```
1  using System;
2  using UnityEngine;
3  using Random = UnityEngine.Random;
4
5  namespace UnityStandardAssets.Vehicles.Car
6  {
7      [RequireComponent(typeof (CarController))]
8      public class CarAudio : MonoBehaviour
9      {
10         // This script reads some of the car's current properties and plays sounds accordingly.
11         // The engine sound can be a simple single clip which is looped and pitched, or it
12         // can be a crossfaded blend of four clips which represent the timbre of the engine
13         // at different RPM and Throttle state.
14
15         // the engine clips should all be a steady pitch, not rising or falling.
16
17         // when using four channel engine crossfading, the four clips should be:
18         // lowAccelClip : The engine at low revs, with throttle open (i.e. begining acceleration at very low speed)
19         // highAccelClip : Thenengine at high revs, with throttle open (i.e. accelerating, but almost at max speed)
20         // lowDecelClip : The engine at low revs, with throttle at minimum (i.e. idling or engine-braking at very low speed)
21         // highDecelClip : Thenengine at high revs, with throttle at minimum (i.e. engine-braking at very high speed)
22
23         // For proper crossfading, the clips pitches should all match, with an octave offset between low and high.
24
25
26         public enum EngineAudioOptions // Options for the engine audio
27         {
28             Simple, // Simple style audio
29             FourChannel // four Channel audio
30         }
31
32         public EngineAudioOptions engineSoundStyle = EngineAudioOptions.FourChannel;// Set the default audio options to be
             four channel
33         public AudioClip lowAccelClip;                                          // Audio clip for low acceleration
34         public AudioClip lowDecelClip;                                          // Audio clip for low deceleration
```

```
35          public AudioClip highAccelClip;                                     // Audio clip for high acceleration
36          public AudioClip highDecelClip;                                     // Audio clip for high deceleration
37          public float pitchMultiplier = 1f;                                  // Used for altering the pitch of audio
              clips
38          public float lowPitchMin = 1f;                                      // The lowest possible pitch for the low
              sounds
39          public float lowPitchMax = 6f;                                      // The highest possible pitch for the low
               sounds
40          public float highPitchMultiplier = 0.25f;                           // Used for altering the pitch of high
              sounds
41          public float maxRolloffDistance = 500;                              // The maximum distance where rollof
              starts to take place
42          public float dopplerLevel = 1;                                      // The mount of doppler effect used in
              the audio
43          public bool useDoppler = true;                                      // Toggle for using doppler
44
45          private AudioSource m_LowAccel; // Source for the low acceleration sounds
46          private AudioSource m_LowDecel; // Source for the low deceleration sounds
47          private AudioSource m_HighAccel; // Source for the high acceleration sounds
48          private AudioSource m_HighDecel; // Source for the high deceleration sounds
49          private bool m_StartedSound; // flag for knowing if we have started sounds
50          private CarController m_CarController; // Reference to car we are controlling
51
52
53          private void StartSound()
54          {
55              // get the carcontroller ( this will not be null as we have require component)
56              m_CarController = GetComponent<CarController>();
57
58              // setup the simple audio source
59              m_HighAccel = SetUpEngineAudioSource(highAccelClip);
60
61              // if we have four channel audio setup the four audio sources
62              if (engineSoundStyle == EngineAudioOptions.FourChannel)
63              {
```

```
64                  m_LowAccel = SetUpEngineAudioSource(lowAccelClip);
65                  m_LowDecel = SetUpEngineAudioSource(lowDecelClip);
66                  m_HighDecel = SetUpEngineAudioSource(highDecelClip);
67              }
68
69              // flag that we have started the sounds playing
70              m_StartedSound = true;
71          }
72
73
74          private void StopSound()
75          {
76              //Destroy all audio sources on this object:
77              foreach (var source in GetComponents<AudioSource>())
78              {
79                  Destroy(source);
80              }
81
82              m_StartedSound = false;
83          }
84
85
86          // Update is called once per frame
87          private void Update()
88          {
89              // get the distance to main camera
90              float camDist = (Camera.main.transform.position - transform.position).sqrMagnitude;
91
92              // stop sound if the object is beyond the maximum roll off distance
93              if (m_StartedSound && camDist > maxRolloffDistance*maxRolloffDistance)
94              {
95                  //StopSound();
96                  if (camDist > maxRolloffDistance * maxRolloffDistance)
97                  {
98                      m_HighAccel.volume = m_HighAccel.volume -= 0.3f * Time.deltaTime;
```

```
 99                      if (m_HighAccel.volume <= 0)
100                      {
101                          m_HighAccel.volume = 0;
102                      }
103                  }
104              }
105
106              // start the sound if not playing and it is nearer than the maximum distance
107              if (!m_StartedSound && camDist < maxRolloffDistance*maxRolloffDistance)
108              {
109                  StartSound();
110                  if(camDist < maxRolloffDistance * maxRolloffDistance)
111                  {
112                      m_HighAccel.volume = m_HighAccel.volume += 0.3f * Time.deltaTime;
113                      if(m_HighAccel.volume >= 1)
114                      {
115                          m_HighAccel.volume = 1;
116                      }
117                  }
118              }
119
120              if (m_StartedSound)
121              {
122                  // The pitch is interpolated between the min and max values, according to the car's revs.
