ANEXOS

ENLACES A LOS ANEXOS 1 Y 2.

ANEXO 1:

Eisenbahnbrücke's Nightmare **(exportado).**

https://mega.nz/#!wdMBBRCb!88UJYUF_1uMuARZ1R7x3OgPqT4RgT0cp0fZvjs1_K-Q

ANEXO 2:

Eisenbahnbrücke's Nightmare **(proyecto en Unity, sin exportar).**

https://mega.nz/#!kB8VBS4Q!Bj7vRTQbbPFfT7S8rX7hp9RinykwsnHAqhtjMnBY92g

ANEXO 3:

Código de programación del videojuego en JavasCript.

Código para pasar de la pantalla inicial mediante el sonido.

```
1.  private var go : GameObject;
2.  private var miControl: MicControl;
3.  private var umbral : float;
4.  private var umbralMax : float = 6.000;
5.  //private var minSens = 0.001;
6.  //private var started : int = 0;
7.
8.  function Start () {
9.      go = GameObject.Find( "FirstScreen" );
10.     miControl = go.GetComponent(MicControl);
11.     miControl.OnApplicationFocus(true);
12.     umbral = miControl.loudness + 0.80;
13.
14.     if ( umbral > umbralMax ) {
15.         umbral = umbralMax;
16.     }
17.
18. }
19.
20. function Update () {
21.
22. //Debug.Log("noiseFirst" + miControl.loudness);
23. Debug.Log("UmbralFirst" + umbral);
24.
25. if ( miControl.loudness > umbral ) {
26.
27.     Application.LoadLevel("Instructions");
28.
29. }
30. }
```

Código para pasar de las instrucciones mediante el sonido.

```
1.  private var go : GameObject;
2.  private var miControl: MicControl;
3.  private var umbral2 : float;
4.  private var umbral2Max : float = 3.500;
5.
6.  function Start () {
7.      go = GameObject.Find( "Instructions" );
8.      miControl = go.GetComponent(MicControl);
9.      miControl.OnApplicationFocus(true);
10.     umbral2 = miControl.loudness + 0.95;
11.
12.     if ( umbral2 > umbral2Max ) {
13.         umbral2 = umbral2Max;
14.     }
15.
16. }
17.
18. function Update () {
19.
20. //Debug.Log("YYY" + miControl.audioSource);
21. Debug.Log("noiseInstructions" + miControl.loudness);
```

```
22. Debug.Log("UmbralInstructions" + umbral2);
23.
24. if ( miControl.loudness > umbral2 ) {
25.
26.     Application.LoadLevel("Level");
27.
28. }
29.
30. }
```

Código de control del personaje.

```
1.  private var animator : Animator;
2.  private var go : GameObject;
3.  private var micControl: MicControl;
4.  private var umbral3 : float;
5.  private var umbral3Max : float = 6.000;
6.
7.  function Start () {
8.
9.      go = GameObject.Find( "Character" );
10.     animator = go.GetComponent( Animator );
11.     micControl = GetComponent(MicControl);
12.     micControl.OnApplicationFocus(true);
13.     umbral3 = micControl.loudness + 0.005;
14.     //GameObject.Find("MicControl").SendMessage("StartMicrophone");
15.
16.     if ( umbral3 > umbral3Max ) {
17.         umbral3 = umbral3Max;
18.     }
19.
20.}
21.
```

```
22. function Update () {
23. //scales the gameObject heigt based on input stream gathered from
    MicControl.loudness
24.
25. //transform.localScale=Vector2(1+MicControl.loudness,1+MicControl.loudnes
    s);
26.     //Debug.Log("YYY" + micControl.audioSource);
27. Debug.Log("noiseGame" + micControl.loudness);
28. Debug.Log("UmbralGame" + umbral3);
29.
30.     if ( micControl.loudness > umbral3 ) {
31.
32.         animator.SetTrigger( "Parar" );
33.
34.
35.     }else{
36.
37.         animator.SetTrigger( "Andar" );
38.
39.
40.     }
41.     Debug.Log("Eisenbahnbrucke's nightmare");
```

```
42.     go.GetComponent.<Rigidbody2D>().AddForce( new Vector2( 1 + ( - 1 *
        micControl.loudness * 2 ), 0 ), ForceMode2D.Force);          // Base velocity,
        sensibility.
43.
44. }
```

