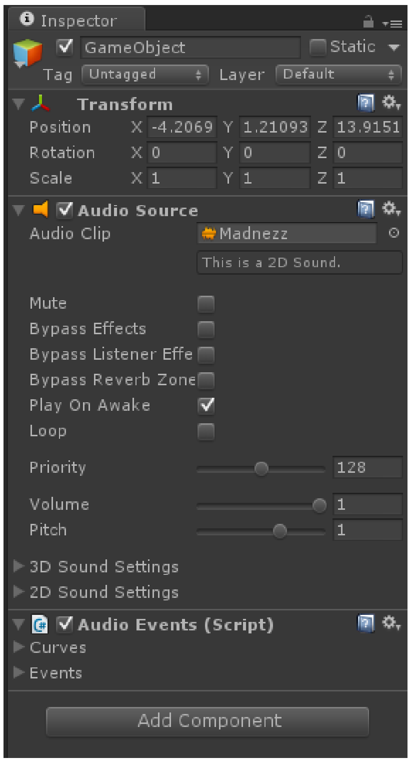
DarkACE : Audio Curves and Events

Part 1: Setting up

In order to set up your scene for usage with DarkACE, you should add the AudioEvents script to the object that should react to the events and that object must have an AudioSource with an AudioClip attached to it. This AudioClip you set here is the clip on which you will be placing curves and events and should not ever be changed. If you really want to change it, remove the AudioEvents script and re-add it after setting the correct clip. This will however remove everything you did with the last track.


Inspector

✓ GameObject Static

Tag Untagged Layer Default

Transform

Position X -4.2069 Y 1.21093 Z 13.9151

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

✓ Audio Source

Audio Clip Madnezz

This is a 2D Sound.

Mute

Bypass Effects

Bypass Listener Effe

Bypass Reverb Zone

Play On Awake ✓

Loop

Priority 128

Volume 1

Pitch 1

3D Sound Settings

2D Sound Settings

✓ Audio Events (Script)

Curves

Events

Add Component

Figure 1: A proper ACE-enabled object

Part 2: The editor

The editor can be found in *Window > ACE Editor*
The editor will only show data when a proper ACE-enabled object is selected.

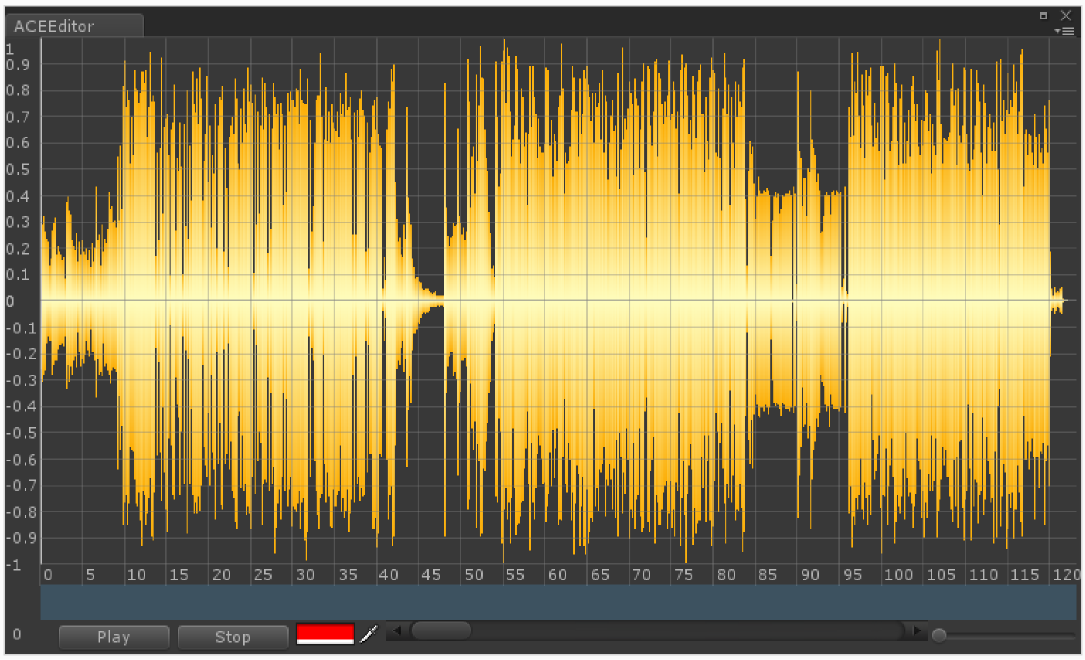


Figure 2: The empty ACE editor

When you first open up the editor it will be empty, without any curves and events.

Let's check out the editor's features!

