360 VR Player Documentation

I. Introduction

360 VR Player is the perfect base to start a VR project which is using 360° images or videos, I tried to make something easy to use even if you're not a developer. It contains a spherical screen, a camera and an UI (an interactive reticle, a bended screen and a look at based interactive UI). I used it for Goodle Cardboard (you can download it here) and GearVR (and this one here) applications, but it could also be used for Oculus Rift or HTV Vive.

The camera rotation is controllable with keyboard, mouse and gyroscope for mobile. The look at based UI allow you to load another picture or video when the user watch the UI for some seconds and I've also made a fade in/out system you can use when you load the video for example.

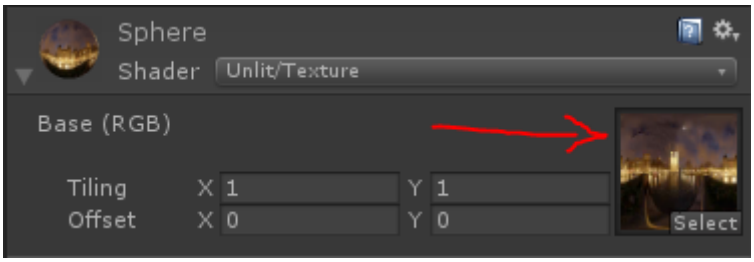
It works on Windows, Mac, Android, iOS and WebGL (I assume it should work on Windows Phone but I couldn't try it).

II. The Spherical Screen

Nothing special about this sphere, just that the normals are inverted to be able to see inside easily. In this example there are just images displayed, but if you want to play videos it's easy, for myself I use Easy Movie Texture and it works perfectly.

If you need 360 videos samples you can download them on YouTube with this software : 4kVideoDownloader.

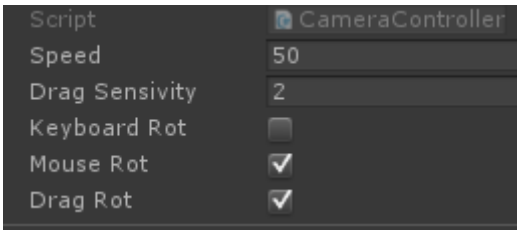
To play a video on the sphere screen (**only for PC & Mac**), you just have to use this video as Movie Texture, replace the image texture by the video:



Then add the script *PlayMovieTexture* (this script doesn't work on mobile platforms), it'll play the video at the start.

III. The Camera

As I said the camera is controllable with keyboard, mouse and gyroscope, it's easy to set up, you can activate or deactivate each function as you want and also change speed of the rotation.



