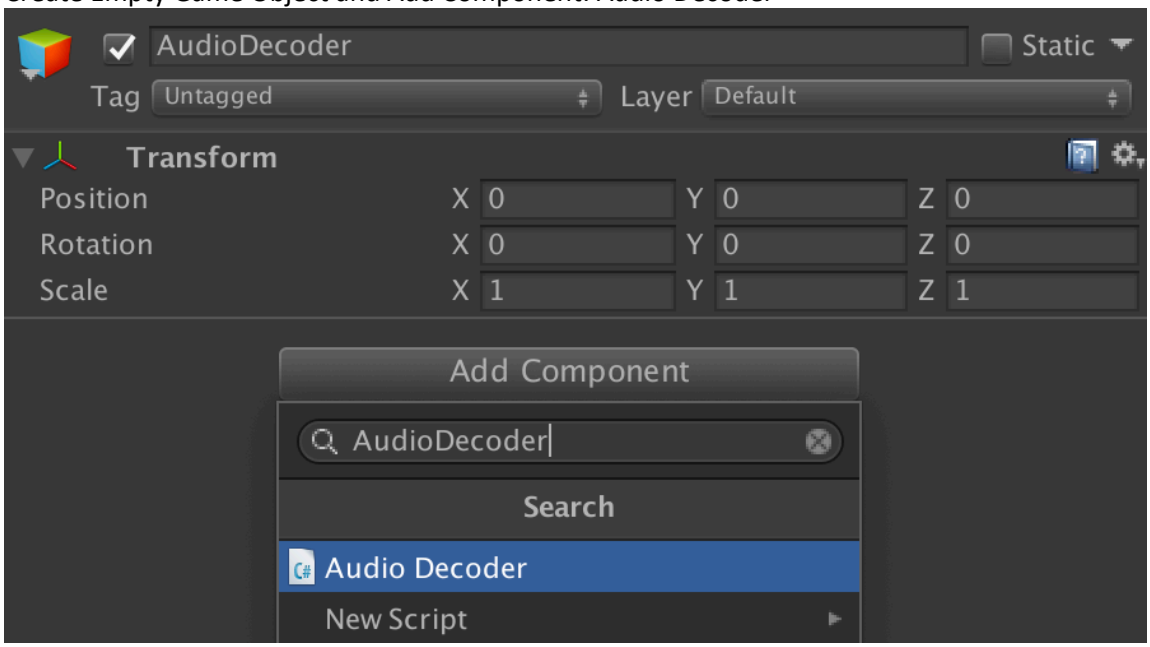
- Create Empty Game Object and Add Component: Mic Encoder



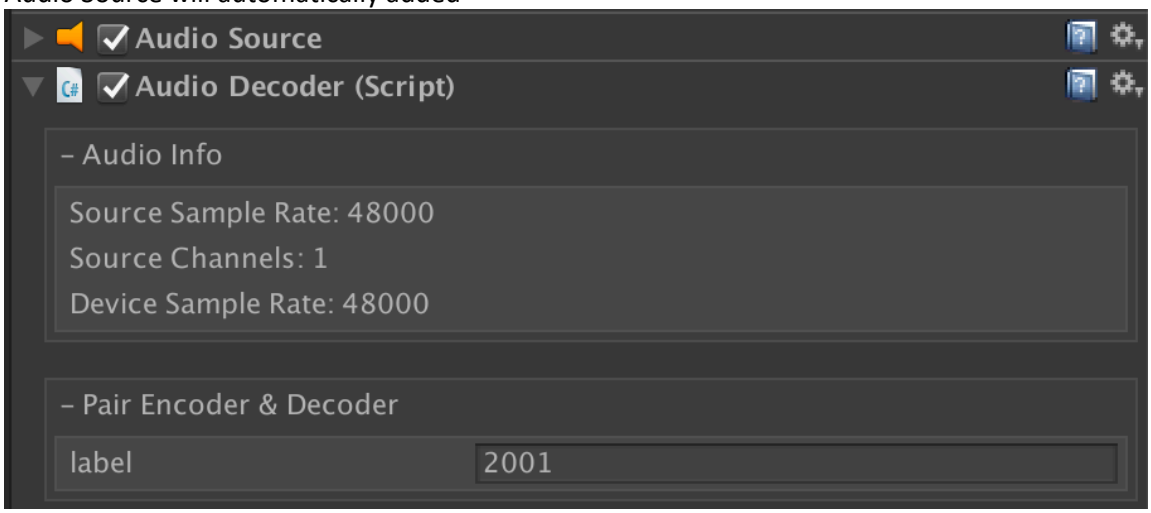
- Assign FMNetworkManager or FMSocketIOManager into “OnDataByteReadyEvent()”
  Pass Byte[] data into FM Network Manager or FMSocketIOManager > SendToOthers()



