Documentation

On

“Augmented Reality”

Submitted By:

Shradhha Bhuju

OCTOBER, 2020

TABLE OF CONTENTS

1. INTRODUCTION

Augmented Reality(AR) is one of the biggest trends right now. AR is an interactive experience of a real-world environment where a virtual object becomes a part of the real world. An application similar to IKEA's Place and an AR portal are two Augmented Reality Applications made during the period of 3 months. Both of the applications were made using Unity and the AR foundation package.

2. OBJECTIVES

- To learn about Augmented reality
- To study how to use AR Foundation and Unity
- To Study how to make Augmented Reality Application for testing furnitures in real time.
- To study how to make an Augmented reality Application for making an AR portal
- To improve 3D modelling skills

3. Setting up Unity for Augmented Reality

There are mainly two types of Augmented reality platforms that can be used for AR development in unity and they are AR Foundation and VuForia. For 2 projects, AR foundation was used.

Setting up AR Foundation in unity

1. We need to install the AR Foundation Package from the Package manager. (Window> Package Manager)
2. After the ARFoundation is Installed, one platform-specific package must be installed. ARKit XR Plugin (Android) or ARCore XR Plugin(iOS)

More about Setting Up ARFoundation In Unity

A Video On Setting Up ARFoundation In Unity (recommended)

4. Some of the classes and methods used

4.1. ARRaycastManager

In the projects ARRaycast manager is used for performing raycast to the detected trackables.The ARraycast manager was used as a way to place the furniture in the Furniture AR project and door in AR Portal project.

More about ARRaycastManager

4.2. Vector 3

Vector 3 is a unity's class that is usually used for storing the 3D position of the of GameObject and even the directions. It has various properties, methods. More about Vector3

4.3. Vector 2

Vector 2 is similar to Vector 3 but Vector 2 represents 2D points and vectors while Vector 3 represents 3D points and vectors. In most of the cases Vector 3 is used instead of Vector 2. More about Vector 2

4.4. Instantiate

This class makes an instance of the object in the game. In other words it makes a clone of the given Object in the game.It is the Static method of the object class

Syntax:Instantiate(Object, Vector 3 position, Quarternion rotation, Transform parent);

More about Instantiate

4.5. Destroy

This does the opposite of the instantiate. it destroys the given object.

Syntax:Destroy(Object, float t = 0.0F);

more about Destroy

4.6. Sprite

Sprites are 2D graphic objects used for characters, props, projectiles and other elements of 2D gameplay.

More about Sprite

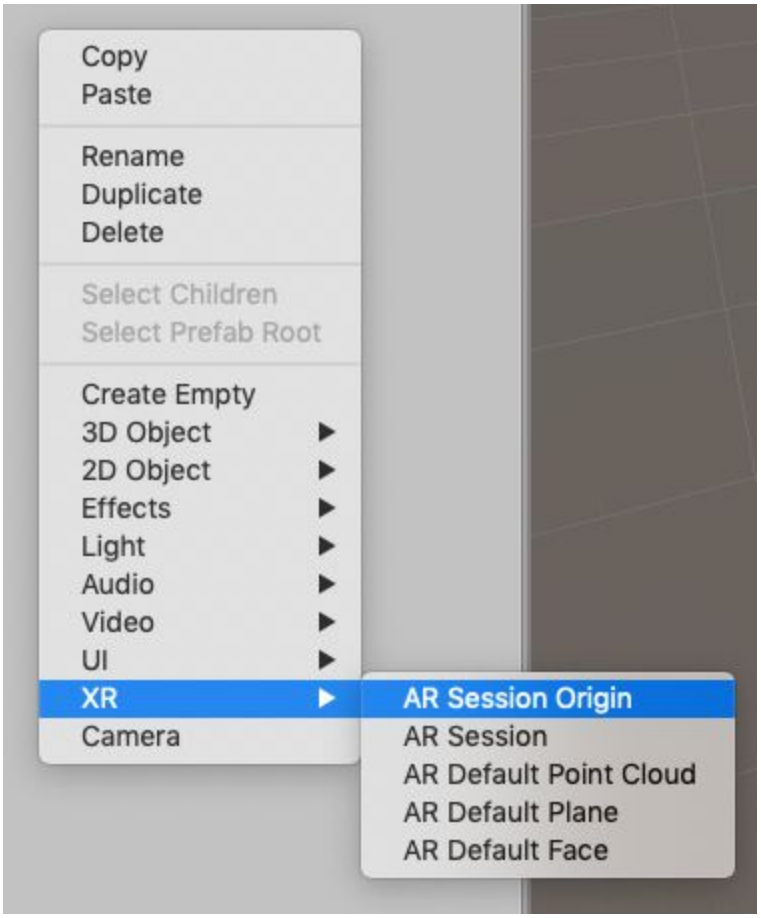
5. PROJECTS

2.1. Augmented Furnitures

This project is similar to IKEA's Place application where users can choose the furniture of their liking and augment it to the real world.

Basic Setup in Unity:

1. ARSession is needed to control the lifecycle of the application.
2. Another object called ARSessionOrigin is used to control the space in the application, this object also consist a camera object called ARCamera or AR Session Origin



3. A script attached to this ARSessionOrigin object to instantiate the furniture.
4. The ARSessionOrigin was renamed to **placementcontroller.cs**
5. placementcontroller.cs handles all the input from the user and place the AR object according to the user's touch position

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System;

[RequireComponent(typeof(ARRaycastManager))]
public class placementcontroller : MonoBehaviour
{



   public static GameObject SpawnedObject;


  // [SerializeField] private Material highlighttmaterial;

   [SerializeField]
   private ARRaycastManager aRRaycastManager;
   static List<ARRaycastHit> hits = new List<ARRaycastHit>();
   static int i = 0;

   void Start()
   {
       aRRaycastManager = GetComponent<ARRaycastManager>();
   }

   void Update()
   {
```

```
        if (!TryGetTheTouchPosition(out Vector2 touchposition))
            return;



                if  (aRRaycastManager.Raycast(touchposition,  hits,
TrackableType.PlaneWithinPolygon))
        {
            if (!IsPointerOverUIObject())
            {
                var hitPose = hits[0].pose;

                if (SpawnedObject == null)
                {

                                                        SpawnedObject   =
Instantiate(DataHandler.Instance.GetFurniture(),   hitPose.position,
hitPose.rotation);

