Interactive 360 Template User Guide

This sample project contains sample scenes, videos, scripts and prefabs that serve as a starting point for building 360 video experiences in Unity.

Scenes

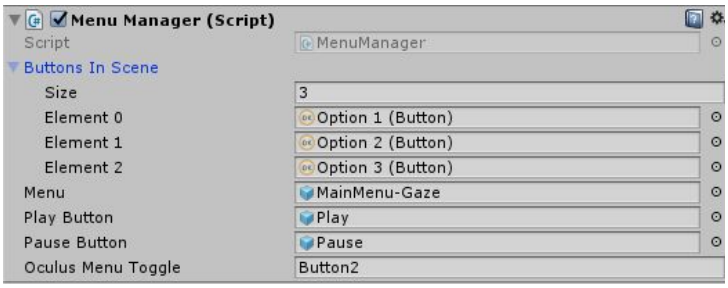
The sample scenes are broken up into 2 seperate folders, Starting Scenes and Video Scenes. Starting Scenes contain scenes for different scenarios with either a menu screen or hotspot interaction. Video Scenes are standalone scenes that playback a 360 video- either in 2D format or 3D format. The individual video scenes are loaded from the starting scenes, using interaction with the menu or hotspot system. To build each scenario, the build settings will need to be updated with the desired starting scene marked as scene 0 before building. By default, the project is set up to build the MainMenu_Gaze starting scene.

Starting Scenes

Hotspot Example_Gaze: This scene demonstrates how to create a “hotspot” (interactive spot) on top of the video. In this case, interacting with the hotspot moves the player to another viewpoint, which is a seperate 360 video scene in our project. The hotspot manager script, which is located on the Hotspots Game Object, takes all the hotspots in the scene and binds them to load the corresponding video scene in the Video Manager. The hotspot assigned to Element 0 in the hotspot manager will load the scene assigned to Element 0 in the Video Manager. In this case, we have 1 hotspot in the scene, HotSpot1. We can see in our Video Manager game object that this 1 hotspot will load the scene called 2D_Lookout. When the user’s gaze interacts with the hotspot, a circular selection bar will become active. This selection bar is attached to MainCamera-Gaze->VRCameraUI->GUIReticle->Normal->Highlighted. Once the bar fills up, we will trigger unity to load the next scene.

MainMenu_Gaze: This scene demonstrates one way to interact with a menu scene using only gaze. The player can navigate through a series of 360 videos by interacting with the menu screen. The MainCamera-Gaze Game Object uses a VR Eye Raycaster,

which detects if the player's gaze is making contact with any items that are interactable. In our scene, only the menu objects are interactable. Those objects can be found under MainMenuManager->MainMenu->ControlUI->VideoSelector. There are 3 options in this sample scene. In order for the object to be interactive, it has a VR Interactive Item script attached to it, a Box Collider, and a Button component. Similar to our other scenes, each Button is linked to a corresponding scene to load and the Menu Manager script, on the MainMenuManager Game Object handles this linking.



MainMenu_Controller: This scene demonstrates how to interact with a menu using Oculus Touch controllers. The camera and controller game objects can be found in the Hierarchy under Player. The RH and LH Game Objects use the Tracked Pose Driver component to get tracking from the devices. The right hand controller uses a Raycast Input script to detect if the raycast is interacting with any buttons in the scene. The primary input is set to the SecondaryIndexTrigger of the touch controller, which is the trigger that the index finger rests on. When the player points their right hand towards a menu object, and pushes the trigger, the button will be clicked. Since our scene is using controllers, and not purely gaze, we are now able to close and open the Menu using a button. The Menu Manager script on the MainMenuManager has a parameter called Oculus Menu Toggle. This Toggle is set to Button2, which corresponds to the B button on the controller.

