Sound Reactor

Manual

TOC

Copyright

- The song: "Don't Make Me" is owned by Ryder and is licensed to Aaron Taylor through **wemakedancemusic.com**. The license allows the original purchaser: Aaron Taylor to distribute inside of projects.
- The song: "SoundReactorShort" is owned by Aaron Taylor and made specifically for Sound Reactor. Permission is hereby granted to use the song in YouTube videos so long as it is demoing work created with Sound Reactor.
- The song: "Synthetic Agony" is owned by Vae. His music can be found at: **soundcloud.com/vaetan/**. Permission to distribute with Sound Reactor was granted by him.
- All other assets are the Copyright of **Little Dreamer Games**. All rights reserved.

Overview

SUMMARY

Sound Reactor is a Unity Extension that makes it easier than ever to make things in Unity react to sound. It is a versatile Unity Extension that is both modular and flexible. With it, you can create stunning visualizers, and drive any property to any Unity AudioSource with ease.

KEY FEATURES

- Easy to use and powerful **Spectrum Builder**
- Property drivers that drive: **color**, **position**, **rotation**, **scale**, **particle emitters**, and **physics forces**
- **Event handler driver** that can be used to drive any public property
- **Expandable property driver class** so you can create your own property drivers
- Remembers changes made during play mode
- A modular design that makes it possible to integrate into any project

BASICS

The setup for Sound Reactor is fairly basic. A typical setup consists of some foundation scripts, some drivers to drive properties, and GameObjects that contain said scripts.

Foundation Scripts

Essential scripts are necessary for any project using Sound Reactor. They are:

- **SpectrumSource**
- **SpectrumFilter**
- **Level**
- One or more drivers that inherit from **PropertyDriver**

Peaks Profile

Currently there is only one external file required by Sound Reactor, and that's the **PeaksProfile**.

Drivers

Drivers are what cause property values to change. The drivers included with Sound Reactor are:

- **PositionDriver**
- **RotationDriver**
- **ScaleDriver**
- **ColorDriver**
- **ForceDriver**
- **ParticleEmitterDriver**
- **EventDriver**

Supplemental

- **EQ**
- **Spectrum Builder**

MODULAR DESIGN

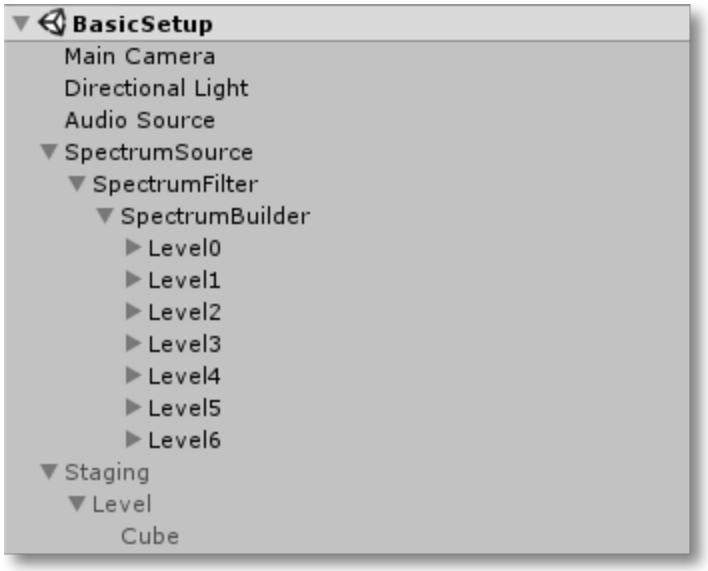
Sound Reactor is designed in such a way that connections between certain scripts can be done manually, or set up in a hierarchy so connections happen automatically. The hierarchy example in the figure below is set up so connections happen automatically.

Automatic Connections

The following scripts are arranged in the order they look for each other at runtime. i.e. **PropertyDriver** looks for **Level**, **Level** looks for **SpectrumFilter**, and so on. It does this by first looking for the script on the GameObject it's attached to, and if it doesn't find one, it travels up through each parent GameObject until it finds what it's looking for.

