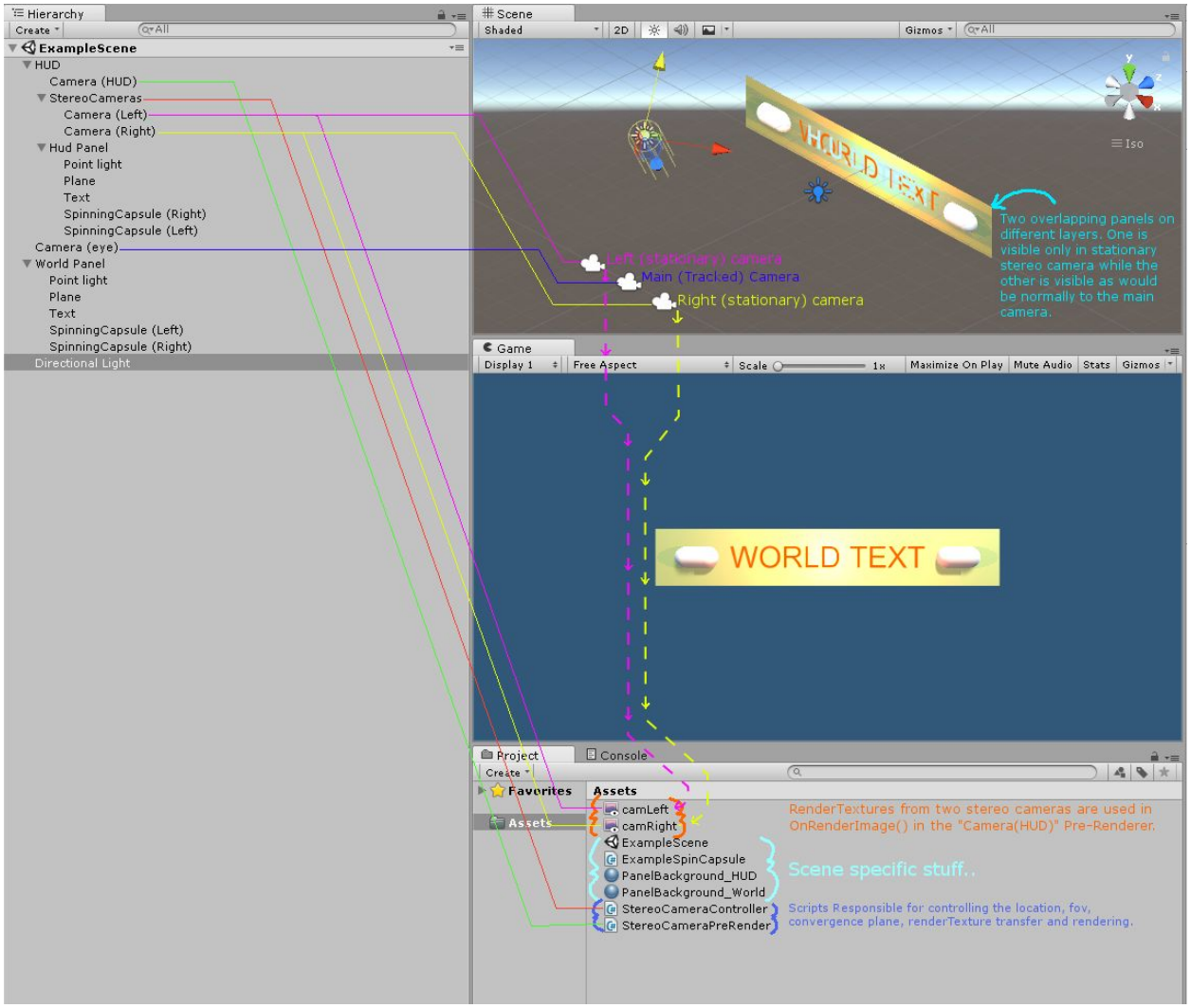
# Stationary Stereo Camera

**Workaround To Disabling Head-Tracking On A Camera.**

Due to some internal limitation with Unity I found that I was unable to stop Unity from stealing all of my cameras and changing their transforms without supplying any option to disable the behavior. So out of frustration I just spent the entire night coming up with my own solution.

