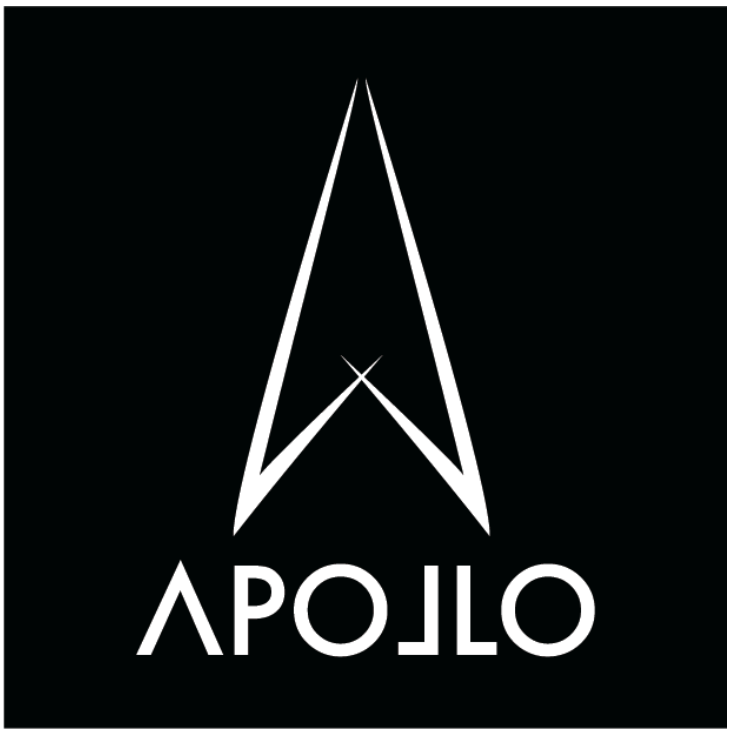


Apollo Visualizer Kit

Beta 0.2

Introduction

Apollo Visualizer Kit is a plugin tool for Unity 3D that lets the user read the intensity of 6 frequency ranges (Sub-bass, bass, low midrange, midrange, upper midrange and presence) on playing AudioSources in the scene in real time, this can be used to affect gameobjects in many ways like changing the the size, color, rotation or even emitting particles.

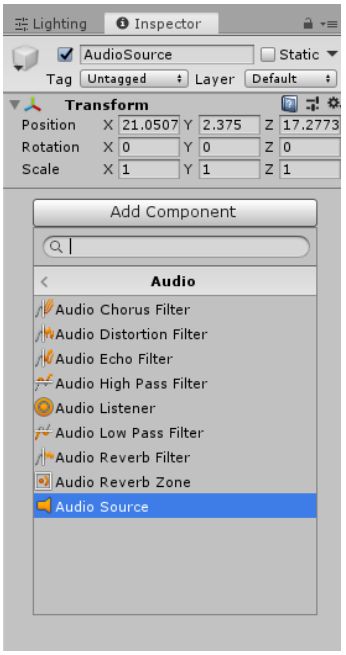
It comes with 7 built-in modifier scripts that will let you get started on making your visualizations and can serve as a starting point to build your own modifier.

You can see the online documentation here.

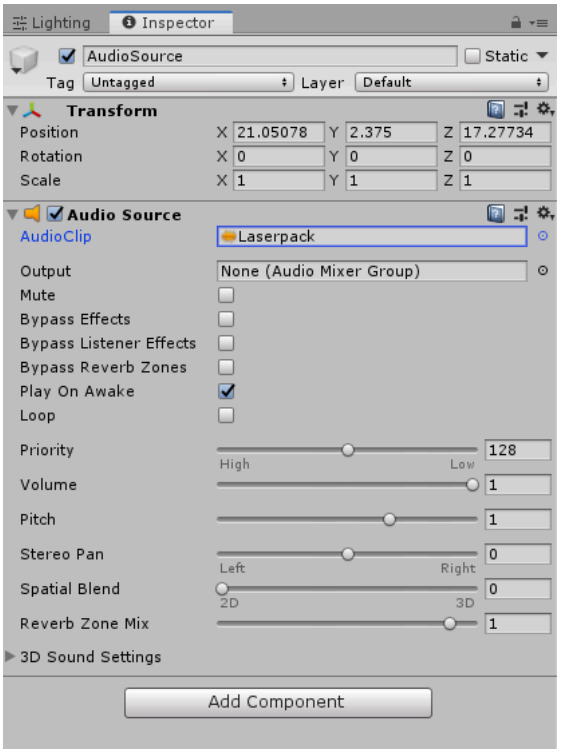
Getting Started

This is a walkthrough for you to start playing with a simple visualization.