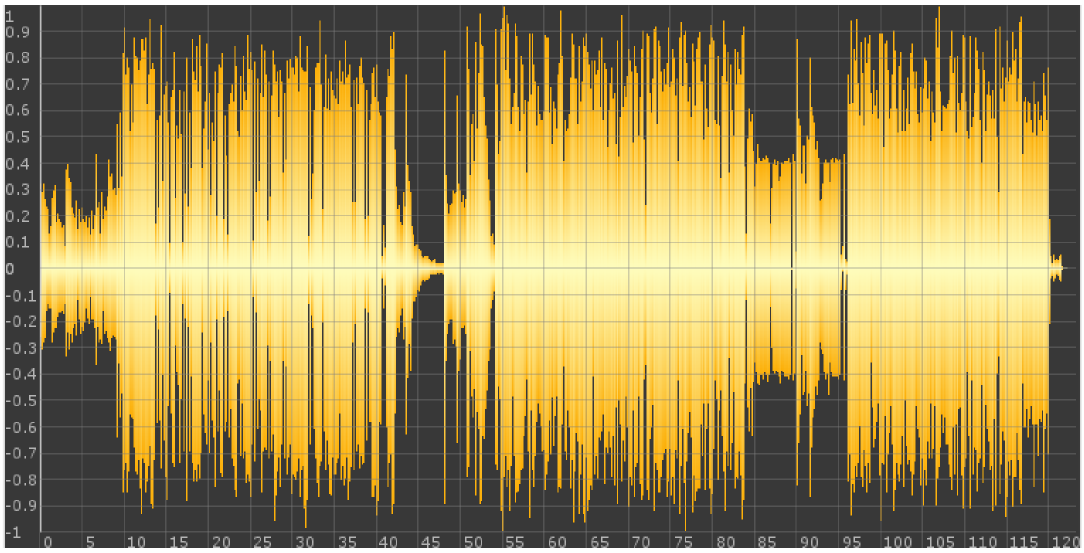


Figure 3: Main audio panel

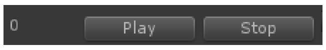


Figure 4: Audio controls: current time (in seconds), Play and Stop/Pause

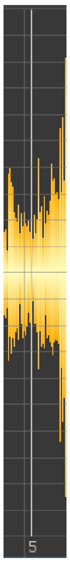


Figure 5: Controllable audio time indicator

Figure 6: Color of selected curve (if curves available)

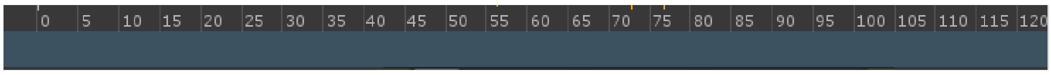


Figure 7: Triggers line

Figure 8: Scroll and zoom

Part 3: Curves

The curves system allows you to manually define and query curves data for your sounds.
This can be useful if, for example, you want the speed of your character to depend on the audio source's time
, or perhaps you might want to make a drugs simulator and you want the audio track to change screen colors, etc. The usage possibilities are endless. :)

You can add a curve by opening the curves dropdown of the AudioEvents component in the inspector and pressing the + button.
You can remove the last curve in the list by pressing the - button.

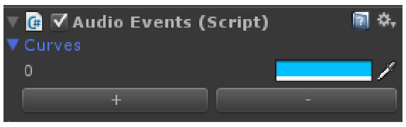


Figure 9: Inspector curves view

Once you add a new curve the change should happen immediately, and there should be a new curve in the ACE Editor.

