FMOD Visualizer

By Dan Gregg

https://danielgregg.carbonmade.com/

https://soundcloud.com/rufferz

https://www.linkedin.com/in/dan-gregg/

http://www.fmod.com

FMOD Visualiser is a platform that enables the user creation of visualisations from raw audio data extracted from FMOD audio.

Version 1.0.0 – September 2019

Contents

Installing FMOD with your Unity Project

Please note this asset requires FMOD to be correctly added to your Unity Project already.

Please follow FMOD's documentation if you are unsure of how to Integrate FMOD into your Unity project.

https://www.fmod.com/resources/documentation-unity?version=2.0&page=user-guide.html

https://www.fmod.com/download

Installation

Installing FMOD Visualiser will create the following folder structure:

FMODVisualiser:

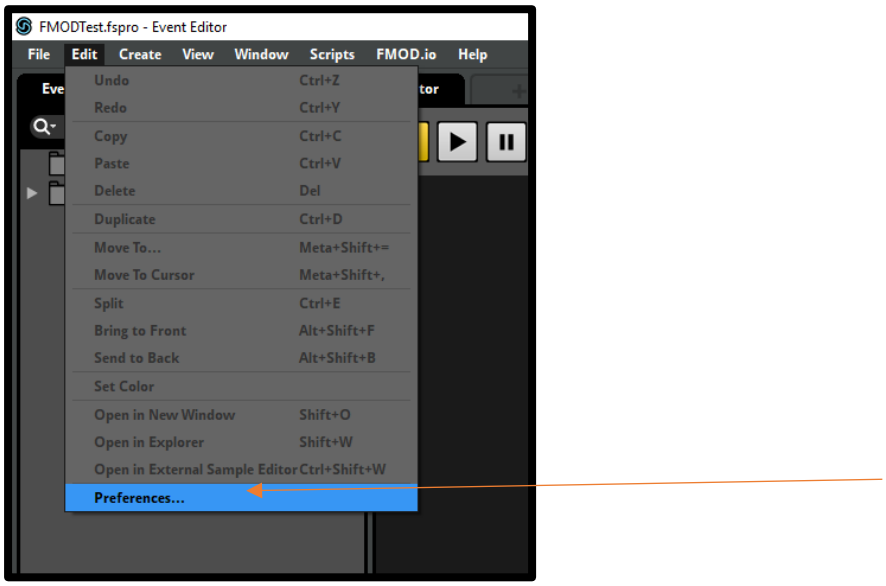
Demo: A Simple demo scene

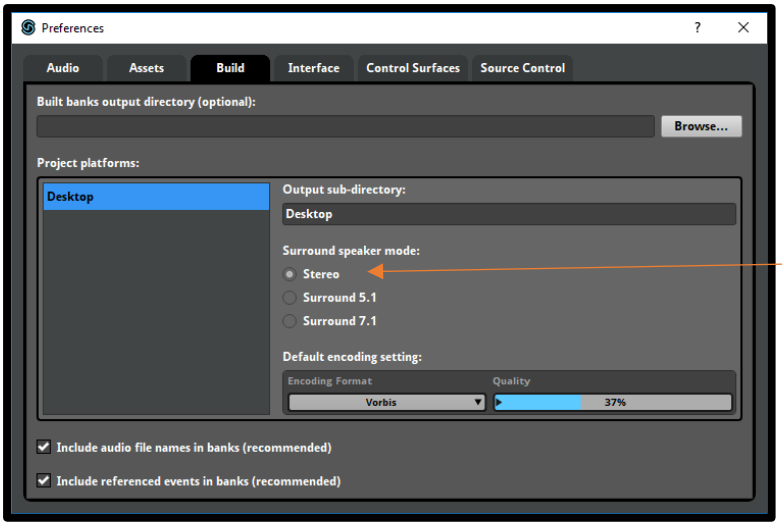
Documentation: FMOD Visualiser documentation

Scripts: FMOD Visualiser source code

Configuring FMOD

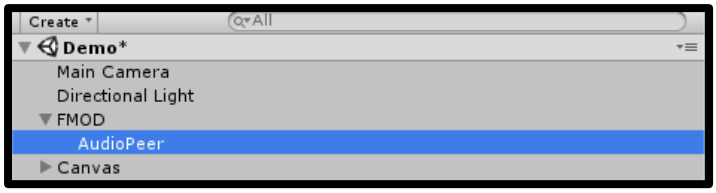
1. Once the package has been run and imported. The current FMOD project that you have configured in Unity, needs to be set to Stereo.
2. This is done within FMOD Studio, go to Edit > Preferences > Build > Project Platforms > [Build target] i.e. Desktop > Surround Speaker Mode > Stereo. **See screenshots below.**