Código para indicar cuándo el personaje llega al puente.

```
1.  function Start () {
2.
3.  }
4.
5.  function Update () {
6.
7.  }
8.
9.  function OnTriggerEnter2D( c : Collider2D ) {
10.
11.     if( c.transform.name == "Character" ) {
12.         Debug.Log( "Finish!" );
13.
14.         // Destroy the object
15.         Destroy(gameObject);
16.
17.         //Load level
18.         //Application.LoadLevel(1);
19.     }
20.
21.}
```

Código para pasar al vídeo de Game Over.

```
1.  function Start () {
2.
3.  }
4.
5.  function Update () {
6.
7.  }
8.
9.  function OnTriggerEnter2D (other : Collider2D) {
10.
11.         Application.LoadLevel ("GameOver");
12.     }
```

Código del temporizador y para pasar al vídeo del Win (si se salva el puente).

```
1.  var maxTime : int = 60;
2.  var timer : int = 0;
3.
4.  function Start () {
5.
6.      timer = maxTime;
7.      InvokeRepeating( "Decrease", 1.0, 1.0 );
8.
9.  }
10.
11. function Update () {
12.
13. }
14.
15. function Decrease () {
16.
17.     timer = timer -1;
18.     gameObject.GetComponent( UI.Text ).text = timer.ToString();
19.
20.     if( timer == 0 ) {
```

```
21.         Application.LoadLevel("GameWin");               // Load the level GameWin.
22.     }
23.
24. }
```

Código para que la cámara siga al personaje.

```
1.  var target : Transform;
2.  private var velocity = Vector3.zero;
3.
4.  function Update () {
5.
6.      if ( target ) {
7.
8.          var point =
    GetComponent.<Camera>().WorldToViewportPoint(target.position);
9.          var delta = target.position -
    GetComponent.<Camera>().ViewportToWorldPoint(new Vector3(0.5f, 0.2f,
    point.z));
10.         var destination = transform.position + delta;
11.         transform.position = Vector3.SmoothDamp(transform.position,
    destination, velocity, 0.15f);
12.
13.     }
14.
15. }
```

Código para los vídeos y para volver a la pantalla inicial tras estos.

```
1.  using UnityEngine;
2.  using System.Collections;
3.
4.  [RequireComponent (typeof(AudioSource))]
5.
6.  public class VideoController : MonoBehaviour {
7.      MovieTexture movie;
8.      // Use this for initialization
9.      void Start () {
10.
11.         movie = GetComponent<Renderer>().material.mainTexture as
    MovieTexture;
12.         GetComponent<AudioSource>().clip = movie.audioClip;
13.
14.         GetComponent<AudioSource>().Play();
15.         movie.Play();
16.
17.     }
18.
19.     // Update is called once per frame
20.     void Update () {
```

```
21.         if (!movie.isPlaying)
22.             Application.LoadLevel("FirstScreen");
23.     }
24. }
```

