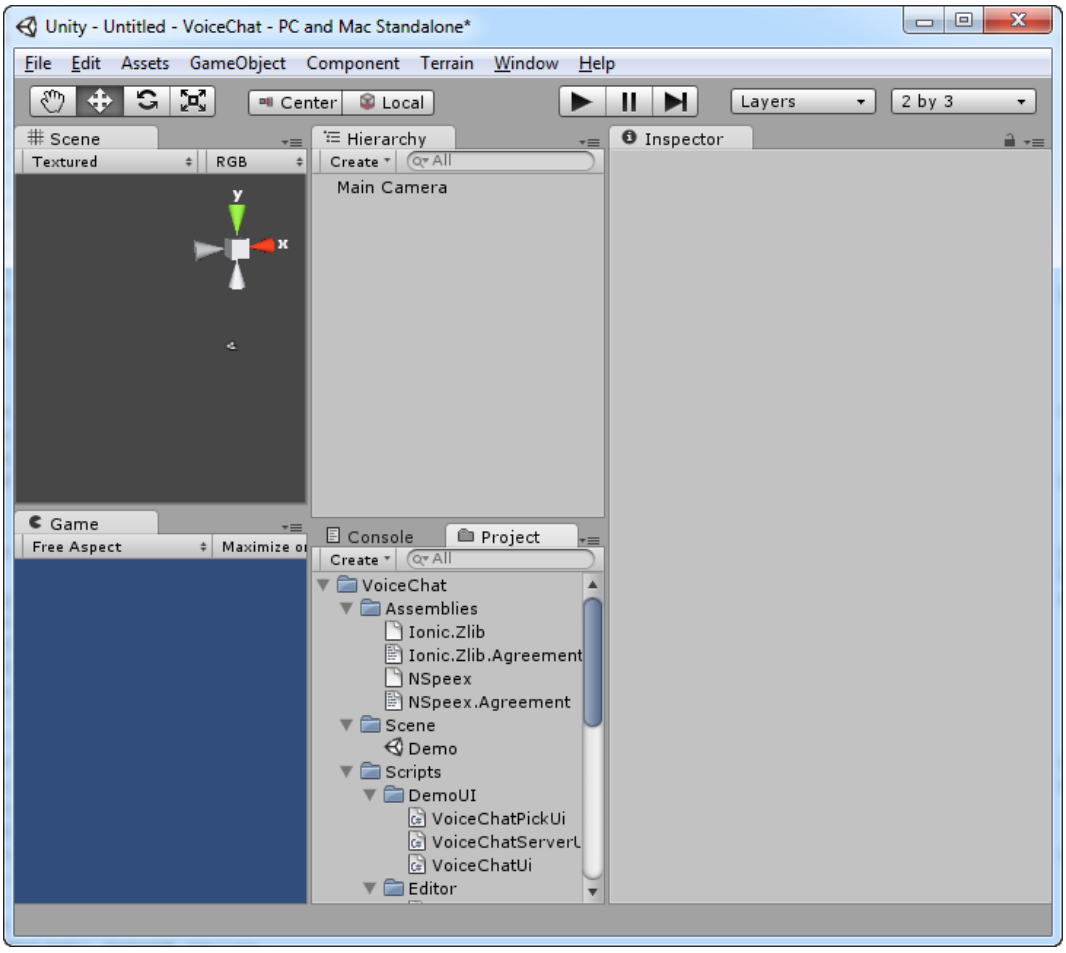
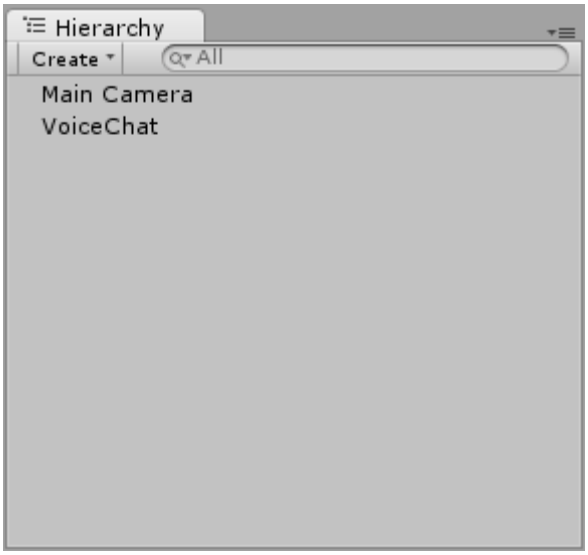
Voice Chat – Setup Manual

