using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

class PostEffectsHelper : MonoBehaviour

{

void OnRenderImage (RenderTexture source, RenderTexture destination)

{

Debug.Log("OnRenderImage in Helper called ...");

}

static void DrawLowLevelPlaneAlignedWithCamera (

float dist ,

RenderTexture source, RenderTexture dest ,

Material material ,

Camera cameraForProjectionMatrix )

{

// Make the destination texture the target for all rendering

RenderTexture.active = dest;

// Assign the source texture to a property from a shader

material.SetTexture("\_MainTex", source);

bool invertY = true; // source.texelSize.y < 0.0f;

// Set up the simple Matrix

GL.PushMatrix();

GL.LoadIdentity();

GL.LoadProjectionMatrix(cameraForProjectionMatrix.projectionMatrix);

float fovYHalfRad = cameraForProjectionMatrix.fieldOfView \* 0.5f \* Mathf.Deg2Rad;

float cotangent = Mathf.Cos(fovYHalfRad) / Mathf.Sin(fovYHalfRad);

float asp = cameraForProjectionMatrix.aspect;

float x1 = asp/-cotangent;

float x2 = asp/cotangent;

float y1 = 1.0f/-cotangent;

float y2 = 1.0f/cotangent;

float sc = 1.0f; // magic constant (for now)

x1 \*= dist \* sc;

x2 \*= dist \* sc;

y1 \*= dist \* sc;

y2 \*= dist \* sc;

float z1 = -dist;

for (int i = 0; i < material.passCount; i++)

{

material.SetPass(i);

GL.Begin(GL.QUADS);

float y1\_; float y2\_;

if (invertY)

{

y1\_ = 1.0f; y2\_ = 0.0f;

}

else

{

y1\_ = 0.0f; y2\_ = 1.0f;

}

GL.TexCoord2(0.0f, y1\_); GL.Vertex3(x1, y1, z1);

GL.TexCoord2(1.0f, y1\_); GL.Vertex3(x2, y1, z1);

GL.TexCoord2(1.0f, y2\_); GL.Vertex3(x2, y2, z1);

GL.TexCoord2(0.0f, y2\_); GL.Vertex3(x1, y2, z1);

GL.End();

}

GL.PopMatrix();

}

static void DrawBorder (

RenderTexture dest ,

Material material )

{

float x1;

float x2;

float y1;

float y2;

RenderTexture.active = dest;

bool invertY = true; // source.texelSize.y < 0.0ff;

// Set up the simple Matrix

GL.PushMatrix();

GL.LoadOrtho();

for (int i = 0; i < material.passCount; i++)

{

material.SetPass(i);

float y1\_; float y2\_;

if (invertY)

{

y1\_ = 1.0f; y2\_ = 0.0f;

}

else

{

y1\_ = 0.0f; y2\_ = 1.0f;

}

// left

x1 = 0.0f;

x2 = 0.0f + 1.0f/(dest.width\*1.0f);

y1 = 0.0f;

y2 = 1.0f;

GL.Begin(GL.QUADS);

GL.TexCoord2(0.0f, y1\_); GL.Vertex3(x1, y1, 0.1f);

GL.TexCoord2(1.0f, y1\_); GL.Vertex3(x2, y1, 0.1f);

GL.TexCoord2(1.0f, y2\_); GL.Vertex3(x2, y2, 0.1f);

GL.TexCoord2(0.0f, y2\_); GL.Vertex3(x1, y2, 0.1f);

// right

x1 = 1.0f - 1.0f/(dest.width\*1.0f);

x2 = 1.0f;

y1 = 0.0f;

y2 = 1.0f;

GL.TexCoord2(0.0f, y1\_); GL.Vertex3(x1, y1, 0.1f);

GL.TexCoord2(1.0f, y1\_); GL.Vertex3(x2, y1, 0.1f);

GL.TexCoord2(1.0f, y2\_); GL.Vertex3(x2, y2, 0.1f);

GL.TexCoord2(0.0f, y2\_); GL.Vertex3(x1, y2, 0.1f);

// top

x1 = 0.0f;

x2 = 1.0f;

y1 = 0.0f;

y2 = 0.0f + 1.0f/(dest.height\*1.0f);

GL.TexCoord2(0.0f, y1\_); GL.Vertex3(x1, y1, 0.1f);

GL.TexCoord2(1.0f, y1\_); GL.Vertex3(x2, y1, 0.1f);

GL.TexCoord2(1.0f, y2\_); GL.Vertex3(x2, y2, 0.1f);

GL.TexCoord2(0.0f, y2\_); GL.Vertex3(x1, y2, 0.1f);

// bottom

x1 = 0.0f;

x2 = 1.0f;

y1 = 1.0f - 1.0f/(dest.height\*1.0f);

y2 = 1.0f;

GL.TexCoord2(0.0f, y1\_); GL.Vertex3(x1, y1, 0.1f);

GL.TexCoord2(1.0f, y1\_); GL.Vertex3(x2, y1, 0.1f);

GL.TexCoord2(1.0f, y2\_); GL.Vertex3(x2, y2, 0.1f);

GL.TexCoord2(0.0f, y2\_); GL.Vertex3(x1, y2, 0.1f);

GL.End();

}

GL.PopMatrix();

}

static void DrawLowLevelQuad ( float x1, float x2, float y1, float y2, RenderTexture source, RenderTexture dest, Material material )

{

// Make the destination texture the target for all rendering

RenderTexture.active = dest;

// Assign the source texture to a property from a shader

material.SetTexture("\_MainTex", source);

bool invertY = true; // source.texelSize.y < 0.0f;

// Set up the simple Matrix

GL.PushMatrix();

GL.LoadOrtho();

for (int i = 0; i < material.passCount; i++)

{

material.SetPass(i);

GL.Begin(GL.QUADS);

float y1\_; float y2\_;

if (invertY)

{

y1\_ = 1.0f; y2\_ = 0.0f;

}

else

{

y1\_ = 0.0f; y2\_ = 1.0f;

}

GL.TexCoord2(0.0f, y1\_); GL.Vertex3(x1, y1, 0.1f);

GL.TexCoord2(1.0f, y1\_); GL.Vertex3(x2, y1, 0.1f);

GL.TexCoord2(1.0f, y2\_); GL.Vertex3(x2, y2, 0.1f);

GL.TexCoord2(0.0f, y2\_); GL.Vertex3(x1, y2, 0.1f);

GL.End();

}

GL.PopMatrix();

}

}

}