Código del MicControl (para la interfaz sonora).

```
1.  @script RequireComponent (AudioSource);
2.  //if true a menu will apear ingame with all the microphones
3.  @HideInInspector
4.  var SelectIngame:boolean=false;
5.  //if false the below will override and set the mic selected in the editor
6.   //Select the microphone you want to use (supported up to 6 to choose from).
    If the device has number 1 in the console, you should select default as it is
    the first defice to be found.
7.  enum Devices {DefaultDevice, Second, Third, Fourth, Fifth, Sixth}
8.
9.  @HideInInspector
10. var ThreeD:boolean=false;
11. @HideInInspector
12. var VolumeFallOff:float=1;
13. @HideInInspector
14. var PanThreshold:float=1;
15.
16. var InputDevice : Devices;
17. private var selectedDevice:String;
18. var audioListener:Transform;
19. var audioSource:AudioSource;
```

```
20. //The maximum amount of sample data that gets loaded in, best is to leave it
    on 256, unless you know what you are doing. A higher number gives more
    accuracy but
21. //lowers performance allot, it is best to leave it at 256.
22. var amountSamples:float=256;
23. static var loudness:float;
24. var sensitivity:float=0.4;
25. var sourceVolume:float=100;
26. private var minFreq: int;
27.  var maxFreq: int=44100;
28.
29. @HideInInspector
30. var Mute:boolean=true;
31. @HideInInspector
32. var debug:boolean=false;
33. @HideInInspector
34. var ShowDeviceName:boolean=false;
35. private var micSelected:boolean=false;
36.
37. private var recording:boolean=true;
38.
39. private var ListenerDistance:float;
```

```
40. private var ListenerPosition:Vector3;
41.
42. private var focused:boolean=false;
43. private var Initialised:boolean=false;
44.
45. function Start () {
46.     // Request permission to use both webcam and microphone.
47.     if(Application.isWebPlayer){
48.         yield Application.RequestUserAuthorization
      (UserAuthorization.Microphone);
49.         if (Application.HasUserAuthorization(UserAuthorization.Microphone)){
50.             InitMic();
51. Initialised=true;
52.         }
53.         else{
54.             return;
55.         }
56.   }
57.   else{
58.   InitMic();
59.   Initialised=true;
60.   }
```

```
61.
62.}
63.
64.
65.//apply the mic input data stream to a float;
66.function Update () {
67.//pause everything when not focused on the app and then re-initialize.
68.if (!focused){
69.            StopMicrophone();
70.            Initialised=false;
71.        }
72.        if (!Application.isPlaying) {
73.        //don't stop the microphone if you are clicking inside the editor
74.            StopMicrophone();
75.            Initialised=false;
76.        }
77.        else{
78.        if(!Initialised){
79.        InitMic();
80.        Initialised=true;
81.        }
82.        }
```

```
83.
84.
85. if(Microphone.IsRecording(selectedDevice)){
86.   loudness = GetDataStream()*sensitivity*(sourceVolume/10);
87.
88.  }
89.   if(debug){
90.   Debug.Log(loudness);
91.  }
92.
93.  //the source volume
94.  if (sourceVolume > 100){
95.       sourceVolume = 100;
96. }
97.
98.  if (sourceVolume < 0){
99.   sourceVolume = 0;
100.              }
101.
102.              //when 3D is enabled adjust the volume based on distance.
103.              if(ThreeD){
104.
```

```
105.                ListenerDistance =
    Vector3.Distance(transform.position,audioListener.position);
106.                ListenerPosition =
    audioListener.InverseTransformPoint(transform.position);
107.
108.              GetComponent.<AudioSource>().volume =
    (sourceVolume/100/(ListenerDistance*VolumeFallOff));
109.              GetComponent.<AudioSource>().panStereo=
    (ListenerPosition.x/PanThreshold);
110.
111.               }
112.               else{
113.              GetComponent.<AudioSource>().volume = (sourceVolume/100);
114.             }
115.
116.
117.            }
118.
119.
120.            function GetDataStream(){
121.            if(Microphone.IsRecording(selectedDevice)){
122.
```

```
123.                var dataStream: float[]  = new float[amountSamples];
124.                    var audioValue: float = 0;
125.
    GetComponent.<AudioSource>().GetOutputData(dataStream,0);
126.
127.                    for(var i in dataStream){
128.                        audioValue += Mathf.Abs(i);
129.                    }
130.                    return audioValue/amountSamples;
131.                    }
132.
133.
134.