- **PropertyDriver**
- **Level**
- **SpectrumFilter**
- **SpectrumSource**
- AudioSource

HIERARCHY EXAMPLE



Basic Setup

SUMMARY

This is the most basic setup using: SpectrumSource, SpectrumFilter, Spectrum Builder, and ScaleDriver

Setting up the hierarchy

1. In the hierarchy view, click: **Create->Audio->Audio Source**
2. Add an audio clip to the Audio Source. If you don't have one, you can search for the "Don't Make Me" song.
3. Right click in an empty place in the Hierarchy and select: **SoundReactor->SpectrumSource**
4. Drag Audio Source into the Audio Source property in *SpectrumSource* (leaving this property empty will cause all the sounds together to become one spectrum)
5. Right click the new *SpectrumSource* GameObject and select: **SoundReactor->SpectrumFilter**
6. Right click the new *SpectrumFilter* GameObject and select: **SoundReactor->Builder**

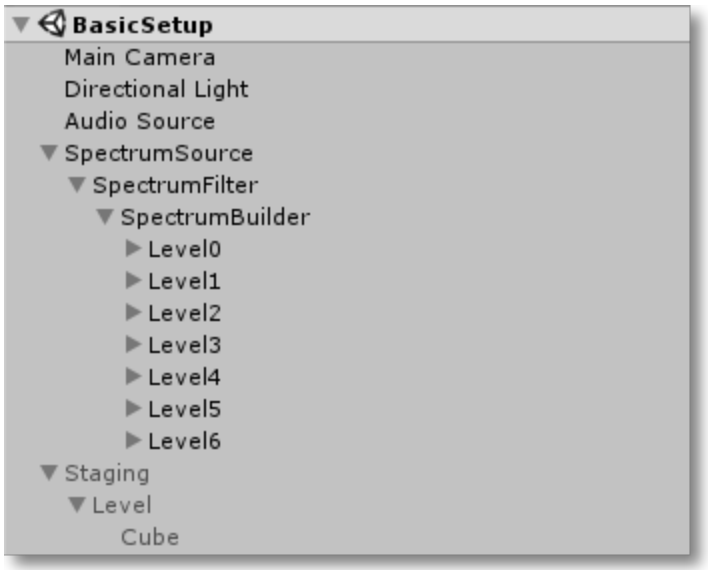
Staging a Level for the Builder

7. Create a GameObject called *Staging* at the root of the hierarchy
8. Right click *Staging* and select: **SoundReactor->Level**
9. Right click *Level* and select: **3D Object->Cube**
10. Click on the *Cube* and select: **Tools->SoundReactor->Driver->Scale**
11. Set the *Y* in *Travel* to 8
12. Disable the *Staging* GameObject since it's not used beyond using it to build up the spectrum.

Building the spectrum visualizer

13. Click on the *SpectrumBuilder*
14. Click and drag the *Level* into the *Level* property in the *SpectrumBuilder*
15. Go into play mode and the cubes should be scaling with the audio.

RESULT



Event Driver Setup

SUMMARY

Setup a scene that drives properties.

EVENT HANDLER

```
using UnityEngine;
using LDG.SoundReactor;

public class EmissionHandler : MonoBehaviour
{
    public Color emissionColor;

    private Material material;
    private int emissionColorID;

    private void Start()
    {
        MeshRenderer meshRenderer;

        if ((meshRenderer = GetComponent<MeshRenderer>()))
        {
            if ((material = meshRenderer.material) != null)
            {
                emissionColorID = Shader.PropertyToID("_EmissionColor");
            }
        }
    }

    public void OnLevel(PropertyDriver driver)
    {
        if (material)
        {
            float level = driver.LevelScalar();

            material.SetColor(emissionColorID, emissionColor * level);
        }
    }
}
```