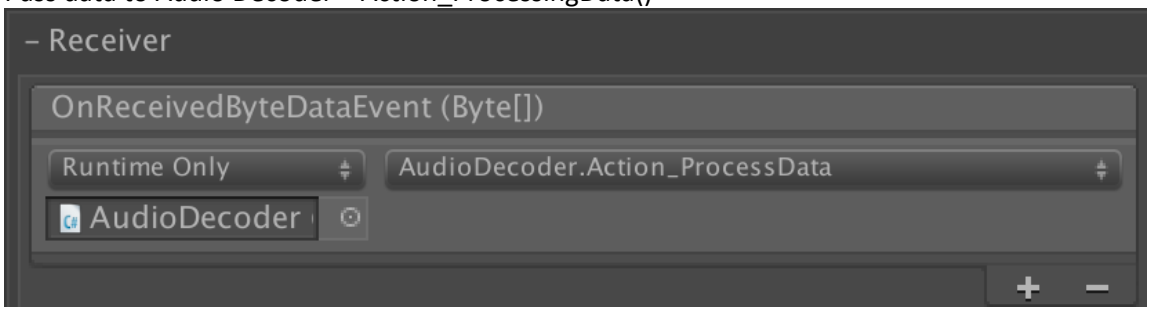
- Create Empty Game Object and Add Component: Audio Decoder



- Audio Source will automatically added



- In FMNetworkManager or FMSocketIOManager:
  Pass data to Audio Decoder > Action_ProcessingData()



FMETP STREAM (Properties in Core Scripts)

Game View Encoder

- Capture Mode: Different methods of capturing game view
- Resolution: streaming resolution
- Quality: streaming quality
- StreamFPS: streaming framerate
- OnDataByteReadyEvent(byte[]): invoke when the streaming bytes are ready
- Label: Pair Encoder & Decoder

Game View Decoder

- Received Texture: decoded Texture2D
- OnReceivedTexture2D(Texture2D): invoke when the received Texture2D is ready
- Label: Pair Encoder & Decoder

Audio Encoder

- Stream Game Sound(bool): capture and stream audio when enable
- Audio Info: information (Output Channels, Output Sample Rate)
- StreamFPS: streaming framerate
- OnDataByteReadyEvent(byte[]): invoke when the streaming bytes are ready
- Label: Pair Encoder & Decoder

Audio Decoder

- Audio Source will be added automatically after adding Audio Decoder
- Audio Info: information (Source Channels, Source Sample Rate, Device Sample Rate)
- Label: Pair Encoder & Decoder

FMETP STREAM (Properties in Core Scripts)