**Here's how it works:**



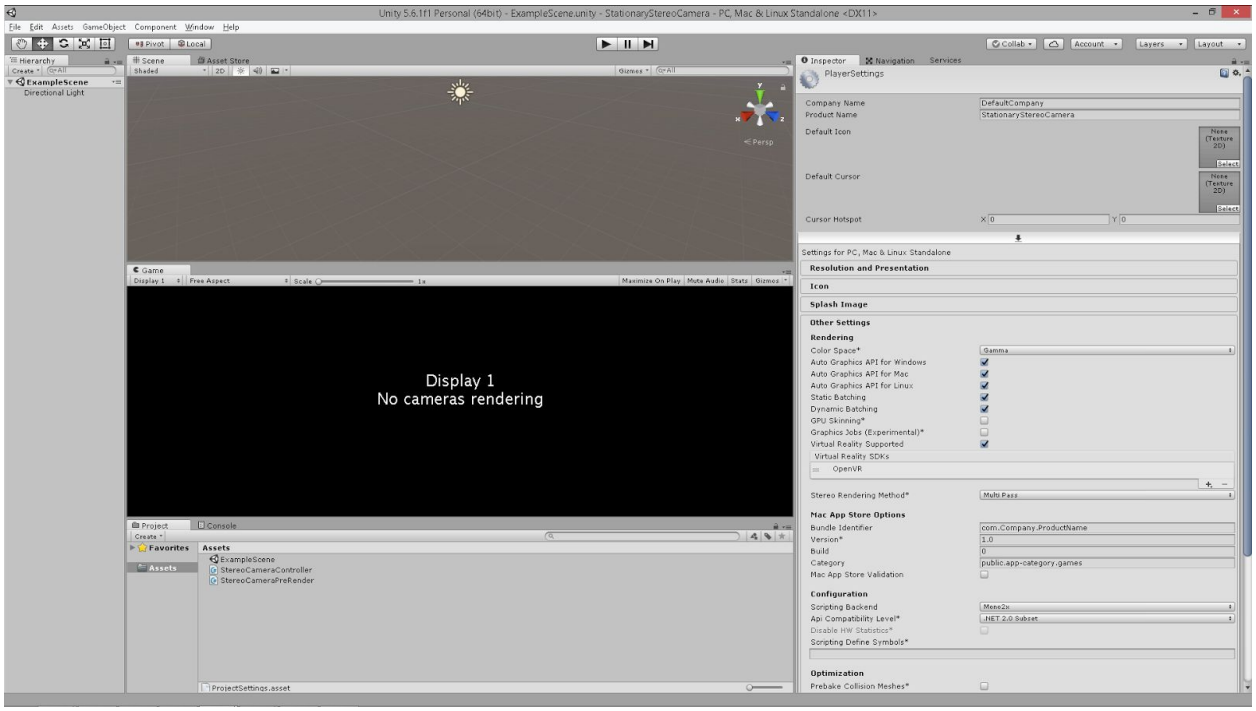

The "Camera (HUD)" is a tracked camera, meaning that Unity will assume that it should follow the gyro of the HMD. This is a way to spoof a tracked camera and feed it renderTextures from our own right and left cameras. So anything that this spoofed and head-tracked camera "sees" is irrelevant and will be completely overridden by renderTextures from the two stationary (left and right) cameras! By setting the spoofed camera's culling mask to "Nothing" and then feeding

in the two renderTextures from our right and left eye cameras, the spoofed head-tracked camera will display the two stationary cameras as the left and right eye of to HMD, while being overlaid on top of the main camera.

This allows building a stereo camera from the ground up and gives more control over parameters such as camera position/rotation, eye separation, field of view, convergence plane, and even some renderTexture specific stuff such as anti-aliasing and resolution.

