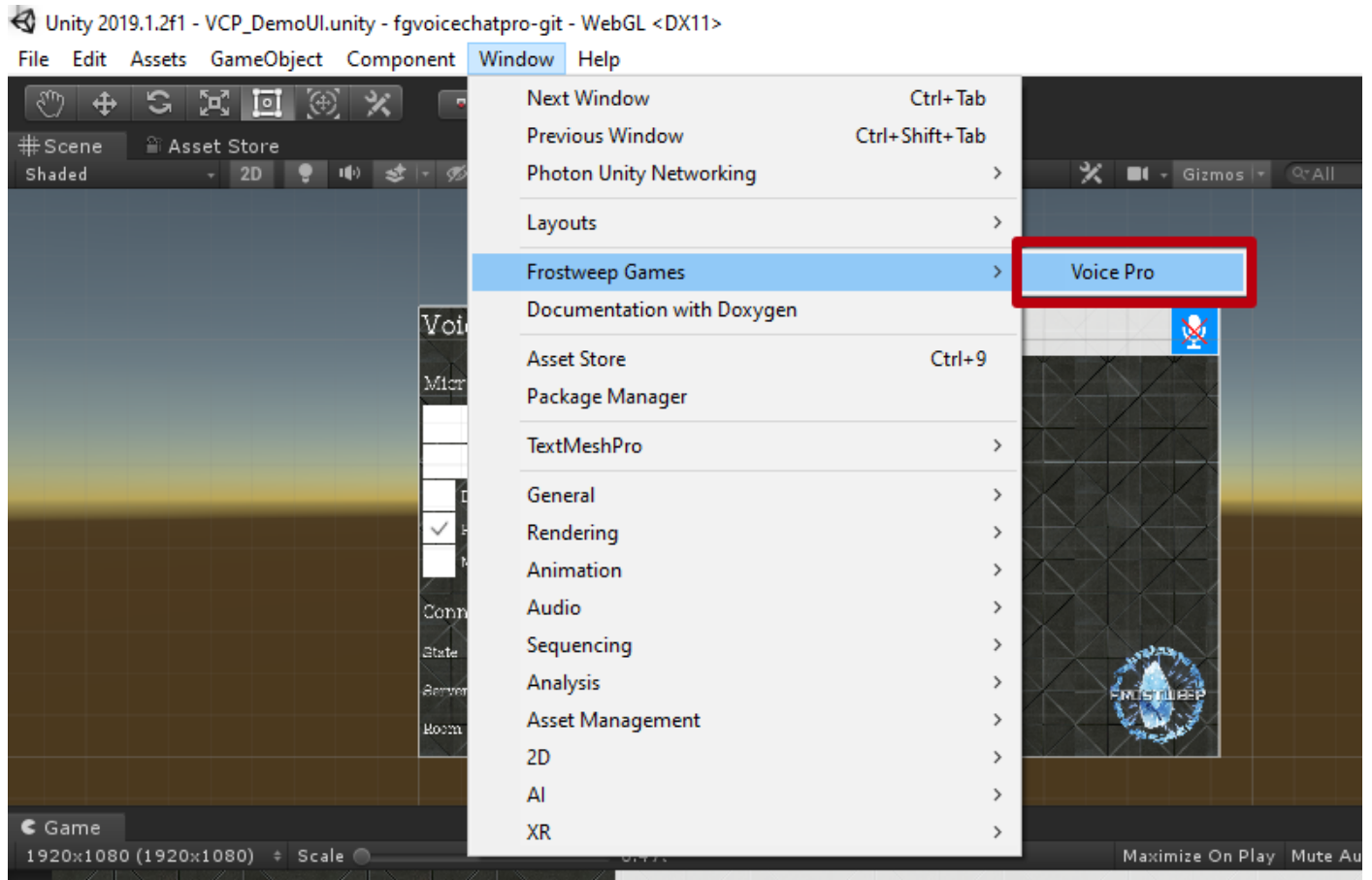


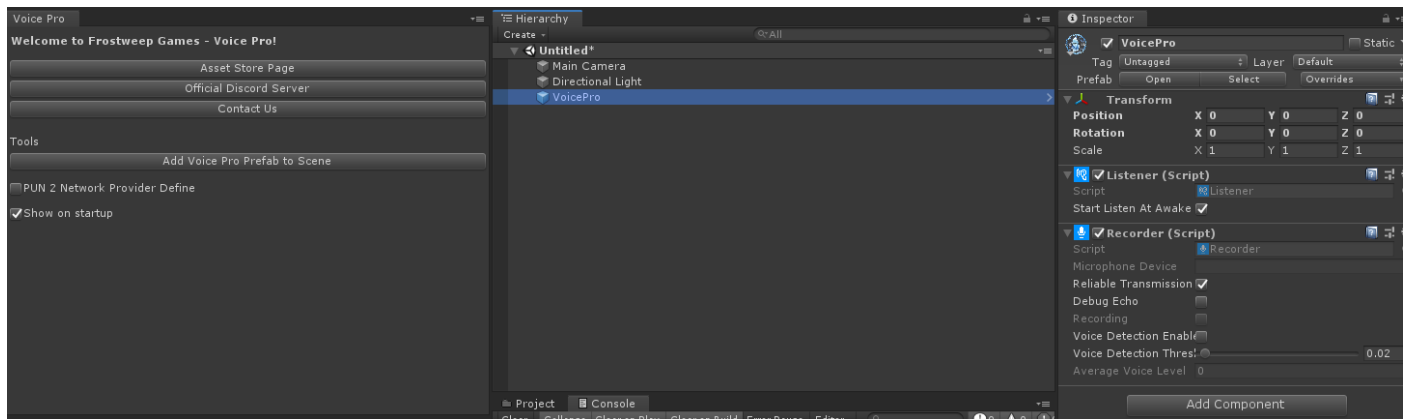
Voice Pro

How to use

First of all open Voice Pro Welcome window if it closed:

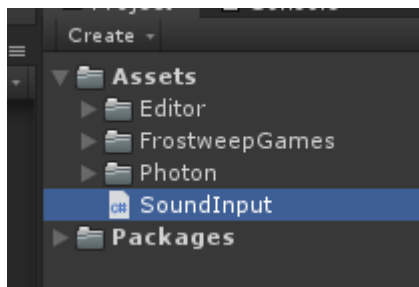


Create a new scene and click on *Add Voice Pro Prefab to Scene* button, it will create a new VoicePro object in scene.

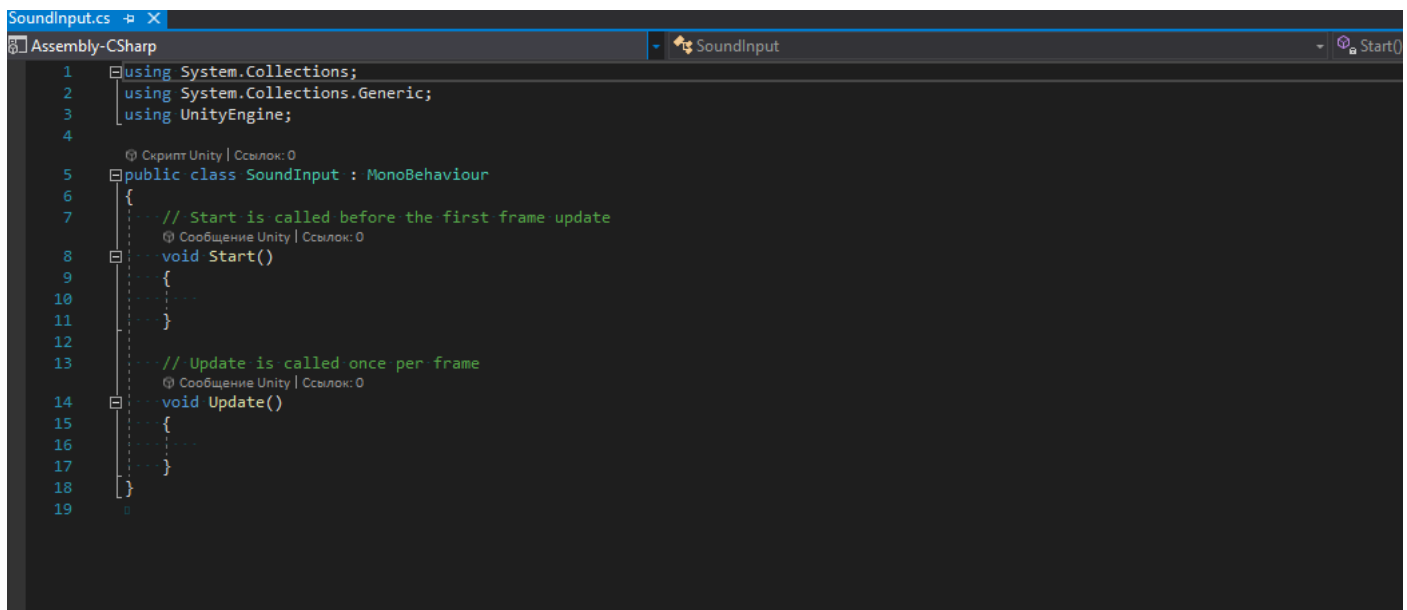


Now you ready to implement simple functionality for sound input.

Create a new script with name SoundInput:



Then open it in scripts editor:



Now we have to declare two variables for recorder and listener:

```
Assembly-CSharp
1 using FrostweepGames.VoicePro;
2 using UnityEngine;
3
4 public class SoundInput : MonoBehaviour
5 {
6     public Recorder recorder;
7
8     public Listener listener;
9
10
11     // Start is called before the first frame update
12     void Start()
13     {
14     }
15
16
17     // Update is called once per frame
18     void Update()
19     {
20     }
21 }
22
23
```

Recorder class need for recording sound from microphone. **Listener** for receiving list of available speakers (Look at VCP_Demo how it could be used).