Create the class

1. Create the above class and call it **EmissionHandler**

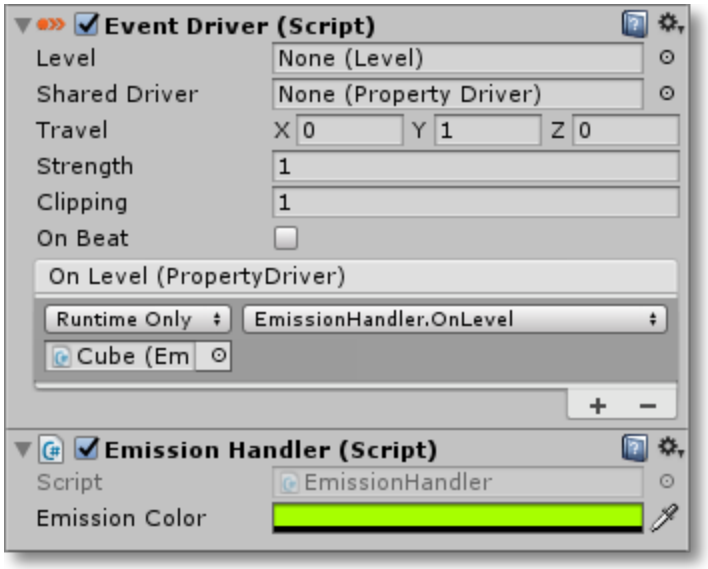
Basic Setup

2. Follow the **Basic Setup**.

Attaching the Handler

3. Attach the **EmissionHandler** class to the *Cube*.
4. Create a new *OnLevel* handler by pressing the '+' button.
5. Attach the *Cube* to the GameObject property.
6. Select *EmissionHandler->OnLevel* from the function drop down menu. Make sure to choose the function located at the top of the list, which is the **dynamic** version.

RESULT


Event Driver (Script)
Level
None (Level)
Shared Driver
None (Property Driver)
Travel
X 0
Y 1
Z 0
Strength
1
Clipping
1
On Beat
On Level (PropertyDriver)
Runtime Only
EmissionHandler.OnLevel
Cube (Em
Emission Handler (Script)
Script
EmissionHandler
Emission Color

Record Peaks

SUMMARY

Peaks are used to scale frequency magnitudes (Levels) to a range somewhere between 0 to 1. Without peaks, the values are to arbitrary to trigger events reliably. For this reason, PeakProfiles can be created and shared among any audio clip. See Audio Peaks for a more detailed explanation.

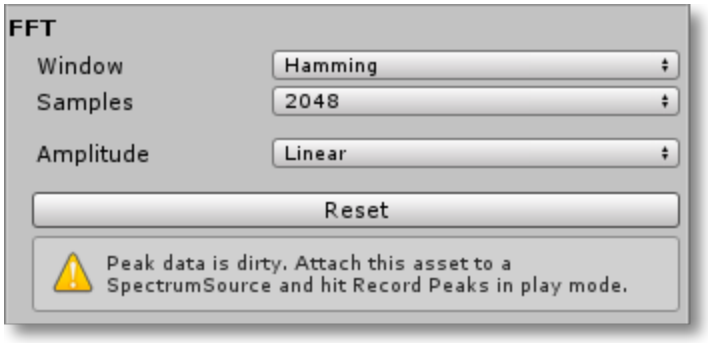
Creating a PeaksProfile

There are multiple ways to create a PeaksProfile file. Here are the most common ways:

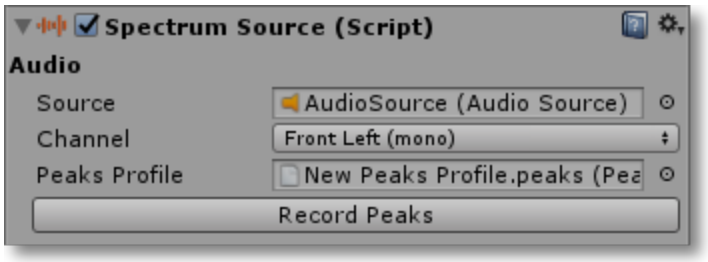
- Right click on an audio file and select: **Create->SoundReactor->Peaks Profile**
- Select: **Create->Assets->SoundReactor->Peaks Profile**
- Select: **Tools->SoundReactor->Peaks Profile**

Recording a PeaksProfile

1. First the profile must be made dirty. To make it dirty, either change one of its settings, or press the Reset button in the inspector.



2. Attach it to a SpectrumSource.
3. Attach an AudioSource that is pointing to an AudioClip you'd like to record peaks for. To generate generic peaks, record the peaks of a Sweep Sound.
4. Go into play mode and a *Record Peaks* button will appear just under where the profile was attached to in the SpectrumSource. Press the button.