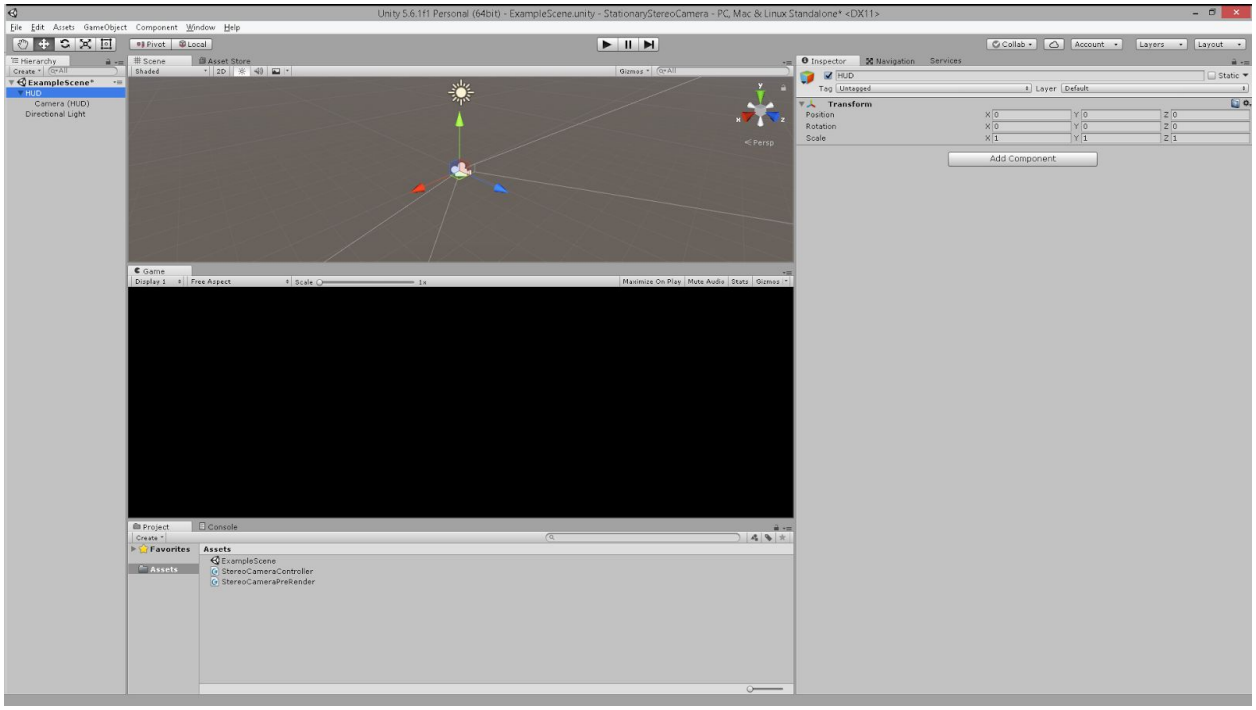
Note that this does support transparency as well as lights and shadows on the HUD layer, so it acts and looks pretty much like any other camera.

**Configure the scene from scratch (OpenVR already installed):**



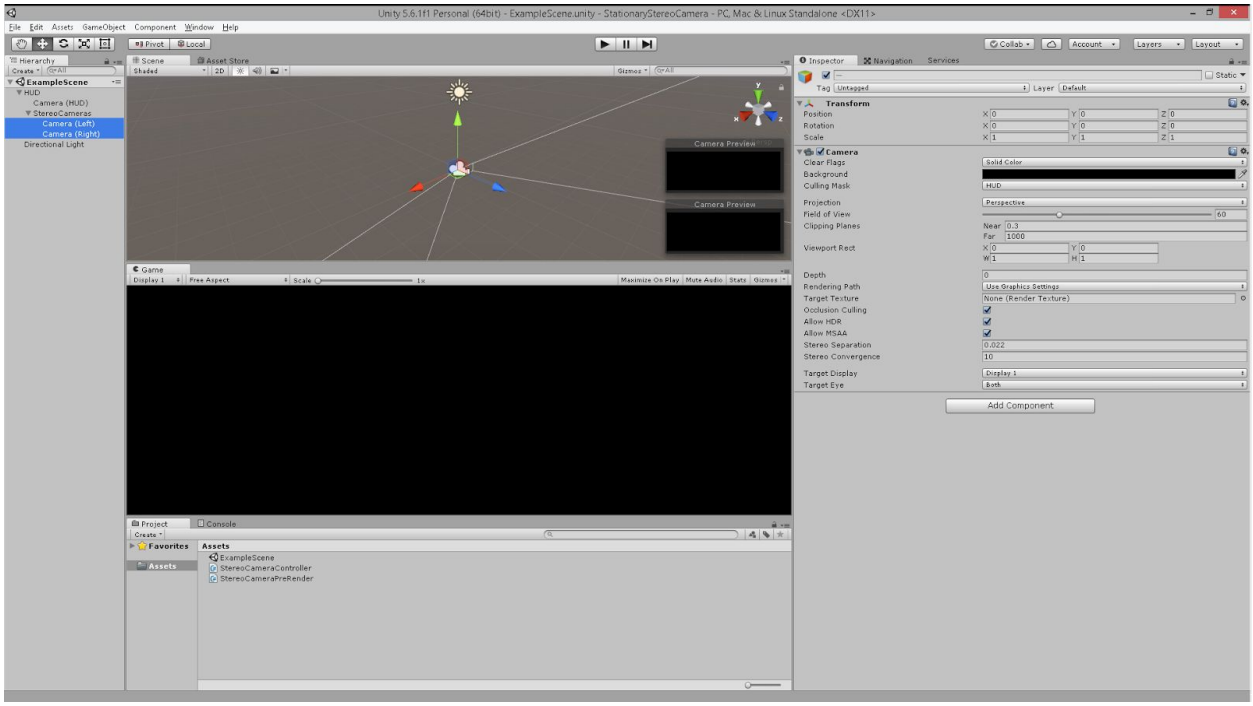

*Step 1 - Scene creating and virtual reality setup.*

- Create a new Unity project and name it StereoCameraOverlay.
- Press Ctrl+S and save the scene as “ExampleScene”
- Delete the “MainCamera” from the project hierarchy.
- Navigate to Edit > Project Settings > Player.
- Check the box “Virtual Reality Supported”.
- In “Virtual Reality SDKs” ensure OpenVR is the only item in the list.



