Unity version to use: **2018.2**

Please stick to this version ONLY, since this is the one that's most stable and works for post-processing.

How To Change Version

1. Open your current version of the project
2. Select Assets - Export Package, check everything and click Export
3. Find and download Unity version 2018.2 from the Unity Hub
4. Open the 2018.2 version
5. Assets – Import Package – Custom Package. Find your exported asset and import in the scene.

Import Animation from MAYA into Unity

The animation file type needs to be FBX. or OBJ. for unity to work. Here, we use FBX. for our project.

1. Open the Maya animation
2. Clean your outline and hierarchy, delete anything that you don't want to include in the scene, which including lights, groups
3. Under the animation tab, select bake animation
4. Clean your history. File – delete by type – delete history
5. Select everything in your hiarchy, and go into File – Export – export everything – click the square and check whether everything looks right
6. Under the file type, find FBX.
7. In the drop down menu, select FBX. 2011. (tested as the most stable version)
8. Other thing should be default, but double check attributes to ensure
9. Export the mesh with the animation as FBX.
10. Now open Unity, you can either drag and drop the file or go into Assets – Import Packege and find the FBX. file you just made.
11. Now everything should be in the project – Assets. Including the mesh, animation, textures, avartar and etc.
12. Note, the texture needs to import separately in this case, just simply drag and drop the unwrapped image into the Asset. Make a folder named: material for these texture files.
13. Under the same folder, right click empty space and Create – material, name it like orca_tex.
14. You can drop the image file in the little space in front of the properties called Albedo, or select it from the menu view, which is the little circle thing.
15. Now drag the orca_tex material you just made on to the whale, HOORAY we got our orca.
16. Do the same for the rest of the texture files.

Put the animation on to the orca.

In order for out orca to animate, we need several things. An Animator, Animator controller, Animation clips.

By default, your orca mesh doesn't have any of these. So, let's create them

- Go to the *Animator* tab. If you can't find it, *Window – Animation – Animator*.
- This is what called a statmachine, where each animation clip is a **state**, connect by **transitions**, and we can setup **parameters** for each transitio.
- Now you can find the animation clip we exported earlier from MAYA, in my case it's called "Take01" and drag it to the statemachine. We now create a state called "Take01"
- A yellow transition is a **default transition**, which means when you hit play, this is the first state to activate no matter what. Usually, the default state is an idle animation for the character.
- In order to make a transition, select a state and right click *Make transition*, and connect to the next state.
- We can make our take01 as the default state by right click this state, and select *Set as layer default state.*
- Try hit play and the orca should have animation take01 playing.

Now, we can create some other animations using Animation tool

- Goto the *Aniamtion* tab. If you can't find it, *Window – Animation – Animation*
- Click Create to create the first animation clip, name it as "swim_towards"
- Then you have a timeline just like in MAYA. We need some keys from the in place animation and manually key the position attributes in the Animation, so our orca can swim towards.
- Find the take001 at the top left hand corner, and you can see we have lots of keys here. Drag and select all the kays, Ctrl+C copy them.
- Switch to your swim_towards timeline and Ctrl+V paste. Now, *Add Property – Transform – Position.*
- Here you can key the start and end position in the direction you want. Click the red rec bottom at the first frame, you can see now you have a key frame at the start. Scroll to the end of the animation and move the orca to the end position. Now we have the second key from. Finish this by click the rec butoom again.
- Now if you try to play, the orca should swim between these two position frames.
- You can now add this "swim_towards" animation to the statemachine and make transitions.

Manually adding the Animator to the any mesh.

- Select the gameobject, and *Add Component*
- Type "Animator" and select the *Animator*
- In the Animator menu, make sure you have the *orca controller* in the *Controller*, and the *Avatar* in the *Avatar*. If you can't find the controller in the menu, simply create one in the Asset by right click, *Create – Animation Controller*, and name it.
- The other things can leave as default.

Adding parameters to control the transitions.

Inside of the Animator, you can create parameters such as *bool, float, triggers* etc.

Here we will add a simple parameter calles “look”, so the orca will play the swim inplace animation(take001) only if we look at it.

- Click the + buttom on the right hand corner, and select Bool. Name it “look”.
- Right click inside of the statemachine, *create state – empty*. This is a state holder that we need when we are not watching the orca. It can be anything, but now we just use an empty state. **Make it the default layer.**
- Make a transition between the default state and the inplace swim state, in my case, it called take001. You need another transition from take001 to the default state too. Because we need this transition when the viewer stop lookin at the orca.
- Now select the transition goes from the default state to take001, in the **Inspector tab**, under the **Conditions** layer, add a new condition.
- Select “look” and set the bool to **true**. As if the transition will activate, when the look is true.
- Do the same for the transition from take001 to default state, only change the look to **false** this time.
- Now you successfully set up your first state. We will code it later.

Now lets set up the scene with the camera rig with the raycaster script.

Your camerarig should have similar hiarchy as this:

- Select the main camera, *Add component – VREyeRaycaster.*
- You are welcome to look and read throught the code, I have added comments and descriptions for important lines.
- Each *private void* method is a function activation that we might need in the future. As for this project we only cares about the *EyeRaycast* method
-

Now lets set up our interactive Item

- Select the gameobject orca, *Add component – VRInteractiveItem*
- This is the code tells the camera which object to react to. The method we care about is called *OnOver*, which means when it knows whenever the viewer look over to the interactive item.

Set up the interact methods and start coding!

We need another code on the **interactive object orca**, it's called VRInteractiveItem, *Add component – ExampleInteractiveItem*. Open the code

- **Public** class is where we set up the object we need for functions. Public, is where we can choose attributes outside of the code, like in the panel. **private** is where the code decide what to look for. Here, we need bunch of classes such as our camera and Gameobject
- All the **void, private void** are functions that we triggers actions
- An **update** function check the status every single frame. we need it for checking distance since the object is moving every single frame, and we need to check that.
- The two functions that we care about are **HandleOver** and **HandleOut**, which are the function the code check whether the viewers are looking or not.
- Inside of the **private void HangleOver**, is where we will set our transition parameter bool value "look" to true, so the transition can go from the default state to take001
- Do the same for function HandleOver, but set the bool to false.
- Try run the game now.