Now need to write two function for start record and stop it. Lets name them accordingly.

```
4 public class SoundInput : MonoBehaviour
5 {
6     public Recorder recorder;
7
8     public Listener listener;
9
10
11     // Start is called before the first frame update
12     void Start()
13     {
14     }
15
16
17     // Update is called once per frame
18     void Update()
19     {
20     }
21
22
23     public void StartRecord()
24     {
25     }
26
27
28     public void StopRecord()
29     {
30     }
31 }
32
33
```

In StartRecord function we have to call record function from Recorder. And in StopRecord call stop record function from Recorder.

```

@ Скрипт Unity | Ссылка: 0
public class SoundInput : MonoBehaviour
{
    public Recorder recorder;

    public Listener listener;

    // Start is called before the first frame update
    @ Сообщение Unity | Ссылка: 0
    void Start()
    {
        ...
    }

    // Update is called once per frame
    @ Сообщение Unity | Ссылка: 0
    void Update()
    {
        ...
    }

    @ Сообщение Unity | Ссылка: 0
    public void StartRecord()
    {
        recorder.StartRecord();
    }

    @ Сообщение Unity | Ссылка: 0
    public void StopRecord()
    {
        recorder.StopRecord();
    }
}

```

Now we should add a new variable named as isRecording with Boolean type to understand do we record now or not:

```

@ Скрипт Unity | Ссылка: 0
4 public class SoundInput : MonoBehaviour
5 {
6     public Recorder recorder;
7
8     public Listener listener;
9
10    public bool isRecording;
11
12

```

Also we need to add logic for network registration. Lets do it:

```

@ Сообщение Unity | Ссылка: 0
void Start()
{
    int id = Random.Range(0, 100);
    NetworkRouter.Instance.Register(id, "Speaker" + id, Enumerators.NetworkType.PUN2);
}
@ Сообщение Unity | Ссылка: 0

```

NetworkRouter.Instance.Register function registers network solution by **NetworkType** (we use PUN2), user **Id** and user **Name**.

Lets add a logic that will start and stop record dependently from pressing of a button with use of isRecording variable:

```
public class SoundInput : MonoBehaviour
{
    public Recorder recorder;
    public Listener listener;
    public bool isRecording;

    // Сообщение Unity | Ссылка: 0
    void Start()
    {
        int id = Random.Range(0, 100);
        NetworkRouter.Instance.Register(id, "Speaker" + id, Enumerators.NetworkType.PUN2);
    }

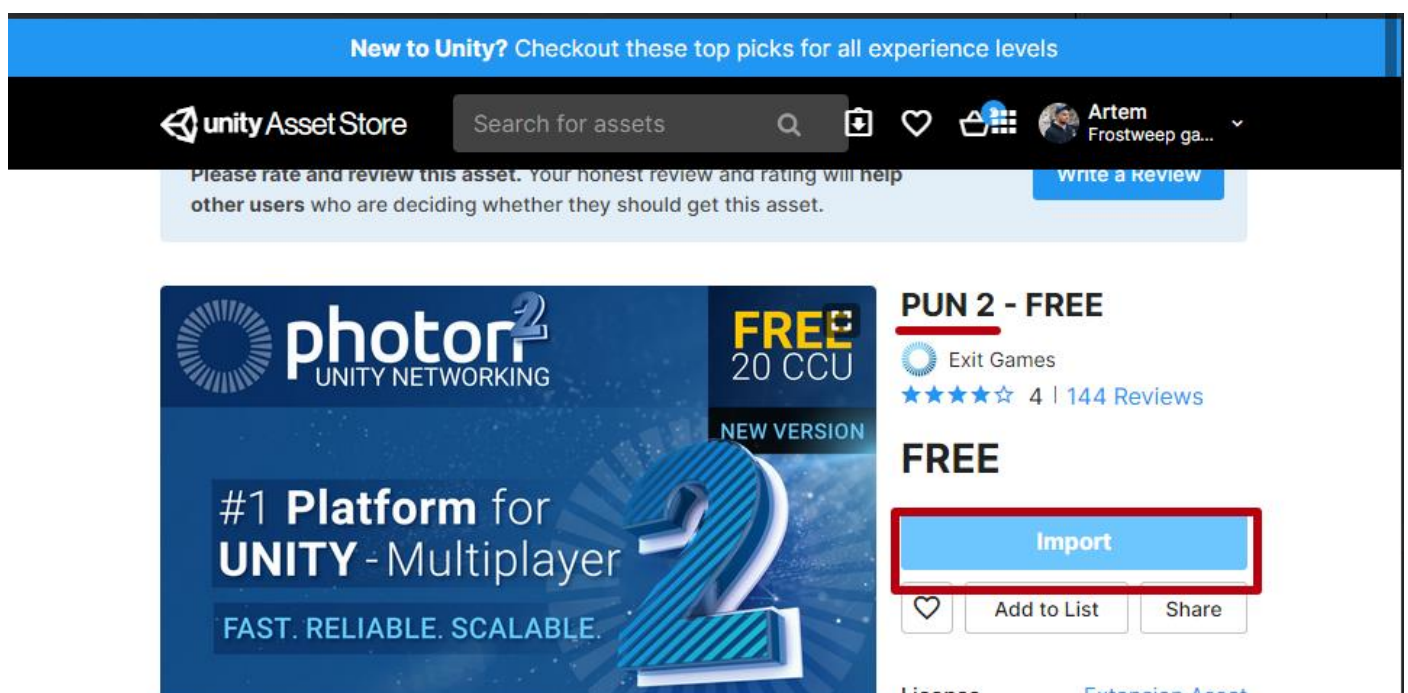
    // Сообщение Unity | Ссылка: 0
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.R) && !isRecording)
        {
            StartRecord();
            isRecording = true;
        }
        else if (Input.GetKeyUp(KeyCode.R) && isRecording)
        {
            StopRecord();
            isRecording = false;
        }
    }

    // Ссылка: 1
    public void StartRecord()
    {
        recorder.StartRecord();
    }

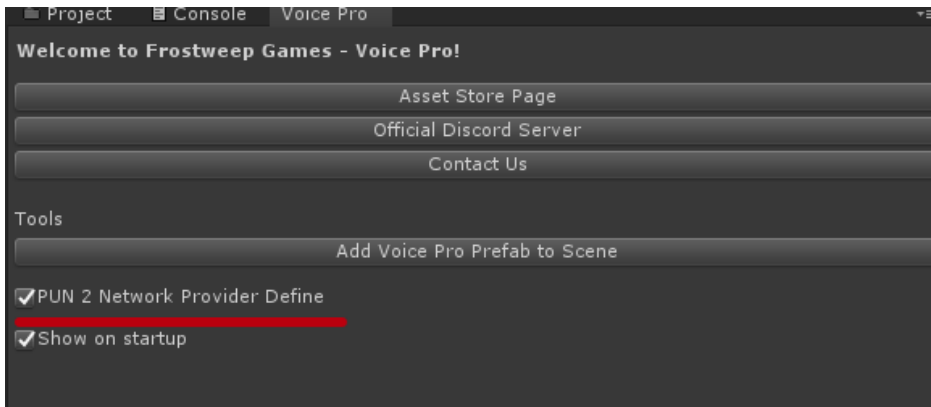
    // Ссылка: 1
    public void StopRecord()
    {
        recorder.StopRecord();
    }
}
```

