using System;

using UnityEditor;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[CustomEditor (typeof(BloomAndFlares))]

class BloomAndFlaresEditor : Editor

{

SerializedProperty tweakMode;

SerializedProperty screenBlendMode;

SerializedObject serObj;

SerializedProperty hdr;

SerializedProperty sepBlurSpread;

SerializedProperty useSrcAlphaAsMask;

SerializedProperty bloomIntensity;

SerializedProperty bloomthreshold;

SerializedProperty bloomBlurIterations;

SerializedProperty lensflares;

SerializedProperty hollywoodFlareBlurIterations;

SerializedProperty lensflareMode;

SerializedProperty hollyStretchWidth;

SerializedProperty lensflareIntensity;

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");

sepBlurSpread = serObj.FindProperty("sepBlurSpread");

useSrcAlphaAsMask = serObj.FindProperty("useSrcAlphaAsMask");

bloomIntensity = serObj.FindProperty("bloomIntensity");

bloomthreshold = serObj.FindProperty("bloomThreshold");

bloomBlurIterations = serObj.FindProperty("bloomBlurIterations");

lensflares = serObj.FindProperty("lensflares");

lensflareMode = serObj.FindProperty("lensflareMode");

hollywoodFlareBlurIterations = serObj.FindProperty("hollywoodFlareBlurIterations");

hollyStretchWidth = serObj.FindProperty("hollyStretchWidth");

lensflareIntensity = serObj.FindProperty("lensflareIntensity");

lensflarethreshold = serObj.FindProperty("lensflareThreshold");

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();

GUILayout.Label("HDR " + (hdr.enumValueIndex == 0 ? "auto detected, " : (hdr.enumValueIndex == 1 ? "forced on, " : "disabled, ")) + (useSrcAlphaAsMask.floatValue < 0.1f ? " ignoring alpha channel glow information" : " using alpha channel glow information"), EditorStyles.miniBoldLabel);

EditorGUILayout.PropertyField (tweakMode, new GUIContent("Tweak mode"));

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

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

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

Camera cam = (target as BloomAndFlares).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);

}

}

if (1 == tweakMode.intValue)

EditorGUILayout.PropertyField (lensflares, new GUIContent("Cast lens flares"));

EditorGUILayout.Separator ();

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

bloomthreshold.floatValue = EditorGUILayout.Slider ("threshold", bloomthreshold.floatValue, -0.05f, 4.0f);

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

sepBlurSpread.floatValue = EditorGUILayout.Slider ("Blur spread", sepBlurSpread.floatValue, 0.1f, 10.0f);

if (1 == tweakMode.intValue)

useSrcAlphaAsMask.floatValue = EditorGUILayout.Slider (new GUIContent("Use alpha mask", "Make alpha channel define glowiness"), useSrcAlphaAsMask.floatValue, 0.0f, 1.0f);

else

useSrcAlphaAsMask.floatValue = 0.0f;

if (1 == tweakMode.intValue) {

EditorGUILayout.Separator ();

if (lensflares.boolValue) {

// further lens flare tweakings

if (0 != tweakMode.intValue)

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

else

lensflareMode.enumValueIndex = 0;

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

EditorGUILayout.PropertyField (lensflareIntensity, new GUIContent("Local intensity"));

lensflarethreshold.floatValue = EditorGUILayout.Slider ("Local threshold", lensflarethreshold.floatValue, 0.0f, 1.0f);

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"));

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

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.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

lensflares.boolValue = false; // disable lens flares in simple tweak mode

serObj.ApplyModifiedProperties();

}

}

}