

# Rhythmator User's Guide

*for v1.0*



<b>Rhythm</b>	<b>3</b>
Markup	3
Layer	3
<b>Rhythmatior</b>	<b>4</b>
<b>Rhythm Listener</b>	<b>4</b>
<b>Rhythm Event Data</b>	<b>5</b>
<b>Editor Controls</b>	<b>10</b>
1  Timeline	10
2  Player controls	11
3  Layer Data	11
4  BPM Options	12
5  Markup Data	13
6  Toolbar	14
Selection Tool	14
Markup Tool	15
7  Markup Visualizer	17
8  Additional Tools	17
Audio Analysis Data Import	18
MIDI Data Import	18
<b>Shortcut Keys</b>	<b>20</b>

# Introduction

Rhythmator is an Unity Plugin aimed for games that needs Rhythmic actions, or actions that follows specific music patterns. You can create markups in a song timeline to fire events in your game, and execute code to handle them. There is a ton of features in the Editor to simplify and make your work faster and more organized. There are layers that can group specific markup types, BPM marks that sends events to you, visual interface that interacts with your markups within the editor, removing the need to run the game to check if the marks are in sync with the song, as also additional parameters that you can send in the events, and much more!

## Components

### Rhythm

The Rhythm object is an Scriptable Object that contains all the informations about one specific song, its layers, markups, and the BPM information if you configure the Rhythm to process BPM markups in the song.

### Markup

An information about a single hit that resides somewhere in the song, it contains a *timer* that informs where in the song it resides, expressed in seconds, and a *length* that informs how much seconds it lasts.

It is important to note that Markups can have length = 0, and for Rhythmator, that simply means the Markup lasts for just one frame, and can be used for events that behave like a single hit.

### Layer

A layer is just a group of Markups, that resides also within the song. The layer was made mainly for organization purposes, as you can use it to identify what type of Markup is being received on code, and also making the Markups less confusing when dealing with a lot of them in the Editor.

# Rhythmator

The Rhythmator is a MonoBehaviour that can be attached to any Object in a Scene. It's purpose is to synchronize the song being played with the markups in a Rhythm, and send event for the specified RhythmListeners. For it to work properly, it needs to have reference for the specified Rhythm, and also an options list of Rhythm Listeners so it can forward the hit events it receives

## Rhythm Listener

The RhythmListener is a MonoBehaviour responsible for reading the hit events passed through the Rhythmator. It contains two methods, *RhythmEvent(RhythmEventData data)*, and *BPMEvent(RhythmEventData data)*.

The way to correctly receive events passed through Rhythmator, is to create a MonoBehaviour component that extends this Rhythm Listener, and implements the two methods cited above.

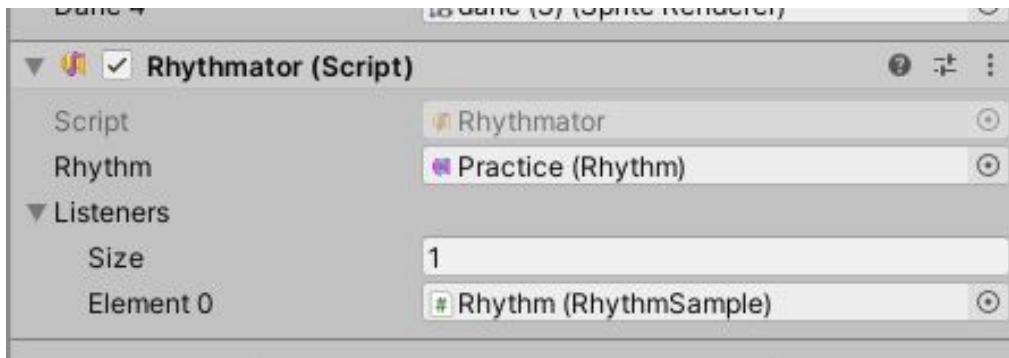
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 referências
public class Sample : RhythmListener
{
    3 referências
    public override void BPMEvent(RhythmEventData data)
    {
    }