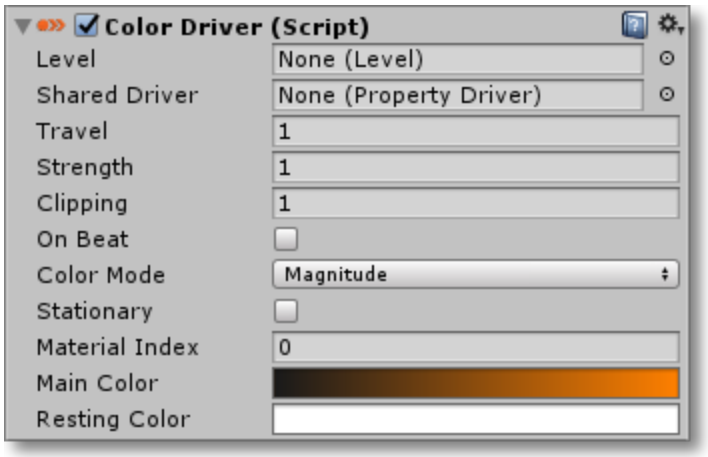
5. The peaks are done recording when the end of the AudioClip has been reached. Sound Reactor will confirm this by hiding the record button and posting a message in the console.

Color Driver

SUMMARY

This driver changes the color of: **materials**, **particles**, and **vectors**.

INSPECTOR



PROPERTIES

Color Mode

The method the color gradient is applied.

- **Magnitude** Sets the gradient color using the level's magnitude.
- **Frequency** Sets the gradient color using the level's frequency.

Stationary

Check this if a Segmented Levels shape was built with the **SpectrumBuilder**. This option only displays when Color Mode is set to Magnitude.

Material Index

The index to a material for the attached mesh.

Main Color

The main color used to colorize the object. This changes the _Color property in the shader. If a _Color property doesn't exist, then the object will not be colorized.

Resting Color

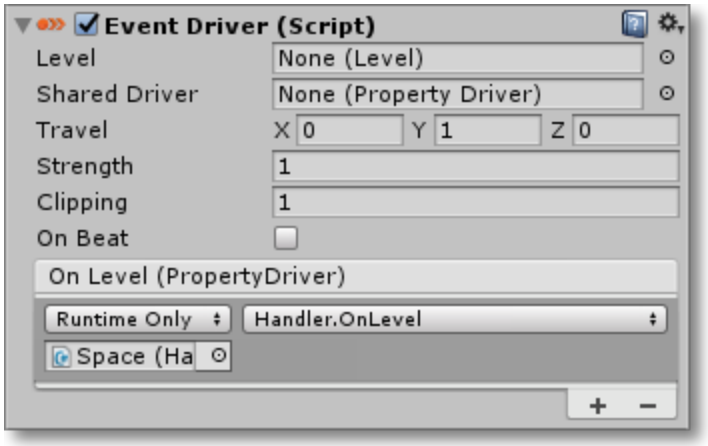
This color is applied to segmented levels, and vector shapes that are anchored.

Event Driver

SUMMARY

An easy way to animate properties in a class.
See **EventDriverSetup** for setup.

INSPECTOR



PROPERTIES

See: **PropertyDriver**

OnLevel

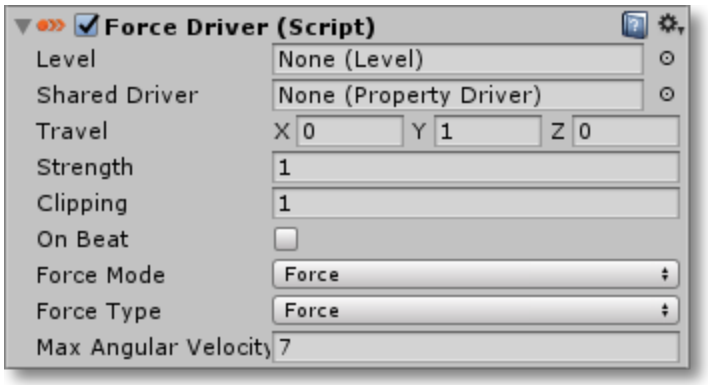
Takes an event handler that can process **PropertyDriver** values.

Force Driver

SUMMARY

Applies a force to a GameObject that has a RigidBody attached to it.

INSPECTOR



PROPERTIES

Force Mode

How the force should be applied to the RigidBody.