123                  float pitch = ULerp(lowPitchMin, lowPitchMax, m_CarController.Revs);
124
125                  // clamp to minimum pitch (note, not clamped to max for high revs while burning out)
126                  pitch = Mathf.Min(lowPitchMax, pitch);
127
128                  if (engineSoundStyle == EngineAudioOptions.Simple)
129                  {
130                      // for 1 channel engine sound, it's oh so simple:
131                      m_HighAccel.pitch = pitch*pitchMultiplier*highPitchMultiplier;
132                      m_HighAccel.dopplerLevel = useDoppler ? dopplerLevel : 0;
133                      if (camDist < maxRolloffDistance * maxRolloffDistance)
```

```
134                     {
135                         m_HighAccel.volume = m_HighAccel.volume += 0.3f * Time.deltaTime;
136                         if (m_HighAccel.volume >= 1)
137                         {
138                             m_HighAccel.volume = 1;
139                         }
140                     }
141                     //m_HighAccel.volume = 1;
142                 }
143                 else
144                 {
145                     // for 4 channel engine sound, it's a little more complex:
146 
147                     // adjust the pitches based on the multipliers
148                     m_LowAccel.pitch = pitch*pitchMultiplier;
149                     m_LowDecel.pitch = pitch*pitchMultiplier;
150                     m_HighAccel.pitch = pitch*highPitchMultiplier*pitchMultiplier;
151                     m_HighDecel.pitch = pitch*highPitchMultiplier*pitchMultiplier;
152 
153                     // get values for fading the sounds based on the acceleration
154                     float accFade = Mathf.Abs(m_CarController.AccelInput);
155                     float decFade = 1 - accFade;
156 
157                     // get the high fade value based on the cars revs
158                     float highFade = Mathf.InverseLerp(0.2f, 0.8f, m_CarController.Revs);
159                     float lowFade = 1 - highFade;
160 
161                     // adjust the values to be more realistic
162                     highFade = 1 - ((1 - highFade)*(1 - highFade));
163                     lowFade = 1 - ((1 - lowFade)*(1 - lowFade));
164                     accFade = 1 - ((1 - accFade)*(1 - accFade));
165                     decFade = 1 - ((1 - decFade)*(1 - decFade));
166 
167                     // adjust the source volumes based on the fade values
168                     m_LowAccel.volume = lowFade*accFade;
```

```
169                     m_LowDecel.volume = lowFade*decFade;
170                     m_HighAccel.volume = highFade*accFade;
171                     m_HighDecel.volume = highFade*decFade;
172 
173                     // adjust the doppler levels
174                     m_HighAccel.dopplerLevel = useDoppler ? dopplerLevel : 0;
175                     m_LowAccel.dopplerLevel = useDoppler ? dopplerLevel : 0;
176                     m_HighDecel.dopplerLevel = useDoppler ? dopplerLevel : 0;
177                     m_LowDecel.dopplerLevel = useDoppler ? dopplerLevel : 0;
178                 }
179 
180                 if(SaveScript.RaceOver == true)
181                 {
182                     m_HighAccel.volume = m_HighAccel.volume -= 2.5f * Time.deltaTime;
183                     if (m_HighAccel.volume <= 0)
184                     {
185                         m_HighAccel.volume = 0;
186                     }
187                 }
188             }
189         }
190 
191 
192         // sets up and adds new audio source to the gane object
193         private AudioSource SetUpEngineAudioSource(AudioClip clip)
194         {
195             // create the new audio source component on the game object and set up its properties
196             AudioSource source = gameObject.AddComponent<AudioSource>();
197             source.clip = clip;
198             source.volume = 0;
199             source.loop = true;
200 
201             // start the clip from a random point
202             source.time = Random.Range(0f, clip.length);
203             source.Play();
```

```
204             source.minDistance = 5;
205             source.maxDistance = maxRolloffDistance;
206             source.dopplerLevel = 0;
207             return source;
208         }
209
210
211         // unclamped versions of Lerp and Inverse Lerp, to allow value to exceed the from-to range
212         private static float ULerp(float from, float to, float value)
213         {
214             return (1.0f - value)*from + value*to;
215         }
216     }
217 }
218
```