Video Chat-Unity3D Documentation

Midnight Status

Notes

- Do not run in the Unity Editor in Web Player mode, you cannot get the proper permissions to access the microphone this way. Using Standalone, WebGL, or other platforms is OK.

- WebGL works in editor but does not currently export due to Microphone conflicts.

- Use Test Mode to assess your machine's ability to process video chat.

- Change PhotonServerSettings (search Project view) asset to:
  Hosting - Best Region
  Enabled Regions - Everything
  Appid - You must register an email with Photon to get your Appid
  Protocol - TCP

Introduction

Video Chat for Unity3D is an extension class that allows audio and video to be processed and queued for network transmission. The VideoChat class does this via the FromAudio() and FromVideo() function calls, typically in the Update callback of a Unity MonoBehaviour.

For example:

```
void Update() {

	VideoChat.PreVideo();

	VideoChat.FromAudio();

	//Run through all AudioPackets and send them across the network connection of your choice

	VideoChat.FromVideo();

	//Run through all VideoPackets and send them across the network connection of your choice
}
```

Depending on the network library employed the transmission and reception can vary. Although earlier versions of Video Chat employed Legacy Unity Network and TNet. We now **RECOMMEND** Photon exclusively. You are still free to send the data with any networking library you see fit to use as long as it can send byte[] (byte arrays).

For example:

```
//For each AudioPacket...
audioView.RPC( "ReceiveAudio", PhotonTargets.Others, currentPacket.position,
currentPacket.length, currentPacket.data );

//For each VideoPacket...
videoView.RPC( "ReceiveVideo", PhotonTargets.Others, currentPacket.x, currentPacket.y,
currentPacket.data );

[RPC]
void ReceiveVideo( int x, int y, byte[] videoData ) {
      VideoChat.ToVideo( x, y, videoData );
}

[RPC]
void ReceiveAudio( int micPosition, int length, byte[] audioData ) {
      VideoChat.ToAudio( micPosition, length, audioData );
}
```

Features

Video Chat-Unity3D can be used as a very simple chat application like Skype or FaceTime. The examples that come with the package demonstrate this type of capability. However, Video Chat can also go far beyond what has typically been done in the past. Because Video Chat can be used in 3D, all manner of effects can be employed to bring Video Chat and the players involved into the game world.

The VideoChat class contains several elements that can help with full 3D integration.

```
//Created in Add() if unassigned
public static GameObject     vcObject;
public static Material       cameraView;

//Created in Init() if unassigned
public static AudioSource    audioIn;
public static AudioSource    audioOut;
public static bool           audioOut3D;
```

The GameObject vcObject (short for Video Chat Object) is a game object from the current scene that VideoChat can render to. If this is unassigned then VideoChat will create a quad for rendering a generic view (not recommended).

The Material cameraView is the material desired for rendering. VideoChat comes with a a CameraView material that can be assigned (or modified). However, If this field remains unassigned VideoChat will still load the shader and material automatically.

The AudioSource's audioIn and audioOut again will be created automatically if unassigned but can be assigned if so desired. The biggest benefit of this feature currently is the ability to assign audioOut as a 3D object. If the audioOut3D boolean flag is set to true then the clip associated with audioOut will also be in 3D.

Class Reference

VideoChat Class

```
//Created in Add() if unassigned
public static GameObject        vcObject;
public static Material          cameraView;
public static Shader            shader;

//Created in Init() if unassigned
public static AudioSource       audioIn;

//The audio source for audio output
public static AudioSource        audioOut;
//Should the audio source be in 3D?
public static bool               audioOut3D;

public static List<AudioPacket>  audioPackets;
public static List<VideoPacket>  videoPackets;
public static List<WebCamDevice> webCamDevices;

//These list can be used for recording incoming data, they can also be cleared if not
public static List<AudioPacket>  receivedAudioPackets = new List<AudioPackets>();
public static List<VideoPacket>  receivedVideoPackets = new List<VideoPackets>();

//Has the other user connected and sent us their video specs?
public static bool               videoPrimed;

//Do we want to see ourselves?
public static bool               localView;

public static void SetVideoQuality( VideoQuality videoQuality ) {
//Pass in VideoQuality.high/mid/low
//Must set deviceIndex (it's a getter/setter) to see changes
}

public static void SetAudioQuality( AudioQuality audioQuality ) {
//Pass in AudioQuality.high/low
//Set in VideoChat.Add() upon initialization, does not work on-the-fly
}

public static void SetCompression( Compression compression ) {
//Pass in Compression.high/low
}
```

```
public static void SetEchoCancellation( EchoCancellation pEchoCancellation ) {
//Pass in EchoCancellation.on/off
}

//The setter here resets the camera. If the value is incremented or decremented it
will switch between available cameras. Some devices report more than one camera device
even if there is only one so sometimes a camera index will point to empty data and you
will see nothing rendered, can call VideoChat.deviceIndex++ to easily cycle through
available devices.
public static  int deviceIndex {
              get{  return _deviceIndex; }
              set{}
}

public static void Add( GameObject addObject = null, AudioSource addAudioIn = null,
AudioSource addAudioOut = null ) {
//This adds an remote view display object (must contain a Renderer or CanvasRenderer)
to the scene, can be done at start or upon network instantiation
}

public static void Init() {
//Creates a new list of video packets and resets the deviceIndex
}

public static void ClearAudioOut() {
//Blanks out the audioOut audio source to get rid of any lingering samples after a
disconnection or end of test mode
}

public static void FromAudio() {
//Collects audio data, packages it, and adds it to the list of audio packets
}

public static void PreVideo() {
//Prepares VideoChat and sets up the local view (if used)
}

public static IEnumerator FromVideoRenderTexture() {
//A coroutine that collects video data from a RenderTexture, packages it, and adds it
to the list of video packets
}

public static void FromVideoTexture2D() {
//Collects video data from a read/write Texture2D, packages it, and adds it to the
list of video packets
}

public static void FromVideo() {
//Collects video data from a WebCamTexture, packages it, and adds it to the list of
video packets
}

public static void ToAudio( int position, int length, byte[] data ) {
//Applies audio data that arrived over the network
```

```
}

public static void ToVideo( int x, int y, byte[] data ) {
//Applies video data that arrived over the network
}
```