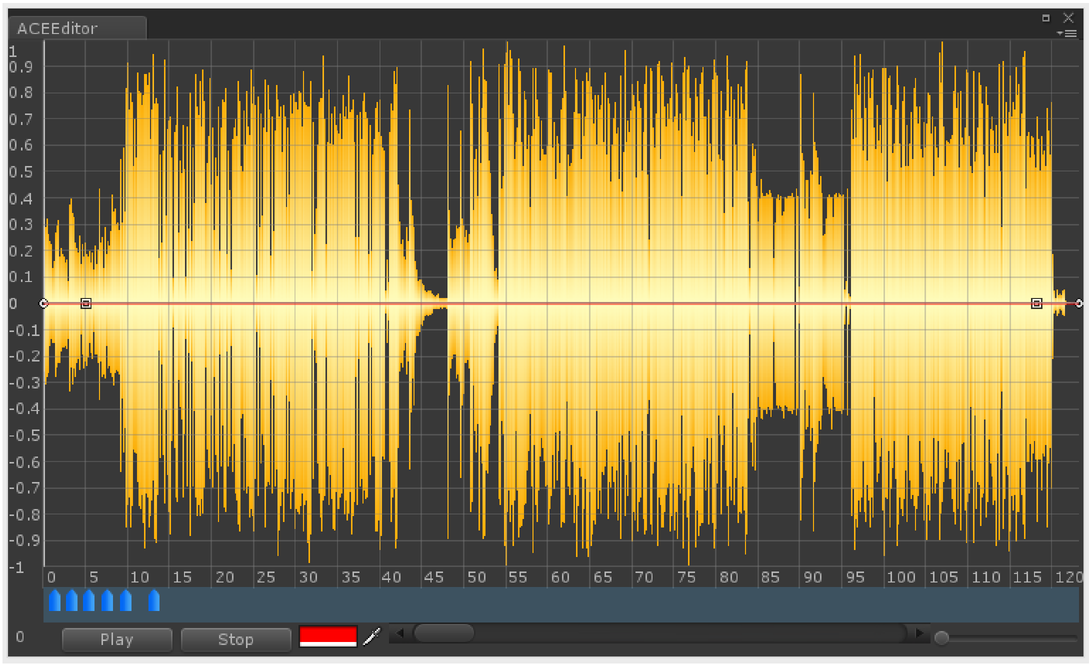


Figure 10: One curve in the editor

We can now start modifying the curve by dragging the points and the handles.
To **select** a curve, click any point that belongs to that curve. The color picker will have the color of the selected curve.
We can **add points** on the selected curve by **shift-clicking** on the desired position of the new point.
Points can be **removed** by **control-clicking** on the point that we want removed.

For proper functioning, your curve **must not** contain S shapes, else funny things could happen.

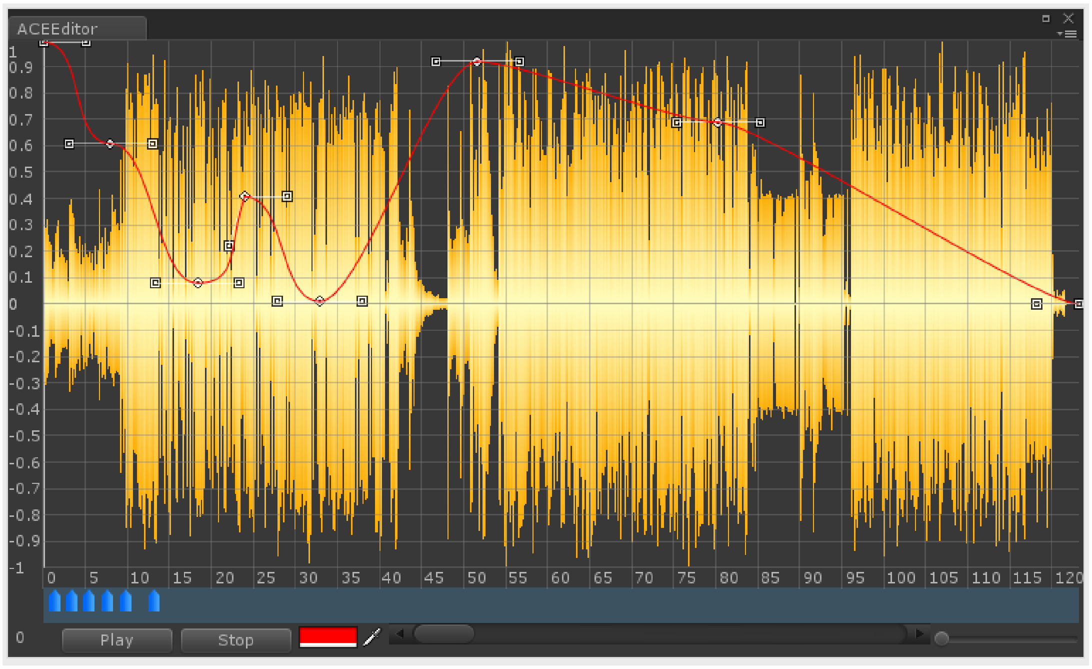


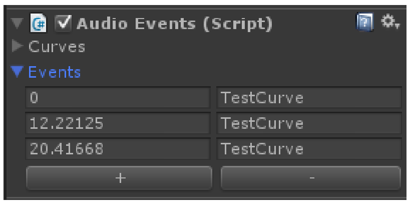
Figure 11: An example curve

Part 4: Trigger Events

The trigger system is made to call your functions whenever the audio clip is at a certain point of its duration that you define.
This can be useful if, for example, you want to make your character dance differently once the music style of a track changes.

There are two ways you can add triggers.

