using System;

using UnityEngine;

namespace UnityStandardAssets.ImageEffects

{

[ExecuteInEditMode]

[RequireComponent (typeof(Camera))]

[AddComponentMenu ("Image Effects/Rendering/Global Fog")]

class GlobalFog : PostEffectsBase

{

[Tooltip("Apply distance-based fog?")]

public bool distanceFog = true;

[Tooltip("Distance fog is based on radial distance from camera when checked")]

public bool useRadialDistance = false;

[Tooltip("Apply height-based fog?")]

public bool heightFog = true;

[Tooltip("Fog top Y coordinate")]

public float height = 1.0f;

[Range(0.001f,10.0f)]

public float heightDensity = 2.0f;

[Tooltip("Push fog away from the camera by this amount")]

public float startDistance = 0.0f;

public Shader fogShader = null;

private Material fogMaterial = null;

public override bool CheckResources ()

{

CheckSupport (true);

fogMaterial = CheckShaderAndCreateMaterial (fogShader, fogMaterial);

if (!isSupported)

ReportAutoDisable ();

return isSupported;

}

[ImageEffectOpaque]

void OnRenderImage (RenderTexture source, RenderTexture destination)

{

if (CheckResources()==false || (!distanceFog && !heightFog))

{

Graphics.Blit (source, destination);

return;

}

Camera cam = GetComponent<Camera>();

Transform camtr = cam.transform;

float camNear = cam.nearClipPlane;

float camFar = cam.farClipPlane;

float camFov = cam.fieldOfView;

float camAspect = cam.aspect;

Matrix4x4 frustumCorners = Matrix4x4.identity;

float fovWHalf = camFov \* 0.5f;

Vector3 toRight = camtr.right \* camNear \* Mathf.Tan (fovWHalf \* Mathf.Deg2Rad) \* camAspect;

Vector3 toTop = camtr.up \* camNear \* Mathf.Tan (fovWHalf \* Mathf.Deg2Rad);

Vector3 topLeft = (camtr.forward \* camNear - toRight + toTop);

float camScale = topLeft.magnitude \* camFar/camNear;

topLeft.Normalize();

topLeft \*= camScale;

Vector3 topRight = (camtr.forward \* camNear + toRight + toTop);

topRight.Normalize();

topRight \*= camScale;

Vector3 bottomRight = (camtr.forward \* camNear + toRight - toTop);

bottomRight.Normalize();

bottomRight \*= camScale;

Vector3 bottomLeft = (camtr.forward \* camNear - toRight - toTop);

bottomLeft.Normalize();

bottomLeft \*= camScale;

frustumCorners.SetRow (0, topLeft);

frustumCorners.SetRow (1, topRight);

frustumCorners.SetRow (2, bottomRight);

frustumCorners.SetRow (3, bottomLeft);

var camPos= camtr.position;

float FdotC = camPos.y-height;

float paramK = (FdotC <= 0.0f ? 1.0f : 0.0f);

fogMaterial.SetMatrix ("\_FrustumCornersWS", frustumCorners);

fogMaterial.SetVector ("\_CameraWS", camPos);

fogMaterial.SetVector ("\_HeightParams", new Vector4 (height, FdotC, paramK, heightDensity\*0.5f));

fogMaterial.SetVector ("\_DistanceParams", new Vector4 (-Mathf.Max(startDistance,0.0f), 0, 0, 0));

var sceneMode= RenderSettings.fogMode;

var sceneDensity= RenderSettings.fogDensity;

var sceneStart= RenderSettings.fogStartDistance;

var sceneEnd= RenderSettings.fogEndDistance;

Vector4 sceneParams;

bool linear = (sceneMode == FogMode.Linear);

float diff = linear ? sceneEnd - sceneStart : 0.0f;

float invDiff = Mathf.Abs(diff) > 0.0001f ? 1.0f / diff : 0.0f;

sceneParams.x = sceneDensity \* 1.2011224087f; // density / sqrt(ln(2)), used by Exp2 fog mode

sceneParams.y = sceneDensity \* 1.4426950408f; // density / ln(2), used by Exp fog mode

sceneParams.z = linear ? -invDiff : 0.0f;

sceneParams.w = linear ? sceneEnd \* invDiff : 0.0f;

fogMaterial.SetVector ("\_SceneFogParams", sceneParams);

fogMaterial.SetVector ("\_SceneFogMode", new Vector4((int)sceneMode, useRadialDistance ? 1 : 0, 0, 0));

int pass = 0;

if (distanceFog && heightFog)

pass = 0; // distance + height

else if (distanceFog)

pass = 1; // distance only

else

pass = 2; // height only

CustomGraphicsBlit (source, destination, fogMaterial, pass);

}

static void CustomGraphicsBlit (RenderTexture source, RenderTexture dest, Material fxMaterial, int passNr)

{

RenderTexture.active = dest;

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

GL.PushMatrix ();

GL.LoadOrtho ();

fxMaterial.SetPass (passNr);

GL.Begin (GL.QUADS);

GL.MultiTexCoord2 (0, 0.0f, 0.0f);

GL.Vertex3 (0.0f, 0.0f, 3.0f); // BL

GL.MultiTexCoord2 (0, 1.0f, 0.0f);

GL.Vertex3 (1.0f, 0.0f, 2.0f); // BR

GL.MultiTexCoord2 (0, 1.0f, 1.0f);

GL.Vertex3 (1.0f, 1.0f, 1.0f); // TR

GL.MultiTexCoord2 (0, 0.0f, 1.0f);

GL.Vertex3 (0.0f, 1.0f, 0.0f); // TL

GL.End ();

GL.PopMatrix ();

}

}

}