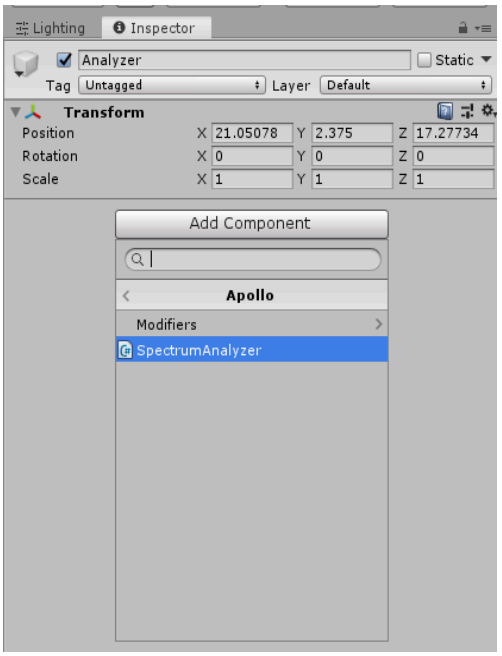
1. Open unity and start with a blank scene.(Ctrl+N)
2. Let's create a new empty GameObject and add the AudioSource (Add Component>Audio>AudioSource) component from the inspector.



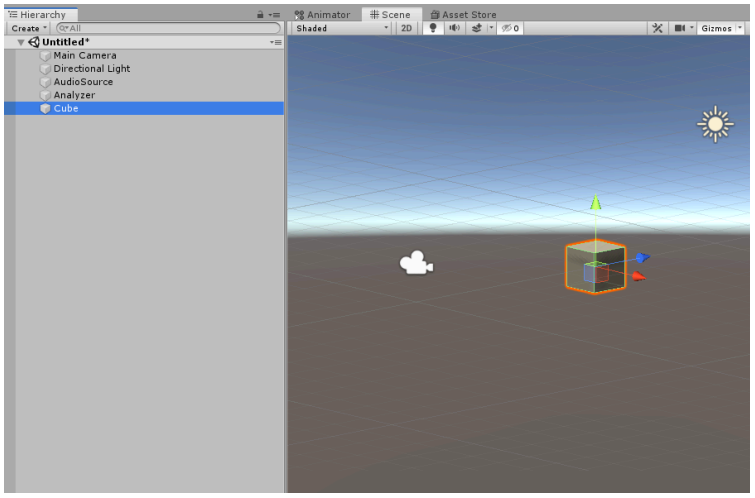
3. Now add Laserpack.mp3 included in the package to the Audio Clip field in the AudioSource component in the inspector.



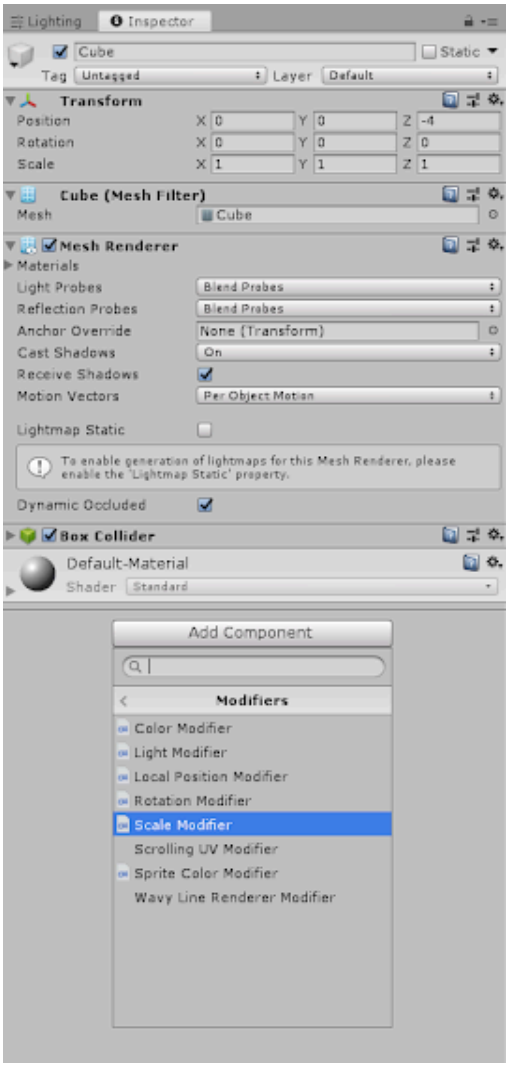
4. Create a new empty GameObject and add the SpectrumAnalizer (Add Component>Apollo>SpectrumAnalizer) component. This GameObject will be the one handling all the audio analysis.