As it's a network solution you have to register networking. Currently our solution provides api for usage of PUN 2 network (**not** PUN Voice).

Lets it import it from asset store:

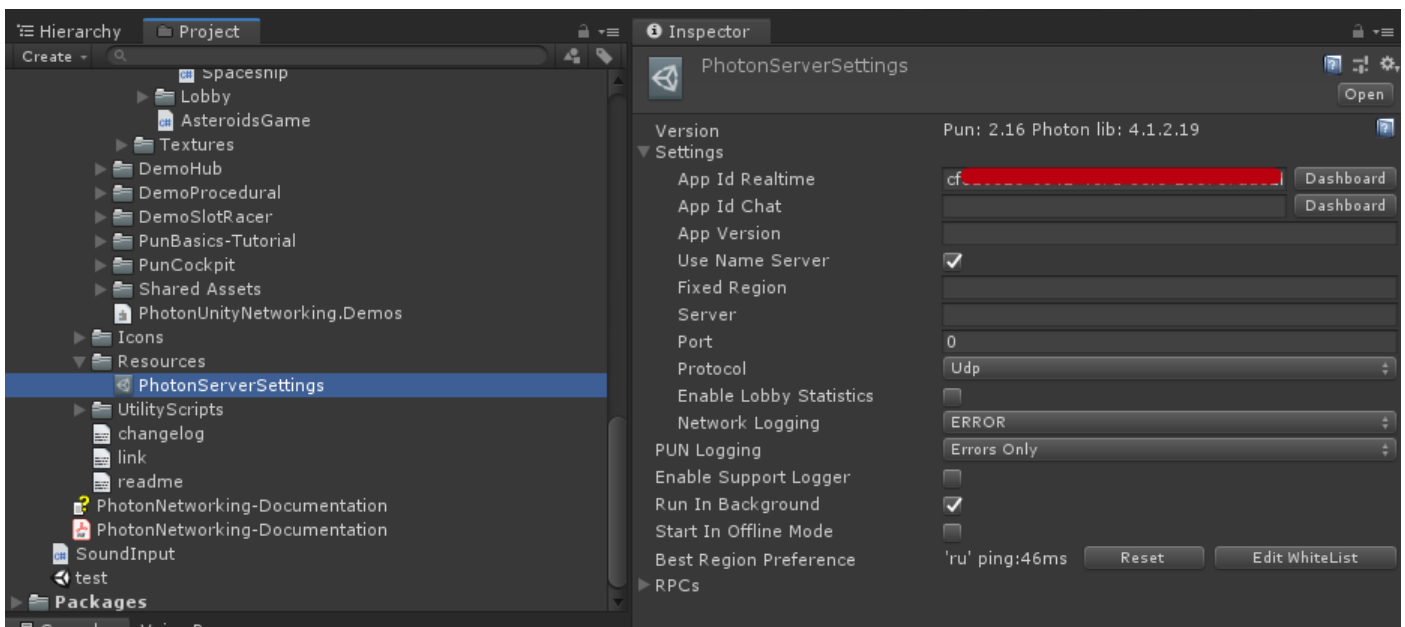



When imported you have to open Voice Pro Welcome window (described at begin of document) and check PUN2 Networking checkbox:



It will add special code define to list of Defines in project settings.

Don't forget to setup PhotonServerSettings. (You have to create Photon account and register an app)



 [PRODUCTS](#) [SDKs](#) [Documentation](#) [Dashboard](#)