                }
                else
                {
                                    SpawnedObject.transform.position  =
hitPose.position;

                }
            }



        }
    }
    bool TryGetTheTouchPosition(out Vector2 touchposition)
    {
        if (Input.touchCount > 0)
        {
            touchposition = Input.GetTouch(0).position;
```

```
            return true;
        }
        touchposition = default;
        return false;
    }
    private bool IsPointerOverUIObject()
    {
                PointerEventData eventDataCurrentPosition = new
PointerEventData(EventSystem.current);
                        eventDataCurrentPosition.position = new
Vector2(Input.mousePosition.x, Input.mousePosition.y);
        List<RaycastResult> results = new List<RaycastResult>();
             EventSystem.current.RaycastAll(eventDataCurrentPosition,
results);
        return results.Count > 0;
    }
}
```

DataHandler.cs

1. This script handles all the data
2. Each prefab will have a unique id
3. Through that itemid datahandler script will assign the prefab to each button.
4. It creates the buttons of the application from which user can choose their furnitures

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.Versioning;
using UnityEngine;
using UnityEngine.UI;

public class DataHandler : MonoBehaviour
```

```
{
    private GameObject furniture;
    [SerializeField] private MenupanelButton buttonPrefab;
    [SerializeField] private GameObject buttonContainer;
    [SerializeField] private List<Item> items;

    private int currentid=0;

    private static DataHandler instance;
    public static DataHandler Instance
    {
        get
        {
            if(instance==null)
            {
                instance = FindObjectOfType<DataHandler>();
            }
            return instance;
        }
    }
     private void Start()
    {
        LoadItem();
        CreateButton();
    }
    void LoadItem()
    {
        var Itemobj = Resources.LoadAll("Items", typeof(Item));
        foreach (var item in Itemobj)
        {
            items.Add(item as Item);
        }
    }
    void CreateButton()
    {
        foreach(Item i in items)
        {
```

```
                    MenupanelButton  b  =  Instantiate(buttonPrefab,
buttonContainer.transform);
            b.Itemid = currentid;
            b.ButtonTexture = i.itemImage;
            currentid++;
        }
    }

    public void SetFurniture( int id)
    {
        Destroy(placementcontroller.SpawnedObject, 100f);
        placementcontroller.SpawnedObject = null;
        furniture = items[id].itemPrefab;
    }

    public GameObject GetFurniture()
    {
        return furniture;
    }

}
```

Item.cs

1. this script was written to make the scriptableObjects of the furnitures
2. scriptableobject is a dynamic asset of unity
3. to make the button of the furniture
4. the furniture's prefab and a sprite image for the button is needed

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[CreateAssetMenu(fileName="Item1", menuName="AddItem/Item")]
```

```
public class Item : ScriptableObject
{
    public GameObject itemPrefab;
    public Sprite itemImage;

}
```

MenupanelButton.cs

1. this script will set the value of the button as the prefabs of the furnitures
2. this script describes what happens when the button is clicked
3. if a button of item id 1 is clicked then, Datahandler's SetFurniturefunction is called, whose main function is to set the prefab at that instance which will be spawned by the placementcontroller script

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Security.Cryptography;
using UnityEngine;
using UnityEngine.UI;

public class MenupanelButton : MonoBehaviour
{
    private Button btn;
    private GameObject furniture;
    [SerializeField]private RawImage Buttonimage;
    private int item_id;
    private Sprite buttontexture;
    public int Itemid
    {
        set
        {
```

```
            item_id = value;
        }
    }
    public Sprite ButtonTexture
    {
        set
        {
            buttontexture = value;
            Buttonimage.texture = buttontexture.texture;
        }
    }
    void Start()
    {
        btn = GetComponent<Button>();
        btn.onClick.AddListener(SelectObject);
    }
    void SelectObject()
    {

        DataHandler.Instance.SetFurniture(item_id);
    }
}
```

2.2. AR Portal

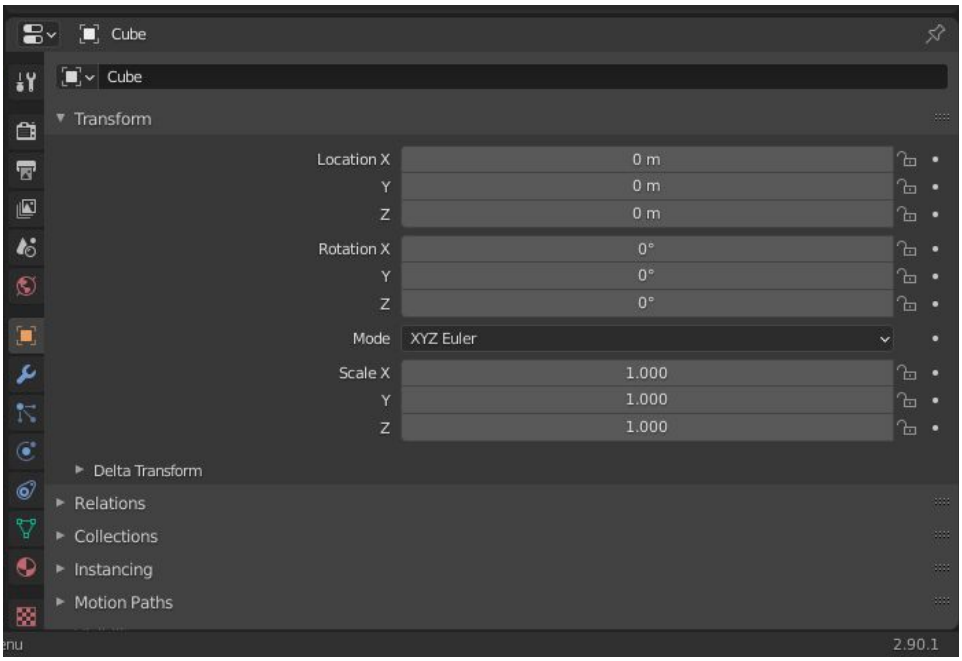
The goal of this project was to augment a world into the real world. There will be a door that will lead the person to another world. We can see this type of Portals in Video games. It is similar to Doraemon's door to anywhere. So to not just limit the Portals to video games and TV Shows this idea came.