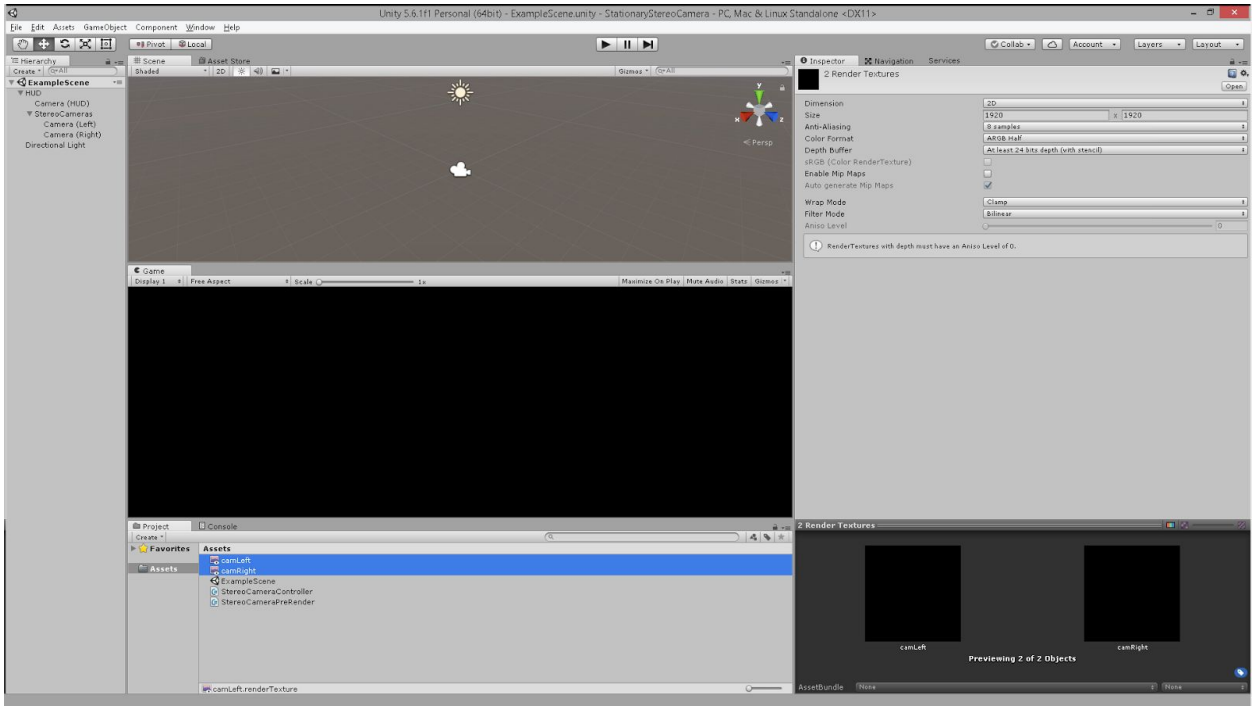

*Step 2 - Creating the HUD parent and camera objects.*

- Add the two scripts (StereoCameraController.cs and StereoCameraPreRendere.cs) to the assets folder.
- Create an empty game object and name it “HUD” (This will be the parent of our HUD world and stationary cameras).
- Ensure the transform position and rotation are all set to zero.
- Create a camera and name it “Camera (HUD)” (this is the tracked hud camera that renders nothing of it’s own and has renderTextues fed into it).
- Remove the “Audio Listener” component (it is redundant and will cause warnings when the main camera is added).
- Drag that object into and make it a child of the “HUD” object.
- Ensure the transform position and rotation are all set to zero.
- Set the Clear Flags to Don’t Clear so it doesn’t render any colors or sky.
- Set Culling Mask to Nothing so it doesn’t render any scene objects.