FM Network Manager

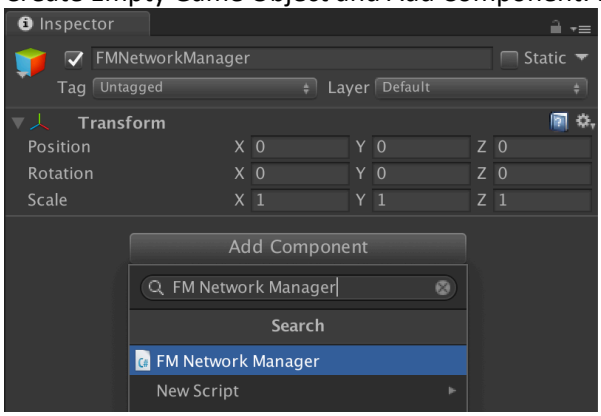
- Auto Init: auto initialize FM Network when enable
  - other functions: Action_InitAsServer(), Action_InitAsClient()
- Network Type: Server/Client
- Server Settings:
  - Server Listen Port: default is 3333
  - Use Async Listener: if disabled, a separated thread will be used
  - Use Main Thread Sender: if disabled, a separated thread will be used
  - Connection Count: the number of connected clients
- Client Settings:
  - Client Listen Port: default is 3334
  - Use Main Thread Sender: if disabled, a separated thread will be used
  - Auto Network Discovery: if disabled, Server IP is required
  - Server IP: target Server IP, or connected Server IP
  - Is Connected: status of connection
- Network Objects
  - Sync transformation of game objects, with the order of array
  - Sync FPS: framerate of syncing via network
- Receiver:
  - OnReceivedByteDataEvent(Byte[]): invoke when received byte data
  - OnReceivedStringDataEvent(String): invoke when received string data
  - GetRawReceivedData(Byte[]): invoke when received any data
- Other functions & Example:
  - FMNetworkManager.instance.SendToAll()
  - FMNetworkManager.instance.SendToOthers()
  - FMNetworkManager.instance.SendToServer()
  - FMNetworkManager.instance.SendToTarget()

FM Socket IO Manager

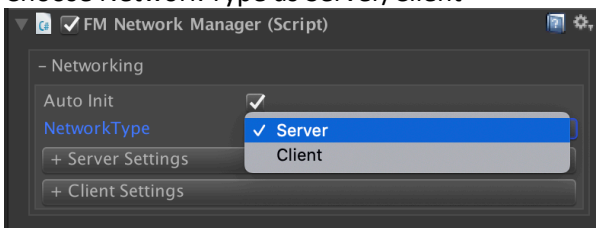
- Auto Init: auto initialize FM Network when enable
  - other functions: Action_InitAsServer(), Action_InitAsClient()
- Network Type: Server/Client
- Settings:
  - IP: IP address of WebSocket server(node.js server)
  - Port: default is 3000
  - Ssl Enabled: true for “https” or “wss”, false for “http” or “ws”
- Ready: true when connected to server
- Receiver:
  - OnReceivedByteDataEvent(Byte[]): invoke when received byte data
  - OnReceivedStringDataEvent(String): invoke when received string data
- Other functions & Example:
  - FMSocketIOManager.instance.SendToAll()
  - FMSocketIOManager.instance.SendToOthers()
  - FMSocketIOManager.instance.SendToServer()

FMETP STREAM (FM Network UDP: Setup & Example)

