using System;

using UnityEngine;

namespace UnityStandardAssets.Cameras

{

public class TargetFieldOfView : AbstractTargetFollower

{

// This script is primarily designed to be used with the "LookAtTarget" script to enable a

// CCTV style camera looking at a target to also adjust its field of view (zoom) to fit the

// target (so that it zooms in as the target becomes further away).

// When used with a follow cam, it will automatically use the same target.

[SerializeField] private float m\_FovAdjustTime = 1; // the time taken to adjust the current FOV to the desired target FOV amount.

[SerializeField] private float m\_ZoomAmountMultiplier = 2; // a multiplier for the FOV amount. The default of 2 makes the field of view twice as wide as required to fit the target.

[SerializeField] private bool m\_IncludeEffectsInSize = false; // changing this only takes effect on startup, or when new target is assigned.

private float m\_BoundSize;

private float m\_FovAdjustVelocity;

private Camera m\_Cam;

private Transform m\_LastTarget;

// Use this for initialization

protected override void Start()

{

base.Start();

m\_BoundSize = MaxBoundsExtent(m\_Target, m\_IncludeEffectsInSize);

// get a reference to the actual camera component:

m\_Cam = GetComponentInChildren<Camera>();

}

protected override void FollowTarget(float deltaTime)

{

// calculate the correct field of view to fit the bounds size at the current distance

float dist = (m\_Target.position - transform.position).magnitude;

float requiredFOV = Mathf.Atan2(m\_BoundSize, dist)\*Mathf.Rad2Deg\*m\_ZoomAmountMultiplier;

m\_Cam.fieldOfView = Mathf.SmoothDamp(m\_Cam.fieldOfView, requiredFOV, ref m\_FovAdjustVelocity, m\_FovAdjustTime);

}

public override void SetTarget(Transform newTransform)

{

base.SetTarget(newTransform);

m\_BoundSize = MaxBoundsExtent(newTransform, m\_IncludeEffectsInSize);

}

public static float MaxBoundsExtent(Transform obj, bool includeEffects)

{

// get the maximum bounds extent of object, including all child renderers,

// but excluding particles and trails, for FOV zooming effect.

var renderers = obj.GetComponentsInChildren<Renderer>();

Bounds bounds = new Bounds();

bool initBounds = false;

foreach (Renderer r in renderers)

{

if (!((r is TrailRenderer) || (r is ParticleRenderer) || (r is ParticleSystemRenderer)))

{

if (!initBounds)

{

initBounds = true;

bounds = r.bounds;

}

else

{

bounds.Encapsulate(r.bounds);

}

}

}

float max = Mathf.Max(bounds.extents.x, bounds.extents.y, bounds.extents.z);

return max;

}

}

}