135.            }
136.
137.
138.
139.
140.
141.            //select device ingame
142.
143.                function OnGUI () {
```

```
144.            if(SelectIngame==true){
145.                if (Microphone.devices.Length > 0 && micSelected == false)//If
    there is more than one device, choose one.
146.                    for (var i:int= 0; i < Microphone.devices.Length; ++i)
147.                        if (GUI.Button(new Rect(400, 100 + (110 * i), 300, 100),
    Microphone.devices[i].ToString())) {
148.                            StopMicrophone();
149.                            selectedDevice = Microphone.devices[i].ToString();
150.                            GetMicCaps();
151.                           StartMicrophone();
152.                            micSelected = true;
153.
154.                        }
155.
156.                if (Microphone.devices.Length < 1 && micSelected == false)
    {//If there is only 1 decive make it default
157.                    selectedDevice = Microphone.devices[0].ToString();
158.                   GetMicCaps();
159.                    micSelected = true;
160.
161.                }
162.            }
```

```
163.
164.                 }
165.
166.             ////////////////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////
167.             //Initialize microphone
168.             private  function InitMic(){
169.             //select audio source
170.             if(!audioSource){
171.               audioSource = transform.GetComponent(AudioSource);
172.                 }
173.
174.              //only Initialize microphone if a device is detected
175.             if(Microphone.devices.Length>=0){
176.
177.             var i=0;
178.             //count amount of devices connected
179.             for(device in Microphone.devices){
180.             i++;
181.             if(ShowDeviceName){
182.             Debug.Log ("Devices number "+i+" Name"+"="+device);
183.
```

```
184.            }
185.            }
186.
187.
188.            if(SelectIngame==false){
189.            //select the device if possible else give error
190.            if(InputDevice==Devices.DefaultDevice){
191.            if(i>=1){
192.            selectedDevice= Microphone.devices[0];
193.            }
194.            else{
195.            Debug.LogError ("No device detected on this slot. Check input
    connection");
196.            }
197.
198.            }
199.
200.
201.            if(InputDevice==Devices.Second){
202.            if(i>=2){
203.            selectedDevice= Microphone.devices[1];
204.            }
```

```
205.            else{
206.            Debug.LogError ("No device detected on this slot. Check input
    connection");
207.            }
208.
209.            }
210.
211.
212.
213.            if(InputDevice==Devices.Third){
214.            if(i>=3){
215.            selectedDevice= Microphone.devices[2];
216.            }
217.            else{
218.            Debug.LogError ("No device detected on this slot. Check input
    connection");
219.            return;
220.            }
221.            }
222.
223.
224.            if(InputDevice==Devices.Fourth){
```

```
225.            if(i>=4){
226.            selectedDevice= Microphone.devices[2];
227.            }
228.            else{
229.            Debug.LogError ("No device detected on this slot. Check input
    connection");
230.            return;
231.            }
232.            }
233.            if(InputDevice==Devices.Fifth){
234.            if(i>=5){
235.            selectedDevice= Microphone.devices[2];
236.            }
237.            else{
238.            Debug.LogError ("No device detected on this slot. Check input
    connection");
239.            return;
240.            }
241.            }
242.
243.            if(InputDevice==Devices.Sixth){
244.            if(i>=6){
```

```
245.            selectedDevice= Microphone.devices[2];
246.            }
247.            else{
248.            Debug.LogError ("No device detected on this slot. Check input
    connection");
249.            return;
250.            }
251.            }
252.
253.            }
254.
255.            //detect the selected microphone
256.            GetComponent.<AudioSource>().clip =
    Microphone.Start(selectedDevice, true, 10, maxFreq);
257.
258.
259.            //loop the playing of the recording so it will be realtime
260.            GetComponent.<AudioSource>().loop = true;