- Create Empty Game Object and Add Component: FM Network Manager



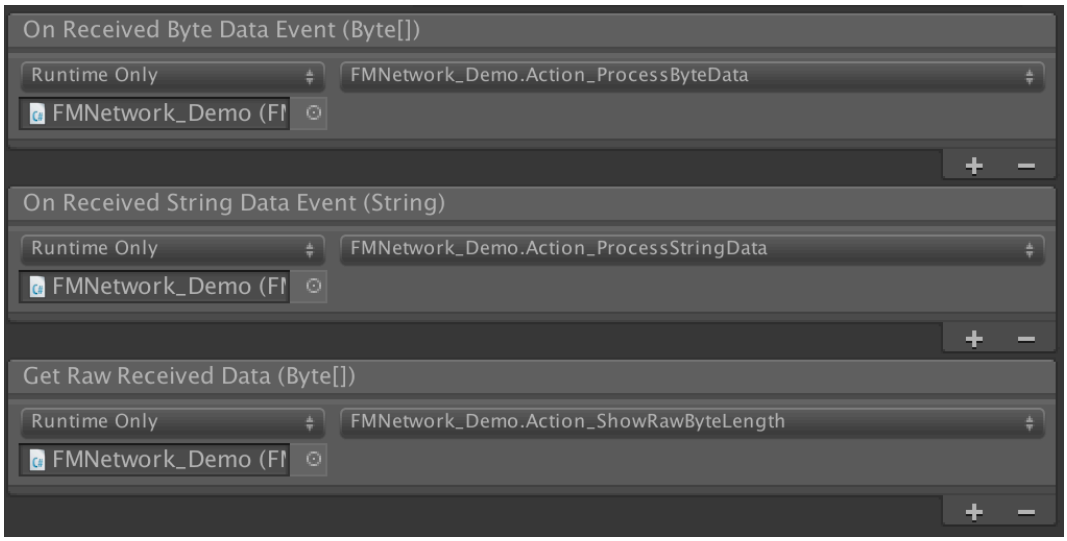
- Choose Network Type as Server/Client



- Example functions for sending Data: Send to (All / Others / Server / Target)

```
string Message = "Hello World!";
FMNetworkManager.instance.SendToAll(Message);
FMNetworkManager.instance.SendToOthers(Message);
FMNetworkManager.instance.SendToServer(Message);
FMNetworkManager.instance.SendToTarget(Message, "127.0.0.1");

byte[] ByteData = new byte[1234];
FMNetworkManager.instance.SendToAll(ByteData);
FMNetworkManager.instance.SendToOthers(ByteData);
FMNetworkManager.instance.SendToServer(ByteData);
FMNetworkManager.instance.SendToTarget(ByteData, "127.0.0.1");
```

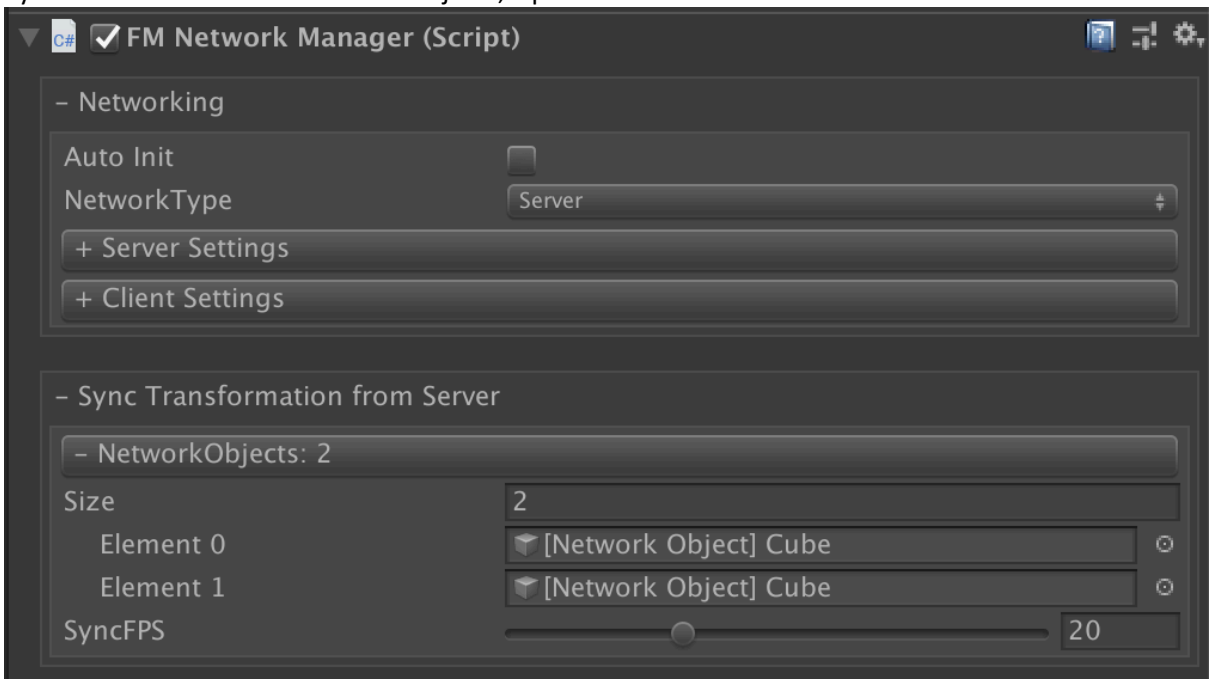
- Events will be invoked when received data