*Step 3 - Create and setting up the stationary stereo camera.*

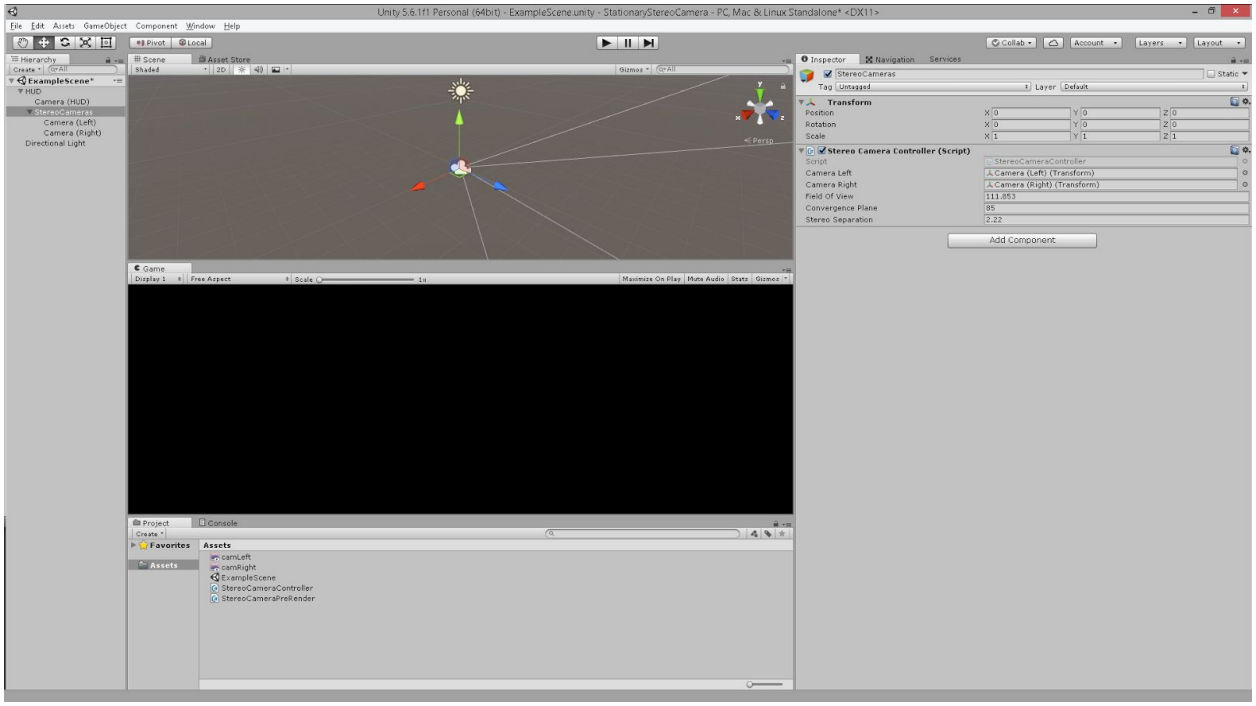
- Create an empty game object and name it “StereoCameras” (This will be the parent of our left and right eye stationary cameras).
- Drag that object into and make it a child of the “HUD” object.
- Add two new cameras and name them as “Camera (Left)” and “Camera (Right)” (these are the cameras that only see things on the HUD layer.
- Drag both cameras into the “StereoCameras” object.
- Ensure the transform positions and rotations are all set to zero (you may separate them on x axis a little if you wish - this will not affect how the script positions them at runtime).
- Remove “GUI Layer”, “Flare Layer” and “Audio Listener” components from both cameras as they are not required.
- Set the Clear Flags to solid color, and select a color of black and no alpha (#00000000).
- Add a new layer called “HUD” (any object on is layer will be visible to these cameras.
- Set Culling mask to the “HUD” layer only.