A. AR application

Importing 3D models into Unity:

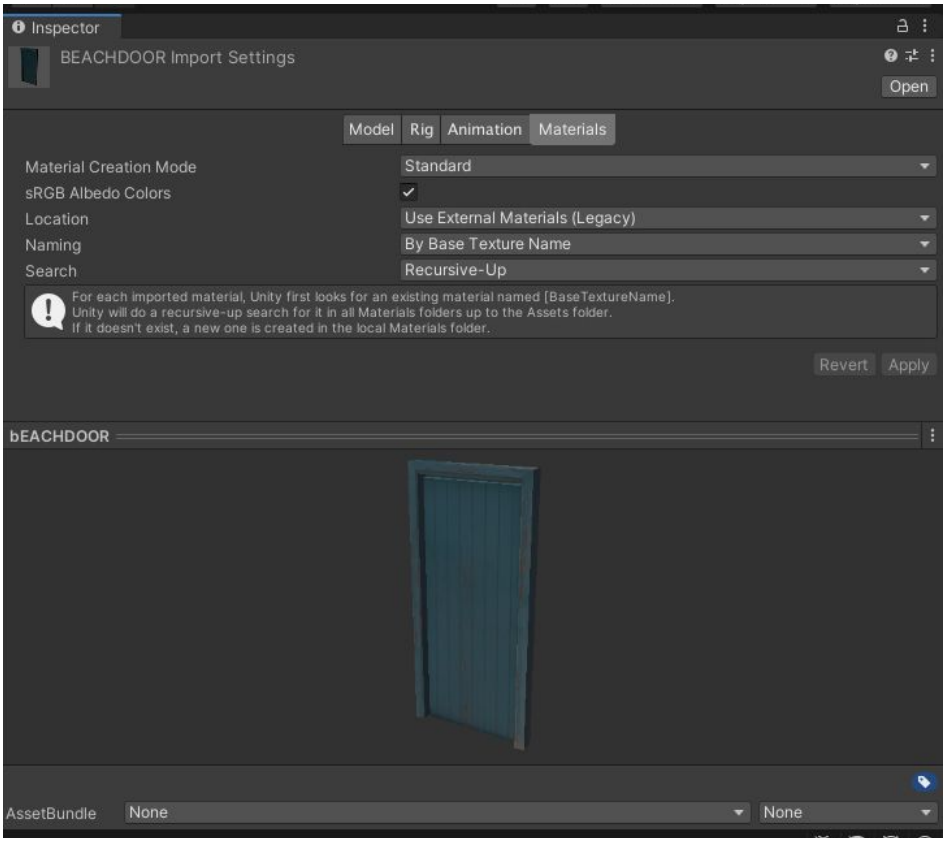
importing 3D models into unity is pretty easy. It is just drag and drop the fbx in unity. But sometimes it may not render in unity as it is rendering in unity. So for better importing of 3D models in Unity we need to follow following steps:

1. File>External Data> Unpack all into files (this will make a copy of texture image in a folder called textures)
2. Then check the location Rotation and Scale of the object, it should be (0,0,0), (0,0,0) and (1,1,1) respectively so that the model is not rotated or huge in the unity editor



3. Select all the objects(Leave the Camera and Light out or delete the camera and light before exporting.)
4. Then, File>Export>fbx
5. Then fbx and the folder texture is imported in to Unity Assets (this will not let us edit materials in future)

6. If the 3D object comes grey even after this then select the model and then Inspector>Materials>Location>Use External Materials (Legacy)>Apply>select the texture folder(From Blender)



7. After applying a new folder called Materials will be created
8. The materials are unpacked from the model and stored in Materials folder
9. The materials can be seen or even edited

Steps to make AR Portal

Animating the Door:

Animating the door is very simple.Steps:

1. Import the models animation too while importing the 3D model
2. Importing animation is done automatically while importing it as 3D model
3. Drag the model to hierarchy

4. then from the model, select the animation and drag to the gameobject containing the model
5. A new component will be added to the model called animator which will have a value called controller
6. Unity automatically creates a controller for the animation, the created controller is attached to controller field in Animator controller
7. By default, the animation is supposed to just run once at the start when the game is started
8. there are options for looping and editing the animation as well

Portal window:

To make an AR Portal, We need to be familiar with StencilLab and Shaders of Unity. So to make the AR portal first we need the window through which we can enter a new world.

Steps to make the window :

1. First, create a new GameObject either Quad or plane. Make its size similiar to the opening of the door.
2. Make a Shader called PortalWindow.shader
3. this will make the portal window transparent

```
Shader "Custom/Portalwindo"
{ //makes the quad transparent
   Subshader
   {
       Zwrite off
   ColorMask 0
   Cull off
   Stencil
   {
       Ref 1
       Pass replace
   }
```

```
        Pass
        {

        }
    }
}
```

4. The material of the Quad is then set to this shader
5. to make this shader quad's material, the shader was dragged into the quad.
6. Manually the materials of the quad could not be changed so the shader had to be dragged onto the quad.