- Example functions of processing received data

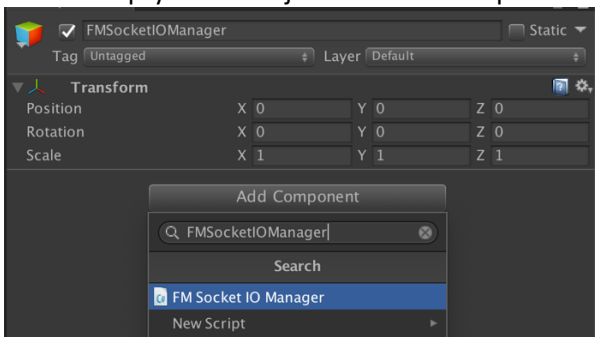
```
public void Action_ProcessStringData(string _string)
{
    if(FMNetworkManager.instance.NetworkType == FMNetworkType.Server)
    {
        if (ServerText != null) ServerText.text = "Server Received : " + _string;
    }
    else
    {
        if (ClientText != null) ClientText.text = "Client Received : " + _string;
    }
}

public void Action_ProcessByteData(byte[] _byte)
{
    if (FMNetworkManager.instance.NetworkType == FMNetworkType.Server)
    {
        if (ServerText != null) ServerText.text = "Server Received byte[] : " + _byte.Length;
    }
    else
    {
        if (ClientText != null) ClientText.text = "Client Received byte[]: " + _byte.Length;
    }
}
```

- Sync the transformation of Game Objects, Option

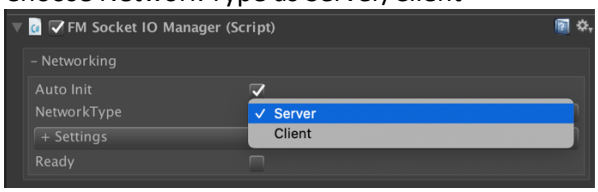


FM Network Manager

- Auto Init: auto initialize FM Network when enable
  - other functions: Action_InitAsServer(), Action_InitAsClient()
- Network Type: Server/Client
- Create Empty Game Object and Add Component: FMNetworkManager



