using System;

using UnityEngine;

namespace UnityStandardAssets.Vehicles.Aeroplane

{

public class AeroplanePropellerAnimator : MonoBehaviour

{

[SerializeField] private Transform m\_PropellorModel; // The model of the the aeroplane's propellor.

[SerializeField] private Transform m\_PropellorBlur; // The plane used for the blurred propellor textures.

[SerializeField] private Texture2D[] m\_PropellorBlurTextures; // An array of increasingly blurred propellor textures.

[SerializeField] [Range(0f, 1f)] private float m\_ThrottleBlurStart = 0.25f; // The point at which the blurred textures start.

[SerializeField] [Range(0f, 1f)] private float m\_ThrottleBlurEnd = 0.5f; // The point at which the blurred textures stop changing.

[SerializeField] private float m\_MaxRpm = 2000; // The maximum speed the propellor can turn at.

private AeroplaneController m\_Plane; // Reference to the aeroplane controller.

private int m\_PropellorBlurState = -1; // To store the state of the blurred textures.

private const float k\_RpmToDps = 60f; // For converting from revs per minute to degrees per second.

private Renderer m\_PropellorModelRenderer;

private Renderer m\_PropellorBlurRenderer;

private void Awake()

{

// Set up the reference to the aeroplane controller.

m\_Plane = GetComponent<AeroplaneController>();

m\_PropellorModelRenderer = m\_PropellorModel.GetComponent<Renderer>();

m\_PropellorBlurRenderer = m\_PropellorBlur.GetComponent<Renderer>();

// Set the propellor blur gameobject's parent to be the propellor.

m\_PropellorBlur.parent = m\_PropellorModel;

}

private void Update()

{

// Rotate the propellor model at a rate proportional to the throttle.

m\_PropellorModel.Rotate(0, m\_MaxRpm\*m\_Plane.Throttle\*Time.deltaTime\*k\_RpmToDps, 0);

// Create an integer for the new state of the blur textures.

var newBlurState = 0;

// choose between the blurred textures, if the throttle is high enough

if (m\_Plane.Throttle > m\_ThrottleBlurStart)

{

var throttleBlurProportion = Mathf.InverseLerp(m\_ThrottleBlurStart, m\_ThrottleBlurEnd, m\_Plane.Throttle);

newBlurState = Mathf.FloorToInt(throttleBlurProportion\*(m\_PropellorBlurTextures.Length - 1));

}

// If the blur state has changed

if (newBlurState != m\_PropellorBlurState)

{

m\_PropellorBlurState = newBlurState;

if (m\_PropellorBlurState == 0)

{

// switch to using the 'real' propellor model

m\_PropellorModelRenderer.enabled = true;

m\_PropellorBlurRenderer.enabled = false;

}

else

{

// Otherwise turn off the propellor model and turn on the blur.

m\_PropellorModelRenderer.enabled = false;

m\_PropellorBlurRenderer.enabled = true;

// set the appropriate texture from the blur array

m\_PropellorBlurRenderer.material.mainTexture = m\_PropellorBlurTextures[m\_PropellorBlurState];

}

}

}

}

}