using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

[AddComponentMenu ("Image Effects/Rendering/Sun Shafts")]

public class SunShafts : PostEffectsBase

{

public enum SunShaftsResolution

{

Low = 0,

Normal = 1,

High = 2,

}

public enum ShaftsScreenBlendMode

{

Screen = 0,

Add = 1,

}

public SunShaftsResolution resolution = SunShaftsResolution.Normal;

public ShaftsScreenBlendMode screenBlendMode = ShaftsScreenBlendMode.Screen;

public Transform sunTransform;

public int radialBlurIterations = 2;

public Color sunColor = Color.white;

public Color sunThreshold = new Color(0.87f,0.74f,0.65f);

public float sunShaftBlurRadius = 2.5f;

public float sunShaftIntensity = 1.15f;

public float maxRadius = 0.75f;

public bool useDepthTexture = true;

public Shader sunShaftsShader;

private Material sunShaftsMaterial;

public Shader simpleClearShader;

private Material simpleClearMaterial;

public override bool CheckResources () {

CheckSupport (useDepthTexture);

sunShaftsMaterial = CheckShaderAndCreateMaterial (sunShaftsShader, sunShaftsMaterial);

simpleClearMaterial = CheckShaderAndCreateMaterial (simpleClearShader, simpleClearMaterial);

if (!isSupported)

ReportAutoDisable ();

return isSupported;

}

void OnRenderImage (RenderTexture source, RenderTexture destination) {

if (CheckResources()==false) {

Graphics.Blit (source, destination);

return;

}

// we actually need to check this every frame

if (useDepthTexture)

GetComponent<Camera>().depthTextureMode |= DepthTextureMode.Depth;

int divider = 4;

if (resolution == SunShaftsResolution.Normal)

divider = 2;

else if (resolution == SunShaftsResolution.High)

divider = 1;

Vector3 v = Vector3.one \* 0.5f;

if (sunTransform)

v = GetComponent<Camera>().WorldToViewportPoint (sunTransform.position);

else

v = new Vector3(0.5f, 0.5f, 0.0f);

int rtW = source.width / divider;

int rtH = source.height / divider;

RenderTexture lrColorB;

RenderTexture lrDepthBuffer = RenderTexture.GetTemporary (rtW, rtH, 0);

// mask out everything except the skybox

// we have 2 methods, one of which requires depth buffer support, the other one is just comparing images

sunShaftsMaterial.SetVector ("\_BlurRadius4", new Vector4 (1.0f, 1.0f, 0.0f, 0.0f) \* sunShaftBlurRadius );

sunShaftsMaterial.SetVector ("\_SunPosition", new Vector4 (v.x, v.y, v.z, maxRadius));

sunShaftsMaterial.SetVector ("\_SunThreshold", sunThreshold);

if (!useDepthTexture) {

var format= GetComponent<Camera>().hdr ? RenderTextureFormat.DefaultHDR: RenderTextureFormat.Default;

RenderTexture tmpBuffer = RenderTexture.GetTemporary (source.width, source.height, 0, format);

RenderTexture.active = tmpBuffer;

GL.ClearWithSkybox (false, GetComponent<Camera>());

sunShaftsMaterial.SetTexture ("\_Skybox", tmpBuffer);

Graphics.Blit (source, lrDepthBuffer, sunShaftsMaterial, 3);

RenderTexture.ReleaseTemporary (tmpBuffer);

}

else {

Graphics.Blit (source, lrDepthBuffer, sunShaftsMaterial, 2);

}

// paint a small black small border to get rid of clamping problems

DrawBorder (lrDepthBuffer, simpleClearMaterial);

// radial blur:

radialBlurIterations = Mathf.Clamp (radialBlurIterations, 1, 4);

float ofs = sunShaftBlurRadius \* (1.0f / 768.0f);

sunShaftsMaterial.SetVector ("\_BlurRadius4", new Vector4 (ofs, ofs, 0.0f, 0.0f));

sunShaftsMaterial.SetVector ("\_SunPosition", new Vector4 (v.x, v.y, v.z, maxRadius));

for (int it2 = 0; it2 < radialBlurIterations; it2++ ) {

// each iteration takes 2 \* 6 samples

// we update \_BlurRadius each time to cheaply get a very smooth look

lrColorB = RenderTexture.GetTemporary (rtW, rtH, 0);

Graphics.Blit (lrDepthBuffer, lrColorB, sunShaftsMaterial, 1);

RenderTexture.ReleaseTemporary (lrDepthBuffer);

ofs = sunShaftBlurRadius \* (((it2 \* 2.0f + 1.0f) \* 6.0f)) / 768.0f;

sunShaftsMaterial.SetVector ("\_BlurRadius4", new Vector4 (ofs, ofs, 0.0f, 0.0f) );

lrDepthBuffer = RenderTexture.GetTemporary (rtW, rtH, 0);

Graphics.Blit (lrColorB, lrDepthBuffer, sunShaftsMaterial, 1);

RenderTexture.ReleaseTemporary (lrColorB);

ofs = sunShaftBlurRadius \* (((it2 \* 2.0f + 2.0f) \* 6.0f)) / 768.0f;

sunShaftsMaterial.SetVector ("\_BlurRadius4", new Vector4 (ofs, ofs, 0.0f, 0.0f) );

}

// put together:

if (v.z >= 0.0f)

sunShaftsMaterial.SetVector ("\_SunColor", new Vector4 (sunColor.r, sunColor.g, sunColor.b, sunColor.a) \* sunShaftIntensity);

else

sunShaftsMaterial.SetVector ("\_SunColor", Vector4.zero); // no backprojection !

sunShaftsMaterial.SetTexture ("\_ColorBuffer", lrDepthBuffer);

Graphics.Blit (source, destination, sunShaftsMaterial, (screenBlendMode == ShaftsScreenBlendMode.Screen) ? 0 : 4);

RenderTexture.ReleaseTemporary (lrDepthBuffer);

}

}

}