Manage Test WebGL voice

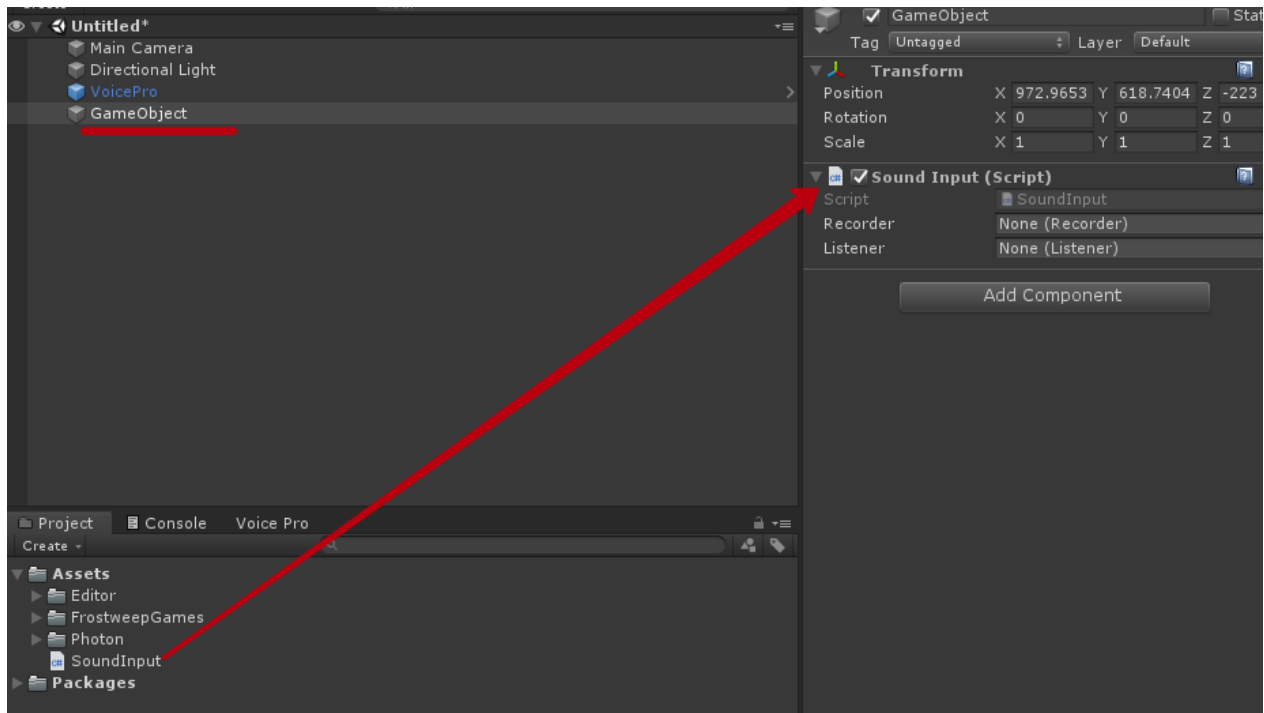
App ID: cf...

[Dashboard](#)

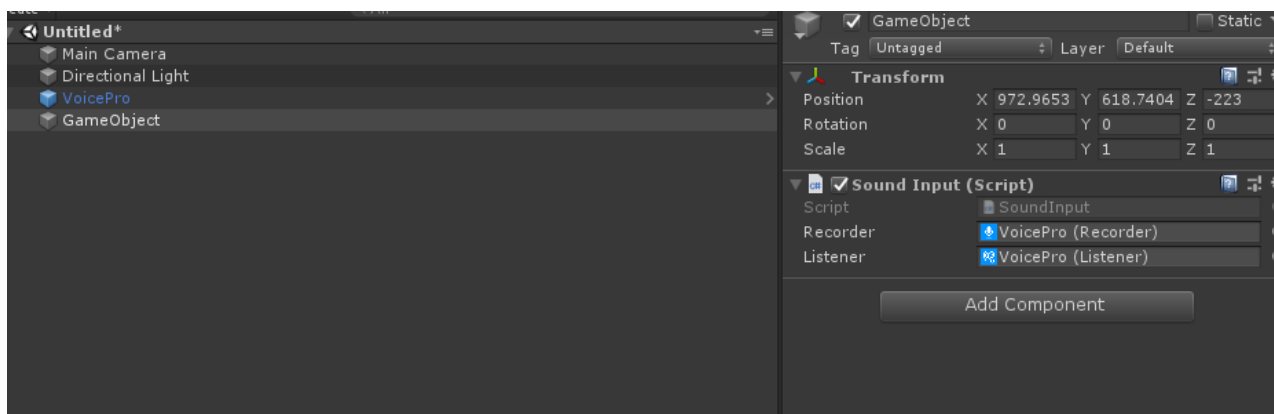
Properties

Name	Concurrent Users
Test WebGL voice	
Subscription	0 CCU
One-Time	0 CCU
Coupon	0 CCU
Total	20 CCU CCU Burst is not allowed.