MainMenu_GVR_Controller: This scene demonstrates how to interact with a menu using Google Daydream View + Daydream Controller. The Google VR SDK is needed as an additional resource to receive input from the mobile controller, which can be found in the project folder under Plugins. The Player game object in the hierarchy contains all of the Prefabs required to receive input and render the Daydream controller. The main Touchpad button on the controller serves as the primary button for interacting with menu objects. The app button on the controller will toggle the menu screen. The home button is not configurable and will always open Google Daydream home. Otherwise, this scene will operate the same as the MainMenu_Controller scene.

MainMenu_Editor: This scene allows you to to test without VR enabled directly in the editor. Toggle VR off by selecting the VR object in the top menu->Disable. There is a CameraEditorControl script added on the Camera Game Object which allows you to

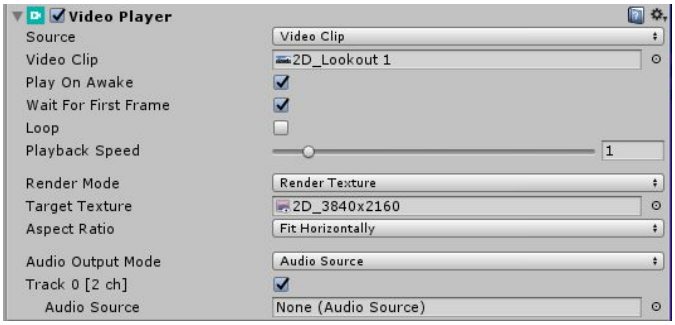
mock VR headset movement using a mouse. Press Play, then the esc key to see your mouse and navigate through the experience in Editor.

Video Scenes

Each video scene contains a unique 360 video. The video asset is played back in the scene using the Video Player component. You can either use a video clip asset or URL to source the video. You can easily swap your own videos in these sample scenes by dragging the video clip asset to the video player. However, you may need to update the Target Texture and Skybox material in your scene depending on your video's pixel dimensions.

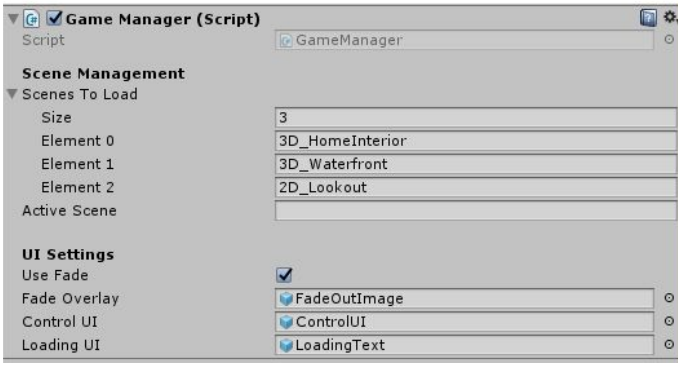
Video Render Textures and Materials

Each video scene is rendered onto a Skybox material using a Render Texture. The Render Mode is defined in the Video Player Component. The Sample Project contains several example Target Textures for common 360 Pixel sizes, in both 3D and 2D format. The size of the Render Texture needs to match the size of the source video. Once the Render Texture has been created, a Skybox material needs to be created that references the Render Texture. More info on creating Render Textures and Materials can be found in the **Setting up Panoramic 2D and 3D Video in Unity** section below.



VideoManager

Each scene has a VideoManager Game Object with a script called Game Manager attached to it. The Game Manager script keeps track of which scenes we may be loading from the current scene. The scenes are stored as strings in the an array. The Game Manager will also tell us which scene is currently active. The Game Manager also handles transitioning in between scenes. Fading between scenes and loading text can be enabled to create a smoother transition. If the Use Fade box is checked, you will need to define the Fade Overlay Image, Control UI and Loading UI in the scene.



Pausing / Playing Videos

Playing and Pausing videos is also handled by the Game Manager. In all of the example scenes, the play/pause buttons call the PlayVideo() and PauseVideo() methods in the Game Manager when the button is clicked. These methods are searching for a video in the scene to pause or play, so if there is no active video in the scene, the buttons will do nothing.

Notes on Mobile Limitations

When building to Android devices, there is max limit of 2GB of assets per scene. This is one of the reasons we keep each video in its own scene. 360 videos tend to be very large and can quickly exceed 2GB. If your video asset is larger than the limit, you will want to load the video via URL or use Asset Bundling.