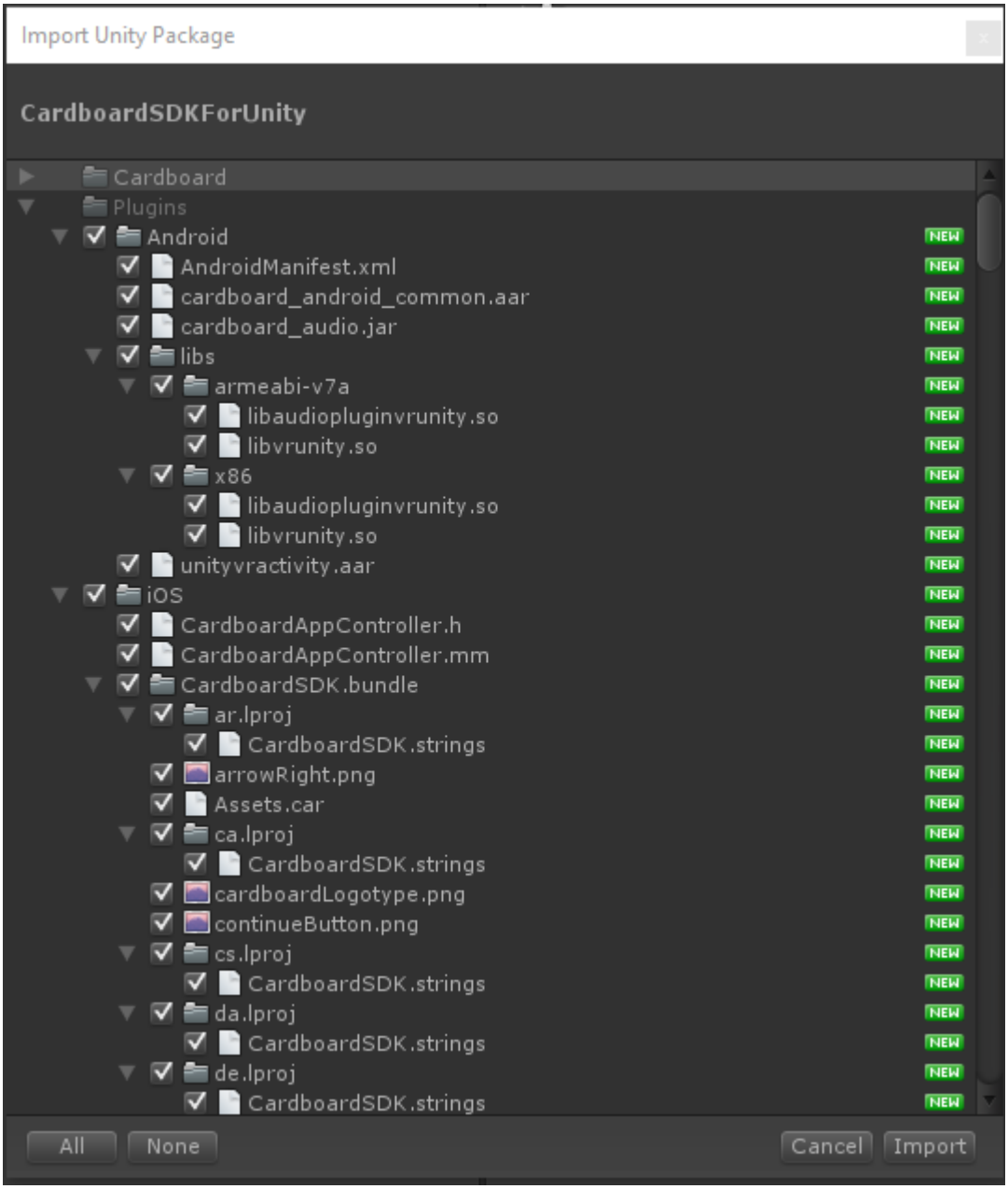
The script "*CamVision*" is used to know where the user is looking at, but I'll talk about it in the interactive UI part.

Little tip: if you want to see what you'll see in game, in the scene tab, select the camera, then in the GameObject tab above, click on "**Align View to Selected**". After that your scene view will be the same that your camera, so if you move your mouse while holding right click, you'll rotate like the camera in game, it can be useful sometimes.

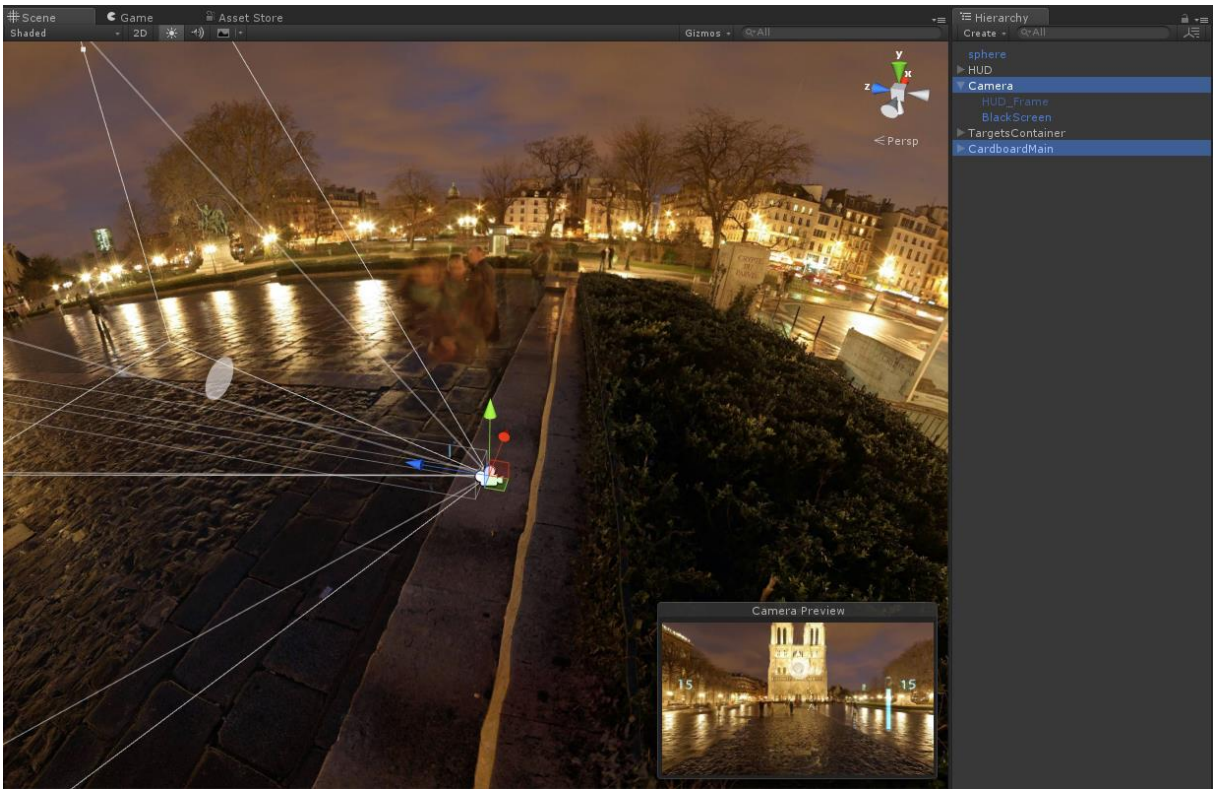
If you want to use this project with a VR camera, it's easy you just have to replace my camera by the camera from the Cardboard SDK (download it here and follow the instruction) or the GearVR SDK (download it here and follow the instruction) for example, but **be careful you can't have both sdk in your project**.