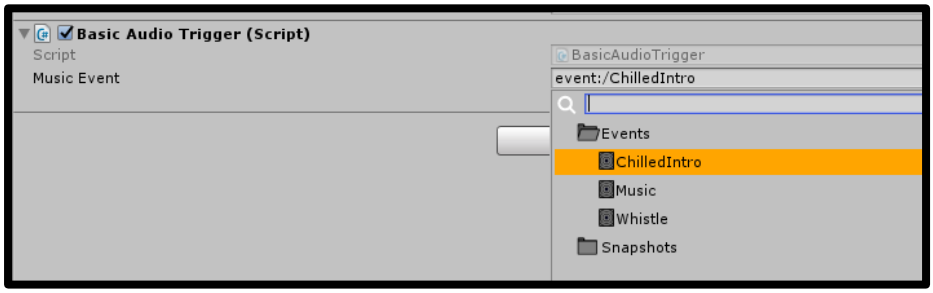




Using the demo Scene

1. Open Scenes/Demo.unity to load up an example of what FMODVisualiser can be used for.
2. Next select a music event from your FMOD Project for an example. Go into the hierarchy and select FMOD/AudioPeer and select a FMOD Music Event in the inspector window. **See screenshots below.**





3. Hit play on the editor window to start the scene and play again on the game view's UI to start playing your selected FMOD Event. **See screenshot on next page.**

Script Reference Guide

DataManager.cs

This script handles the main functionality of FMOD Visualizer, this retrieves the raw data from FMOD and then handles it and converts the data into usable variables for you to create visualizations from your FMOD project.

Parameters

- fft – FMOD's digital signal processor used for encoding live signals to digital signal.
- windowSize – Constant digital window size for retrieving data from the source.
- samples – Contains the amount of data samples in a float array.
- samplesBuffer – Contains a buffer which is delayed behind the samples to smooth out visuals.
- freqBand – Contains a band of frequencies set in the CreateFreqBands() function.
- bandBuffer – Contains a buffer which is delayed behind the freqBand to smooth out visuals.
- audioBand – 0-7 audioBands see CreateAudioBands() function.
- audioBandBuffer – Contains a buffer for audioBands
- audioProfile – This is used at the start of runtime to make the freqBandHighest array more stable for the first few frames so that you have fluid visuals from the start. The higher the number this number the less scaleVisuals will react to audio.
- samplesBufferDecrease – Contains an array of floats which the samplesBuffer uses to decrease the samples value.
- bufferDecrease – Same as above for bandBuffer.
- freqBandHighest – Contains the highest frequency value for each frequency band.
- unmanagedData – Contains the raw data retrieved from your FMOD audio.
- Length – This is used to set the length of spectrum data you want to retrieve from FMOD.
- Spectrum – This array holds the managed spectrum data extracted from FMOD once it has been managed.
- bufferDecreaseSpeed – Used to set the speed of the buffer decrease. (The shrinking speed of objects) for example.

- amplitude – Contains the average intensity of the track you are playing in FMOD.
- amplitudeBuffer – Contains a buffer for amplitude.
- amplitudeHighest – The highest amplitude value is stored here.

Functions

void AudioProfile(float audioProfile)

Set the each frequencyBandHighest at the start to equal audioProfile. This is to allow fluid visuals at the start and prevent some glitchy visuals for the first few frames.

void GetSpectrum()

Getting spectrum data from FMOD in the form of unmanaged data, then use DSP_PARAMETER_FFT to manage the data correctly. Store the data in the Spectrum array of floats.

Check that there are channels currently contained in FMOD. If so assign the data from the spectrum to the array of floats.

void SampleBuffer()

This function is used to handle the array of 512 samples and can be used for large number of visuals. If samples are bigger than the buffer increase the buffer size, if samples are smaller than the buffer, reduce by the buffer decrease.

void GetAmplitude()

Function used to calculate the average amplitude of all buffers and clamp it within bounds.

void CreateAudioBands()

Function which creates the audioBands.

void BandBuffer()

Used to smooth out values of the freqBand.

If samples are bigger than the buffer increase the buffer size, if samples are smaller than the buffer, reduce by the buffer decrease.

void CreateFreqBands()

Function used to create the frequency bands using loops.

Cubes512.cs (Example Visualization)

this script spawns the ring of cubes around the track and then updates the scale depending on the music playing.

Parameters

- sampleCubePrefab – Holds the prefab of cubes to be instantiated.
- maxScale – Caps the scale of the cubes.
- cubeAmount – Amount of cubes to instantiate.
- useBuffer – Set to true to use buffer to smooth out visuals using a buffer. If off the visuals decrease instantly if the sample is lower.

- startScale – Sets the starting scale of the cubes.
- scaleMultiplier – Used to increase the scale size variation depending on the sample.
- sampleCube – Contains an array of gameObjects which will contain each cube at runtime.