*Step 4 - Create renderTextures for left and right eyes.*

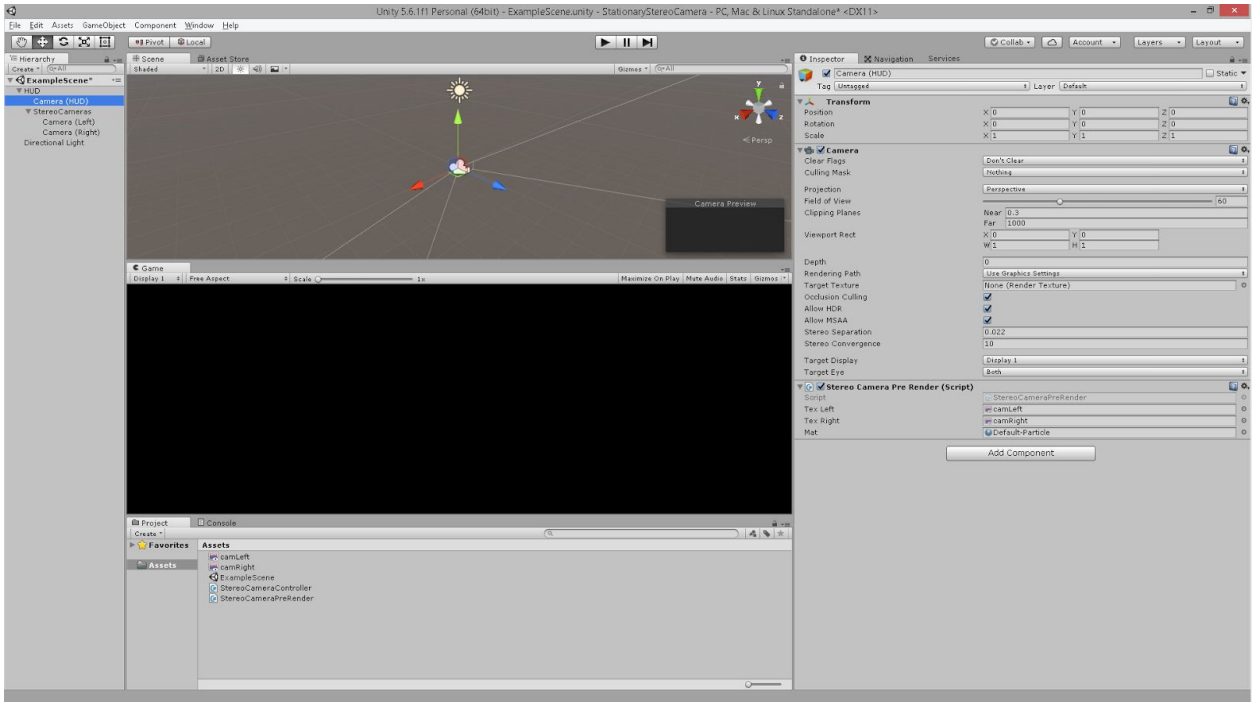
- Right click in your project's asset folder and create a new renderTexture (we will copy it for both eyes later on).
- Set the Size parameter to 1920x1920 (This is what I chose but can be changed as desired).
- Set Anti-Aliasing to some value (I chose 8 samples but again, you can decide).
- Set Color Format to ARGB Half, (I believe this is the native format used by cameras??).
- Now, just copy that renderTexture (Ctrl+d) and name the two as "camLeft" and "camRight".
- Finally, select each of the two cameras in StereoCameras and add the renderTextures to the to their Target Texture, ensure the "camRight" texture is placed in "Camera (Right)" and "camLeft" is placed in "Camera (Left)".

*Now that those two cameras have renderTextures attached, Unity will no longer try to change their positions or any other parameters. So all we have to do now is use those renderTextures as the inputs for our tracked camera that is sending video to the HMD.*



