using System;

using UnityEditor;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[CustomEditor (typeof(SunShafts))]

class SunShaftsEditor : Editor

{

SerializedObject serObj;

SerializedProperty sunTransform;

SerializedProperty radialBlurIterations;

SerializedProperty sunColor;

SerializedProperty sunThreshold;

SerializedProperty sunShaftBlurRadius;

SerializedProperty sunShaftIntensity;

SerializedProperty useDepthTexture;

SerializedProperty resolution;

SerializedProperty screenBlendMode;

SerializedProperty maxRadius;

void OnEnable () {

serObj = new SerializedObject (target);

screenBlendMode = serObj.FindProperty("screenBlendMode");

sunTransform = serObj.FindProperty("sunTransform");

sunColor = serObj.FindProperty("sunColor");

sunThreshold = serObj.FindProperty("sunThreshold");

sunShaftBlurRadius = serObj.FindProperty("sunShaftBlurRadius");

radialBlurIterations = serObj.FindProperty("radialBlurIterations");

sunShaftIntensity = serObj.FindProperty("sunShaftIntensity");

resolution = serObj.FindProperty("resolution");

maxRadius = serObj.FindProperty("maxRadius");

useDepthTexture = serObj.FindProperty("useDepthTexture");

}

public override void OnInspectorGUI () {

serObj.Update ();

EditorGUILayout.BeginHorizontal();

EditorGUILayout.PropertyField (useDepthTexture, new GUIContent ("Rely on Z Buffer?"));

if ((target as SunShafts).GetComponent<Camera>())

GUILayout.Label("Current camera mode: "+ (target as SunShafts).GetComponent<Camera>().depthTextureMode, EditorStyles.miniBoldLabel);

EditorGUILayout.EndHorizontal();

// depth buffer need

/\*

bool newVal = useDepthTexture.boolValue;

if (newVal != oldVal) {

if (newVal)

(target as SunShafts).camera.depthTextureMode |= DepthTextureMode.Depth;

else

(target as SunShafts).camera.depthTextureMode &= ~DepthTextureMode.Depth;

}

\*/

EditorGUILayout.PropertyField (resolution, new GUIContent("Resolution"));

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

EditorGUILayout.Separator ();

EditorGUILayout.BeginHorizontal();

EditorGUILayout.PropertyField (sunTransform, new GUIContent("Shafts caster", "Chose a transform that acts as a root point for the produced sun shafts"));

if ((target as SunShafts).sunTransform && (target as SunShafts).GetComponent<Camera>()) {

if (GUILayout.Button("Center on " + (target as SunShafts).GetComponent<Camera>().name)) {

if (EditorUtility.DisplayDialog ("Move sun shafts source?", "The SunShafts caster named "+ (target as SunShafts).sunTransform.name +"\n will be centered along "+(target as SunShafts).GetComponent<Camera>().name+". Are you sure? ", "Please do", "Don't")) {

Ray ray = (target as SunShafts).GetComponent<Camera>().ViewportPointToRay(new Vector3(0.5f,0.5f,0));

(target as SunShafts).sunTransform.position = ray.origin + ray.direction \* 500.0f;

(target as SunShafts).sunTransform.LookAt ((target as SunShafts).transform);

}

}

}

EditorGUILayout.EndHorizontal();

EditorGUILayout.Separator ();

EditorGUILayout.PropertyField (sunThreshold, new GUIContent ("Threshold color"));

EditorGUILayout.PropertyField (sunColor, new GUIContent ("Shafts color"));

maxRadius.floatValue = 1.0f - EditorGUILayout.Slider ("Distance falloff", 1.0f - maxRadius.floatValue, 0.1f, 1.0f);

EditorGUILayout.Separator ();

sunShaftBlurRadius.floatValue = EditorGUILayout.Slider ("Blur size", sunShaftBlurRadius.floatValue, 1.0f, 10.0f);

radialBlurIterations.intValue = EditorGUILayout.IntSlider ("Blur iterations", radialBlurIterations.intValue, 1, 3);

EditorGUILayout.Separator ();

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

serObj.ApplyModifiedProperties();

}

}

}