    6 referências
    public override void RhythmEvent(RhythmEventData data)
    {
    }
}
```

And use this component in any object in the Scene.

The Rhythm Listener object then needs to be subscribed to Rhythmator's listener list, as Rhythmator Object contains a list of Rhythm Listeners that it uses to forward event information

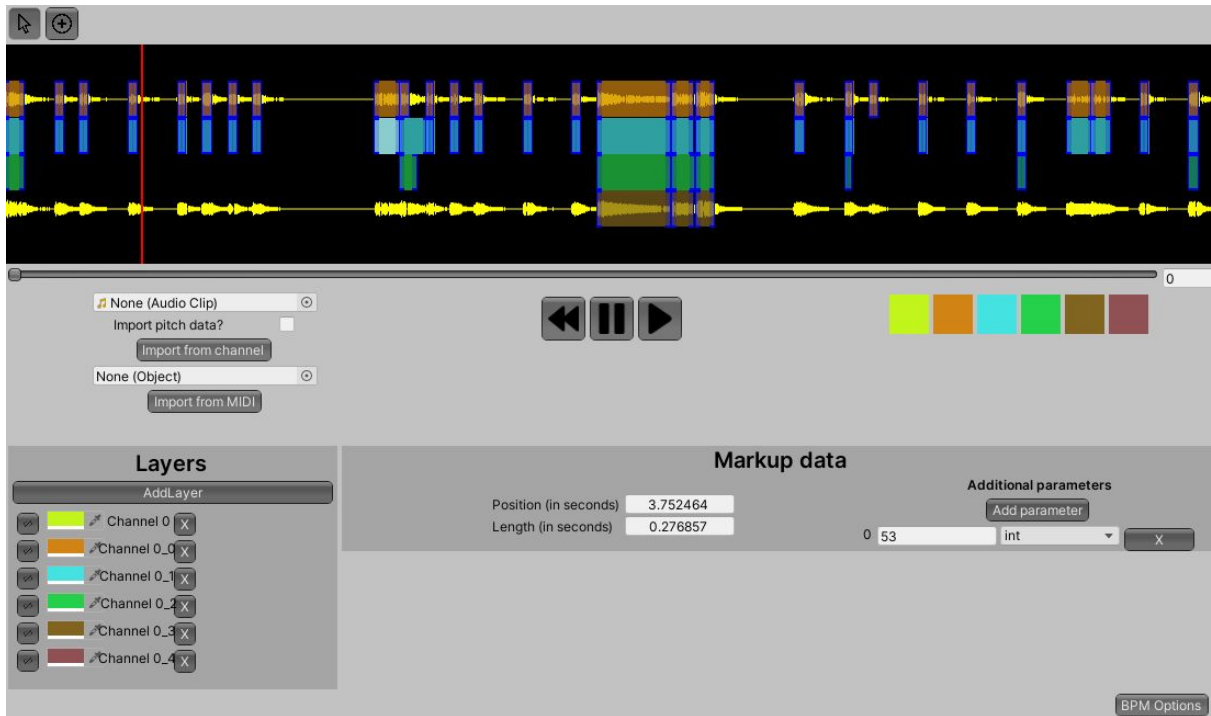


## Rhythm Event Data

This object is created in Rhythmatior when a hit is triggered, then passed to all listeners, that forward that to your specific code to handle them in any way you want. It contains all the information about that specific hit

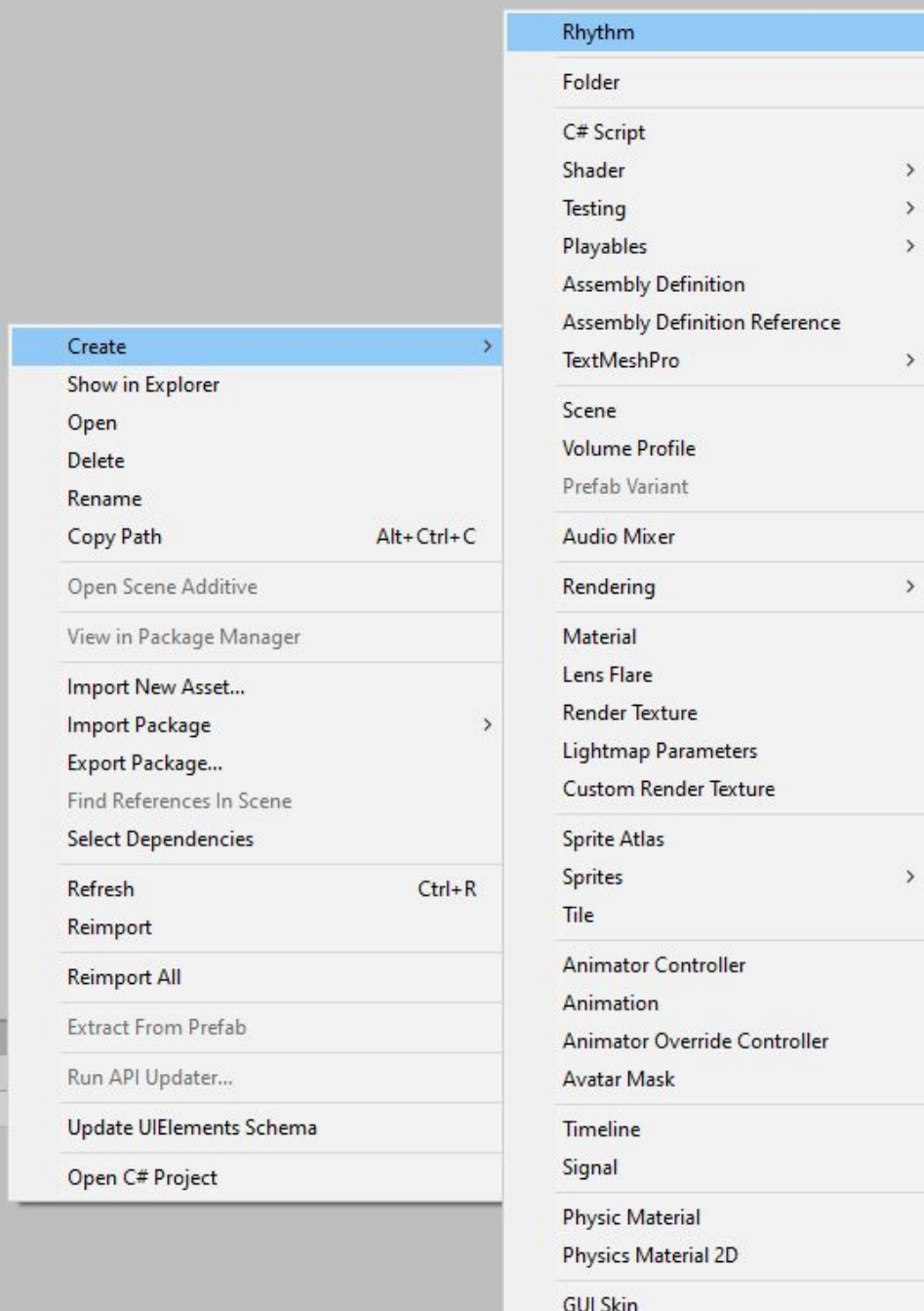
- *parent* is the Rhythm Object that contains the specified Markup triggered by this event
- *layer* is the Layer object that contains that specific Markup object
- *markup* is the Markup object that is being triggered by the event
- *type* consists of the type of the hit, being *Hit*, *Begin*, *End*, *Stay* and *BPM*
- *factor* tells you about the current progress of the hit, assuming it has a length > 0
- *objects* is the list of additional parameters informed in the Editor if it has any

# The Editor

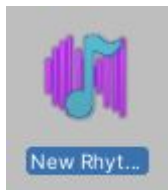


The Rhythm Editor is a powerful resource that allows you to create your own Rhythms as you mark up specific parts of the song to be triggered in the game.

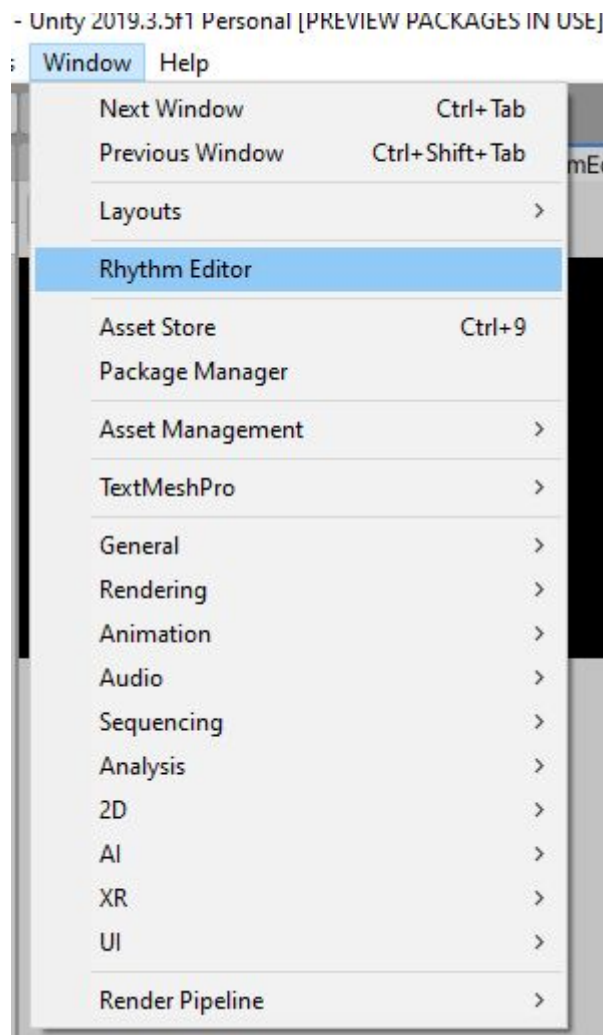
To set up and use the editor, the first thing you need is a Rhythm to be edited. It can be created by right-clicking in the explorer and *Create -> Rhythm*



A Rhythm will be created in the specified location, and will show up like this:



The next step is opening the Editor, which can be found in *Window -> Rhythm Editor*

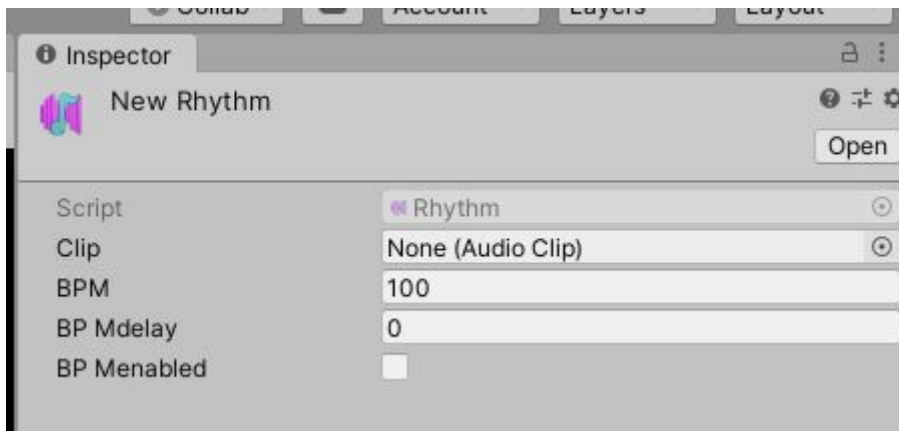


It is advised to dock this window in the same area as the *Game* and *Scene* windows as it makes the workflow easier.

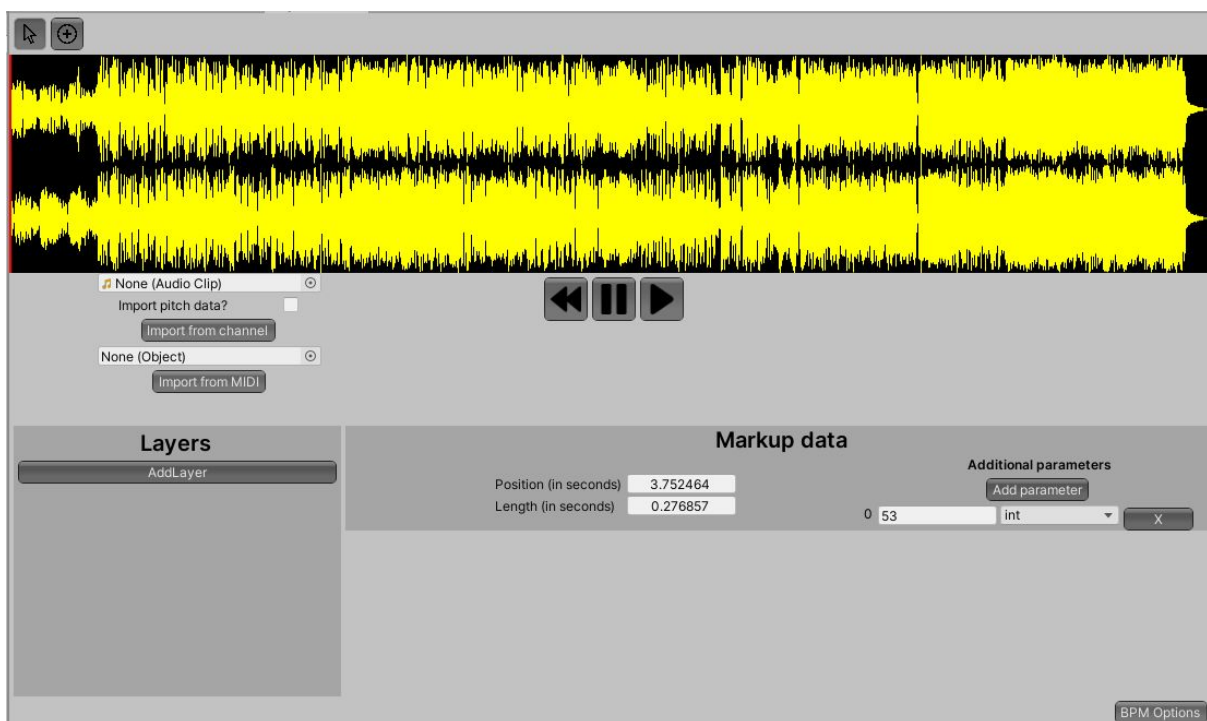
Upon opening the window, and selecting the Rhythm in the explorer, you will see a blank screen with a black bar on top, that means you haven't specified a song for the Rhythm.

You can do this by simply dragging an audio file to the *Clip* property in the Rhythm Inspector



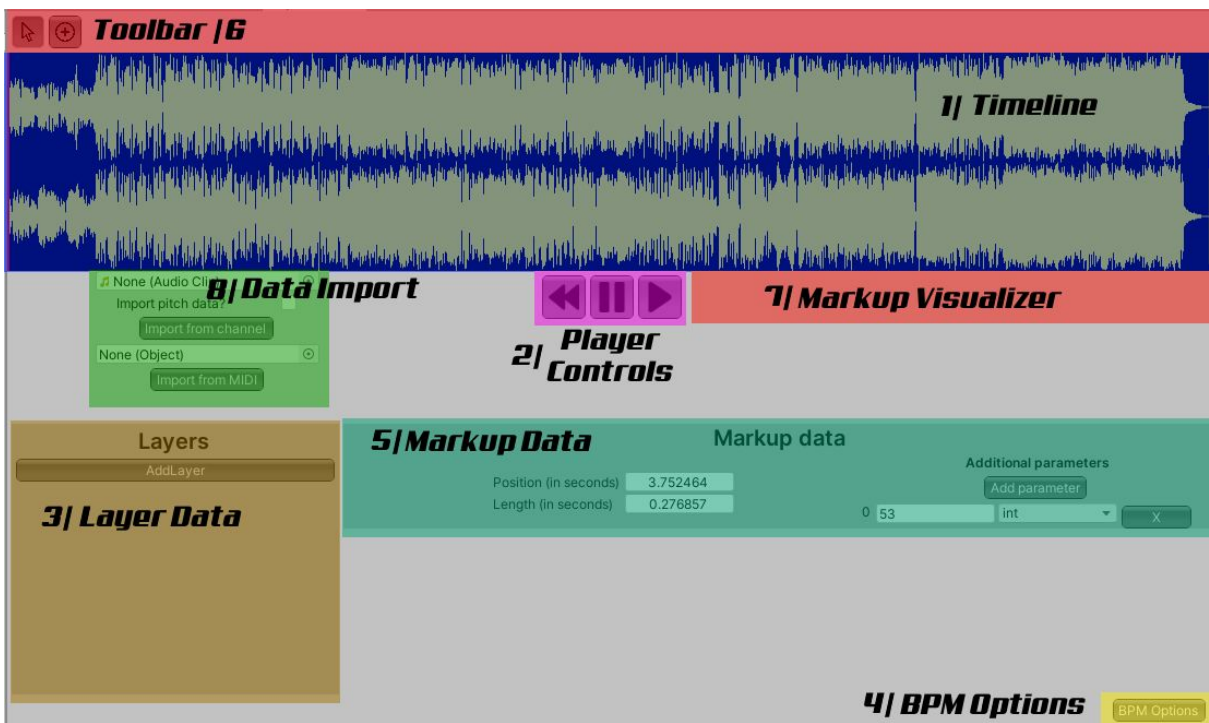


After selecting a Clip, the Editor will update showing the Song timeline



# Editor Controls

The Editor is divided in some sections, highlighted below:






## 1| Timeline

The timeline shows the current selected song waveform and can be controlled by the following commands:

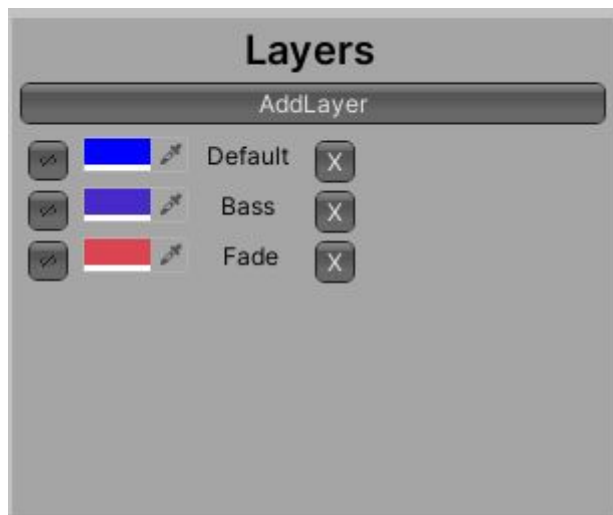
Mouse Scroll	Zooms in / out
Middle Mouse drag	Move the timeline
Ctrl + Mouse Scroll	Move the timeline
Slider change	Move the timeline

## 2| Player controls

The player controls can control how the song is played and previewed. You can control the needle position in the timeline with the right mouse button, and from that, you can use the buttons to play specific portions of the song.





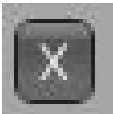
	Position the needle at the start of the song
	Pause the preview, if it is currently playing
	Plays the song starting at the needle position

## 3| Layer Data

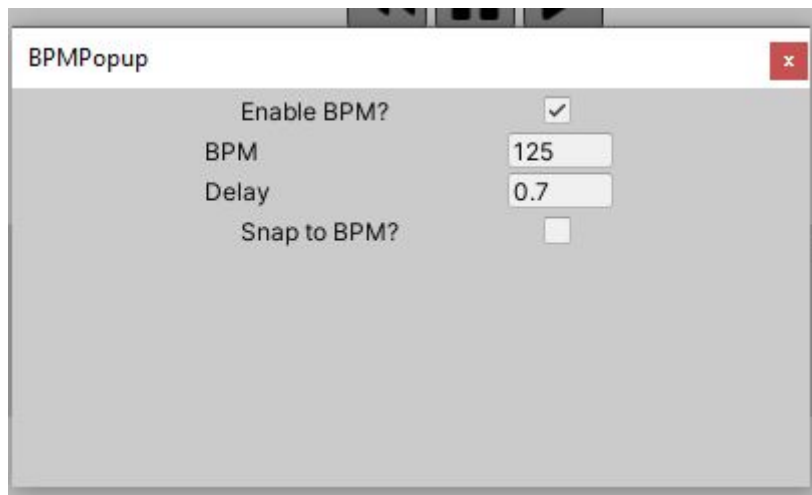


The layer data gives you control and information about the current layers in the Rhythm. To put a Markup in the song, it needs to be attributed in a layer. If you try to put a Markup in the song without having any layers configured, the editor will automatically create a new layer called "Default".