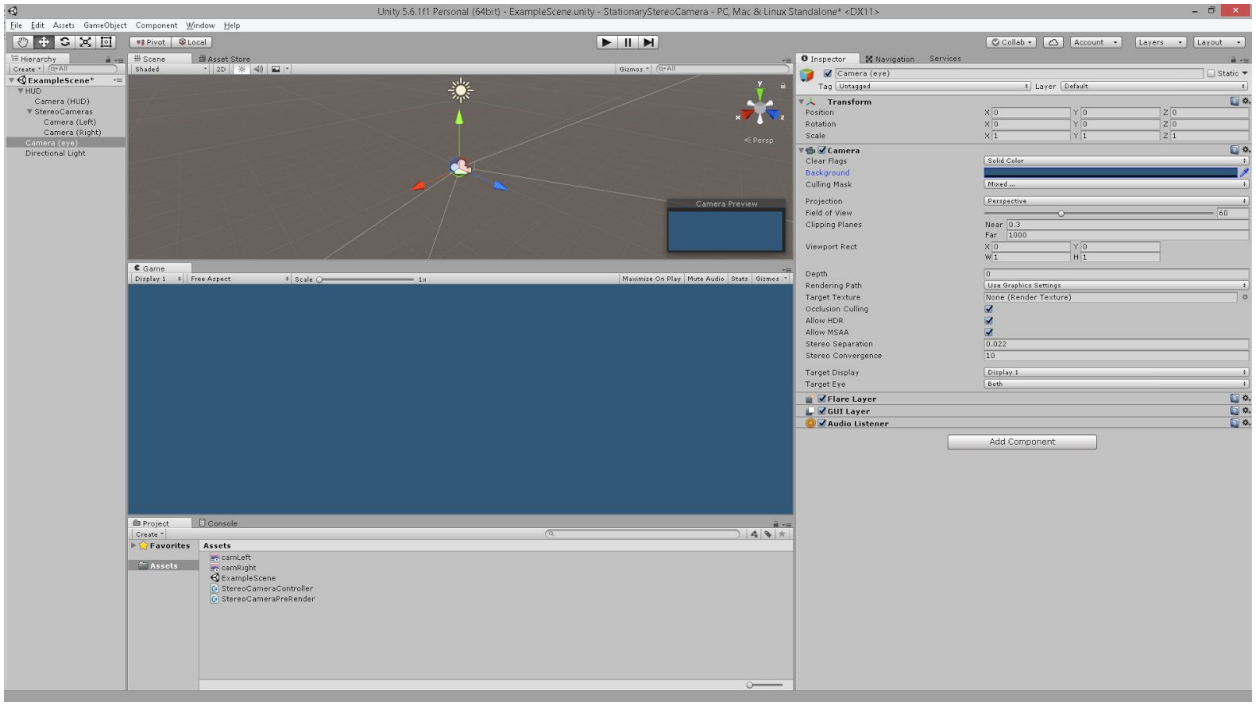

*Step 5 - Adding the StereoCameraController script and linking the two stereo cameras.*

- Place the StereoCameraController.cs script onto the StereoCameras object.
- Drag each Camera (Left) and Camera (Right) and place them in the script's camera variables.
- The other default values are there to be adjusted but should work ok for most projects (as well as this example).



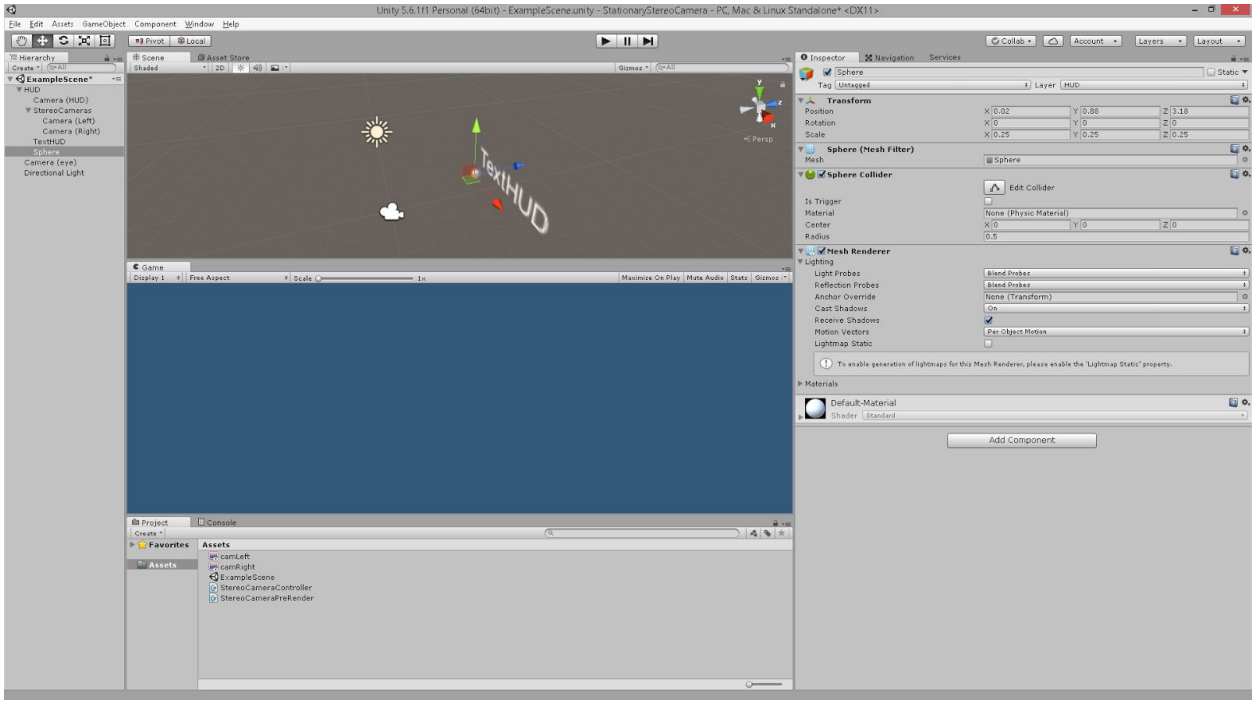

*Step 6 - Adding the StereoCameraPreRenderer script and linking the two renderTextures.*

- Drag the StereoCameraPreRendere.cs script onto the “Camera (HUD)” object.
- Drag each “camLeft” and “camRight” renderTextures and place them in the script’s corresponding “TexLeft“ and “TexRight” variables.
- Click the small circle button to the right of “Mat” and select “Default-Particle”.



