using System;

using UnityEngine;

namespace UnityStandardAssets.Vehicles.Aeroplane

{

public class AeroplaneAudio : MonoBehaviour

{

[Serializable]

public class AdvancedSetttings // A class for storing the advanced options.

{

public float engineMinDistance = 50f; // The min distance of the engine audio source.

public float engineMaxDistance = 1000f; // The max distance of the engine audio source.

public float engineDopplerLevel = 1f; // The doppler level of the engine audio source.

[Range(0f, 1f)] public float engineMasterVolume = 0.5f; // An overall control of the engine sound volume.

public float windMinDistance = 10f; // The min distance of the wind audio source.

public float windMaxDistance = 100f; // The max distance of the wind audio source.

public float windDopplerLevel = 1f; // The doppler level of the wind audio source.

[Range(0f, 1f)] public float windMasterVolume = 0.5f; // An overall control of the wind sound volume.

}

[SerializeField] private AudioClip m\_EngineSound; // Looped engine sound, whose pitch and volume are affected by the plane's throttle setting.

[SerializeField] private float m\_EngineMinThrottlePitch = 0.4f; // Pitch of the engine sound when at minimum throttle.

[SerializeField] private float m\_EngineMaxThrottlePitch = 2f; // Pitch of the engine sound when at maximum throttle.

[SerializeField] private float m\_EngineFwdSpeedMultiplier = 0.002f; // Additional multiplier for an increase in pitch of the engine from the plane's speed.

[SerializeField] private AudioClip m\_WindSound; // Looped wind sound, whose pitch and volume are affected by the plane's velocity.

[SerializeField] private float m\_WindBasePitch = 0.2f; // starting pitch for wind (when plane is at zero speed)

[SerializeField] private float m\_WindSpeedPitchFactor = 0.004f; // Relative increase in pitch of the wind from the plane's speed.

[SerializeField] private float m\_WindMaxSpeedVolume = 100; // the speed the aircraft much reach before the wind sound reaches maximum volume.

[SerializeField] private AdvancedSetttings m\_AdvancedSetttings = new AdvancedSetttings();// container to make advanced settings appear as rollout in inspector

private AudioSource m\_EngineSoundSource; // Reference to the AudioSource for the engine.

private AudioSource m\_WindSoundSource; // Reference to the AudioSource for the wind.

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

private Rigidbody m\_Rigidbody;

private void Awake()

{

// Set up the reference to the aeroplane controller.

m\_Plane = GetComponent<AeroplaneController>();

m\_Rigidbody = GetComponent<Rigidbody>();

// Add the audiosources and get the references.

m\_EngineSoundSource = gameObject.AddComponent<AudioSource>();

m\_EngineSoundSource.playOnAwake = false;

m\_WindSoundSource = gameObject.AddComponent<AudioSource>();

m\_WindSoundSource.playOnAwake = false;

// Assign clips to the audiosources.

m\_EngineSoundSource.clip = m\_EngineSound;

m\_WindSoundSource.clip = m\_WindSound;

// Set the parameters of the audiosources.

m\_EngineSoundSource.minDistance = m\_AdvancedSetttings.engineMinDistance;

m\_EngineSoundSource.maxDistance = m\_AdvancedSetttings.engineMaxDistance;

m\_EngineSoundSource.loop = true;

m\_EngineSoundSource.dopplerLevel = m\_AdvancedSetttings.engineDopplerLevel;

m\_WindSoundSource.minDistance = m\_AdvancedSetttings.windMinDistance;

m\_WindSoundSource.maxDistance = m\_AdvancedSetttings.windMaxDistance;

m\_WindSoundSource.loop = true;

m\_WindSoundSource.dopplerLevel = m\_AdvancedSetttings.windDopplerLevel;

// call update here to set the sounds pitch and volumes before they actually play

Update();

// Start the sounds playing.

m\_EngineSoundSource.Play();

m\_WindSoundSource.Play();

}

private void Update()

{

// Find what proportion of the engine's power is being used.

var enginePowerProportion = Mathf.InverseLerp(0, m\_Plane.MaxEnginePower, m\_Plane.EnginePower);

// Set the engine's pitch to be proportional to the engine's current power.

m\_EngineSoundSource.pitch = Mathf.Lerp(m\_EngineMinThrottlePitch, m\_EngineMaxThrottlePitch, enginePowerProportion);

// Increase the engine's pitch by an amount proportional to the aeroplane's forward speed.

// (this makes the pitch increase when going into a dive!)

m\_EngineSoundSource.pitch += m\_Plane.ForwardSpeed\*m\_EngineFwdSpeedMultiplier;

// Set the engine's volume to be proportional to the engine's current power.

m\_EngineSoundSource.volume = Mathf.InverseLerp(0, m\_Plane.MaxEnginePower\*m\_AdvancedSetttings.engineMasterVolume,

m\_Plane.EnginePower);

// Set the wind's pitch and volume to be proportional to the aeroplane's forward speed.

float planeSpeed = m\_Rigidbody.velocity.magnitude;

m\_WindSoundSource.pitch = m\_WindBasePitch + planeSpeed\*m\_WindSpeedPitchFactor;

m\_WindSoundSource.volume = Mathf.InverseLerp(0, m\_WindMaxSpeedVolume, planeSpeed)\*m\_AdvancedSetttings.windMasterVolume;

}

}

}