Functions

void Start()
Creates 512 cubes by looping through 512 times and instantiating each cube from the sampleCube prefab. Setting the position to the object this script is attached to, parent and name.

Using pi to calculate rotation and position of where the cube should be placed so that it forms a circle. Set pos.

private void ScaleUpdate()

Adjusting the height of each cube depending on the sample or sampleBuffer depending on whether useBuffer is true or not.

DynamicLight.cs (Example Visualization)

This script changes the intensity of a light depending on the amplitude of a specific audioBand.

Parameters

- minIntensity, maxIntensity – Clamping light intensity.
- audioBand – Set which audio band you would like this light to react to. Please see [1]
- colourLerp – Set to true if you want the light to lerp through colours as well.
- lerpSpeed – Set the speed of the lerp to be multiplied by audioBand value.
- colourMin, colourMax – Set the min intensity colour and max intensity colour for the light to lerp through.
- Light – Contains the light instance.

Functions

void Update()
Check the bandBuffer value is above 0, If so call the LightAudioIntensity function

void LightAudioIntensity()

Check if colourLerp is true, if so lerp through the colours colourMin and colourMax, using the DataManager.audioBandBuffer[_audioBand] * lerpSpeed as the parameter.

Change light intensity according to the value calculated by multiplying the audioBandBuffer by maxIntensity – minIntensity and then adding minIntensity to the sum.

[1] Audio Bands = 0-8 20-60hz 60-250hz 250-500hz 500-2000hz 2000-4000hz 4000-6000hz 6000-20000hz

GraphicEQ.cs (Example Visualization)

This script is an example of using an audioBand value to increase the .y vector of a Cube to simulate a graphic Equaliser.

Parameter

- startScale – Sets the scale you wish your object to start from.
- scaleMultiplier – Set the multiplier of the scale.
- audioBand – Set which audio band you would like this light to react to. Please see [1] above
- useBuffer – Set to true to use buffer to smooth out visuals using a buffer. If off the visuals decrease instantly if the sample is lower.

Functions

private void HeightScaleUpdate()

Adjust height of the object depending on the value in the audioBandBuffer according to the audioBand entered into this script. This is retrieved from the DataManager and is multiplied by scaleMultiplier to emphasise the visual reaction. .x and .z remain the same in this script.

ScaleOnAmplitude.cs (Example Visualization)

This script changes the scale of an object depending on the amplitude of the track playing. (This is the average intensity of all audio calculated in DataManager.GetAmplitude()).

Parameter

- startScale – Sets the scale you wish your object to start from.
- maxScale – Clamps the scale so that the object cannot scale larger than this value.
- useBuffer – Set to true to use buffer to smooth out visuals using a buffer. If off the visuals decrease instantly if the sample is lower.
- colourLerp – Set to true if you want the light to lerp through colours as well.
- lerpSpeed – Set the speed of the lerp to be multiplied by amplitude value.
- colourMin, colourMax – Set the min intensity colour and max intensity colour for the light to lerp through.
- Material – Allows us set the colour of the object this script is attached to.

Functions

private void ScaleUpdate()

Check the useBuffer boolean, and make sure that the average amplitude of the track is above 0 i.e. there is a track or sound effect playing currently. If not, return.

Adjust local scale of this GameObject according to the amplitude or amplitudeBuffer retrieved from DataManager.

If colourLerp is true lerp through the colours, again reacting to the average amplitude of the track playing.

ScaleOnBand.cs (Example Visualization)

This script changes the scale of an object depending on audio band of the track playing.

Parameter

- startScale – Sets the scale you wish your object to start from.
- maxScale – Clamps the scale so that the object cannot scale larger than this value.
- audioBand – Set which audio band you would like this light to react to. Please see [1] above
- useBuffer – Set to true to use buffer to smooth out visuals using a buffer. If off the visuals decrease instantly if the sample is lower.
- colourLerp – Set to true if you want the light to lerp through colours as well.
- lerpSpeed – Set the speed of the lerp to be multiplied by audioBand value.
- colourMin, colourMax – Set the min intensity colour and max intensity colour for the light to lerp through.
- Material – Allows us set the colour of the object this script is attached to.

Functions

private void ScaleUpdate()

Check the useBuffer boolean, and make sure that the audioBand or audioBandBuffer of the track is above 0 i.e. there is a track or sound effect playing currently. If not, return.

Adjust local scale of this GameObject according to the amplitude or audioBandBuffer or audioBand retrieved from DataManager.

If colourLerp is true lerp through the colours, again reacting to the average amplitude of the track playing.

WaveLineRender.cs (Example Visualization)

Please note this script does not require DataManager.
This script is a basic example of using a line renderer that creates and updates a waveform by using the gathered data from FMOD.
You can use this to ensure the spectrum data is being correctly gathered. If your FMOD project is correctly configured then the line will update.

See comments within script.

License Info