using System;

using UnityEditor;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[CustomEditor (typeof(Bloom))]

class BloomEditor : Editor

{

SerializedProperty tweakMode;

SerializedProperty screenBlendMode;

SerializedObject serObj;

SerializedProperty hdr;

SerializedProperty quality;

SerializedProperty sepBlurSpread;

SerializedProperty bloomIntensity;

SerializedProperty bloomThresholdColor;

SerializedProperty bloomThreshold;

SerializedProperty bloomBlurIterations;

SerializedProperty hollywoodFlareBlurIterations;

SerializedProperty lensflareMode;

SerializedProperty hollyStretchWidth;

SerializedProperty lensflareIntensity;

SerializedProperty flareRotation;

SerializedProperty lensFlareSaturation;

SerializedProperty lensflareThreshold;

SerializedProperty flareColorA;

SerializedProperty flareColorB;

SerializedProperty flareColorC;

SerializedProperty flareColorD;

SerializedProperty lensFlareVignetteMask;

void OnEnable () {

serObj = new SerializedObject (target);

screenBlendMode = serObj.FindProperty("screenBlendMode");

hdr = serObj.FindProperty("hdr");

quality = serObj.FindProperty("quality");

sepBlurSpread = serObj.FindProperty("sepBlurSpread");

bloomIntensity = serObj.FindProperty("bloomIntensity");

bloomThreshold = serObj.FindProperty("bloomThreshold");

bloomThresholdColor = serObj.FindProperty("bloomThresholdColor");

bloomBlurIterations = serObj.FindProperty("bloomBlurIterations");

lensflareMode = serObj.FindProperty("lensflareMode");

hollywoodFlareBlurIterations = serObj.FindProperty("hollywoodFlareBlurIterations");

hollyStretchWidth = serObj.FindProperty("hollyStretchWidth");

lensflareIntensity = serObj.FindProperty("lensflareIntensity");

lensflareThreshold = serObj.FindProperty("lensflareThreshold");

lensFlareSaturation = serObj.FindProperty("lensFlareSaturation");

flareRotation = serObj.FindProperty("flareRotation");

flareColorA = serObj.FindProperty("flareColorA");

flareColorB = serObj.FindProperty("flareColorB");

flareColorC = serObj.FindProperty("flareColorC");

flareColorD = serObj.FindProperty("flareColorD");

lensFlareVignetteMask = serObj.FindProperty("lensFlareVignetteMask");

tweakMode = serObj.FindProperty("tweakMode");

}

public override void OnInspectorGUI () {

serObj.Update();

EditorGUILayout.LabelField("Glow and Lens Flares for bright screen pixels", EditorStyles.miniLabel);

EditorGUILayout.PropertyField (quality, new GUIContent("Quality", "High quality preserves high frequencies with bigger blurs and uses a better blending and down-/upsampling"));

EditorGUILayout.Separator ();

EditorGUILayout.PropertyField (tweakMode, new GUIContent("Mode"));

EditorGUILayout.PropertyField (screenBlendMode, new GUIContent("Blend"));

EditorGUILayout.PropertyField (hdr, new GUIContent("HDR"));

EditorGUILayout.Separator ();

// display info text when screen blend mode cannot be used

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

if (cam != null) {

if (screenBlendMode.enumValueIndex==0 && ((cam.hdr && hdr.enumValueIndex==0) || (hdr.enumValueIndex==1))) {

EditorGUILayout.HelpBox("Screen blend is not supported in HDR. Using 'Add' instead.", MessageType.Info);

}

}

EditorGUILayout.PropertyField (bloomIntensity, new GUIContent("Intensity"));

bloomThreshold.floatValue = EditorGUILayout.Slider ("Threshold", bloomThreshold.floatValue, -0.05f, 4.0f);

if (1 == tweakMode.intValue) {

EditorGUILayout.PropertyField(bloomThresholdColor, new GUIContent(" RGB Threshold"));

}

EditorGUILayout.Separator ();

bloomBlurIterations.intValue = EditorGUILayout.IntSlider ("Blur Iterations", bloomBlurIterations.intValue, 1, 4);

sepBlurSpread.floatValue = EditorGUILayout.Slider (" Sample Distance", sepBlurSpread.floatValue, 0.1f, 10.0f);

EditorGUILayout.Separator ();

if (1 == tweakMode.intValue) {

// further lens flare tweakings

if (0 != tweakMode.intValue)

EditorGUILayout.PropertyField (lensflareMode, new GUIContent("Lens Flares"));

else

lensflareMode.enumValueIndex = 0;

EditorGUILayout.PropertyField (lensflareIntensity, new GUIContent(" Local Intensity", "0 disables lens flares entirely (optimization)"));

lensflareThreshold.floatValue = EditorGUILayout.Slider ("Threshold", lensflareThreshold.floatValue, 0.0f, 4.0f);

if (Mathf.Abs(lensflareIntensity.floatValue) > Mathf.Epsilon) {

if (lensflareMode.intValue == 0) {

// ghosting

EditorGUILayout.BeginHorizontal ();

EditorGUILayout.PropertyField (flareColorA, new GUIContent(" 1st Color"));

EditorGUILayout.PropertyField (flareColorB, new GUIContent(" 2nd Color"));

EditorGUILayout.EndHorizontal ();

EditorGUILayout.BeginHorizontal ();

EditorGUILayout.PropertyField (flareColorC, new GUIContent(" 3rd Color"));

EditorGUILayout.PropertyField (flareColorD, new GUIContent(" 4th Color"));

EditorGUILayout.EndHorizontal ();

}

else if (lensflareMode.intValue == 1) {

// hollywood

EditorGUILayout.PropertyField (hollyStretchWidth, new GUIContent(" Stretch width"));

EditorGUILayout.PropertyField (flareRotation, new GUIContent( " Rotation"));

hollywoodFlareBlurIterations.intValue = EditorGUILayout.IntSlider (" Blur Iterations", hollywoodFlareBlurIterations.intValue, 1, 4);

EditorGUILayout.PropertyField (lensFlareSaturation, new GUIContent(" Saturation"));

EditorGUILayout.PropertyField (flareColorA, new GUIContent(" Tint Color"));

}

else if (lensflareMode.intValue == 2) {

// both

EditorGUILayout.PropertyField (hollyStretchWidth, new GUIContent(" Stretch width"));

hollywoodFlareBlurIterations.intValue = EditorGUILayout.IntSlider (" Blur Iterations", hollywoodFlareBlurIterations.intValue, 1, 4);

EditorGUILayout.PropertyField (lensFlareSaturation, new GUIContent(" Saturation"));

EditorGUILayout.BeginHorizontal ();

EditorGUILayout.PropertyField (flareColorA, new GUIContent(" 1st Color"));

EditorGUILayout.PropertyField (flareColorB, new GUIContent(" 2nd Color"));

EditorGUILayout.EndHorizontal ();

EditorGUILayout.BeginHorizontal ();

EditorGUILayout.PropertyField (flareColorC, new GUIContent(" 3rd Color"));

EditorGUILayout.PropertyField (flareColorD, new GUIContent(" 4th Color"));

EditorGUILayout.EndHorizontal ();

}

EditorGUILayout.PropertyField(lensFlareVignetteMask, new GUIContent(" Mask", "This mask is needed to prevent lens flare artifacts"));

}

}

serObj.ApplyModifiedProperties();

}

}

}