261.            //if you only need the data stream values  check Mute, if you want
    to hear yourself ingame don't check Mute.
262.            GetComponent.<AudioSource>().mute = Mute;
263.
```

```
264.            //don't do anything until the microphone started up
265.            while (!(Microphone.GetPosition(selectedDevice) > 0)){
266.            if(debug){
267.            Debug.Log("Awaiting connection");
268.            }
269.            }
270.            if(debug){
271.            Debug.Log("Connected");
272.            }
273.
274.            //Put the clip on play so the data stream gets ingame on realtime
275.            GetComponent.<AudioSource>().Play();
276.            recording=true;
277.             }
278.
279.             }
280.            ////////////////////////////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////////////
281.
282.
283.
284.                //for the above control the mic start or stop
```

```
285.
286.
287.            public function StartMicrophone () {
288.                GetComponent.<AudioSource>().clip =
    Microphone.Start(selectedDevice, true, 10, maxFreq);//Starts recording
289.                while (!(Microphone.GetPosition(selectedDevice) > 0)){} //
    Wait until the recording has started
290.                GetComponent.<AudioSource>().Play(); // Play the audio
    source!
291.
292.            }
293.
294.
295.            public function StopMicrophone () {
296.                GetComponent.<AudioSource>().Stop();//Stops the audio
297.                Microphone.End(selectedDevice);//Stops the recording of the
    device
298.
299.            }
300.
301.              function GetMicCaps () {
```

```
302.                    Microphone.GetDeviceCaps(selectedDevice,  minFreq,
     maxFreq);//Gets the frequency of the device
303.                    if ((minFreq + maxFreq) == 0)//These 2 lines of code are
    mainly for windows computers
304.                        maxFreq = 44100;
305.
306.                }
307.
308.
309.
310.
311.
312.                //Create a gui button in another script that calls to this script
313.                    public function MicDeviceGUI (left:float , top:float, width:float,
    height:float, buttonSpaceTop:float, buttonSpaceLeft:float) {
314.                if (Microphone.devices.Length > 1 && micSelected == false)//If
    there is more than one device, choose one.
315.                    for (var i:int=0; i < Microphone.devices.Length; ++i)
316.                        if (GUI.Button(new Rect(left + (buttonSpaceLeft * i), top +
    (buttonSpaceTop * i), width, height), Microphone.devices[i].ToString())) {
317.                            StopMicrophone();
318.                            selectedDevice = Microphone.devices[i].ToString();
```

```
319.                         GetMicCaps();
320.                         StartMicrophone();
321.                         micSelected = true;
322.                     }
323.                 if (Microphone.devices.Length < 2 && micSelected == false) {//If
    there is only 1 decive make it default
324.                     selectedDevice = Microphone.devices[0].ToString();
325.                     GetMicCaps();
326.                     micSelected = true;
327.                 }
328.                 }
329.
330.
331.
332.
333.
334.
335.
336.
337.
338.
339.
```

```
340.
341.            //flush the date through a custom created audio clip, this controls
    the data flow of that clip
342.            // Creates a 1 sec long audioclip, with a 440hz sinoid
343.            private var position: int = 0;
344.            private var sampleRate : int = 0;
345.            private var frequency : float = 440;
346.            function OnAudioRead(data:float[]){
347.                for (var count = 0; count < data.Length; count++)
348.                {
349.                    data[count] = Mathf.Sign(Mathf.Sin(2 * Mathf.PI * frequency
    * position / sampleRate));
350.                    position++;
351.                }
352.            }
353.
354.            function OnAudioSetPosition(newPosition:int){
355.                position = newPosition;
356.            }
357.
358.            //start or stop the script from running when the state is paused or
    not.
```

```
359.        function OnApplicationFocus(focus: boolean) {
360.            focused = focus;
361.        }
362.
363.        function OnApplicationPause(focus: boolean) {
364.            focused = focus;
365.        }
```