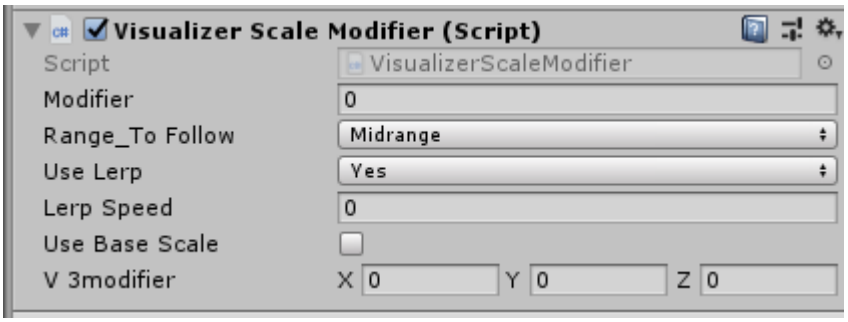
5. Now with the SpectrumAnalyzer setup we can create our objects and add modifiers to them. Let’s create a cube (Right click on the hierarchy>Create>3D Object>Cube) and position it in front of your Main Camera.



6. Now let's add a Scale Modifier component to the cube (Add Component>Apollo>Modifiers>)



7. Now let's edit the values on the ScaleModifier component in the cube. Here you will see multiple values to edit:



   a. **Modifier:** This value will let you know in real time what's the intensity of the frequency range. This will update on realtime, no need to change it.
   b. **Range to follow:** This dropdown will show all the frequency ranges available to follow. For now let's leave it on Midrange.
   c. **Use Lerp:** This checkbox is used to tell the script to use Linear interpolation and smooth how the value changes in time. Check it.
   d. **Lerp Speed:** This determines how fast the linear interpolation will work. The higher the number the snappier will be the changes. Let's make it 1.
   e. **Use Base Scale:** This checkbox lets you choose if you want the object to change size in relation to their current size or by the modifier raw value. Let's check it.
   f. **V3 Modifier:** This Vector 3 value is key in order to make the object move, it will let you choose with which intensity you want the object to change in each axis.

8. Now you're set! click play and see the cube dance.

Modifier Reference

Visualizer Local Position Modifier

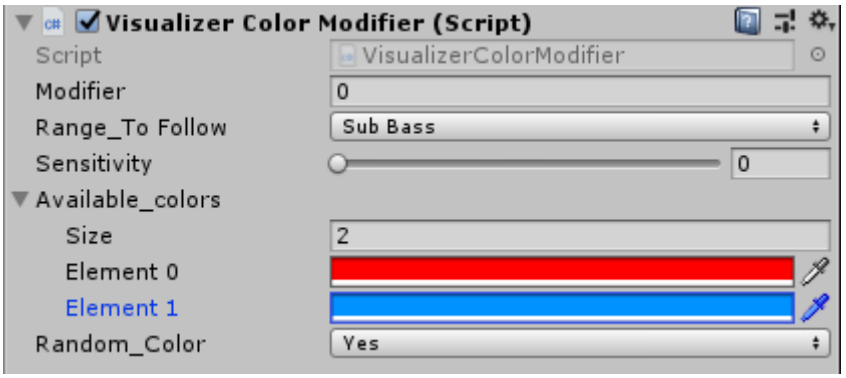
This scripts lets you change the local position of a GameObject according to their range to follow.

1. **Modifier:** This value will let you know in real time what's the intensity of the frequency range. This will update on realtime, no need to change it.