Interdimensionaltransport.cs

1. Interdimensionaltransport.cs is needed to program the AR portal such that when the user is out of the portal they can see the other world from the door.
2. interdimensianaltransport.cs is attached to the quad(Portalwindow GameObject).
3. and when the user is inside the world the user can see the real world outside.
4. Before this, the ARCamera of the application was made a rigidbody and also was given a box collider so that the code can run after the ARcamera collides with the Quad(Portal Window)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Rendering;
public class Interdimensionaltransport : MonoBehaviour
{
    public GameObject InnerWorld;
    //This materials matter needs to be optimizated!
       public  Material[]  materials;//  all  the  materials  of  the
InnerWorld
```

```
    private Vector3 camPostionInPortalSpace;
    bool wasInFront;
    bool inOtherWorld;
    bool hasCollided;
    // Start is called before the first frame update
    void Start()
    {
        SetMaterials(false);
    }
    void SetMaterials(bool fullRender)
    {
          var stencilTest = fullRender ? CompareFunction.NotEqual :
CompareFunction.Equal;
        foreach (var mat in materials)
        {
            mat.SetInt("_StencilComp", (int)stencilTest);
        }
    }
    //Set bidirectional function
    bool GetIsInFront()
    {
                                        GameObject      MainCamera      =
GameObject.FindGameObjectWithTag("MainCamera");
                Vector3  worldPos  =  MainCamera.transform.position  +
MainCamera.transform.forward * Camera.main.nearClipPlane;
                                           camPostionInPortalSpace      =
transform.InverseTransformPoint(worldPos);
        return camPostionInPortalSpace.y >= 0 ? true : false;
    }
    private void OnTriggerEnter(Collider collider)
    {
                                        GameObject      MainCamera      =
GameObject.FindGameObjectWithTag("MainCamera");
        if (collider.transform != MainCamera.transform)
            return;
        wasInFront = GetIsInFront();
        hasCollided = true;
```

```
    }
    // Update is called once per frame
    void OnTriggerExit(Collider collider)
    {
                                        GameObject      MainCamera      =
GameObject.FindGameObjectWithTag("MainCamera");
        if (collider.transform != MainCamera.transform)
            return;
        hasCollided = false;
    }
    void whileCameraColliding()
    {
        if (!hasCollided)
            return;
        bool isInFront = GetIsInFront();
              if  ((isInFront  &&    !wasInFront)  ||  (wasInFront  &&
!isInFront))
        {
            inOtherWorld = !inOtherWorld;
            SetMaterials(inOtherWorld);
        }
        wasInFront = isInFront;
    }
    private void OnDestroy()
    {
        SetMaterials(true);
    }
    private void Update()
    {
        whileCameraColliding();
    }
}
```

InputController.cs

1. InputController Script is attached to the same GameObject which has the ARSession origin script in it

2. This will instantiate the inner world and door when the user touches the position in the AR plane detected by the application.
3.

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
/// <summary>
/// Listens for touch events and performs an AR raycast from the
screen touch point.
///  AR raycasts will only hit detected trackables like feature
points and planes.
///
/// If a raycast hits a trackable, the <see cref="placedPrefab"/> is
instantiated
/// and moved to the hit position.
/// </summary>
[RequireComponent(typeof(ARRaycastManager))]
public class InputController : MonoBehaviour
{
     static List<ARRaycastHit> Hits = new List<ARRaycastHit>();
   ARRaycastManager ARRaycastManager;
   [SerializeField]
     [Tooltip("Instantiates this prefab on a plane at the touch
location.")]
   GameObject PlacedPrefab;
   /// <summary>
   /// The prefab to instantiate on touch.
   /// </summary>
   public GameObject placedPrefab
   {
       get { return PlacedPrefab; }
       set { PlacedPrefab = value; }
   }
   /// <summary>
```

```
    /// The object instantiated as a result of a successful raycast
intersection with a plane.
    /// </summary>
    public GameObject spawnedObject { get; private set; }
    void Awake()
    {
        ARRaycastManager = GetComponent<ARRaycastManager>();
    }
    bool TryGetTouchPosition(out Vector2 touchPosition)
    {
#if UNITY_EDITOR
        if (Input.GetMouseButton(0))
        {
            var mousePosition = Input.mousePosition;
                      touchPosition  =  new  Vector2(mousePosition.x,
mousePosition.y);
            return true;
        }
#else
        if (Input.touchCount > 0)
        {
            touchPosition = Input.GetTouch(0).position;
            return true;
        }
#endif
        touchPosition = default;
        return false;
    }
    void Update()
    {
        if (!TryGetTouchPosition(out Vector2 touchPosition))
            return;
                  if   (ARRaycastManager.Raycast(touchPosition,   Hits,
TrackableType.PlaneWithinPolygon))
        {
            // Raycast hits are sorted by distance, so the first one
            // will be the closest hit.
```

```
            var hitPose = Hits[0].pose;
            if ((spawnedObject == null))
            {
                        spawnedObject = Instantiate(PlacedPrefab,
hitPose.position, hitPose.rotation);
            }
        }
    }
 }
```

SpecularStencilFilter.shader and StandardStencilShader.shader

1. A built in shader was downloaded from https://unity3d.com/get-unity/download/archive
2. Then specular shader and standard shader was imported into unity
3. Then some more lines of code were added to both of the shaders
4. these shaders were attached to the materials of the innerworld

