Synesthesure

Synesthesure's AudioVisual component listens to incoming audio, and processes it into dynamics that are easy to use in scripts, and can even control parameters of other game object components without programming. Synesthesure enables vFX creators to focus on the artistic and creative-exploration process of creating visual effects, instead of FFT signal processing and "massaging" audio values to get "workable" baselines.

It is important to understand that humans *perceive* sound levels different than they actually are in reality. Human ears are not as sensitive to bass sounds as the mid or high pitched sounds. Synesthesure compensates for this discrepancy with some "magic numbers" used in its calculations for its Bass, Mid, Treble, and Sibilance frequency bands output.

Synesthesure outputs normalized values of **0 – 1** for its frequency bands' dynamics.
Values of 0 – 1 are very easy to work with, vs. their actual amplitude/power values.
Dynamics provided for **Bass, Mid, Treble, Sibilance, and Volume**:

- **Raw** – the raw value tracks very literally with the level.
- **Smoothed** – the raw value smoothed a bit, so it's not so "rough".
- **Average** – the raw value averaged over a small amount of time.
- **Chase** – chases the raw value and set to be more/less sensitive to up/down changes.
- **Decay** – pushed up by raw level and decays down after a bit if not kept pushed up.
- **Attack** – spikes with the initial "hit" of sound, and then decays until the actual level has gotten low enough to allow it to be hit harder again. Great for accenting beats, the hits (but not holds) of chords/notes, or the syllables in words.

Access a dynamic's value by: float value = Bass.Attack;

Synesthesure also provides a more fine-grained method for getting audio info:
It divides the audio spectrum into **64 frequency "bins"** – from bass to a higher frequency.
The value for each bin is accessible in its raw form, but can also be fed into Synesthesure's **FrequencyResponseBands** component to get output levels that have been adjusted by an animation curve. The FrequencyResponseBands component can also use a saved frequency response curve (scriptable object) to configure it.

Synesthesure has some basic effect components that use the FrequencyResponseBands component: *AudioSignal, AudioSignalArc, MeterBars, MeterBarsArc, GradientModulation*
These components are good examples to examine if you want to create effects using the whole frequency spectrum, instead of only the bass, mid, treble, and sibilance bands.

SETUP

Add the **AudioVisual** component to a game object. (only add *one* to the scene)
If the AudioVisual component doesn't have its audio source set, it will use the audio listener, or alternatively the AudioSource component on the same game object as the AudioVisual component.

```
using Synesthesure;
AudioVisual AV;
private void Start()
{
    AV = AudioVisual.AV;
{
 …..  float value = AV.Bass.Attack;
```

COMMAND EVENTS

Synesthesure dynamically reacts to audio input, but it can also trigger time-synced events. The **CommandEvents** component sends string-based commands at predetermined times. It monitors the clip of an AudioSource for the time-sync.

The format for the command events is: *time (in seconds)* | *command*.
Example:
20 | MeterBarHeight | 1900
20 | CameraLocation | 2, 0.5
20.5 | CameraShake | false
20.5 | ChangeColor | 1

The value after the timestamp gets sent over the CommandsEvents' *Send Commands* event.
The timestamp, first "|" character, and any leading and ending spaces are not sent.

NOTE: The demos scenes and their scripts use an additional "|" (as shown in the example) as a convention to pass (and separately parse for) parameters for the command.

Lines with no timestamp followed by a "|" character are considered comment lines, are not processed, and will not cause errors. Use them to make your event routines clear and organized.

A text file can be used to define the commands for the CommandsEvents component.
Commands are triggered in the order of their timestamps – not the order listed in the component or text file. If commands have the same timestamp, the commands with matching timestamps are executed in the order they were defined.

The **AudioSourceControlHelper** component is useful for controlling the playback of a clip in an AudioSource component during Play Mode in the editor. It is useful for "dialing" in the times used for commands' timestamps to sync events with things in a song/clip.

Gradient Library

Color gradients can be stored in a GradientLibrary scriptable object.
The gradients in a gradient library are accessible at run-time.

The *ColorModulation, GradientModulation, LightModulation, MaterialColorModulation* and components use gradient libraries.

Beat Detection

Beat detection works in real-time, so it's not possible for Synesthesure to analyze audio for beats by looking ahead of the actual playback time in a file. "Beats" are determined to have occurred when the amount of power for a band significantly increases over its average, and a minimum time has elapsed.