2. **Range to follow:** This dropdown will show all the frequency ranges available to follow.
3. **Use Lerp:** This checkbox is used to tell the script to use Linear interpolation and smooth how the value changes in time.
4. **Lerp Speed:** This determines how fast the linear interpolation will work. The higher the number the snappier will be the changes.
5. **Use Base position:** This checkbox lets you choose if you want the object to change size in relation to their current size or by the modifier raw value.
6. **V3 Modifier:** This Vector 3 value is key in order to make the object move, it will let you choose with which intensity you want the object to change in each axis.

Visualizer Color Modifier

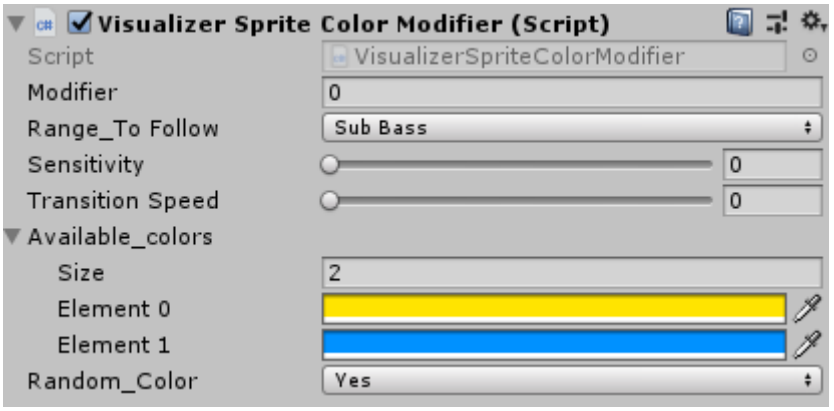
This scripts allows you to change the material.color property in your 3d object to a random color in a list or just follow it in order.



1. **Modifier:** This value will let you know in real time what's the intensity of the frequency range. This will update on realtime, no need to change it.
2. **Range to follow:** This dropdown will show all the frequency ranges available to follow.
3. **Sensitivity:** In order to mark a color change, the modifier finds differences between last modifier value and the current one. The sensitivity value lets you define how big the difference between current and last value has to be in order to trigger a color change.
4. **Available colors:** This is an array of multiple colors that you can set from the inspector.
5. **Random color:** This lets you choose whether select Available colors in order or just in a random fashion.

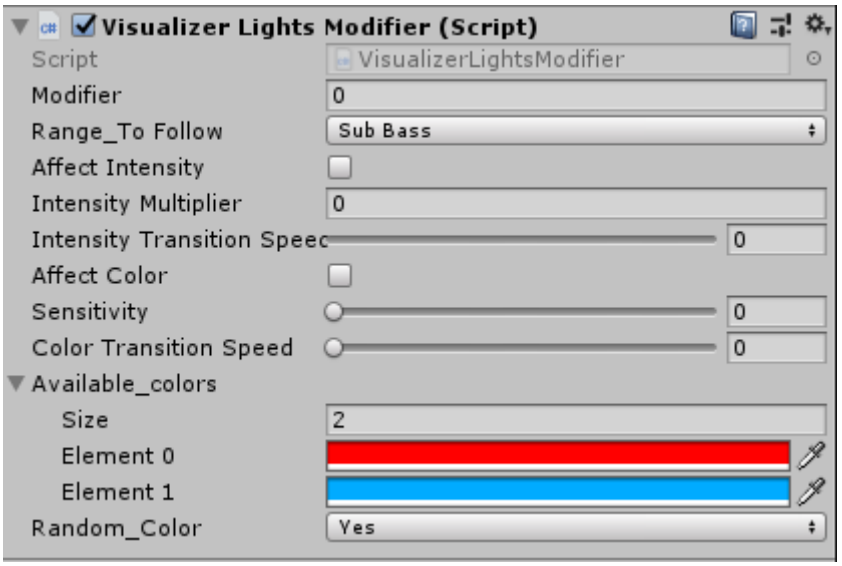
Visualizer Sprite Color Modifier

This component is made in order to control the color of a 2D sprite renderer according to the range to follow.