SpecularStencilFilter.shader

```
// Unity built-in shader source. Copyright (c) 2016 Unity
Technologies. MIT license (see license.txt)

Shader "Custom/SpecularStencilFilter"
{
    Properties
    {
        _Color("Color", Color) = (1,1,1,1)
        _MainTex("Albedo", 2D) = "white" {}

        _Cutoff("Alpha Cutoff", Range(0.0, 1.0)) = 0.5

        _Glossiness("Smoothness", Range(0.0, 1.0)) = 0.5
         _GlossMapScale("Smoothness Factor", Range(0.0, 1.0)) =
1.0
```

```
                    [Enum(Specular   Alpha,0,Albedo   Alpha,1)]
_SmoothnessTextureChannel   ("Smoothness   texture   channel",
Float) = 0

        _SpecColor("Specular", Color) = (0.2,0.2,0.2)
        _SpecGlossMap("Specular", 2D) = "white" {}
          [ToggleOff] _SpecularHighlights("Specular Highlights",
Float) = 1.0
           [ToggleOff] _GlossyReflections("Glossy Reflections",
Float) = 1.0

        _BumpScale("Scale", Float) = 1.0
        [Normal] _BumpMap("Normal Map", 2D) = "bump" {}

        _Parallax ("Height Scale", Range (0.005, 0.08)) = 0.02
        _ParallaxMap ("Height Map", 2D) = "black" {}

        _OcclusionStrength("Strength", Range(0.0, 1.0)) = 1.0
        _OcclusionMap("Occlusion", 2D) = "white" {}

        _EmissionColor("Color", Color) = (0,0,0)
        _EmissionMap("Emission", 2D) = "white" {}

        _DetailMask("Detail Mask", 2D) = "white" {}

        _DetailAlbedoMap("Detail Albedo x2", 2D) = "grey" {}
        _DetailNormalMapScale("Scale", Float) = 1.0
        [Normal] _DetailNormalMap("Normal Map", 2D) = "bump" {}

            [Enum(UV0,0,UV1,1)] _UVSec ("UV Set for secondary
textures", Float) = 0
             [Enum(Equal,3,NotEqual,6)] _StencilTest ("Stencil
Test", int) = 6


        // Blending state
        [HideInInspector] _Mode ("__mode", Float) = 0.0
```

```
        [HideInInspector] _SrcBlend ("__src", Float) = 1.0
        [HideInInspector] _DstBlend ("__dst", Float) = 0.0
        [HideInInspector] _ZWrite ("__zw", Float) = 1.0
    }

    CGINCLUDE
        #define UNITY_SETUP_BRDF_INPUT SpecularSetup
    ENDCG

    SubShader
    {
                              Tags    {    "RenderType"="Opaque"
"PerformanceChecks"="False" }
        LOD 300


                                                                  //
------------------------------------------------------------------
----
          //   Base forward pass (directional light, emission,
lightmaps, ...)
        Pass
        {
            Stencil{
            Ref 1
            Comp [_StencilTest]
        }

            Name "FORWARD"
            Tags { "LightMode" = "ForwardBase" }

            Blend [_SrcBlend] [_DstBlend]
            ZWrite [_ZWrite]

            CGPROGRAM
            #pragma target 3.0
```

```
            // ------------------------------------

            #pragma shader_feature_local _NORMALMAP
                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _SPECGLOSSMAP
            #pragma shader_feature_local _DETAIL_MULX2
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                    #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _GLOSSYREFLECTIONS_OFF
            #pragma shader_feature_local _PARALLAXMAP

            #pragma multi_compile_fwdbase
            #pragma multi_compile_fog
            #pragma multi_compile_instancing
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertBase
            #pragma fragment fragBase
            #include "UnityStandardCoreForward.cginc"

            ENDCG
        }
                                                                    //
------------------------------------------------------------------
----
        //  Additive forward pass (one light per pass)
        Pass
        {
            Name "FORWARD_DELTA"
            Tags { "LightMode" = "ForwardAdd" }
```

```
            Blend [_SrcBlend] One
               Fog { Color (0,0,0,0) } // in additive pass fog
should be black
            ZWrite Off
            ZTest LEqual

            CGPROGRAM
            #pragma target 3.0

            // -------------------------------------

            #pragma shader_feature_local _NORMALMAP
                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _SPECGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                    #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature_local _PARALLAXMAP

            #pragma multi_compile_fwdadd_fullshadows
            #pragma multi_compile_fog
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertAdd
            #pragma fragment fragAdd
            #include "UnityStandardCoreForward.cginc"

            ENDCG
        }
```

```
                                                                    //
------------------------------------------------------------------
----
        //  Shadow rendering pass
        Pass {
            Name "ShadowCaster"
            Tags { "LightMode" = "ShadowCaster" }

            ZWrite On ZTest LEqual

            CGPROGRAM
            #pragma target 3.0

            // -------------------------------------


                    #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _SPECGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma shader_feature_local _PARALLAXMAP
            #pragma multi_compile_shadowcaster
            #pragma multi_compile_instancing
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertShadowCaster
            #pragma fragment fragShadowCaster

            #include "UnityStandardShadow.cginc"

            ENDCG
        }
```

```
                                                                    //
------------------------------------------------------------------
----
        //  Deferred pass
        Pass
        {
            Name "DEFERRED"
            Tags { "LightMode" = "Deferred" }

            CGPROGRAM
            #pragma target 3.0
            #pragma exclude_renderers nomrt


            // -------------------------------------

            #pragma shader_feature_local _NORMALMAP
                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _SPECGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                    #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature_local _PARALLAXMAP

            #pragma multi_compile_prepassfinal
            #pragma multi_compile_instancing
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertDeferred
            #pragma fragment fragDeferred
```

```
            #include "UnityStandardCore.cginc"

            ENDCG
        }

                                                                    //
------------------------------------------------------------------
----
        // Extracts information for lightmapping, GI (emission,
albedo, ...)
        // This pass it not used during regular rendering.
        Pass
        {
            Name "META"
            Tags { "LightMode"="Meta" }

            Cull Off

            CGPROGRAM
            #pragma vertex vert_meta
            #pragma fragment frag_meta

            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _SPECGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature EDITOR_VISUALIZATION

            #include "UnityStandardMeta.cginc"
            ENDCG
        }
    }

    SubShader
    {
```

```
                            Tags    {    "RenderType"="Opaque"
"PerformanceChecks"="False" }
       LOD 150



                                                                //
------------------------------------------------------------------
----
         //   Base forward pass (directional light, emission,
lightmaps, ...)
       Pass
       {
            Stencil{
           Ref 1
           Comp [_StencilTest]
       }
           Name "FORWARD"
           Tags { "LightMode" = "ForwardBase" }

           Blend [_SrcBlend] [_DstBlend]
           ZWrite [_ZWrite]

           CGPROGRAM
           #pragma target 2.0

           #pragma shader_feature_local _NORMALMAP
                 #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
           #pragma shader_feature _EMISSION
           #pragma shader_feature_local _SPECGLOSSMAP
                                #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
           #pragma shader_feature_local _GLOSSYREFLECTIONS_OFF
           #pragma shader_feature_local _DETAIL_MULX2
```

```
                // SM2.0: NOT SUPPORTED shader_feature_local
_PARALLAXMAP

                        #pragma skip_variants SHADOWS_SOFT
DYNAMICLIGHTMAP_ON DIRLIGHTMAP_COMBINED

            #pragma multi_compile_fwdbase
            #pragma multi_compile_fog

            #pragma vertex vertBase
            #pragma fragment fragBase
            #include "UnityStandardCoreForward.cginc"

            ENDCG
        }
                                                                //
------------------------------------------------------------------
----
        // Additive forward pass (one light per pass)
        Pass
        {
            Name "FORWARD_DELTA"
            Tags { "LightMode" = "ForwardAdd" }
            Blend [_SrcBlend] One
               Fog { Color (0,0,0,0) } // in additive pass fog
should be black
            ZWrite Off
            ZTest LEqual

            CGPROGRAM
            #pragma target 2.0

            #pragma shader_feature_local _NORMALMAP
                  #pragma shader_feature_local _ _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _SPECGLOSSMAP
```

```
                                #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _DETAIL_MULX2
                  // SM2.0: NOT SUPPORTED shader_feature_local
_PARALLAXMAP
            #pragma skip_variants SHADOWS_SOFT

            #pragma multi_compile_fwdadd_fullshadows
            #pragma multi_compile_fog

            #pragma vertex vertAdd
            #pragma fragment fragAdd
            #include "UnityStandardCoreForward.cginc"

            ENDCG
        }
                                                                    //
------------------------------------------------------------------
----
        // Shadow rendering pass
        Pass {
            Name "ShadowCaster"
            Tags { "LightMode" = "ShadowCaster" }

            ZWrite On ZTest LEqual

            CGPROGRAM
            #pragma target 2.0

                  #pragma shader_feature_local _ _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _SPECGLOSSMAP
                                #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma skip_variants SHADOWS_SOFT
```

```
            #pragma multi_compile_shadowcaster

            #pragma vertex vertShadowCaster
            #pragma fragment fragShadowCaster

            #include "UnityStandardShadow.cginc"

            ENDCG
        }
                                                                    //
------------------------------------------------------------------
----
        // Extracts information for lightmapping, GI (emission,
albedo, ...)
        // This pass it not used during regular rendering.
        Pass
        {
            Name "META"
            Tags { "LightMode"="Meta" }

            Cull Off

            CGPROGRAM
            #pragma vertex vert_meta
            #pragma fragment frag_meta

            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _SPECGLOSSMAP
                                #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature EDITOR_VISUALIZATION

            #include "UnityStandardMeta.cginc"
            ENDCG
        }
    }
```

```
    //FallBack "VertexLit"
    //CustomEditor "StandardShaderGUI"
}
```