The Layer Data has a couple controls and informations:

	Creates a new layer and selects it
	Defines a color for the specific layer, that is translated as the markups color. It is used for organization purposes and has no effect in the Rhythmotor component while in Game
	The layer name
	A toggle that hides and shows the markups of this layer in timeline. Used for organization purposes and has no effect on the game. Layers that are hidden in the editor will <b>still</b> send events to the Listeners
	Removes the layer from the list, and with it, all the Markups attached to it

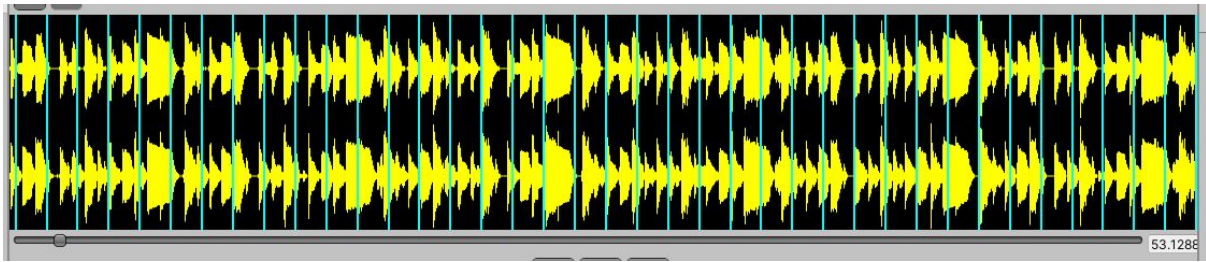
## 4| BPM Options



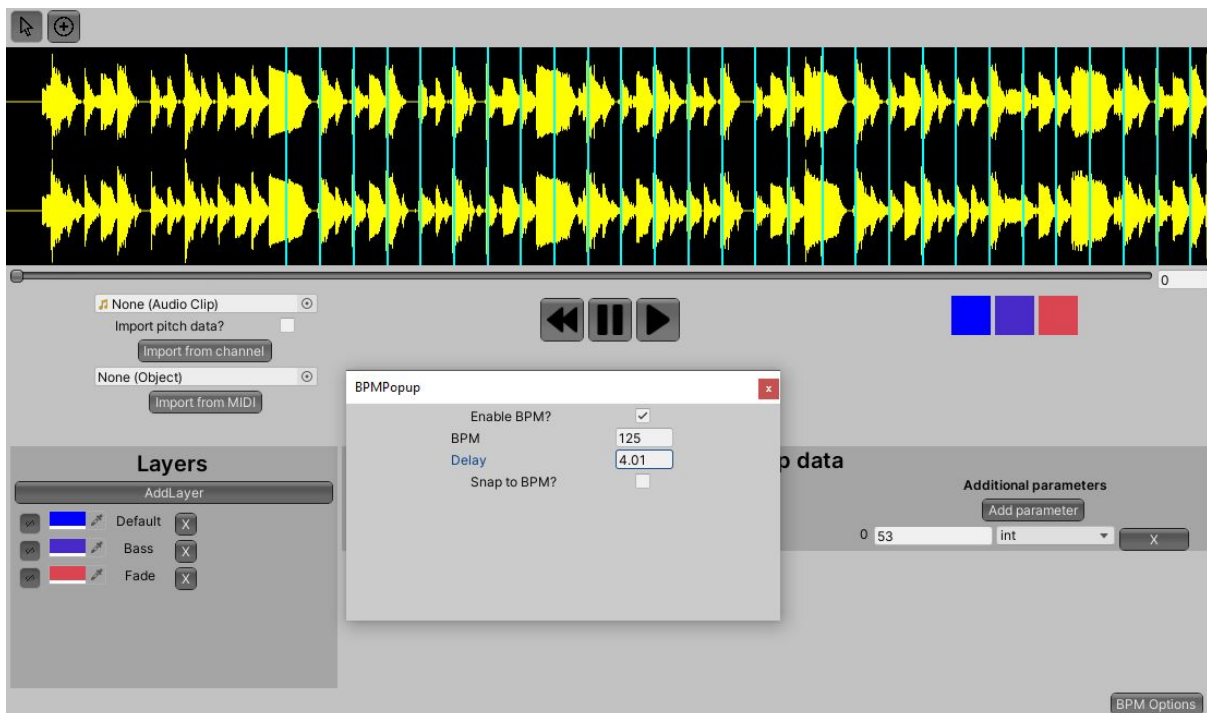
Upon clicking the BPM Options button, a Popup will show up with BPM information

The BPM Options window allows you to specify a BPM for the song, so it can receive BPM data as beat events. It needs to be enabled to use the controls of the BPM.

Upon enabling the BPM, marks will show at the timeline for that specific BPM

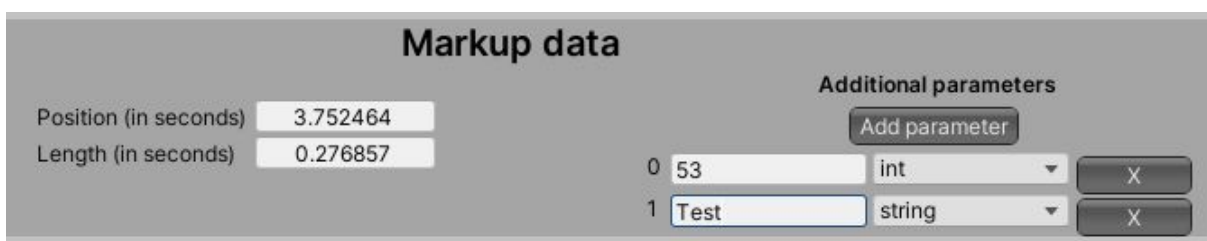


The marks can be controlled by specifying a BPM and a Delay. The delay is used to delay the count of BPM and is expressed in seconds from the start of the song



The *Snap to BPM?* Button allows you to snap you Markups to the nearest BPM, if near enough. It can be used when you want to set you Markup to near perfect BPM hits on the song

## 5| Markup Data



The Markup Data tab gives you information about the selected Markup in the timeline, and it has some sections:

Position (in seconds) specify the position of the markup expressed in seconds in the timeline, and can be changed from within this area

Length (in seconds) specify the duration of the markup in seconds. If this is 0, that the Markup is a Hit type markup

Additional Parameters is a section where you can specify additional data that can be sent within this specific Markup. Each Markup has its own Additional Parameters list, and the objects that can be used as Additional Parameters are of type:

- int
- float
- string
- bool

At the left most part of the additional data row, a number can be seen. This number is the index of the additional data, and can be retrieved in the *RhythmEventData* by the field *RhythmEventData.objects[index]*

## 6| Toolbar

The toolbar has currently two tools available:



### Selection Tool



The selection tool is used to select, drag and modify the Markups already in the timeline and has these controls:

Left click Markup	Selects the Markup
Left click outside any Markup	Unselects all Markups
Left click + Drag center of Markup	Moves all the selected markups in time relative to the dragged markup
Left click + Drag end / start of Markup (highlighted in blue)	Changes the dragged markup length. Note that this can only be achieved if a markup has length > 0
Left click + Drag outside any markup	Creates a selection rectangle that selects all markups within its bounds
Ctrl + Left Click Markup	Adds the clicked Markup to the selected Markups

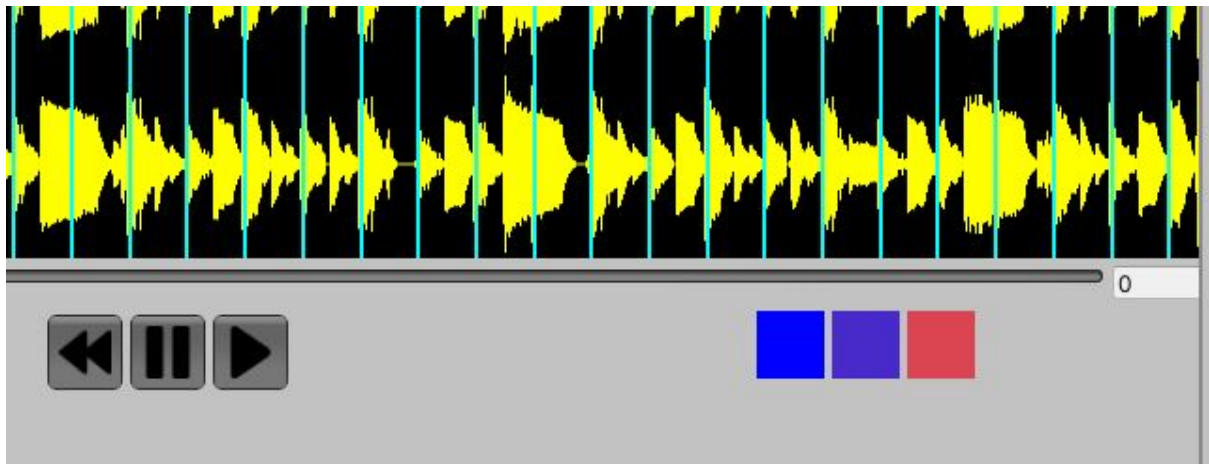
## Markup Tool



The Markup tool is used to create and place Markups in the timeline and has these controls:

Left Click in the timeline	Creates a Hit Type Markup in the specified location. The layer that the markup is attached is based on the vertical position of the mouse click, aligned to the Markups of the same layer
Left Click + Drag the timeline	Creates a Markup with a length defined by how much you have dragged the mouse. The layer rule also applies here as the point of initial click
Ctrl + Left click in Markup	Deletes the clicked markup. This also works by dragging the mouse

## 7| Markup Visualizer



The Markup Visualizer provides visual feedback of the current markups configured in the timeline.

For each layer created, a square with the layer color will show at the Right part of the screen, under the Timeline. While the music is not playing, all of them will be showing with full opacity.

When starting the song, these markups will have the target color as invisible, but upon reaching a Hit Type Mark, they will flash, providing visual feedback of how in sync your markup is, When the Hit Type is a long one, it will remain lit for the whole time the needle is still in the markup, turning off when outside.

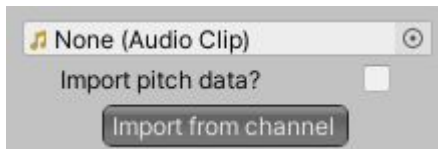
These visual detectors does not influence the Data, and exists only for testing purposes. They also are not affected by hiding a layer, as they will still flash even if the layer is not being shown

## 8| Additional Tools

Rhythmator Editor provides also some additional tools that can speed up the sync work by using Audio Analysis to automatically create markups in the song, and currently provides 2 ways of data generation:



## Audio Analysis Data Import



The Audio Analysis Data Import analyses the audio clip provided in the Object Field, and generate markups based on calculated peaks. It uses the Aubio Audio Analysis library.