- Force
- Impulse
- Velocity Change
- Acceleration

Force Type

What type of force to apply

- Force
- Torque
- Relative Force
- Relative Torque

Max Angular Velocity

Limits the angular velocity

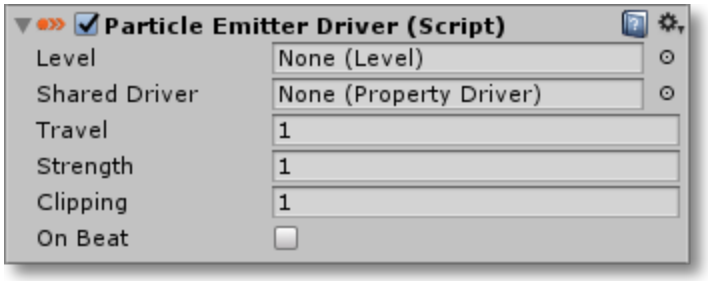
Particle Emitter Driver

SUMMARY

Emits specified number of particles.

See: **PropertyDriver** for more info.

INSPECTOR



PROPERTIES

See: **PropertyDriver**

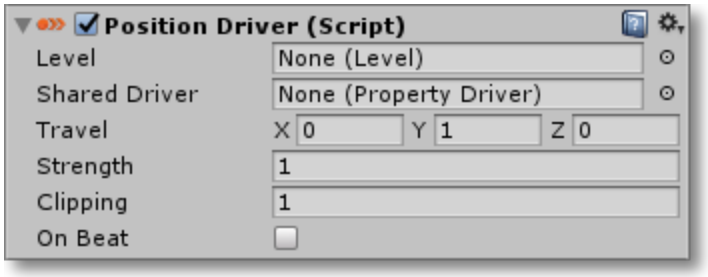
Position Driver

SUMMARY

Moves an object.

See: **PropertyDriver** for more info.

INSPECTOR



PROPERTIES

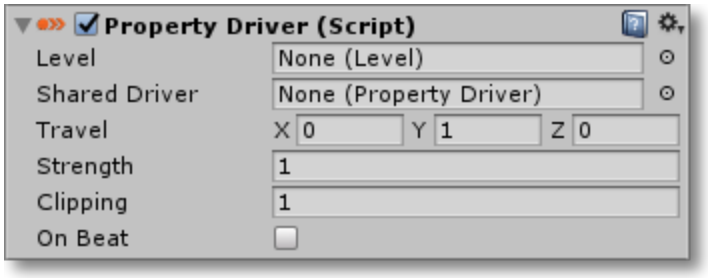
See: **PropertyDriver**

Property Driver

SUMMARY

PropertyDrivers use the magnitude of a frequency stored in a **Level** to drive property values. This class is meant to be inherited from. All its properties are common and editable in the following scripts included with Sound Reactor: **PositionDriver**, **RotationDriver**, **ScaleDriver**, **ParticleEmitterDriver**, **ColorDriver**, **ForceDriver**, and **EventDriver**.

INSPECTOR



PROPERTIES

Level

The **Level** this driver uses to calculate the travel distance. If this is set to None, then it'll try to find a Level at runtime on the GameObject it's attached to by looking up through the hierarchy.

Shared Driver

Travel, Max Level, Strength, and On Beat are all overridden by this driver.

Travel

Defines the travel distance. If a level reaches 100% of its magnitude, and the travel is set to 10, then the travel distance will be 10.

Strength

Scales the level's magnitude. This is useful to get values to reach their max travel distance sooner.

Clipping

Clips the level's magnitude. A value of 1 will allow the level to reach 100% of its magnitude.

On Beat

Causes the Travel to reach 100% once per frame per beat. This is useful for physics and particle emitters.

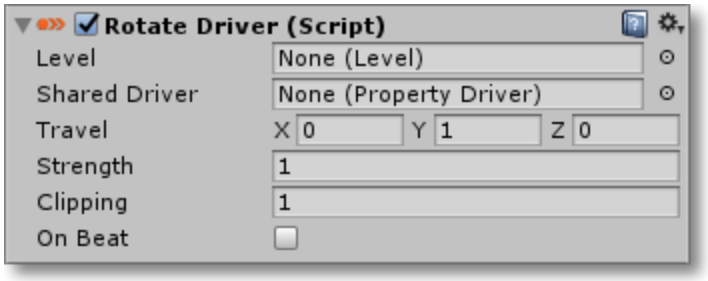
Rotation Driver

SUMMARY

Rotates object.

See: **PropertyDriver** for more info.

INSPECTOR



PROPERTIES

See: **PropertyDriver**

Scale Driver

SUMMARY

Scales object.

See: **PropertyDriver** for more info.