Add Cardboard SDK, step by step:

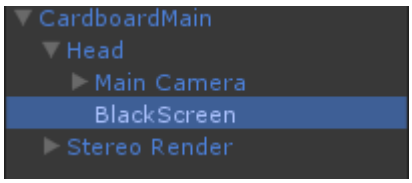
1. Download the Cardboard SDK here.
2. Import everything in the .unitypackage



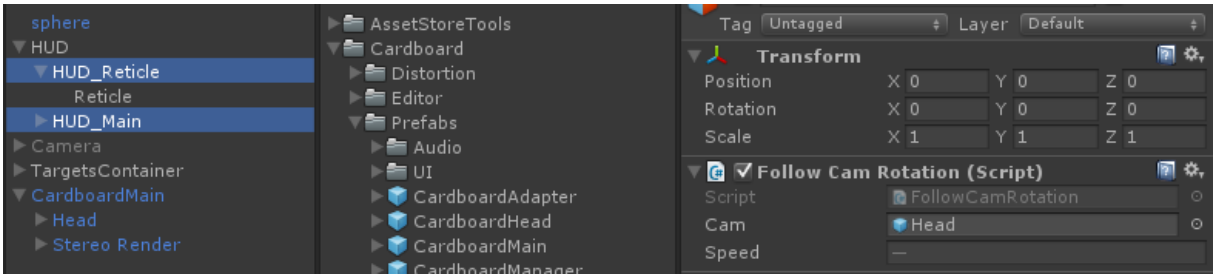
3. Add CardboardMain prefab in the scene (Cardboard/Prefabs/CardboardMain) at the same position that my camera (0, 0, 0).



4. Put the BlackScreen GameObject (from my camera) as a child of the CardboardMain's Head (see image below). **This "Head" is the part which will rotate when the head tracking is enabled** (it's enabled by defaut)

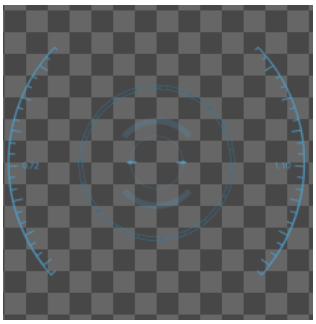


5. Add the *CamVision* script to this Head asset, without this the UI won't work with the Cardboard camera. Replace the Cam target in the *FollowCamRotation* script (for HUD_Reticle and HUD_Main) by this Head.