Setting up recording

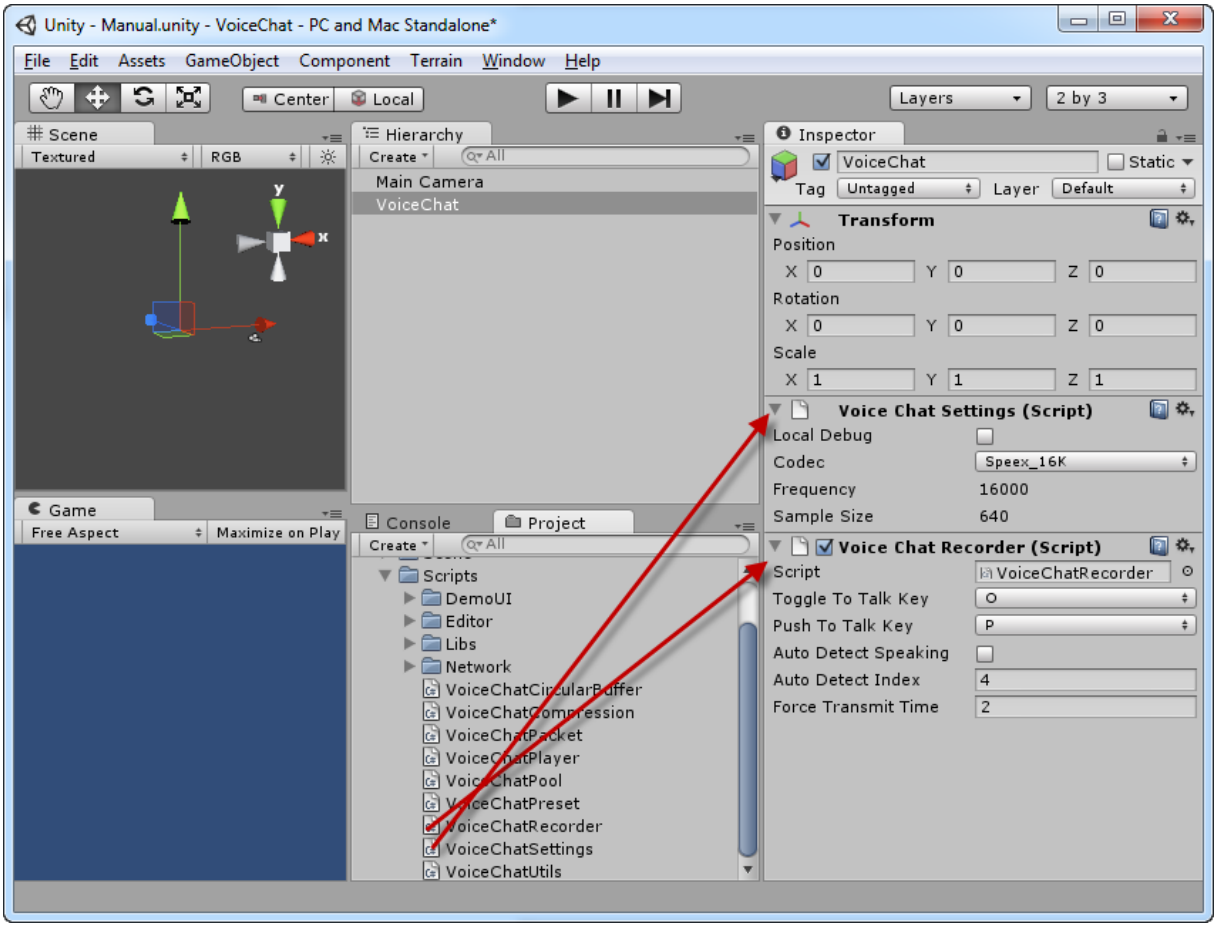
The first thing we're going to do is create a new, completely empty scene:



Then create a new game object, and call it Voice Chat:



After that drag one instance of the VoiceChatSettings script and one instance of the VoiceChatRecorder script to the game object:



Let's go through the options available on each script, let's start with VoiceChatSettings:

- Local Debug – If checked allows you to attach a VoiceChatPlayer script to the same object, and it will listen to whatever the VoiceChatRecorder transmits, this is for local debugging and basically enables you to talk to yourself.
- Codec – This is the main setting that controls the quality and size of the audio sent, Speex_16k is the default and is a very good compromise between size and quality. There is one other codec available called Alaw, but for all intents and purposes Speex_8k or Speex_16k will give you the best performance and audio.
- Frequency – Not editable, shows the frequency of the current codec
- Sample Size – Not editable, shows the sample size of the current codec

The settings on VoiceChatRecorder are as follows:

- Toggle To Talk Key – Press and release this key to enable talking, Press and release again to disable.
- Push To Talk Key – Hold down this key to enable talking, release to disable
- Auto Detect Speaking – Allows the recorder to auto-detect when you're speaking, so no key needs to be held down. Requires some CPU.