INSPECTOR

PROPERTIES

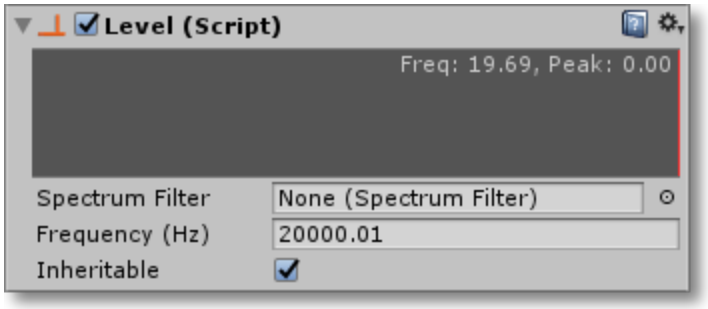
See: **PropertyDriver**

Level

SUMMARY

A Level points to a specific frequency in **SpectrumSource**, and its value is the magnitude of that frequency after it has been **normalized**. The level itself is always falling at a certain rate, which is defined by the **SpectrumFilter** it points to.

INSPECTOR



PROPERTIES

Spectrum Filter

The filter to grab spectrum data from. If this is set to None, then it'll try to find a **SpectrumFilter** at runtime on the GameObject it's attached to, then by looking up through the hierarchy for the first **SpectrumFilter** it finds.

Frequency (Hz)

The audio frequency that the level is tracking. You can either set the frequency directly in the edit field, or by left clicking the mouse inside the frequency window. When using the **SpectrumBuilder**, the frequency is automatically set for each level created.

Inheritable

Tells a child **PropertyDriver** if it can inherit a Level from another GameObject or not. The only exception is if the **PropertyDriver** is sharing a GameObject with the **Level**.

Peaks Profile

SUMMARY

The PeaksProfile contains peaks used for normalizing spectrum magnitudes. See the manual for setup.
See **Record Peaks** for recording custom peaks.

INSPECTOR

PROPERTIES

Window

The quality of the spectrum data. The list is sorted so that low quality starts at the top, with Blackman Harris being highest quality is at the bottom. The higher the quality the lower the performance, and vise versa.

Samples

The number of samples. The higher the number the higher the quality. The higher the quality the lower the performance, and vise versa.

Amplitude

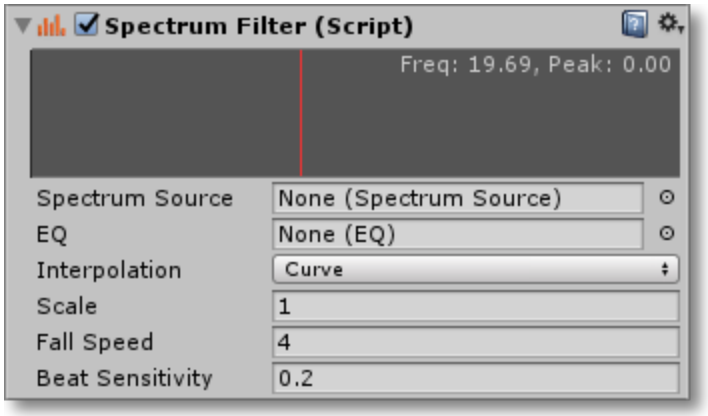
Sets whether to use decibel or linear magnitude.

Spectrum Filter

SUMMARY

The SpectrumFilter is used to modify the shape and scale of the spectrum. It also allows you to set the falling speed and beat detection sensitivity of all levels parented to this filter.

INSPECTOR



PROPERTIES

Frequency Window

Red Line Represents the frequency.
Green Line Represents the normalized spectrum.
Blue Line Represents the falling level.
Black Line Will change positions whenever the threshold under *Beat Sensitivity* is reached.

Spectrum Source

The **SpectrumSource** to filter. If this is set to None, then it'll try to find a **SpectrumSource** at runtime on the GameObject it's attached to, then by looking up through the hierarchy for the first **SpectrumSource** it finds.

EQ

This gives you a little more control over how the spectrum is scaled. See **EQ** in this document for more details.

Interpolation

Sets whether values in between levels are interpolated as straight or curved lines.

Scale

Scales all incoming band magnitudes acquired from the attached **SpectrumSource**.

Fall Speed

The speed at which the levels should fall. This is in **normalized space**, so setting the value to 1 will bring the level from full height down to zero in 1 second. Setting this value to 2 would bring it all the way down in a half a second.