Código para la interfaz sonora.

```
1.  using UnityEngine;
2.  using System.Collections;
3.
4.  [RequireComponent(typeof(AudioSource))]
5.  public class MicControlC : MonoBehaviour {
6.      public enum micActivation {
7.          HoldToSpeak,
8.          PushToSpeak,
9.          ConstantSpeak
10.     }
11.
12.     [HideInInspector]
13.     public bool virual3D = false;
14.     [HideInInspector]
15.     public float volumeFallOff = 0.3f;
16.     [HideInInspector]
17.     public float listenerDistance { get; private set; }
18.     [HideInInspector]
19.     public bool ableToHearMic = false;
20.     public float sensitivity = 100;
21.     [Range(0,100)]
```

```
22.     public float sourceVolume = 100;//Between 0 and 100
23.     [HideInInspector]
24.     public bool GuiSelectDevice = true;
25.     public micActivation micControl;
26.     //
27.     public string selectedDevice { get; private set; }
28.     public float loudness { get; private set; } //dont touch
29.     //
30.     private bool micSelected = false;
31.     private int amountSamples = 256; //increase to get better average, but will
    decrease performance. Best to leave it
32.     private int minFreq, maxFreq;
33.
34.     private bool focused = true;
35.
36.     void Start() {
37.         GetComponent<AudioSource>().loop = true; // Set the AudioClip to loop
38.         GetComponent<AudioSource>().mute = false; // Mute the sound, we
    don't want the player to hear it
39.         selectedDevice = Microphone.devices[0].ToString();
40.         micSelected = true;
41.         GetMicCaps();
```

```
42.
43.         if (Application.isWebPlayer) {
44.
    Application.RequestUserAuthorization(UserAuthorization.Microphone);
45.             if (Application.HasUserAuthorization(UserAuthorization.Microphone)) {
46.                 selectedDevice = Microphone.devices[0].ToString();
47.                 GetMicCaps();
48.                 StartMicrophone();
49.                 micSelected = true;
50.             }
51.             else return;
52.         }
53.     }
54.     void OnGUI() {
55.         MicDeviceGUI((Screen.width/2)-150, (Screen.height/2)-75, 300, 100, 10,
    -300);
56.     }
57.     public void MicDeviceGUI (float left, float top, float width, float height, float
    buttonSpaceTop, float buttonSpaceLeft) {
58.         if (Microphone.devices.Length > 1 && GuiSelectDevice == true ||
    micSelected == false)//If there is more than one device, choose one.
59.             for (int i = 0; i < Microphone.devices.Length; ++i)
```

```
60.             if (GUI.Button(new Rect(left + ((width + buttonSpaceLeft) * i), top +
    ((height + buttonSpaceTop) * i), width, height),
    Microphone.devices[i].ToString())) {
61.                 StopMicrophone();
62.                 selectedDevice = Microphone.devices[i].ToString();
63.                 GetMicCaps();
64.                 StartMicrophone();
65.                 micSelected = true;
66.             }
67.         if (Microphone.devices.Length < 2 && micSelected == false) {//If there is
    only 1 decive make it default
68.             selectedDevice = Microphone.devices[0].ToString();
69.             GetMicCaps();
70.             micSelected = true;
71.         }
72.     }
73.     public void GetMicCaps () {
74.         Microphone.GetDeviceCaps(selectedDevice, out minFreq, out
    maxFreq);//Gets the frequency of the device
75.         if ((minFreq + maxFreq) == 0)//These 2 lines of code are mainly for
    windows computers
76.             maxFreq = 44100;
```

```
77.    }
78.    public void StartMicrophone () {
79.        GetComponent<AudioSource>().clip =
    Microphone.Start(selectedDevice, true, 10, maxFreq);//Starts recording
80.        while (!(Microphone.GetPosition(selectedDevice) > 0)){} // Wait until the
    recording has started
81.        GetComponent<AudioSource>().Play(); // Play the audio source!
82.    }
83.    public void StopMicrophone () {
84.        GetComponent<AudioSource>().Stop();//Stops the audio
85.        Microphone.End(selectedDevice);//Stops the recording of the device
86.    }
87.    void Update() {
88.        if (!focused)
89.            StopMicrophone();
90.
91.        if (!Application.isPlaying) {
92.            StopMicrophone();
93.        }
94.
95.        if(virual3D){
```

```
96.             listenerDistance = Vector3.Distance(transform.position,
    GetComponent<AudioSource>().transform.position);
97.             GetComponent<AudioSource>().volume = (sourceVolume / 100 /
    (listenerDistance * volumeFallOff));
98.         }
99.         else {
100.                        GetComponent<AudioSource>().volume = (sourceVolume /
    100);
101.                        loudness = GetAveragedVolume() * sensitivity *
    (sourceVolume / 10);
102.                    }
103.                    //Hold To Speak!!
104.                    if (micControl == micActivation.HoldToSpeak) {
105.                        if (Microphone.IsRecording(selectedDevice) &&
    Input.GetKey(KeyCode.T) == false)
106.                            StopMicrophone();
107.                        //
108.                        if (Input.GetKeyDown(KeyCode.T)) //Push to talk
109.                            StartMicrophone();
110.                        //
111.                        if (Input.GetKeyUp(KeyCode.T))
112.                            StopMicrophone();
```

```
113.                 //
114.             }
115.
116.             //Push To Talk!!
117.             if (micControl == micActivation.PushToSpeak) {
118.                 if (Input.GetKeyDown(KeyCode.T)) {
119.                     if (Microphone.IsRecording(selectedDevice))
120.                         StopMicrophone();
121.
122.                     else if (!Microphone.IsRecording(selectedDevice))
123.                         StartMicrophone();
124.                 }
125.                 //
126.             }
127.
128.             //Constant Speak!!
129.             if (micControl == micActivation.ConstantSpeak)
130.                 if (!Microphone.IsRecording(selectedDevice))
131.                     StartMicrophone();
132.
133.             //Mic Slected = False!!
134.             if (Input.GetKeyDown(KeyCode.G))
```

```
135.                     micSelected = false;
136.                 }
137.
138.                 float GetAveragedVolume() {
139.                     float[] data = new float[amountSamples];
140.                     float a = 0;
141.                     GetComponent<AudioSource>().GetOutputData(data,0);
142.                     foreach(float s in data) {
143.                         a += Mathf.Abs(s);
144.                     }
145.                     return a/amountSamples;
146.                 }
147.
148.                 void OnApplicationFocus(bool focus) {
149.                     focused = focus;
150.                 }
151.
152.                 void OnApplicationPause(bool focus) {
153.                     focused = focus;
154.                 }
155.             }
```