- Auto Detect Index – How “common” the highest frequency must be in the recorder sample to active transmission of audio. The lower it is the harder it is for the auto transmission to get activated.
- Force Transmit Time – How long, in seconds, that transmission shall be enabled after auto-detecting speech.

Recording audio

To start recording audio two things must be done, when your game is running grab the VoiceChatRecorder instance by accessing the static property VoiceChatRecorder.Instance. The first thing you need to do is to set the VoiceChatRecorder.Instance.Device property, this has to be either null (default) or one of the devices available in VoiceChatRecorder.Instance.AvailableDevices.

The second thing you must do is to call VoiceChatRecorder.Instance.StartRecording(), do note that if you’re not connected to the network or you have not enabled local debugging, StartRecording will fail and return false. If you want to stop recording just call StopRecording() instead.

Note that Voice Chat makes a difference between recording (recording audio locally) and transmitting (sending audio to other players). When you call StartRecording you will not start sending audio to anyone, you still need to hold down the push to talk key, toggle talking or use speech auto detection to start sending audio.

Setting up the default networking

The package comes with pre-built support for the built in Unity Networking, and we’re going to start by configuring it. If you’re planning to use Voice Chat with a third party networking solution, skip to the part titled “Integrating with third party networking”.

The most common way will be that you need to integrate it into your existing game, which is incredibly simple. All you have to do is call the VoiceChatUtils.CreateProxy method, without any arguments, when you’re connected to the server (or whenever you want to give you the ability to transmit audio through Voice Chat). In most cases it will look something like this:

```
VoiceChatNetworkProxy proxy;

void OnConnectedToServer()
{
    proxy = VoiceChatUtils.CreateProxy();
}

void OnDisconnectedFromServer(NetworkDisconnection info)
{
    GameObject.Destroy(proxy.gameObject);
}
```

This code is actually copied straight out of the Demo/VoiceChatUnitClient.cs script, which is an example/default script that just connects to a Unity networking server. The

Demo/VoiceChatUnityServer.cs script also exists which is equally simple and just sets up a default unity server.

This is really all there is to it for setting up networking.

Integrating with third party networking

If you want to use Voice Chat with a third party networking solution like Photon, uLink, Smartfox or SlimNet there are a few more things that need doing.

The first thing you need to do is to set the VoiceChatRecorder.Instance.NetworkId property, this should be done as soon as the client connects to the server, and before StartRecording() is called the id should be unique per client as it's used to identify different speaking clients. Most networking libraries supply you with a player id or client id that you can use for this, if not you can have the server send you a unique increment integer. For an example on how this can be done check the Start() method of the Network/VoiceChatNetworkProxy.cs script which does this for the default unity networking.

The second thing we want to do is to transmit our audio to the other clients, this is done by attaching an event listener to the VoiceChatRecorder.Instance.NewSample event, the event has one argument which is a struct of the type VoiceChatPacket, which looks like this:

```
public struct VoiceChatPacket
{
    public VoiceChatCompression Compression;
    public int Length;
    public byte[] Data;
    public int NetworkId;
}
```

This struct needs to be transmitted to all other players, how this is done is specific to your networking solution. All fields in this struct needs to be transmitted, the part of the Data byte array that needs to be sent is from index 0 and the amount of elements is the same as the Length property.

The third thing that needs to be done is playing the audio you receive, if you've done things correctly you should get a VoiceChatPacket sent from each transmitting client with the latest audio from them, what you need to do is to create one instance of the Network/Resources/VoiceChat_Player prefab for each NetworkId, grab the VoiceChatPlayer component on it and call OnNewSample(packet);

That's it for setting up the third party networking, a quick recap:

1. Set VoiceChatRecorder.Instance.NetworkId to a unique number on each client
2. Subscribe to the VoiceChatRecorder.Instance.NewSample event
3. When the NewSample events get triggered, send the VoiceChatPacket data to all other clients.
4. On the client read the data back, and make sure you have one instance of the Network/Resources/VoiceChat_Player prefab, grab the VoiceChatPlayer script and call OnNewSample(packet) on it.