Beat Sensitivity

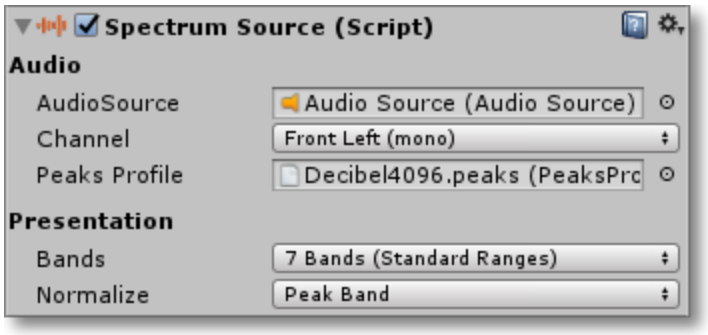
Just like fall speed, this value is in **normalized space**. Setting this value to 0.75 would cause a beat to happen when a level descends 75%. Setting the value to 0.5 would cause a beat to happen when the level descends 50%.

Spectrum Source

SUMMARY

Converts an AudioSource to spectrum data, and then processes it for output to **SpectrumFilter**.

INSPECTOR



PROPERTIES

AudioSource

The audio source to pull spectrum data from. If this is set to *None*, then a spectrum will be created from all the audio sources currently playing.

Channel

The audio channel to pull spectrum data for. If an AudioSource is attached and points to a valid AudioClip, this channel will be restricted to the highest supported channel for that clip.

Peaks Profile

A profile that stores peak data along with sample rate and amplitude type.

Bands

This sets how many bands the spectrum will be divided into.

Normalize

Normalizes the attached levels to a range of 0 to 1.

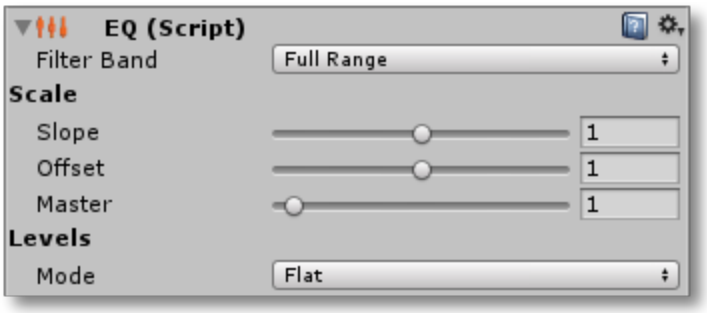
- **Peak** Normalizes all levels based on the spectrum's peak.
- **Peak Band** Normalizes each level based on its own peak.
- **Raw** No scaling whatsoever, i.e. the unprocessed data straight from Unity's GetSpectrumData function.

EQ

SUMMARY

Modifies the **normalized magnitudes** of the bands coming out of **SpectrumSource**.

INSPECTOR



PROPERTIES

Filter Band

All bands outside the specified range will be scaled to zero.

Slope

Scales the slope defined in Mode.

Offset

Offsets all the levels.

Master

Scales the levels

Mode

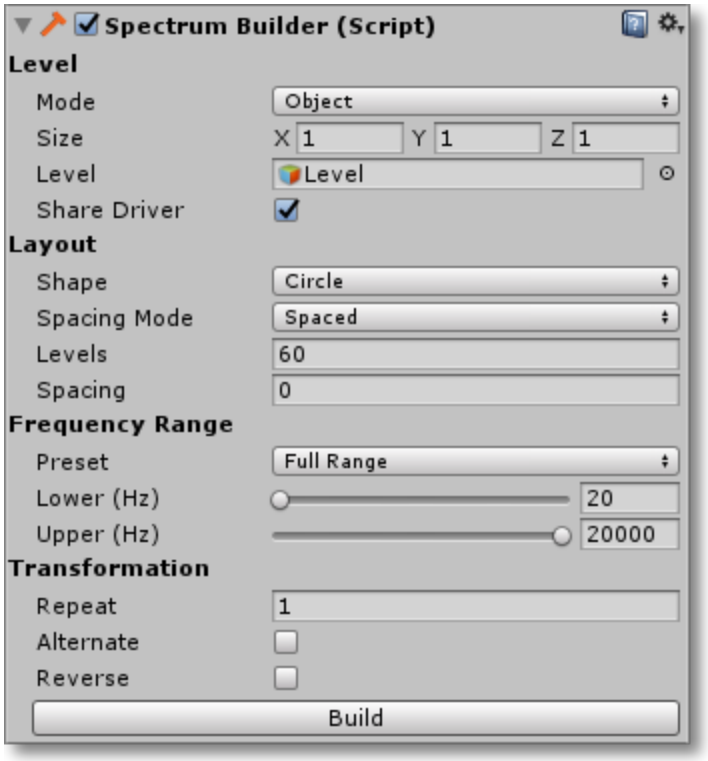
Defines a shape for the levels.

