using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

public class PostEffectsBase : MonoBehaviour

{

protected bool supportHDRTextures = true;

protected bool supportDX11 = false;

protected bool isSupported = true;

protected Material CheckShaderAndCreateMaterial ( Shader s, Material m2Create)

{

if (!s)

{

Debug.Log("Missing shader in " + ToString ());

enabled = false;

return null;

}

if (s.isSupported && m2Create && m2Create.shader == s)

return m2Create;

if (!s.isSupported)

{

NotSupported ();

Debug.Log("The shader " + s.ToString() + " on effect "+ToString()+" is not supported on this platform!");

return null;

}

else

{

m2Create = new Material (s);

m2Create.hideFlags = HideFlags.DontSave;

if (m2Create)

return m2Create;

else return null;

}

}

protected Material CreateMaterial (Shader s, Material m2Create)

{

if (!s)

{

Debug.Log ("Missing shader in " + ToString ());

return null;

}

if (m2Create && (m2Create.shader == s) && (s.isSupported))

return m2Create;

if (!s.isSupported)

{

return null;

}

else

{

m2Create = new Material (s);

m2Create.hideFlags = HideFlags.DontSave;

if (m2Create)

return m2Create;

else return null;

}

}

void OnEnable ()

{

isSupported = true;

}

protected bool CheckSupport ()

{

return CheckSupport (false);

}

public virtual bool CheckResources ()

{

Debug.LogWarning ("CheckResources () for " + ToString() + " should be overwritten.");

return isSupported;

}

protected void Start ()

{

CheckResources ();

}

protected bool CheckSupport (bool needDepth)

{

isSupported = true;

supportHDRTextures = SystemInfo.SupportsRenderTextureFormat(RenderTextureFormat.ARGBHalf);

supportDX11 = SystemInfo.graphicsShaderLevel >= 50 && SystemInfo.supportsComputeShaders;

if (!SystemInfo.supportsImageEffects || !SystemInfo.supportsRenderTextures)

{

NotSupported ();

return false;

}

if (needDepth && !SystemInfo.SupportsRenderTextureFormat (RenderTextureFormat.Depth))

{

NotSupported ();

return false;

}

if (needDepth)

GetComponent<Camera>().depthTextureMode |= DepthTextureMode.Depth;

return true;

}

protected bool CheckSupport (bool needDepth, bool needHdr)

{

if (!CheckSupport(needDepth))

return false;

if (needHdr && !supportHDRTextures)

{

NotSupported ();

return false;

}

return true;

}

public bool Dx11Support ()

{

return supportDX11;

}

protected void ReportAutoDisable ()

{

Debug.LogWarning ("The image effect " + ToString() + " has been disabled as it's not supported on the current platform.");

}

// deprecated but needed for old effects to survive upgrading

bool CheckShader (Shader s)

{

Debug.Log("The shader " + s.ToString () + " on effect "+ ToString () + " is not part of the Unity 3.2+ effects suite anymore. For best performance and quality, please ensure you are using the latest Standard Assets Image Effects (Pro only) package.");

if (!s.isSupported)

{

NotSupported ();

return false;

}

else

{

return false;

}

}

protected void NotSupported ()

{

enabled = false;

isSupported = false;

return;

}

protected 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();

}

}

}