using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

public enum AAMode

{

FXAA2 = 0,

FXAA3Console = 1,

FXAA1PresetA = 2,

FXAA1PresetB = 3,

NFAA = 4,

SSAA = 5,

DLAA = 6,

}

[ExecuteInEditMode]

[RequireComponent(typeof (Camera))]

[AddComponentMenu("Image Effects/Other/Antialiasing")]

public class Antialiasing : PostEffectsBase

{

public AAMode mode = AAMode.FXAA3Console;

public bool showGeneratedNormals = false;

public float offsetScale = 0.2f;

public float blurRadius = 18.0f;

public float edgeThresholdMin = 0.05f;

public float edgeThreshold = 0.2f;

public float edgeSharpness = 4.0f;

public bool dlaaSharp = false;

public Shader ssaaShader;

private Material ssaa;

public Shader dlaaShader;

private Material dlaa;

public Shader nfaaShader;

private Material nfaa;

public Shader shaderFXAAPreset2;

private Material materialFXAAPreset2;

public Shader shaderFXAAPreset3;

private Material materialFXAAPreset3;

public Shader shaderFXAAII;

private Material materialFXAAII;

public Shader shaderFXAAIII;

private Material materialFXAAIII;

public Material CurrentAAMaterial()

{

Material returnValue = null;

switch (mode)

{

case AAMode.FXAA3Console:

returnValue = materialFXAAIII;

break;

case AAMode.FXAA2:

returnValue = materialFXAAII;

break;

case AAMode.FXAA1PresetA:

returnValue = materialFXAAPreset2;

break;

case AAMode.FXAA1PresetB:

returnValue = materialFXAAPreset3;

break;

case AAMode.NFAA:

returnValue = nfaa;

break;

case AAMode.SSAA:

returnValue = ssaa;

break;

case AAMode.DLAA:

returnValue = dlaa;

break;

default:

returnValue = null;

break;

}

return returnValue;

}

public override bool CheckResources()

{

CheckSupport(false);

materialFXAAPreset2 = CreateMaterial(shaderFXAAPreset2, materialFXAAPreset2);

materialFXAAPreset3 = CreateMaterial(shaderFXAAPreset3, materialFXAAPreset3);

materialFXAAII = CreateMaterial(shaderFXAAII, materialFXAAII);

materialFXAAIII = CreateMaterial(shaderFXAAIII, materialFXAAIII);

nfaa = CreateMaterial(nfaaShader, nfaa);

ssaa = CreateMaterial(ssaaShader, ssaa);

dlaa = CreateMaterial(dlaaShader, dlaa);

if (!ssaaShader.isSupported)

{

NotSupported();

ReportAutoDisable();

}

return isSupported;

}

public void OnRenderImage(RenderTexture source, RenderTexture destination)

{

if (CheckResources() == false)

{

Graphics.Blit(source, destination);

return;

}

// ----------------------------------------------------------------

// FXAA antialiasing modes

if (mode == AAMode.FXAA3Console && (materialFXAAIII != null))

{

materialFXAAIII.SetFloat("\_EdgeThresholdMin", edgeThresholdMin);

materialFXAAIII.SetFloat("\_EdgeThreshold", edgeThreshold);

materialFXAAIII.SetFloat("\_EdgeSharpness", edgeSharpness);

Graphics.Blit(source, destination, materialFXAAIII);

}

else if (mode == AAMode.FXAA1PresetB && (materialFXAAPreset3 != null))

{

Graphics.Blit(source, destination, materialFXAAPreset3);

}

else if (mode == AAMode.FXAA1PresetA && materialFXAAPreset2 != null)

{

source.anisoLevel = 4;

Graphics.Blit(source, destination, materialFXAAPreset2);

source.anisoLevel = 0;

}

else if (mode == AAMode.FXAA2 && materialFXAAII != null)

{

Graphics.Blit(source, destination, materialFXAAII);

}

else if (mode == AAMode.SSAA && ssaa != null)

{

// ----------------------------------------------------------------

// SSAA antialiasing

Graphics.Blit(source, destination, ssaa);

}

else if (mode == AAMode.DLAA && dlaa != null)

{

// ----------------------------------------------------------------

// DLAA antialiasing

source.anisoLevel = 0;

RenderTexture interim = RenderTexture.GetTemporary(source.width, source.height);

Graphics.Blit(source, interim, dlaa, 0);

Graphics.Blit(interim, destination, dlaa, dlaaSharp ? 2 : 1);

RenderTexture.ReleaseTemporary(interim);

}

else if (mode == AAMode.NFAA && nfaa != null)

{

// ----------------------------------------------------------------

// nfaa antialiasing

source.anisoLevel = 0;

nfaa.SetFloat("\_OffsetScale", offsetScale);

nfaa.SetFloat("\_BlurRadius", blurRadius);

Graphics.Blit(source, destination, nfaa, showGeneratedNormals ? 1 : 0);

}

else

{

// none of the AA is supported, fallback to a simple blit

Graphics.Blit(source, destination);

}

}

}

}