using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

[AddComponentMenu ("Image Effects/Camera/Vignette and Chromatic Aberration")]

public class VignetteAndChromaticAberration : PostEffectsBase

{

public enum AberrationMode

{

Simple = 0,

Advanced = 1,

}

public AberrationMode mode = AberrationMode.Simple;

public float intensity = 0.375f; // intensity == 0 disables pre pass (optimization)

public float chromaticAberration = 0.2f;

public float axialAberration = 0.5f;

public float blur = 0.0f; // blur == 0 disables blur pass (optimization)

public float blurSpread = 0.75f;

public float luminanceDependency = 0.25f;

public float blurDistance = 2.5f;

public Shader vignetteShader;

public Shader separableBlurShader;

public Shader chromAberrationShader;

private Material m\_VignetteMaterial;

private Material m\_SeparableBlurMaterial;

private Material m\_ChromAberrationMaterial;

public override bool CheckResources ()

{

CheckSupport (false);

m\_VignetteMaterial = CheckShaderAndCreateMaterial (vignetteShader, m\_VignetteMaterial);

m\_SeparableBlurMaterial = CheckShaderAndCreateMaterial (separableBlurShader, m\_SeparableBlurMaterial);

m\_ChromAberrationMaterial = CheckShaderAndCreateMaterial (chromAberrationShader, m\_ChromAberrationMaterial);

if (!isSupported)

ReportAutoDisable ();

return isSupported;

}

void OnRenderImage (RenderTexture source, RenderTexture destination)

{

if ( CheckResources () == false)

{

Graphics.Blit (source, destination);

return;

}

int rtW = source.width;

int rtH = source.height;

bool doPrepass = (Mathf.Abs(blur)>0.0f || Mathf.Abs(intensity)>0.0f);

float widthOverHeight = (1.0f \* rtW) / (1.0f \* rtH);

const float oneOverBaseSize = 1.0f / 512.0f;

RenderTexture color = null;

RenderTexture color2A = null;

if (doPrepass)

{

color = RenderTexture.GetTemporary (rtW, rtH, 0, source.format);

// Blur corners

if (Mathf.Abs (blur)>0.0f)

{

color2A = RenderTexture.GetTemporary (rtW / 2, rtH / 2, 0, source.format);

Graphics.Blit (source, color2A, m\_ChromAberrationMaterial, 0);

for(int i = 0; i < 2; i++)

{ // maybe make iteration count tweakable

m\_SeparableBlurMaterial.SetVector ("offsets",new Vector4 (0.0f, blurSpread \* oneOverBaseSize, 0.0f, 0.0f));

RenderTexture color2B = RenderTexture.GetTemporary (rtW / 2, rtH / 2, 0, source.format);

Graphics.Blit (color2A, color2B, m\_SeparableBlurMaterial);

RenderTexture.ReleaseTemporary (color2A);

m\_SeparableBlurMaterial.SetVector ("offsets",new Vector4 (blurSpread \* oneOverBaseSize / widthOverHeight, 0.0f, 0.0f, 0.0f));

color2A = RenderTexture.GetTemporary (rtW / 2, rtH / 2, 0, source.format);

Graphics.Blit (color2B, color2A, m\_SeparableBlurMaterial);

RenderTexture.ReleaseTemporary (color2B);

}

}

m\_VignetteMaterial.SetFloat ("\_Intensity", intensity); // intensity for vignette

m\_VignetteMaterial.SetFloat ("\_Blur", blur); // blur intensity

m\_VignetteMaterial.SetTexture ("\_VignetteTex", color2A); // blurred texture

Graphics.Blit (source, color, m\_VignetteMaterial, 0); // prepass blit: darken & blur corners

}

m\_ChromAberrationMaterial.SetFloat ("\_ChromaticAberration", chromaticAberration);

m\_ChromAberrationMaterial.SetFloat ("\_AxialAberration", axialAberration);

m\_ChromAberrationMaterial.SetVector ("\_BlurDistance", new Vector2 (-blurDistance, blurDistance));

m\_ChromAberrationMaterial.SetFloat ("\_Luminance", 1.0f/Mathf.Max(Mathf.Epsilon, luminanceDependency));

if (doPrepass) color.wrapMode = TextureWrapMode.Clamp;

else source.wrapMode = TextureWrapMode.Clamp;

Graphics.Blit (doPrepass ? color : source, destination, m\_ChromAberrationMaterial, mode == AberrationMode.Advanced ? 2 : 1);

RenderTexture.ReleaseTemporary (color);

RenderTexture.ReleaseTemporary (color2A);

}

}

}