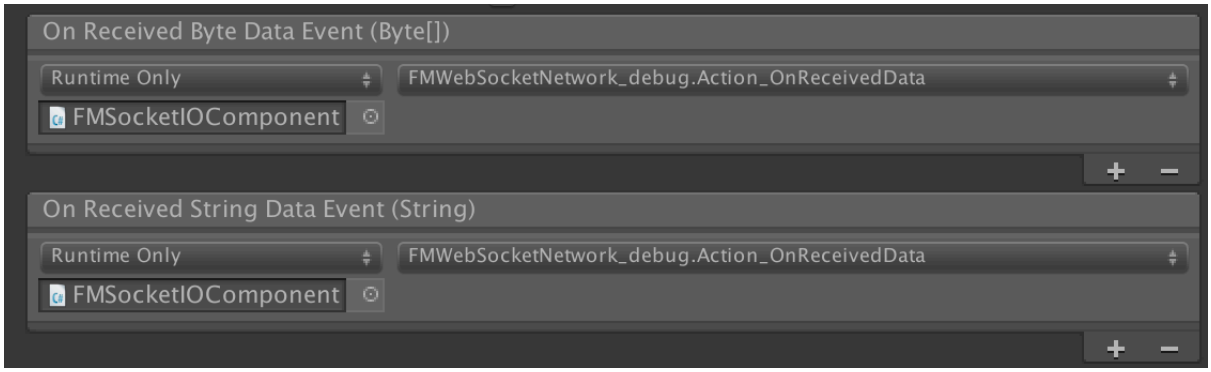
- Choose Network Type as Server/Client



- Example functions for sending Data: Send to (All / Others / Server)

```
string Message = "Hello World!";
FMSocketIOManager.instance.SendToAll(Message);
FMSocketIOManager.instance.SendToServer(Message);
FMSocketIOManager.instance.SendToOthers(Message);

byte[] ByteData = new byte[1234];
FMSocketIOManager.instance.SendToAll(ByteData);
FMSocketIOManager.instance.SendToServer(ByteData);
FMSocketIOManager.instance.SendToOthers(ByteData);
```

- Events will be invoked when received data



- Example functions of processing received data

```
public void Action_OnReceivedData(string _string)
{
    debugText.text = "received: " + _string;
}
public void Action_OnReceivedData(byte[] _byte)
{
    debugText.text = "received(byte): " + _byte.Length;
}
```

Node.js Server setup:

- 1) Install npm + node.js
  - download and install all necessary components: https://nodejs.org/en/download/
- 2) install socket io
  - open terminal/cmd and type:
  npm install socket.io
- 3) Install express
  - open terminal/cmd and type:
  npm init
  //press Enter…
  npm install express --save
- 4) Finish & Test on localhost: demo server is in FMWebSocket/“TestServer.zip”
  - Copy & Unzip the demo server into other location, cannot be in Asset folder
  - open terminal/cmd and type:
  node /[path]/index.js
- 5) IP & Port of node.js server should match the settings in FMSocketIOManager.

- Step-by-step Video Tutorial: https://youtu.be/Zjm5KGHyceU

General Questions:

- Mobile App crash when turn on webcam
  - iOS: add Camera Usage Description in Player Setting
  - Android: check your webcam permission on devices
- FM Network doesn't work, cannot connect in local network
  - Windows 10: disable firewall, enable network discovery
  - Router: some advanced security setting used in University/Company router, will filter the data
  - Virtual Machines: it may cause connection trouble.

Known Issues:

- Auto Network Discovery doesn't work, when FMNetworkUDP server is on Google Pixel2, Pixel3
  - you have to grant multi-broadcast permission from devices. Otherwise, the server on Google Pixel Phones cannot receive UDP broadcast message from clients.

- EncodeToJPG() not working on iOS
  - As we applied native encoding method, which does not match Unity3D default jpeg library version. Thus, it will crash due to mismatching.
  - The alternative solution will be using FMEncodeToJPG(), we will try to fix this bug in future.

- Unity2020 WebGL(FM WebSocket) connection issue:
  - FM WebSocket may not work properly in WebGL Build from Unity2020, due to missing "gameInstance" or "unityInstance" is not defined.
  - Solution:
  - Ref: https://forum.unity.com/threads/unity-2020-1-sendmessage-no-longer-works-help.842209/

  1) After building WebGL from Unity, please open index.html, and add below line.

```
window.gameInstance = unityInstance;
```

  2) after adding this line:

```
var script = document.createElement("script");
    script.src = loaderUrl;
    script.onload = () => {
      createUnityInstance(canvas, config, (progress) => {
        progressBarFull.style.width = 100 * progress + "%";
      }).then((unityInstance) => {
        window.gameInstance = unityInstance;
        loadingBar.style.display = "none";
        fullscreenButton.onclick = () => {
          unityInstance.SetFullscreen(1);
        };
      }).catch((message) => {
        alert(message);
      });
    };
```