Setting up Panoramic 2D and 3D Video in Unity

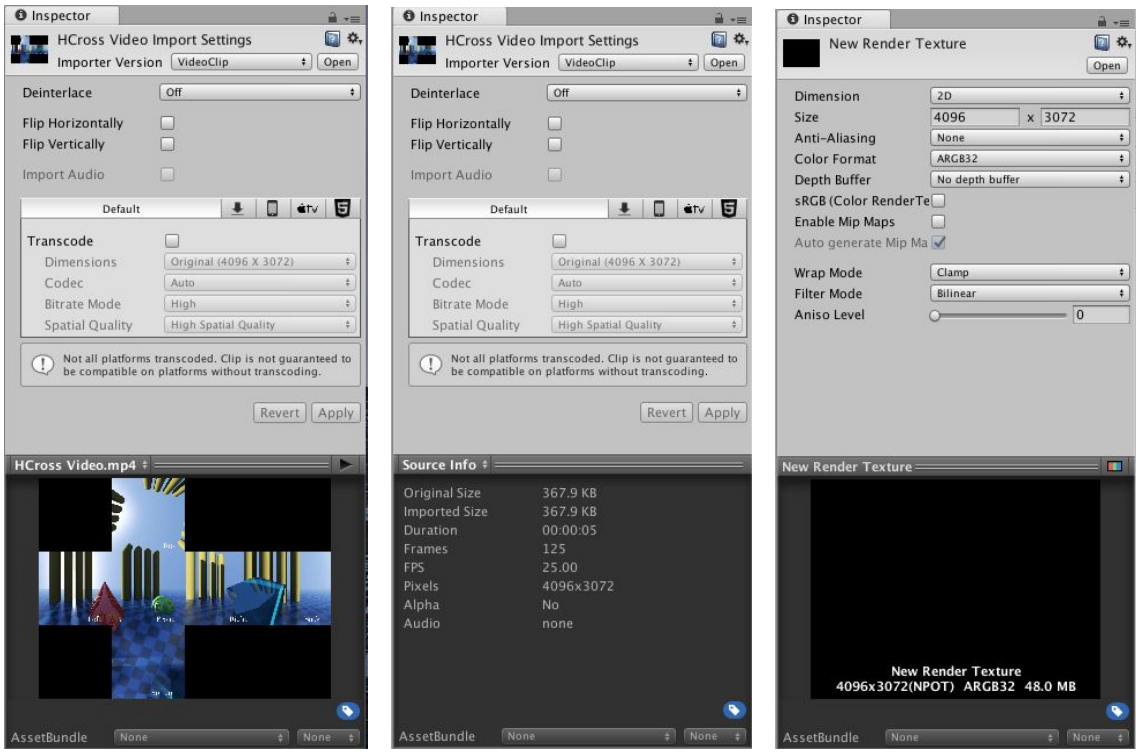
Unity supports 180 and 360 degree videos in either equirectangular (latitude longitude) or cubemap (6 frames) layouts.

Conceptually, displaying panoramic video in Unity can be achieved in 3 steps regardless of the type of panoramic video that you may have.

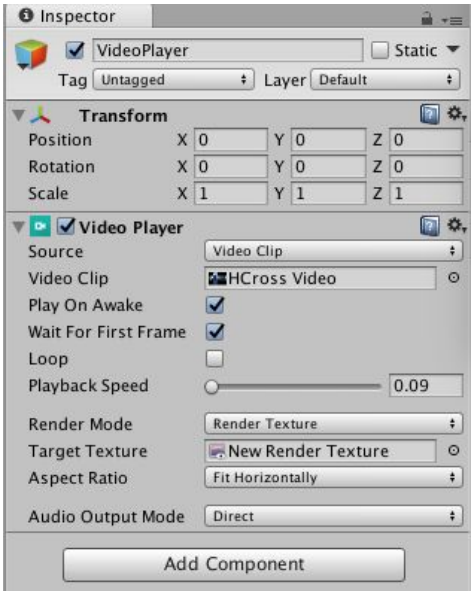
1. Set up a Video Player to play back the video source to a Render Texture.
2. Set up a Skybox Material which will receive the RenderTexture.
3. Set the scene to use the Skybox Material.