- **Shift-clicking on the Triggers Line** (see Figure 7) where you want the new trigger to be
- Clicking the plus sign on the Audio Events component



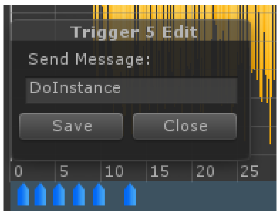
0	TestCurve
12.22125	TestCurve
20.41668	TestCurve

Figure 12: Adding trigger through the Audio Events component

You can drag around your new triggers in the editor to change their position, or you could manually enter the time in seconds in the Audio Events component.
To remove an event, alt+click it.

After placing the components where you want them to be, you should tell them which message to send.
There are two ways of achieving this:

- **Right clicking or control-clicking** on the event in the ACE Editor, and typing the function name there.
- Typing the function name in the second column of the Audio Events component.



Send Message: DoInstance

Figure 13: Changing trigger message through the ACE Editor

Part 5: Scripting Trigger events

Curves

The common use case for curves is getting the curve value at a specific time. This can either be the current audio time of the ACE-Enabled object, or an arbitrary time that you need.
For this the functions

```
GetCurrentValue(int curve)
```

and

```
GetValueAtTime(int curve)
```

exist in every instance of the AudioEvents class.

Here's an example that changes the camera's background color given three curves to signify red, green and blue color.

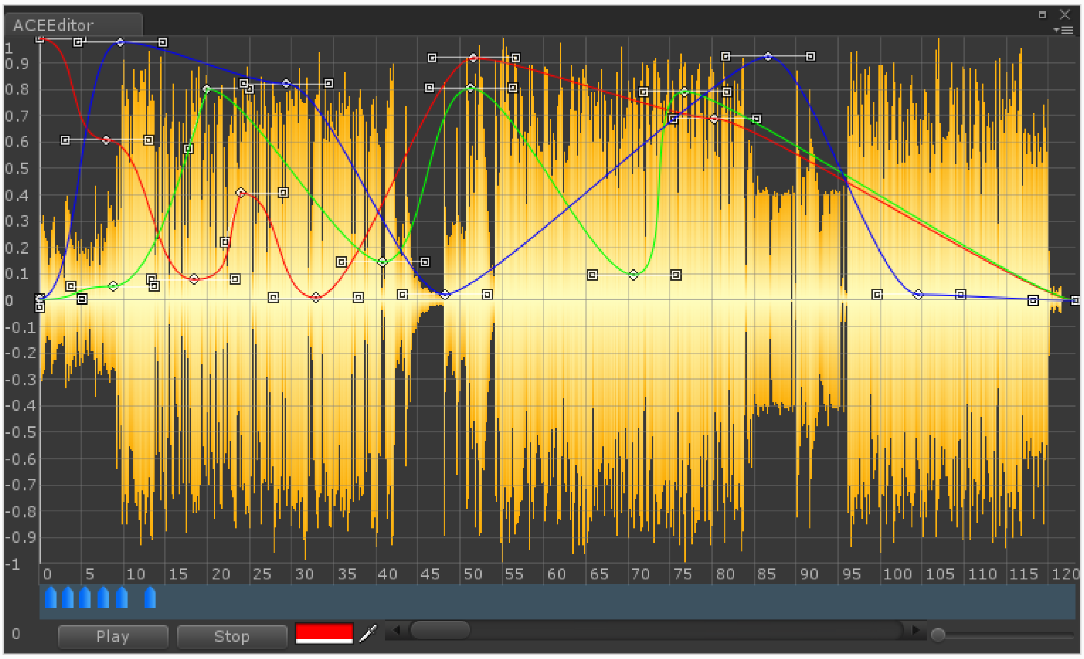


Figure 14: Example curves

Here's code for the script attached to the ACE-Enabled object:

```
using UnityEngine;
using System.Collections;

public class CameraBackground : MonoBehaviour {

        AudioEvents audioEvents;

        // Use this for initialization
        void Start () {
                audioEvents = GetComponent();
        }

        // Update is called once per frame
        void Update () {
                float r = audioEvents.GetCurrentValue(0);
                float g = audioEvents.GetCurrentValue(1);
                float b = audioEvents.GetCurrentValue(2);

                Camera.main.backgroundColor = new Color(r, g, b);
        }
}
```

Triggers

Triggers are a little different to implement because they don't have a state. They just send a message to the ACE-Enabled gameobject they belong to when they occur.

Here's an example which instantiates a given object at a random place close to 0, 0, 0:

```
using UnityEngine;
using System.Collections;

public class CreateInstance : MonoBehaviour {

        public GameObject original;

        void DoInstance(){
                GameObject.Instantiate(original, new Vector3(Random.value, Random.value, Random.value), Quaternion.identity);
        }
}
```

Most of times you would probably want to just add a proxy script that forwards the message to selected other objects in the scene.