Keep in mind that this data import need to be feeded an audio clip, and does **not** uses the Rhythm clip automatically. The reason why is because the Audio Analysis Data Import works better for instrument isolated clips. For example, if you have your current song split into multiple instruments, isolated into unique tracks, you can import the data from each track to automatically create markups synchronized with the instrument provided.

The Markups generated are allocated in a layer with the same name as the audio clip provided. **Note that if a layer with the Audio clip name already exists, it will override all data, deleting all the markups of that layer.**

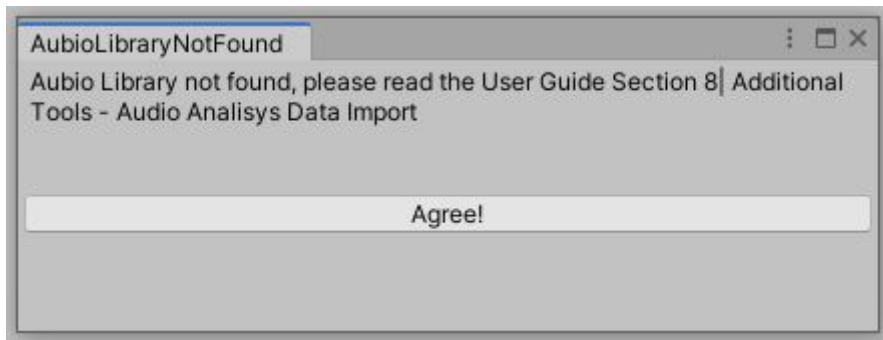
Audio Analysis Data Import can also gather pitch information, in the form of Semi-notes. These informations are packed into the generated Markup in the Additional Parameters object list, being the index 0, an Integer with the provided Note, if it could be detected. Not all note pitches can be detected, based on the audio quality. Note also that enabling pitch data can make the import less accurate, so if the audio quality is not good enough, we recommend disabling this option for markup accuracy.

By default, the Audio Analysis Data Import is **not** currently included into the package, since it depends on the Aubio external library. Since Unity Packages are not allowed to contain executable files, the user need to install manually the library:

First follow [this link](#) to download the library zipped folder. Extract the file into the folder *Rhythmator > Libraries*, so the hierarchy can be like this:

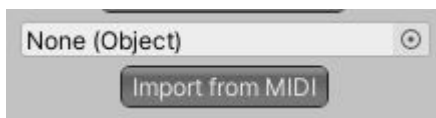