Código para controlar el sonido que entra por el micro, la sensibilidad y otras opciones.

```
1.  using UnityEditor;
2.  using UnityEngine;
3.  using System.Collections;
4.
5.  [CustomEditor(typeof(MicControlC))]
6.  public class VolumeBarC : Editor {
7.
8.      private bool microphoneDeviceFound = false;
9.
10.     public override void OnInspectorGUI () {
11.         if (!microphoneDeviceFound) {
12.             EditorGUILayout.HelpBox("No Microphone Device Was Found
    Connected To Your Computer.", MessageType.Warning);
13.
14.             if (Microphone.devices.Length >= 1)
15.                 microphoneDeviceFound = true;
16.         }
17.
18.         MicControlC micCon  = (MicControlC) target;
19.
```

```
20.         if (microphoneDeviceFound) {
21.             float micInputValue = micCon.loudness;
22.             VolumeReader (micInputValue / 100, "Loudness " + micInputValue);
23.
24.             micCon.virual3D = EditorGUILayout.Toggle (new GUIContent("Enable
    Virtual 3D", "Uses a falloff system to simulate virtual 3D"), micCon.virual3D);
25.             micCon.volumeFallOff = EditorGUILayout.FloatField (new
    GUIContent("Volume Falloff", "Set the rate at wich audio volume gets
    lowered. A lower value will have a slower falloff and thus hearable from a
    greater distance, while a higher value will make the audio degrate faster and
    dissapear from a shorter distance"), micCon.volumeFallOff);
26.             micCon.GuiSelectDevice = EditorGUILayout.Toggle (new
    GUIContent("Gui Selection", "Select the microphone ingame using a GUI
    menu"), micCon.GuiSelectDevice);
27.             EditorGUI.BeginChangeCheck();
28.             micCon.ableToHearMic = EditorGUILayout.Toggle (new
    GUIContent("Audio Mute", "Select whether you can hear yourself talking or
    not"), micCon.ableToHearMic);
29.             if (EditorGUI.EndChangeCheck()) {
30.                 micCon.GetComponent<AudioSource>().mute =
    micCon.ableToHearMic;
31.             }
```

```
32.
33.             micCon.sensitivity = EditorGUILayout.FloatField(new
    GUIContent("Mic Sensitivity", "The sensitivity that the audio is recieved from
    the microphone"), micCon.sensitivity);
34.             //micCon.ramFlushSpeed = EditorGUILayout.FloatField(new
    GUIContent("Ram Flush Speed", "The interval time between when the
    microphone audioclip is reset"), micCon.ramFlushSpeed);
35.             micCon.sourceVolume = EditorGUILayout.FloatField(new
    GUIContent("Volume", "Volume of the audio that comes out of audiosouce,
    basically the volume of the microphone"), micCon.sourceVolume);
36.             micCon.micControl =
    (MicControlC.micActivation)EditorGUILayout.EnumPopup(new
    GUIContent("Mic Control Type", "Changes the type of method for using the
    microphone"), micCon.micControl);
37.
38.             EditorUtility.SetDirty(target);
39.         }
40.     }
41.
42.     void VolumeReader (float value, string label) {
43.         EditorGUILayout.Space ();
44.         Rect vRect = GUILayoutUtility.GetRect (18, 18, "TextField");
```

```
45.         EditorGUI.ProgressBar (vRect, value, label);
46.         EditorGUILayout.Space ();
47.     }
48.}
```