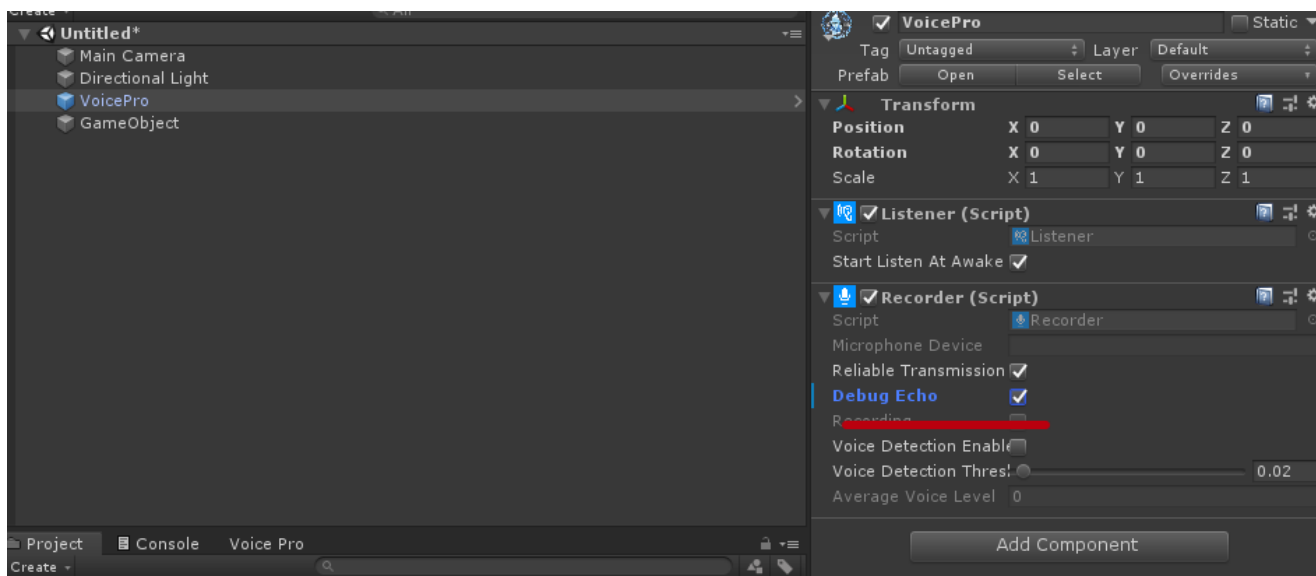
Last things that we have to do its create an empty object on created scene and attach SoundInput script we created on it(don't forget to save script before).



Now we should set our variables. Lets drag & drop VoicePro object from scene to inspector on both fields:



To make local test you could enable DebugEcho on Recorder component. Lets check DebugEcho checkbox:

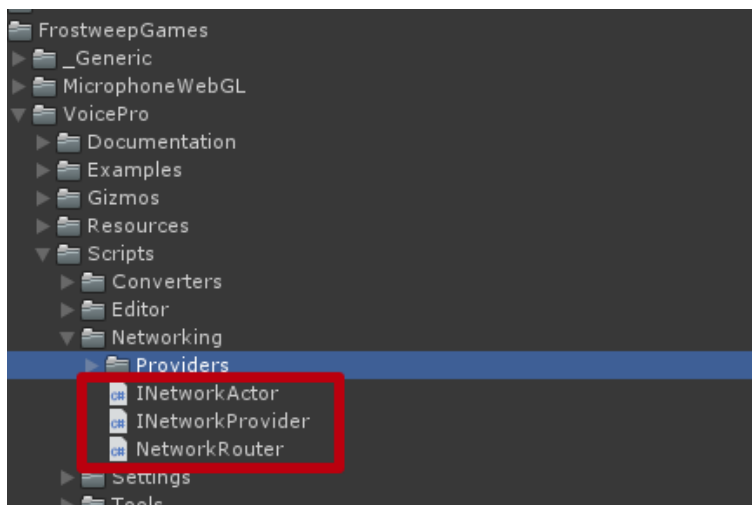


Now save scene in project folder and play scene. During pressing on R button, - it will record and transmit data.

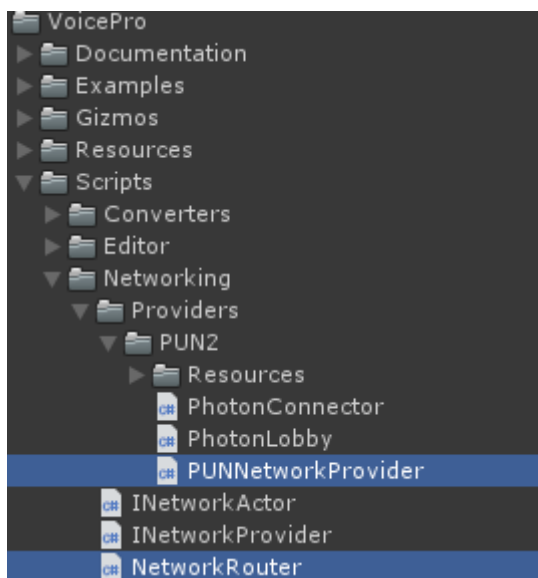
Advanced

As we implemented only PUN network solution you also could implement own.

We made few scripts that handles most of functionality for easy integration different networks.



How to use them you could look at PUNNetworkProvider and NetworkRouter scripts:



For example this is the list of function PUNNetworkProvider implements from interface:

```

/// <summary>
/// Photon PUN 2 Network Provider for data transmission
/// </summary>
/// Ссылка: 2
public class PUNNetworkProvider : INetworkProvider
{
    /// <summary>
    /// Code of network event that uses for voice data transition
    /// </summary>
    private const byte VoiceEventCode = 199;

    public event Action<INetworkActor, byte[]> NetworkDataReceivedEvent;

    private INetworkActor _networkActor;

    private GameObject _eventsHandler;

    /// Ссылка: 2
    public void Dispose()...

    /// Ссылка: 2
    public void Init(INetworkActor networkActor)...

    /// Ссылка: 2
    public void SendNetworkData(NetworkRouter.NetworkParameters parameters, byte[] bytes)...

    /// Ссылка: 2
    public string GetNetworkState()...

    /// Ссылка: 2
    public string GetConnectionToServer()...

    /// Ссылка: 2
    public string GetCurrentRoomName()...