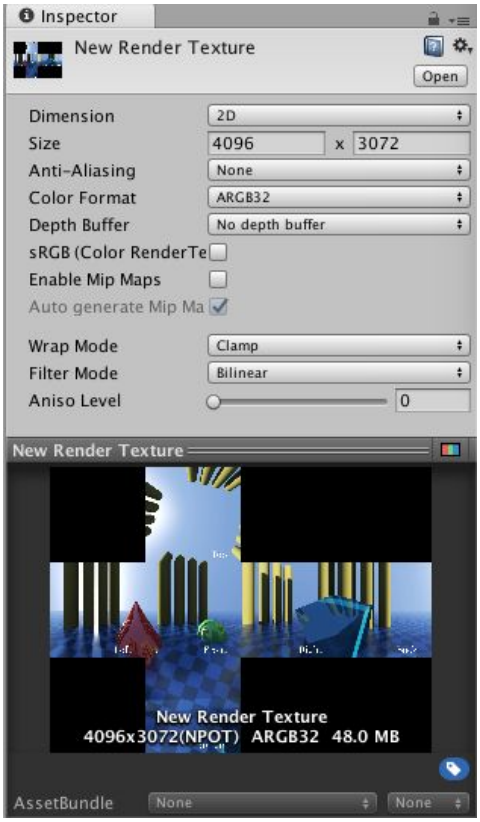
Video Player Setup

Once the video has been imported into Unity as an Asset, a Video Player can be created for it by simply dragging the video asset to an empty area of the scene hierarchy. By default, this will set up the component to play the video fullscreen for the default camera which is not exactly what we want (go ahead and try it by pressing Play). We will change this behaviour so that it is rendered to a Render Texture so we can control exactly how it is displayed. Do so by first creating a Render Texture from the Assets->Create menu. Set the Render Texture's Size to match that of the video exactly (the dimensions of the video are given under the "Source Info" section in the Preview UI in the Video Asset Inspector) and set the Depth Buffer option to No Depth Buffer.



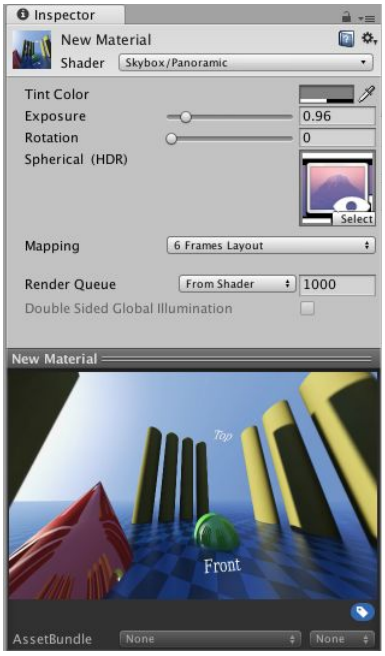
Now, back in the Video Player Inspector, switch the Render Mode to Render Texture and drag the newly created Render Texture from the Asset view to the Target Texture slot. You can verify that this is functioning correctly by entering Play mode. Nothing will be rendered to the Game view at this point, but if you select the Render Texture asset, you should see that its content is being updated with the video frames.





Create a Skybox Material

In order to render the panoramic video as a backdrop to our scene, we will replace the default Skybox with the video content. Do so by creating a new Material from the Assets->Create menu. Set the Material's Shader to Skybox/Panoramic. Drag the Render Texture from the Asset view to the texture slot in the new material. In order for the panoramic video to be properly displayed, you will need to correctly identify the type of content in the video. For cubemap content (such as a cross and strip layout as is common for static skybox textures) select the 6 Frames Layout Mapping. For equirectangular, choose Latitude Longitude Layout Mapping and then either the 360 or 180 degree sub-option depending on if the video covers a full 360 degree view, or just a front-facing 180 degree view. Once correctly set up, you should be able to pan around in the Preview at the bottom of the Material inspector and check that the 360 or 180 content looks correct.