StandarStencilFilter.shader

```
// Unity built-in shader source. Copyright (c) 2016 Unity
Technologies. MIT license (see license.txt)

Shader "Custom/StandardStencil"
{
    Properties
    {
        _Color("Color", Color) = (1,1,1,1)
        _MainTex("Albedo", 2D) = "white" {}

        _Cutoff("Alpha Cutoff", Range(0.0, 1.0)) = 0.5

        _Glossiness("Smoothness", Range(0.0, 1.0)) = 0.5
          _GlossMapScale("Smoothness Scale", Range(0.0, 1.0)) =
1.0
                    [Enum(Metallic Alpha,0,Albedo Alpha,1)]
_SmoothnessTextureChannel ("Smoothness texture channel",
Float) = 0

        [Gamma] _Metallic("Metallic", Range(0.0, 1.0)) = 0.0
        _MetallicGlossMap("Metallic", 2D) = "white" {}

          [ToggleOff] _SpecularHighlights("Specular Highlights",
Float) = 1.0
            [ToggleOff] _GlossyReflections("Glossy Reflections",
Float) = 1.0
```

```
        _BumpScale("Scale", Float) = 1.0
        [Normal] _BumpMap("Normal Map", 2D) = "bump" {}

        _Parallax ("Height Scale", Range (0.005, 0.08)) = 0.02
        _ParallaxMap ("Height Map", 2D) = "black" {}

        _OcclusionStrength("Strength", Range(0.0, 1.0)) = 1.0
        _OcclusionMap("Occlusion", 2D) = "white" {}

        _EmissionColor("Color", Color) = (0,0,0)
        _EmissionMap("Emission", 2D) = "white" {}

        _DetailMask("Detail Mask", 2D) = "white" {}

        _DetailAlbedoMap("Detail Albedo x2", 2D) = "grey" {}
        _DetailNormalMapScale("Scale", Float) = 1.0
        [Normal] _DetailNormalMap("Normal Map", 2D) = "bump" {}

            [Enum(UV0,0,UV1,1)] _UVSec ("UV Set for secondary
textures", Float) = 0
             [Enum(Equal,3,NotEqual,6)] _StencilTest ("Stencil
Test", int) = 6

        // Blending state
        [HideInInspector] _Mode ("__mode", Float) = 0.0
        [HideInInspector] _SrcBlend ("__src", Float) = 1.0
        [HideInInspector] _DstBlend ("__dst", Float) = 0.0
        [HideInInspector] _ZWrite ("__zw", Float) = 1.0
    }

    CGINCLUDE
        #define UNITY_SETUP_BRDF_INPUT MetallicSetup
    ENDCG

    SubShader
    {
```

```
                            Tags    {    "RenderType"="Opaque"
"PerformanceChecks"="False" }
       LOD 300




                                                                //
------------------------------------------------------------------
----
         //   Base forward pass (directional light, emission,
lightmaps, ...)
       Pass
       {
           Stencil{
           Ref 1
           Comp [_StencilTest]
       }
           Name "FORWARD"
           Tags { "LightMode" = "ForwardBase" }

           Blend [_SrcBlend] [_DstBlend]
           ZWrite [_ZWrite]

           CGPROGRAM
           #pragma target 3.0

           // -------------------------------------

           #pragma shader_feature_local _NORMALMAP
                 #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
           #pragma shader_feature _EMISSION
           #pragma shader_feature_local _METALLICGLOSSMAP
           #pragma shader_feature_local _DETAIL_MULX2
                                  #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
```

```
                                #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _GLOSSYREFLECTIONS_OFF
            #pragma shader_feature_local _PARALLAXMAP

            #pragma multi_compile_fwdbase
            #pragma multi_compile_fog
            #pragma multi_compile_instancing
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertBase
            #pragma fragment fragBase
            #include "UnityStandardCoreForward.cginc"

            ENDCG
        }
                                                                  //
-----------------------------------------------------------------
----
        //  Additive forward pass (one light per pass)
        Pass
        {
            Name "FORWARD_DELTA"
            Tags { "LightMode" = "ForwardAdd" }
            Blend [_SrcBlend] One
               Fog { Color (0,0,0,0) } // in additive pass fog
should be black
            ZWrite Off
            ZTest LEqual

            CGPROGRAM
            #pragma target 3.0

            // ------------------------------------
```

```
            #pragma shader_feature_local _NORMALMAP
                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _METALLICGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                    #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature_local _PARALLAXMAP

            #pragma multi_compile_fwdadd_fullshadows
            #pragma multi_compile_fog
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertAdd
            #pragma fragment fragAdd
            #include "UnityStandardCoreForward.cginc"

            ENDCG
        }
                                                                    //
------------------------------------------------------------------
----
        //  Shadow rendering pass
        Pass {
            Name "ShadowCaster"
            Tags { "LightMode" = "ShadowCaster" }

            ZWrite On ZTest LEqual

            CGPROGRAM
```

```
            #pragma target 3.0

            // -------------------------------------


                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _METALLICGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma shader_feature_local _PARALLAXMAP
            #pragma multi_compile_shadowcaster
            #pragma multi_compile_instancing
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertShadowCaster
            #pragma fragment fragShadowCaster

            #include "UnityStandardShadow.cginc"

            ENDCG
        }
                                                                    //
-------------------------------------------------------------------
----
        //  Deferred pass
        Pass
        {
            Name "DEFERRED"
            Tags { "LightMode" = "Deferred" }

            CGPROGRAM
            #pragma target 3.0
            #pragma exclude_renderers nomrt
```

```
            // ------------------------------------

            #pragma shader_feature_local _NORMALMAP
                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _METALLICGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                    #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature_local _PARALLAXMAP

            #pragma multi_compile_prepassfinal
            #pragma multi_compile_instancing
             // Uncomment the following line to enable dithering
LOD crossfade. Note: there are more in the file to uncomment
for other passes.
            //#pragma multi_compile _ LOD_FADE_CROSSFADE

            #pragma vertex vertDeferred
            #pragma fragment fragDeferred

            #include "UnityStandardCore.cginc"

            ENDCG
      }

                                                                  //
------------------------------------------------------------------
----
       // Extracts information for lightmapping, GI (emission,
albedo, ...)
      // This pass it not used during regular rendering.
```

```
        Pass
        {
            Name "META"
            Tags { "LightMode"="Meta" }

            Cull Off

            CGPROGRAM
            #pragma vertex vert_meta
            #pragma fragment frag_meta

            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _METALLICGLOSSMAP
                                #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature EDITOR_VISUALIZATION

            #include "UnityStandardMeta.cginc"
            ENDCG
        }
    }

    SubShader
    {
                               Tags    {    "RenderType"="Opaque"
"PerformanceChecks"="False" }
        LOD 150

                                                                    //
------------------------------------------------------------------
----
          //   Base forward pass (directional light, emission,
lightmaps, ...)
        Pass
        {
             Stencil{
```

```
            Ref 1
            Comp [_StencilTest]
        }

            Name "FORWARD"
            Tags { "LightMode" = "ForwardBase" }

            Blend [_SrcBlend] [_DstBlend]
            ZWrite [_ZWrite]

            CGPROGRAM
            #pragma target 2.0

            #pragma shader_feature_local _NORMALMAP
                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _METALLICGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                    #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _GLOSSYREFLECTIONS_OFF
                  // SM2.0: NOT SUPPORTED shader_feature_local
_DETAIL_MULX2
                  // SM2.0: NOT SUPPORTED shader_feature_local
_PARALLAXMAP

                              #pragma  skip_variants  SHADOWS_SOFT
DIRLIGHTMAP_COMBINED

            #pragma multi_compile_fwdbase
            #pragma multi_compile_fog

            #pragma vertex vertBase
            #pragma fragment fragBase
            #include "UnityStandardCoreForward.cginc"
```

```
            ENDCG
        }
                                                                    //
------------------------------------------------------------------
----
        //  Additive forward pass (one light per pass)
        Pass
        {
            Name "FORWARD_DELTA"
            Tags { "LightMode" = "ForwardAdd" }
            Blend [_SrcBlend] One
               Fog { Color (0,0,0,0) } // in additive pass fog
should be black
            ZWrite Off
            ZTest LEqual

            CGPROGRAM
            #pragma target 2.0

            #pragma shader_feature_local _NORMALMAP
                  #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _METALLICGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
                                    #pragma   shader_feature_local
_SPECULARHIGHLIGHTS_OFF
            #pragma shader_feature_local _DETAIL_MULX2
                  // SM2.0: NOT SUPPORTED shader_feature_local
_PARALLAXMAP
            #pragma skip_variants SHADOWS_SOFT

            #pragma multi_compile_fwdadd_fullshadows
            #pragma multi_compile_fog

            #pragma vertex vertAdd
```