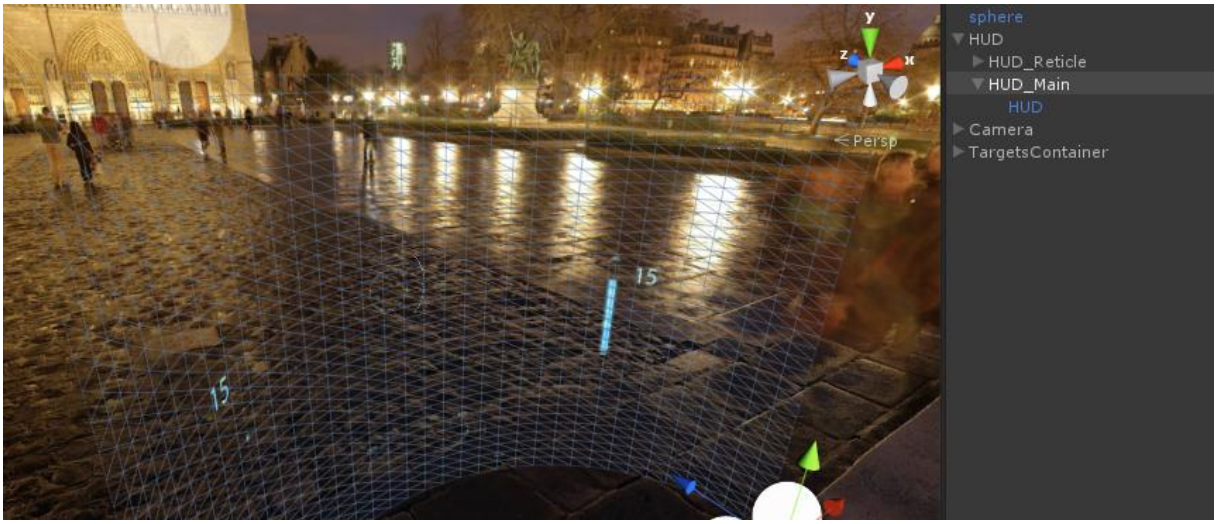
6. Desactivate or delete my camera.

IV. The User Interface

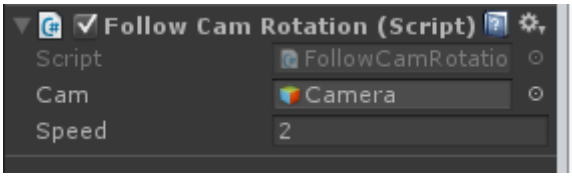


The reticle is png sequence with an opening, an idle state and an interaction state. The interaction state is launched when the animator receive the "*LookSomething*" trigger.

There is also a bended screen where you can display images, images sequences or videos.



On those assets there is a script named "*FollowCamRotation*" which makes them follow the camera rotation with latency.



The more speed is high, the more the HUD will follow the camera quickly.

And finally there is an interactive target, for the example it's just two disks, one for the background and the other one which scale when the user look at it. There is a script on it, "*ChangeRoom*" which change the sphere's texture when the foreground disk is about 90% of the background one.

```
55      IEnumerator FadeChangeRoom (){
56          isChanging = true;
57          transform.GetChild(0).localScale = new Vector3(0,0,0);
58          StartCoroutine(blkScreen.FadeOut(0.5f));
59          yield return new WaitForSeconds(0.5f);
60          sphereScreen.GetComponent<Renderer>().material.mainTexture = textureSphere[i];
61          if(i == textureSphere.Count-1)
62              i=0;
63          else i++;
64          isChanging = false;
65          StartCoroutine(blkScreen.FadeIn(0.5f));
66          yield return null;
67      }
```

If you use videos, you can easily load another one by changing the line 60, for example with Easy Movie Texture you just have to change it by something like *'MediaPlayerCtrl.Load("Video02.mp4");'*.

As you can see it on lines 58 and 65, there is a *FadeOut*/*FadeIn* function to hide the texture change, it's about 1sec here (0.5s + 0.5s) but you can change it or suppress it as you want (everything which concern those function is in the *BlackScreen* class).

If you want to play an animation when the user look at an object, you just have to add the script "*PlayAnimOnSight*" on it, add a collider, change his Tag for "*AnimTarget*" (for the targets which scale it's "*ScaleTarget*") and set up a Trigger on its Animator with the name "*OnSight*".

Those tags are managed in the camera script "*CamVision*":

```
14          if (Physics.Raycast(transform.position, transform.forward*50, out hit)) {
15              if(hit.transform.tag == "ScaleTarget")
16                  hit.transform.GetComponent<ChangeRoom>().IncreaseChildSize();
17              if(hit.transform.tag == "AnimTarget")
18                  hit.transform.GetComponent<PlayAnimOnSight>().OnSightEnter();
19          }
```