Render the Panoramic Video to the Skybox

The final step is to connect the Skybox material just created to the scene. This is done by opening up the Settings panel from the Window->Lighting menu. Simply drag and drop the new Skybox material asset to the first slot under Environment. At this point, pressing Play should show the 360 video as a scene backdrop on the Skybox. Changing the scene camera orientation will show a different portion of the Skybox and therefore a different portion of the panoramic video.

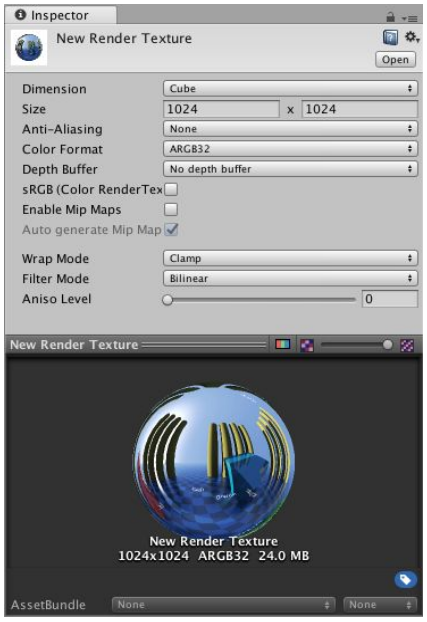
Notes on 3D

If you have Virtual Reality Support turned on in the Player Settings, an extra 3D Layout option will be available in the Skybox/Panoramic material. If your source video has stereo content this option should be turned on. Use the Side by Side settings if the video contains the left eye's content on the left and the right eye's content on the right, or, choose Over Under if the left and right content are positioned above and below one another in the video. The correct half of the video will be used when rendering each eye's content in VR.

For non-360 3D video (where a Skybox material wouldn't be used), the same 3D Layout option is available directly in the Video Player component when using the Camera Near/Far Plane Render Modes.

Alternate Render Texture Type for Cubemap Videos

An alternative to the aforementioned steps is available when working cube map videos. Rather than have the VideoPlayer render to a 2D Render Texture preserving the exact cube map layout as seen in the original video, it is instead possible to have the Video Player render directly to Render Texture Cube. This is achieved by simply changing the RenderTexture asset's dimension from 2D to Cube. In this case, the Render Texture's Size should be set to exactly the dimensions of the individual faces of the source video (for example, if you have a 4x3 horizontal cross cubemap layout video with dimensions 4096x3072, the Render Texture's Size should be set to 1024x1024). When the Video Player renders to such a Cube Target Texture, it will assume that the source video contains a



cube map in either cross or strip layout (automatically determined by the video aspect ratio) and fill out the Render Texture's faces with the correct cube faces. The resulting Render Texture Cube can then be used as a Skybox by using a Skybox/Cubemap material in place of the Skybox/Panoramic material described above.

Notes on Video Dimensions and Transcoding

Cubemap (6 Frame Layout) 2D videos should have aspect ratio 1:6, 3:4, 4:3, or 6:1 depending on face layout. Equirectangular (Latitude Longitude Layout) 2D videos should have an aspect ratio of exactly 2:1 for 360 content or 1:1 for 180 content. Including 3D content will require double either the width or the height of the video (corresponding to Side-by-Side or Over-Under layouts). Keep in mind that many desktop hardware video decoders are limited to 4K resolutions and mobile hardware video decoders are often limited to 2K or less which limits the resolution that can be played back in realtime on those platforms. Video transcoding can be used to produce lower resolution versions of panoramic videos but precautions should be taken to avoid introducing bleeding at the edge between left and right 3D content, or, between cube map faces and adjacent black areas. In general, authoring video in power-of-two dimensions and transcoding to other power-of-two dimensions is suggested to reduce visual artifacts.