using System;

using UnityEngine;

namespace UnityStandardAssets.Vehicles.Aeroplane

{

[RequireComponent(typeof (ParticleSystem))]

public class JetParticleEffect : MonoBehaviour

{

// this script controls the jet's exhaust particle system, controlling the

// size and colour based on the jet's current throttle value.

public Color minColour; // The base colour for the effect to start at

private AeroplaneController m\_Jet; // The jet that the particle effect is attached to

private ParticleSystem m\_System; // The particle system that is being controlled

private float m\_OriginalStartSize; // The original starting size of the particle system

private float m\_OriginalLifetime; // The original lifetime of the particle system

private Color m\_OriginalStartColor; // The original starting colout of the particle system

// Use this for initialization

private void Start()

{

// get the aeroplane from the object hierarchy

m\_Jet = FindAeroplaneParent();

// get the particle system ( it will be on the object as we have a require component set up

m\_System = GetComponent<ParticleSystem>();

// set the original properties from the particle system

m\_OriginalLifetime = m\_System.startLifetime;

m\_OriginalStartSize = m\_System.startSize;

m\_OriginalStartColor = m\_System.startColor;

}

// Update is called once per frame

private void Update()

{

// update the particle system based on the jets throttle

m\_System.startLifetime = Mathf.Lerp(0.0f, m\_OriginalLifetime, m\_Jet.Throttle);

m\_System.startSize = Mathf.Lerp(m\_OriginalStartSize\*.3f, m\_OriginalStartSize, m\_Jet.Throttle);

m\_System.startColor = Color.Lerp(minColour, m\_OriginalStartColor, m\_Jet.Throttle);

}

private AeroplaneController FindAeroplaneParent()

{

// get reference to the object transform

var t = transform;

// traverse the object hierarchy upwards to find the AeroplaneController

// (since this is placed on a child object)

while (t != null)

{

var aero = t.GetComponent<AeroplaneController>();

if (aero == null)

{

// try next parent

t = t.parent;

}

else

{

return aero;

}

}

// controller not found!

throw new Exception(" AeroplaneContoller not found in object hierarchy");

}

}

}