After this, you can use the Audio Analysis Data Import correctly. If the library cannot be found, it will show this pop-up error:



If after manually installing the library, and checking for the correct hierarchy, as stated in the image, the editor still shows this window, please contact our support

## MIDI Data Import

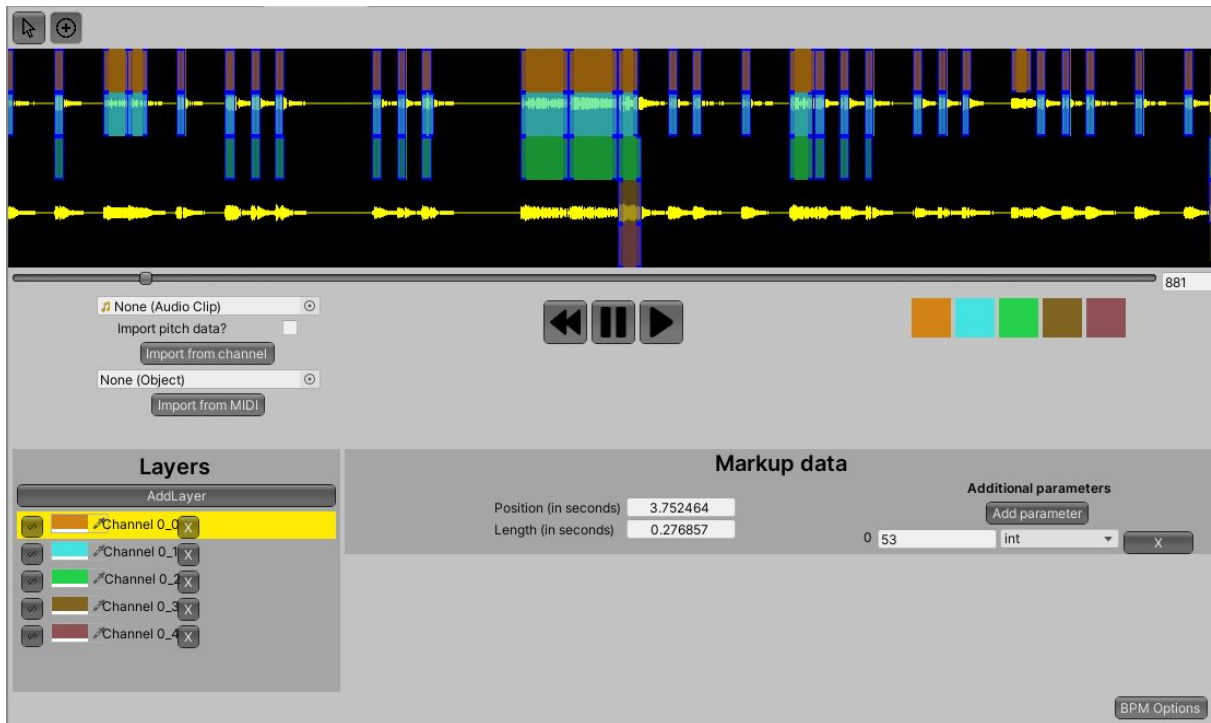


MIDI Data Import uses a MIDI file to import note information to the editor, and packages them into the following layer syntax:

*Channel [channel\_id]\_[note\_group\_index]*

The *channel\_id* is the MIDI instrument that the note is currently on, and *note\_group\_index* refers to the possible situation where there are 2 or more notes starting at the same time within the same channel. So Rhythmator can identify these, and not put 2 or more markups in the same position, it creates additional layers so all notes can be visible.

This is a MIDI Data Import generated example where there is only one channel, but some notes are chords, meaning more that one note start at a specific time:



OBS: Midi Import does not import pitch bending data, as it only imports note position, length, and pitch

# Shortcut Keys

The Rhythmator Editor contains some shortcuts to some functions:

Space Bar	Plays / Pauses the song in the Timeline
Left Arrow	Advances the Needle position in 1 (one) second
Right Arrow	Rewinds the Needle position in 1 (one) second
A	Selects the Selection Tool
D	Selects the Markup Tool
Ctrl + C	Copy the selected Markups
Ctrl + V	Paste the selected Markups in the current needle position
Ctrl + Z	Undos the last action
Ctrl + Y	Redo the last action
F2	If there is a selected layer, renames it