Código para más configuraciones de sonido.

```
1.  @CustomEditor (MicControl)
2.  class VolumeBarJ extends Editor  {
3.
4.  var ListenToMic = Selection.activeGameObject;
5.
6.
7.  ////////////////////////////////////////////////////////////////////////////////////////////
8.          function OnInspectorGUI() {
9.  var micInputValue=MicControl.loudness;
10. ProgressBar (micInputValue, "Loudness");
11.
12.
13. //show other variables
14.
15. //Redirect 3D toggle
16. ListenToMic.GetComponent(MicControl).ThreeD=GUILayout.Toggle(ListenTo
    Mic.GetComponent(MicControl).ThreeD,GUIContent ("3D sound","Should the
    streamed audio be a 3D sound? (Only enable this if you are using the
    controller to stream sound (VOIP) "));
17. //when 3D audio is enabled show the fall off settings
18. if(ListenToMic.GetComponent(MicControl).ThreeD){
```

19. ListenToMic.GetComponent(MicControl).VolumeFallOff = EditorGUILayout.FloatField(GUIContent ("Volume falloff","Set the rate at wich audio volume gets lowered. A lower value will have a slower falloff and thus hearable from a greater distance, while a higher value will make the audio degrate faster and dissapear from a shorter distance"), ListenToMic.GetComponent(MicControl).VolumeFallOff);
20. ListenToMic.GetComponent(MicControl).PanThreshold = EditorGUILayout.FloatField(GUIContent ("PanThreshold","Set the rate at wich audio PanThreshold gets switched between left or right ear. A lower value will have a faster transition and thus a faster switch, while a higher value will make the transition slower and smoothly switch between the ears. Don't go to smooth though as this will turn your audio to mono channel"), ListenToMic.GetComponent(MicControl).PanThreshold);
21. }
22.
23.
24. //Redirect select ingame
25. ListenToMic.GetComponent(MicControl).SelectIngame=GUILayout.Toggle(ListenToMic.GetComponent(MicControl).SelectIngame,GUIContent ("Select in game","select the audio source through a GUI ingame"));
26.
27. //Redirect Mute ingame

```
28. ListenToMic.GetComponent(MicControl).Mute=GUILayout.Toggle(ListenToMi
    c.GetComponent(MicControl).Mute,GUIContent ("Mute","when dissabled you
    can listen to a playback of the microphone"));
29.
30. //Redirect debug ingame
31. ListenToMic.GetComponent(MicControl).debug=GUILayout.Toggle(ListenTo
    Mic.GetComponent(MicControl).debug,GUIContent ("Debug","This will write
    the gathered Loudness value to the console during playmode. This is handy if
    you want if statements to listen at a specific value."));
32.
33. //Redirect ShozDeviceName ingame
34. ListenToMic.GetComponent(MicControl).ShowDeviceName=GUILayout.Togg
    le(ListenToMic.GetComponent(MicControl).ShowDeviceName,GUIContent
    ("Show Device name(s)","When selected all detected devices will be written
    to the console during play mode"));
35.
36.
37. EditorUtility.SetDirty(target);
38.
39.     // Show default inspector property editor
40.     DrawDefaultInspector ();
41.     }
```

```
42.
43.
44.
45.         // Custom GUILayout progress bar.
46.     function ProgressBar (value : float, label : String) {
47.         // Get a rect for the progress bar using the same margins as a textfield:
48.         var rect : Rect = GUILayoutUtility.GetRect (18, 18, "TextField");
49.         EditorGUI.ProgressBar (rect, value, label);
50.         EditorGUILayout.Space ();
51.     }
52.
53.}
```