```

Also there implemented PUNNetworkActor that uses for understanding of actors in network:

```

public class PUNNetworkActor : INetworkActor
{
    // Ссылка: 4
    public int Id { get; private set; }

    // Ссылка: 4
    public string Name { get; private set; }

    // Ссылка: 2
    public PUNNetworkActor(int id, string name) {}

    // Ссылка: 16
    public override string ToString() {}

    // Ссылка: 1
    public static PUNNetworkActor FromString(string data) {}
}

```

In NetworkRouter you have to implement registration of your network solution:

```

public void Register(int id, string name, Enumerators.NetworkType networkType)
{
    NetworkType = networkType;

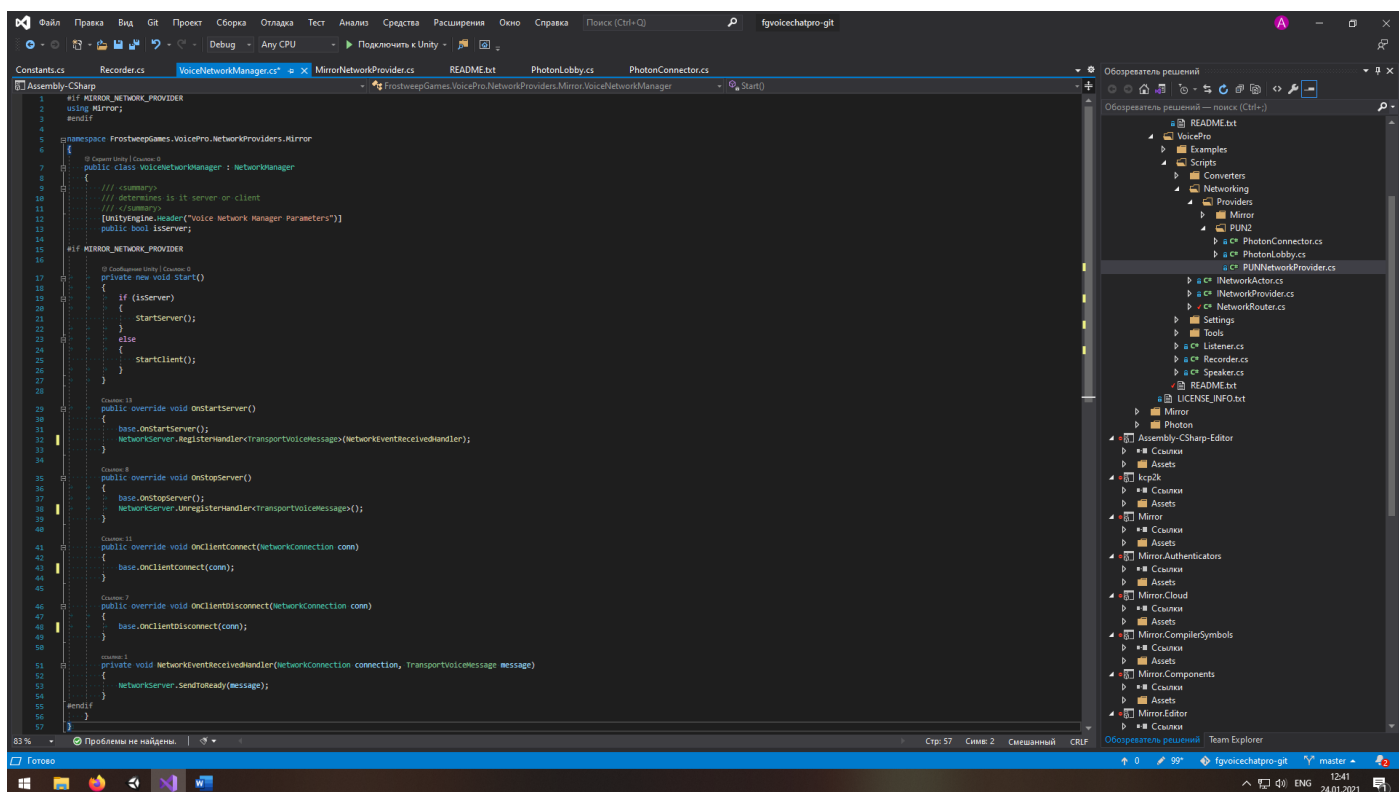
    switch (networkType)
    {
        case Enumerators.NetworkType.PUN2:
        {
            #if PUN2_NETWORK_PROVIDER
            _networkProvider = new PUNNetworkProvider();
            _networkProvider.Init(new PUNNetworkProvider.PUNNetworkActor(id, name));
            _networkProvider.NetworkDataReceivedEvent += ReceiveNetworkDataActionHandler;
            #endif
            break;
        }
        default:
        {
            throw new NotImplementedException("Network didn't registered! Network type didn't implemented.");
        }
    }
}

```

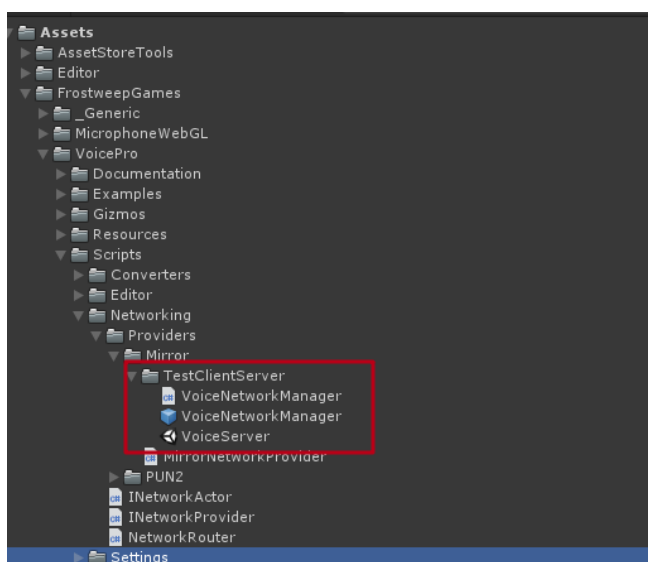
All connections to network written in other scripts such as lobby and connector.