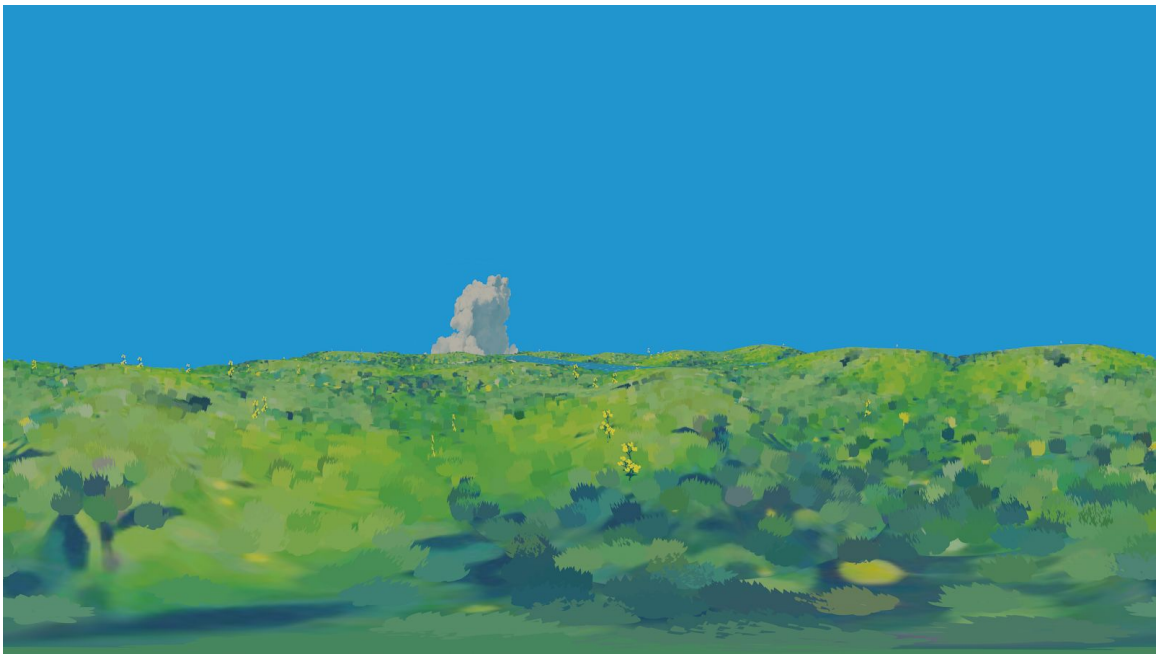
```
            #pragma fragment fragAdd
            #include "UnityStandardCoreForward.cginc"

            ENDCG
        }
                                                                    //
------------------------------------------------------------------
----
        //  Shadow rendering pass
        Pass {
            Name "ShadowCaster"
            Tags { "LightMode" = "ShadowCaster" }

            ZWrite On ZTest LEqual

            CGPROGRAM
            #pragma target 2.0

                    #pragma  shader_feature_local  _  _ALPHATEST_ON
_ALPHABLEND_ON _ALPHAPREMULTIPLY_ON
            #pragma shader_feature_local _METALLICGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma skip_variants SHADOWS_SOFT
            #pragma multi_compile_shadowcaster

            #pragma vertex vertShadowCaster
            #pragma fragment fragShadowCaster

            #include "UnityStandardShadow.cginc"

            ENDCG
        }

                                                                    //
------------------------------------------------------------------
----
```

```
        // Extracts information for lightmapping, GI (emission,
albedo, ...)
        // This pass it not used during regular rendering.
        Pass
        {
            Name "META"
            Tags { "LightMode"="Meta" }

            Cull Off

            CGPROGRAM
            #pragma vertex vert_meta
            #pragma fragment frag_meta

            #pragma shader_feature _EMISSION
            #pragma shader_feature_local _METALLICGLOSSMAP
                                    #pragma   shader_feature_local
_SMOOTHNESS_TEXTURE_ALBEDO_CHANNEL_A
            #pragma shader_feature_local _DETAIL_MULX2
            #pragma shader_feature EDITOR_VISUALIZATION

            #include "UnityStandardMeta.cginc"
            ENDCG
        }
    }


    //FallBack "VertexLit"
    //CustomEditor "StandardShaderGUI"
}
```

