using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

[AddComponentMenu ("Image Effects/Bloom and Glow/Bloom (Optimized)")]

public class BloomOptimized : PostEffectsBase

{

public enum Resolution

{

Low = 0,

High = 1,

}

public enum BlurType

{

Standard = 0,

Sgx = 1,

}

[Range(0.0f, 1.5f)]

public float threshold = 0.25f;

[Range(0.0f, 2.5f)]

public float intensity = 0.75f;

[Range(0.25f, 5.5f)]

public float blurSize = 1.0f;

Resolution resolution = Resolution.Low;

[Range(1, 4)]

public int blurIterations = 1;

public BlurType blurType= BlurType.Standard;

public Shader fastBloomShader = null;

private Material fastBloomMaterial = null;

public override bool CheckResources ()

{

CheckSupport (false);

fastBloomMaterial = CheckShaderAndCreateMaterial (fastBloomShader, fastBloomMaterial);

if (!isSupported)

ReportAutoDisable ();

return isSupported;

}

void OnDisable ()

{

if (fastBloomMaterial)

DestroyImmediate (fastBloomMaterial);

}

void OnRenderImage (RenderTexture source, RenderTexture destination)

{

if (CheckResources() == false)

{

Graphics.Blit (source, destination);

return;

}

int divider = resolution == Resolution.Low ? 4 : 2;

float widthMod = resolution == Resolution.Low ? 0.5f : 1.0f;

fastBloomMaterial.SetVector ("\_Parameter", new Vector4 (blurSize \* widthMod, 0.0f, threshold, intensity));

source.filterMode = FilterMode.Bilinear;

var rtW= source.width/divider;

var rtH= source.height/divider;

// downsample

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

rt.filterMode = FilterMode.Bilinear;

Graphics.Blit (source, rt, fastBloomMaterial, 1);

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

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

{

fastBloomMaterial.SetVector ("\_Parameter", new Vector4 (blurSize \* widthMod + (i\*1.0f), 0.0f, threshold, intensity));

// vertical blur

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

rt2.filterMode = FilterMode.Bilinear;

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

RenderTexture.ReleaseTemporary (rt);

rt = rt2;

// horizontal blur

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

rt2.filterMode = FilterMode.Bilinear;

Graphics.Blit (rt, rt2, fastBloomMaterial, 3 + passOffs);

RenderTexture.ReleaseTemporary (rt);

rt = rt2;

}

fastBloomMaterial.SetTexture ("\_Bloom", rt);

Graphics.Blit (source, destination, fastBloomMaterial, 0);

RenderTexture.ReleaseTemporary (rt);

}

}

}