How to setup Mirror Networking

For test purposes you could use our basic implementation of Mirror Server. We made VoiceNetworkManager which handles connection to server for clients and server starting functionality.



You can find this class and demo scene in:

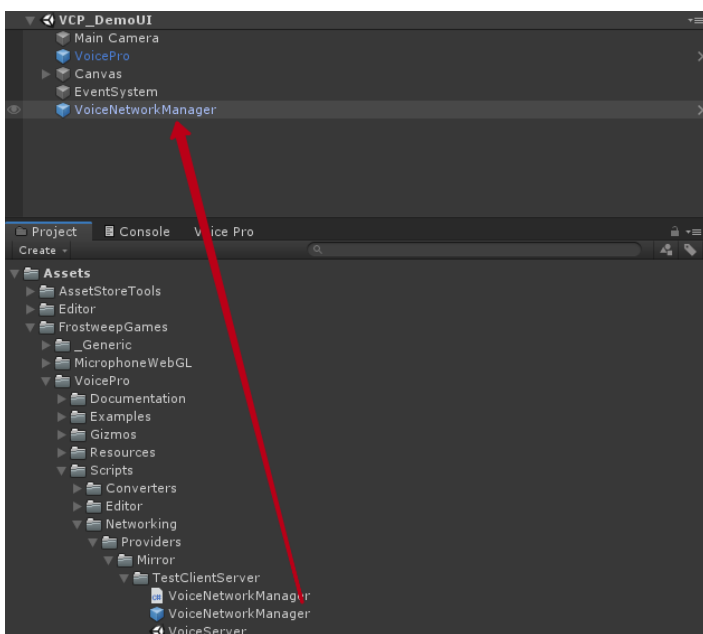


VoiceServer scene already includes prefab which contains VoiceNetworkManager configured for server

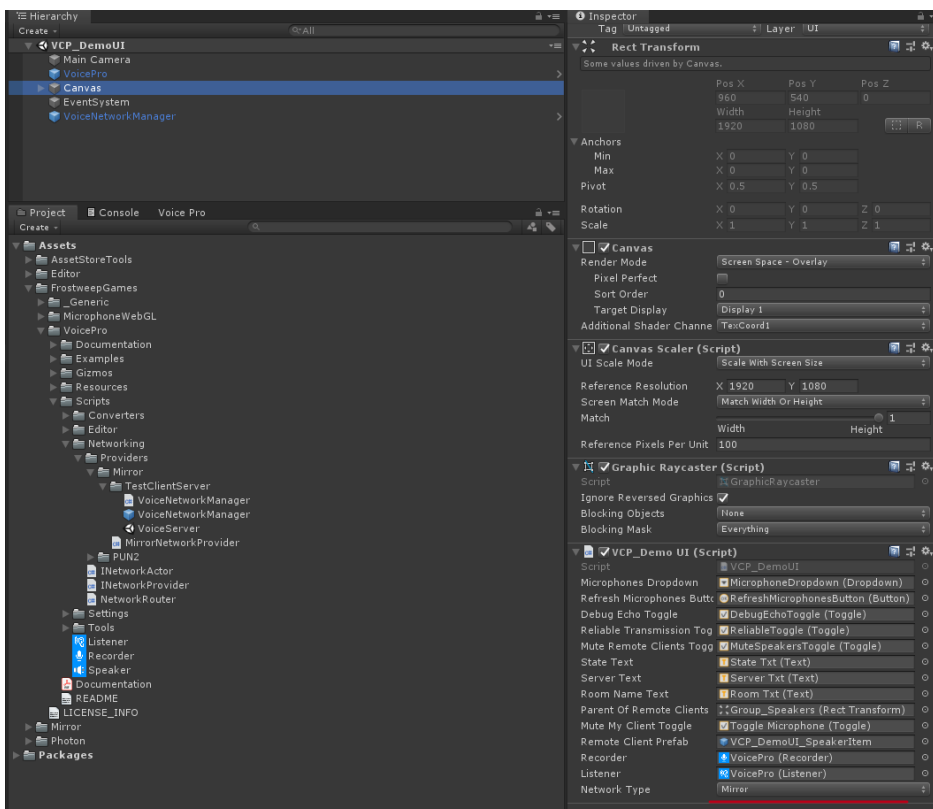
To start server build only VoiceServer scene and run it.

To make demo scene of Voice Pro workable with Mirror, you have to configure few things:

First of all you need to open demo scene and place configured prefab on the scene:



Then you need to select networking in Example script:



Now start the scene and enjoy!

API Reference available at: <https://unitydemos.frostweepgames.com/assetstore/voiceprodemo/api/>

WebGL demo: <https://unitydemos.frostweepgames.com/assetstore/voiceprodemo/>

Please read **README** file!