1. **Modifier:** This value will let you know in real time what's the intensity of the frequency range. This will update on real-time, no need to change it.
2. **Range to follow:** This dropdown will show all the frequency ranges available to follow.
3. **Sensitivity:** In order to mark a color change, the modifier finds differences between last modifier value and the current one. The sensitivity value lets you define how big the difference between current and last value has to be in order to trigger a color change.
4. **Transition speed:** This determines how fast the color changes will execute. The higher the number the snappier will be the changes.
5. **Available colors:** This is an array of multiple colors that you can set from the inspector.
6. **Random color:** This lets you choose whether select Available colors in order or just in a random fashion.

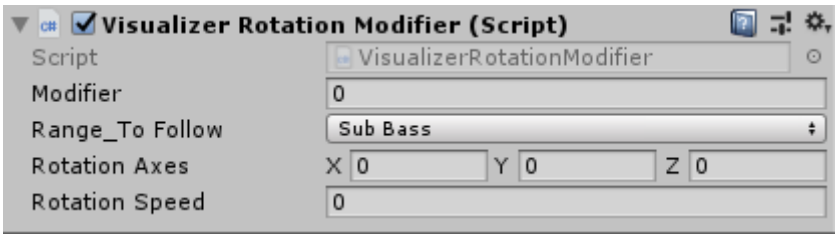
Visualizer Lights Modifier



1. **Modifier:** This value will let you know in real time what's the intensity of the frequency range. This will update on realtime, no need to change it.
2. **Range to follow:** This dropdown will show all the frequency ranges available to follow.
3. **Affect intensity:** This option turns the intensity change on the object.
4. **Intensity multiplier:** This value is multiplied by the modifier value and added to the intensity of the light.
5. **Intensity transition speed:** This determines how fast the intensity changes will execute. The higher the number the snappier will be the changes.
6. **Affect color:** This option turns on the color change on the object.
7. **Sensitivity:** In order to mark a color change, the modifier finds differences between last modifier value and the current one. The sensitivity value lets you define how big the difference between current and last value has to be in order to trigger a color change.
8. **Available colors:** This is an array of multiple colors that you can set from the inspector.
9. **Random color:** This lets you choose whether select Available colors in order or just in a random fashion.

Visualizer Rotation Modifier

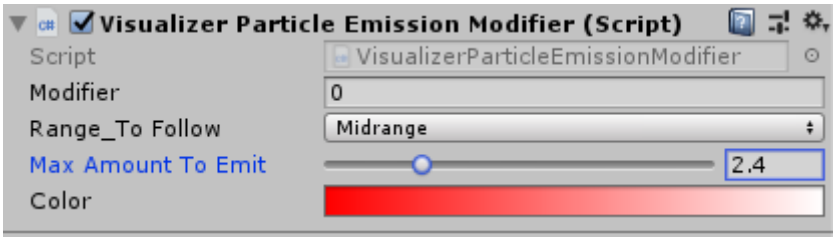
This component lets you rotate a game object according to the range to follow.



1. **Modifier:** This value will let you know in real time what's the intensity of the frequency range. This will update on real-time, no need to change it.
2. **Range to follow:** This dropdown will show all the frequency ranges available to follow.
3. **Multiplier:** This will let you choose with which intensity you want the object to change in each axis.
4. **Rotation Speed:** A multiplier for the rotation axes that determines the speed of the rotation.

Visualizer Particle Emission Modifier

This component will let you control the emission of a particle system according to the range to follow.



1. **Modifier:** This value will let you know in real time what's the intensity of the frequency range. This will update on real-time, no need to change it.
2. **Range to follow:** This dropdown will show all the frequency ranges available to follow.
3. **Max Amount to Emit:** This value will be multiplied by the modifier and will be the amount of particles the system will emit on each frame.
4. **Color:** This lets you choose a main gradient for the particle system.

Build your own modifier

In order to build your own modifiers you need to create a new script and make it inherit from **VisualizerObjectBase** and then call **EvaluateRange()** inside your Update() function , this will change the value of **Modifier** that you can use for whatever you need it.

This documentation is a work in progress and I'll be adding more guides and reference to the rest of the Modifiers. Feel free to extend the **VisualizerObjectBase** class to build your own.

Contact

Web: Kichex.itch.io
Forums: https://kichex.itch.io/apollovisualizer/community
Support email: Ekike797@gmail.com
Join our Discord server: Here