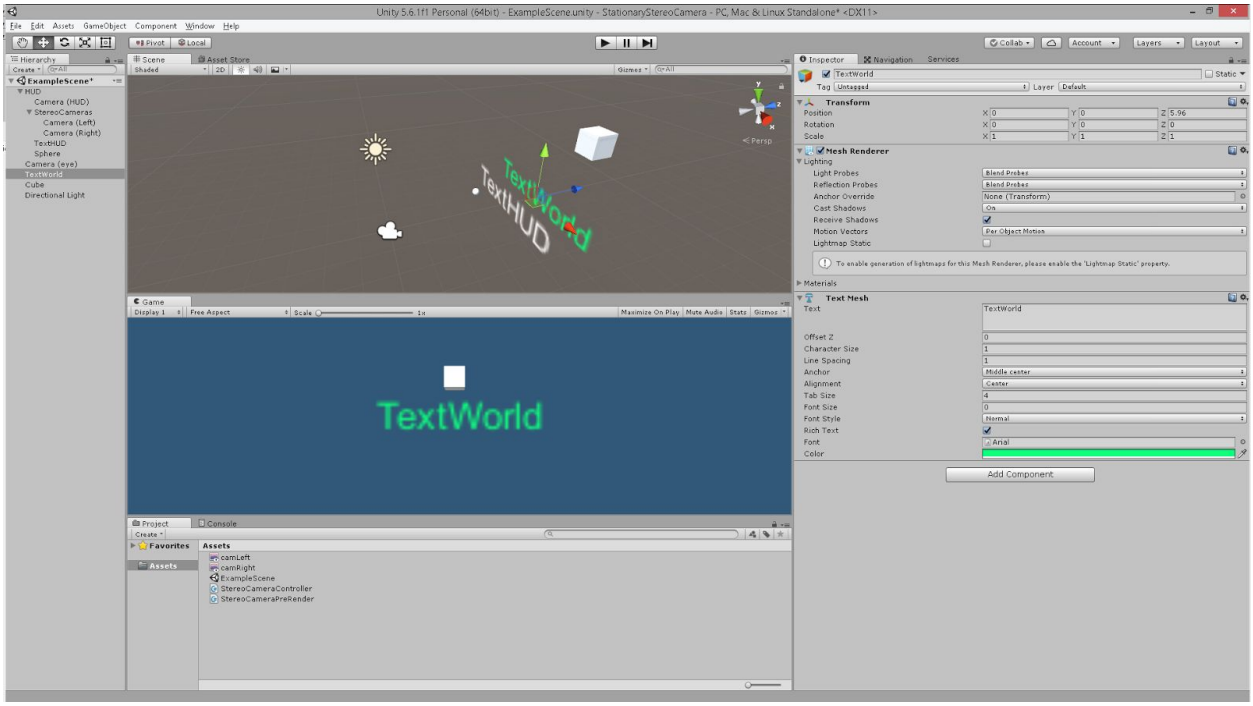

*Step 7 - Adding the main camera.*

- Create a camera and name it “Camera (eye)” (this is the tracked main camera that renders the world to the HMD).
- Set the Culling Mask however you wish except make sure that the “HUD” layer is not selected.
- The rest of that camera’s configuration is totally up to you.




*Step 8 - Adding some visible HUD objects.*

- *Add some 3dText or meshes to your scene, then for each of them simply change their layer to “HUD”. This will make them visible to only the two stereo cameras.*
- *Make those object children of the “HUD” object in the hierarchy (optional really..).*
- *Remember that this camera is stationary (in this example), so you must place all of your HUD object in front of the camera so they are in it’s field of view.*




*Step 9 - Adding some visible world objects.*

- Just as you normally would add objects to your scene, do so and place then anywhere in the scene (I have placed them near my HUD to demonstrate how the two cameras can't see the various layers).
- Now press the Play button and if everything went well you should see that TextHUD is visible overtop of the main camera and best of all, it's completely stationary and free from any noticeable jerky motion when rotating your head quickly.

Something to consider when it come to lighting is that you may want to manage the Culling Masks for lights and make sure your main world's lights do not include the HUD layer (especially the Directional Lights and area lamps). Additionally you can add lighting to you HUD layer by setting the lamp's Culling Mask to only HUD layer.

Please feel free to use this example any way you wish (without restriction of any kind) as well as redistribute, modify and share it with all of your friends and co-workers.

Legal notice:
By downloading or using any resource from this example you agree that I (the author) am not liable for any losses or damages due to the use of any part(s) of the content in this example. It is distributed as is and without any warranty or guarantees.

Project by: Grant Olsen (jython.scripts@gmail.com)
Creation year: 2017