Spectrum Builder

SUMMARY

The SpectrumBuilder is the quickest and easiest way to build a visualizer. It makes it possible to create frequency ranges in a variety of shapes, and even allows you to apply simple transforms to those frequencies to further customize their arrangement along the shape.

INSPECTOR



PROPERTIES

Mode

Tells the level to be used as an object or a vector.

Size

The **Level** is scaled by this size.

Level

The GameObject containing **PropertyDrivers**. When Mode is set to Object.

Share Driver

Tells the builder to attach the **PropertyDrivers** from the **Level** to the instanced Levels. When Mode is set to Object.

Color Driver

The **ColorDriver** to use for coloring the vector. When Mode is set to Vector.

Travel

The distance the vertices should move. When Mode is set to Vector.

Anchored

Anchors the bottom or inside edge of the vector. When Mode is set to Vector.

Anchored Diam.

Diameter of the inside edge of the vector. When Mode is set to Vector and Anchored is checked.

Shape

The shape to arrange the Levels into.

Spacing Mode

How the **Levels** should be spaced. **Levels** can either be spaced evenly apart from each other, or they can be divided evenly based on the layout size.

Levels

The number of levels to build up the spectrum with.

Preset

Frequency range presets.

Lower (Hz)

Lower frequency of the range.

Upper (Hz)

Upper frequency of the range.

Repeat

The number of times to repeat the frequency range along the shape.

Alternate

Tells the frequency to reverse the frequency every time it repeats.

Reverse

Causes the levels to be assigned frequencies starting with the highest frequency first.

Flip Level

Only works with segmented levels. Causes the levels to display upside down.

Audio Peaks

SUMMARY

Audio peaks are an important part of visualizing spectrums, and even more important when you want things to react to the magnitude of a frequency. This topic exists to help developers understand what peaks are and how they are used to make spectrum values useful.

Peaks

A **peak** is considered the highest possible magnitude of a frequency for a given sound. If we divide a magnitude by it's peak, the result is a value that ranges from 0 all the way up to 1. This technique is called normalization, and it is very useful when visualizing sounds, or writing events that react to sounds.

Raw Spectrum

Unity returns spectrum data at a very small scale, around 0.01, and those values are not very useful to us. Moreover, the scale changes slightly depending on the sample rate, FFT window, and magnitude type, i.e. dB vs. Linear. What we need to do then is save the peaks of a sound so they can be used later to normalize the sound, and Sound Reactor let's you do just that by allowing you to save peaks to a file called a **PeaksProfile**.

Peaks Profile

A **PeaksProfile** stores: sample rate, FFT window, magnitude type, peaks for 30 bands, and one peak for the entire spectrum. There are two types of peaks profiles that can be recorded:

- **Unique Peaks** peaks used to normalize the magnitudes of one sound. Recording a peaks profile like this guarantees that the highest magnitude of that sound will always reach a value of 1, but it means recording the peaks of every sound you intend to play.
- **Generic Peaks** peaks used to normalize any sound. This method doesn't guarantee that the highest magnitude will reach a value of 1, but it does means that you only need a minimum of one profile.

Generic Peaks

In order to record generic peaks you need a special sound file called a *sweep tone*. A *sweep tone* is a sound that plays a tone from 20Hz all the way up to 20,000Hz at a constant dB, which is essentially creating a ceiling of peaks. The result is a curve that looks like the following:

Sound Reactor already comes with a few profiles that can be used, just look for: **Decibel2048.peaks**. However, if you need to record your own, you can use the provided sound file, or generate one from a website that generates sweep tones, then follow this **recording guide**.

Bands

Last but not least. Sound Reactor doesn't record the peaks of every sample in the spectrum, it only records the number of bands specified in **SpectrumSource**. Furthermore, when the peaks are saved to a **PeaksProfile**, only 30 bands are saved. These peaks are automatically interpolated when saving, and when loading.