B. 360 Video

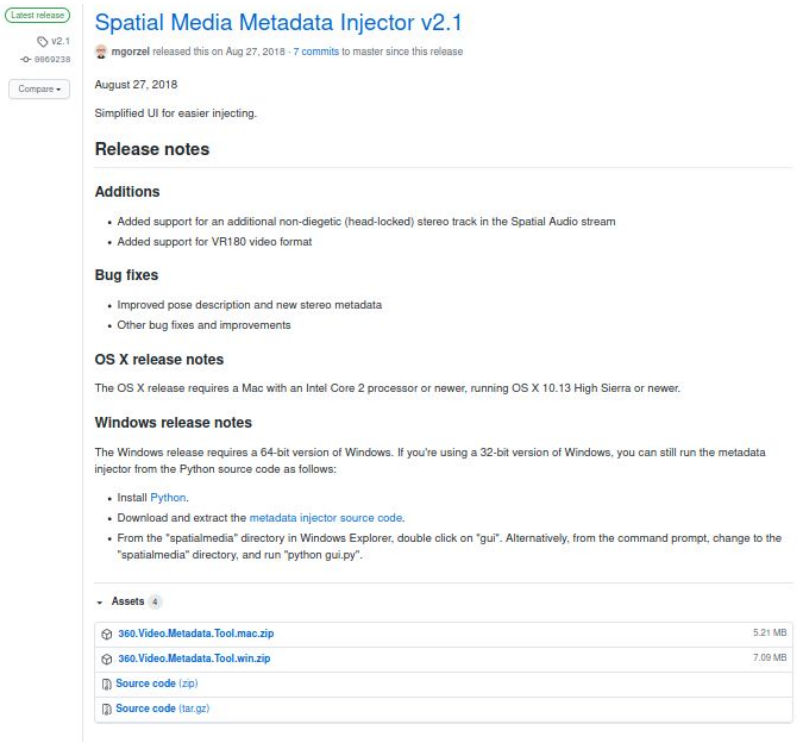
Steps to make 360 video:

1. A equirectangular image () is converted into a video



Equirectangular Image

2. The Video is injected with spatial media metadata



Spatial Media Metadata

3. That will help YouTube or Facebook to recognize that it is a 360 video.