using System;

using UnityEditor;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[CustomEditor (typeof(Tonemapping))]

class TonemappingEditor : Editor

{

SerializedObject serObj;

SerializedProperty type;

// CURVE specific parameter

SerializedProperty remapCurve;

SerializedProperty exposureAdjustment;

// REINHARD specific parameter

SerializedProperty middleGrey;

SerializedProperty white;

SerializedProperty adaptionSpeed;

SerializedProperty adaptiveTextureSize;

void OnEnable () {

serObj = new SerializedObject (target);

type = serObj.FindProperty ("type");

remapCurve = serObj.FindProperty ("remapCurve");

exposureAdjustment = serObj.FindProperty ("exposureAdjustment");

middleGrey = serObj.FindProperty ("middleGrey");

white = serObj.FindProperty ("white");

adaptionSpeed = serObj.FindProperty ("adaptionSpeed");

adaptiveTextureSize = serObj.FindProperty("adaptiveTextureSize");

}

public override void OnInspectorGUI () {

serObj.Update ();

GUILayout.Label("Mapping HDR to LDR ranges since 1982", EditorStyles.miniLabel);

Camera cam = (target as Tonemapping).GetComponent<Camera>();

if (cam != null) {

if (!cam.hdr) {

EditorGUILayout.HelpBox("The camera is not HDR enabled. This will likely break the Tonemapper.", MessageType.Warning);

}

else if (!(target as Tonemapping).validRenderTextureFormat) {

EditorGUILayout.HelpBox("The input to Tonemapper is not in HDR. Make sure that all effects prior to this are executed in HDR.", MessageType.Warning);

}

}

EditorGUILayout.PropertyField (type, new GUIContent ("Technique"));

if (type.enumValueIndex == (int) Tonemapping.TonemapperType.UserCurve) {

EditorGUILayout.PropertyField (remapCurve, new GUIContent ("Remap curve", "Specify the mapping of luminances yourself"));

} else if (type.enumValueIndex == (int) Tonemapping.TonemapperType.SimpleReinhard) {

EditorGUILayout.PropertyField (exposureAdjustment, new GUIContent ("Exposure", "Exposure adjustment"));

} else if (type.enumValueIndex == (int) Tonemapping.TonemapperType.Hable) {

EditorGUILayout.PropertyField (exposureAdjustment, new GUIContent ("Exposure", "Exposure adjustment"));

} else if (type.enumValueIndex == (int) Tonemapping.TonemapperType.Photographic) {

EditorGUILayout.PropertyField (exposureAdjustment, new GUIContent ("Exposure", "Exposure adjustment"));

} else if (type.enumValueIndex == (int) Tonemapping.TonemapperType.OptimizedHejiDawson) {

EditorGUILayout.PropertyField (exposureAdjustment, new GUIContent ("Exposure", "Exposure adjustment"));

} else if (type.enumValueIndex == (int) Tonemapping.TonemapperType.AdaptiveReinhard) {

EditorGUILayout.PropertyField (middleGrey, new GUIContent ("Middle grey", "Middle grey defines the average luminance thus brightening or darkening the entire image."));

EditorGUILayout.PropertyField (white, new GUIContent ("White", "Smallest luminance value that will be mapped to white"));

EditorGUILayout.PropertyField (adaptionSpeed, new GUIContent ("Adaption Speed", "Speed modifier for the automatic adaption"));

EditorGUILayout.PropertyField (adaptiveTextureSize, new GUIContent ("Texture size", "Defines the amount of downsamples needed."));

} else if (type.enumValueIndex == (int) Tonemapping.TonemapperType.AdaptiveReinhardAutoWhite) {

EditorGUILayout.PropertyField (middleGrey, new GUIContent ("Middle grey", "Middle grey defines the average luminance thus brightening or darkening the entire image."));

EditorGUILayout.PropertyField (adaptionSpeed, new GUIContent ("Adaption Speed", "Speed modifier for the automatic adaption"));

EditorGUILayout.PropertyField (adaptiveTextureSize, new GUIContent ("Texture size", "Defines the amount of downsamples needed."));

}

GUILayout.Label("All following effects will use LDR color buffers", EditorStyles.miniBoldLabel);

serObj.ApplyModifiedProperties();

}

}

}