using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

[AddComponentMenu ("Image Effects/Blur/Blur (Optimized)")]

public class BlurOptimized : PostEffectsBase

{

[Range(0, 2)]

public int downsample = 1;

public enum BlurType {

StandardGauss = 0,

SgxGauss = 1,

}

[Range(0.0f, 10.0f)]

public float blurSize = 3.0f;

[Range(1, 4)]

public int blurIterations = 2;

public BlurType blurType= BlurType.StandardGauss;

public Shader blurShader = null;

private Material blurMaterial = null;

public override bool CheckResources () {

CheckSupport (false);

blurMaterial = CheckShaderAndCreateMaterial (blurShader, blurMaterial);

if (!isSupported)

ReportAutoDisable ();

return isSupported;

}

public void OnDisable () {

if (blurMaterial)

DestroyImmediate (blurMaterial);

}

public void OnRenderImage (RenderTexture source, RenderTexture destination) {

if (CheckResources() == false) {

Graphics.Blit (source, destination);

return;

}

float widthMod = 1.0f / (1.0f \* (1<<downsample));

blurMaterial.SetVector ("\_Parameter", new Vector4 (blurSize \* widthMod, -blurSize \* widthMod, 0.0f, 0.0f));

source.filterMode = FilterMode.Bilinear;

int rtW = source.width >> downsample;

int rtH = source.height >> downsample;

// downsample

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

rt.filterMode = FilterMode.Bilinear;

Graphics.Blit (source, rt, blurMaterial, 0);

var passOffs= blurType == BlurType.StandardGauss ? 0 : 2;

for(int i = 0; i < blurIterations; i++) {

float iterationOffs = (i\*1.0f);

blurMaterial.SetVector ("\_Parameter", new Vector4 (blurSize \* widthMod + iterationOffs, -blurSize \* widthMod - iterationOffs, 0.0f, 0.0f));

// vertical blur

RenderTexture rt2 = RenderTexture.GetTemporary (rtW, rtH, 0, source.format);

rt2.filterMode = FilterMode.Bilinear;

Graphics.Blit (rt, rt2, blurMaterial, 1 + passOffs);

RenderTexture.ReleaseTemporary (rt);

rt = rt2;

// horizontal blur

rt2 = RenderTexture.GetTemporary (rtW, rtH, 0, source.format);

rt2.filterMode = FilterMode.Bilinear;

Graphics.Blit (rt, rt2, blurMaterial, 2 + passOffs);

RenderTexture.ReleaseTemporary (rt);

rt = rt2;

}

Graphics.Blit (rt, destination);

RenderTexture.ReleaseTemporary (rt);

}

}

}