Tazman-Audio

Fabric Manual

FABRIC

Table of contents

Introduction 4
Installation 5
Getting started 6
Fabric audio hierarchy 6
FabricManager 7
Components 8
Component properties 9
Propagation of component properties 10
Audio Component 14
Dialog Audio Component 15
Timeline component 16
Random Component 16
Sequence Component 17
IntroLoopOutro Component 17
Switch Component 18
Blend Component 18
Group Component 19
Volume Meter Component 20
Loudness Meter Component 21
Side chain Component 22
DSP Components 23
Dynamic Mixer 25
Events 28
EventManager 28
Listener 29
Trigger 30
Debug log 31
AssetLoader 31
Fabric windows 32

Fabric v2.1.7

Timeline 33
Runtime parameter (RTP) Window 38
Languages 39
Graph View 40
Event Log 41
Mixer 42
Multi edit Audio 43
Previewer 43
ConvertSamplesToSec 44
Modular Synth Extension 45
Synth manager 45
Synth Graph 46
Synth 46
Synth window 47
Adding modules to the graph 48
Linking module pins 49
Deleting module pins 50
Modules 50
FdpToFtpConverter Tool 53
Quick walkthrough 55
Setting up the scene 55
Creating the audio hierarchy 55
Creating the events 57

Introduction

Fabric extends Unity's existing audio functionality and provides an extensive set of high level audio components that allows the creation of complex and rich audio behaviours.

Key benefits of Fabric are:

- **It allows the design of many types of audio behaviours quickly and easily.**
- **All game audio assets are located under one hierarchy making it easy to manage and locate.**
- **Quick integration with the game using an easy and intuitive event base system. Trigger audio in the editor or from code.**
- **Custom user interfaces allow to quickly iterate and focus on the audio functionality that is important.**
- **Multiple instances of the same audio behaviour can be triggered from different Game Objects.**
- **Written entirely with Unity's scripting language, doesn't require external native plug-ins and can be used on any platform Unity supports.**

Installation

To install Fabric you simple need to import the Fabric.package into your project.

The package comes with a number of DLL assemblies, documentation, a set of tutorials and the FdpToFtpConverter tool.

Getting started

Here we will outline some the key concepts and components of Fabric.

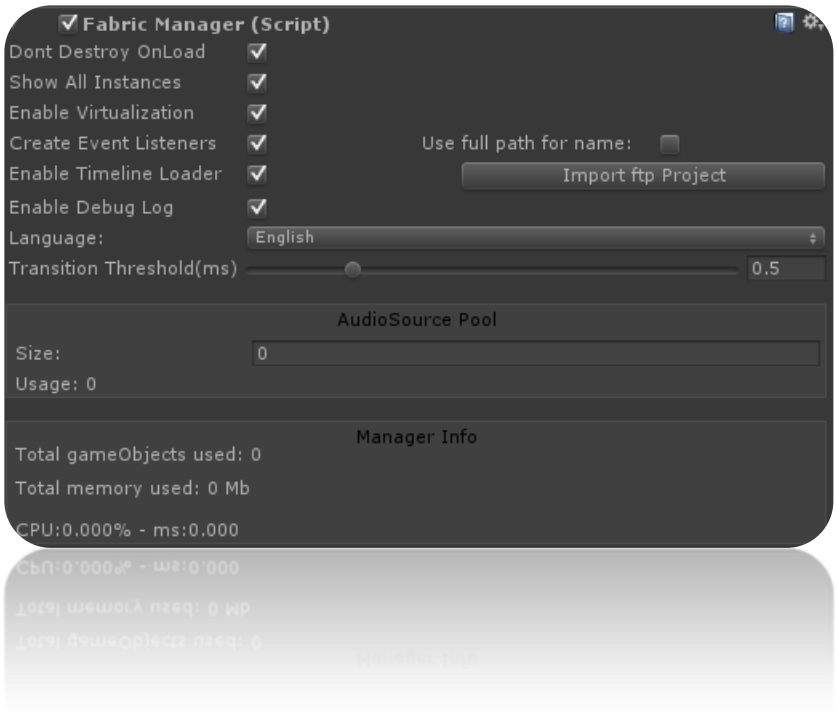
Fabric audio hierarchy

Fabric uses Unity's object based hierarchical structure in order to build complex audio behaviours.

Each game object must have a Fabric component within it in order to be able to expand the hierarchy.

FabricManager

Fabric manager component must always be at the top of the hierarchy. Its main responsibility is to manage the component hierarchy.



Audio Source Pool

The manager also provides a pool of audio sources that are used by the audio components instead of having to create one for each instance. For example if there is an audio component that has a very big number Fabric's default behaviour is to create an audio source for each instance. However, using the audio pool the number of audio sources is pre-allocated.

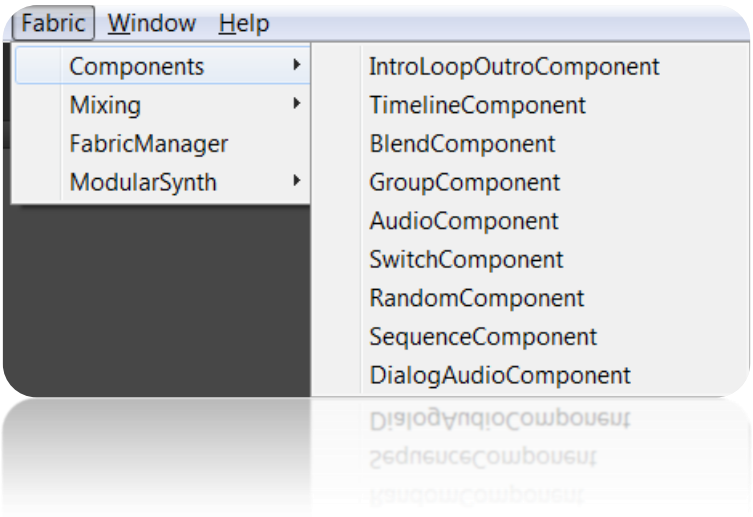
NOTE: At the moment if the maximum number of audio sources are used the pool will fail to allocate a new one.

Usage value shows the number of audio sources that are in use.

Components

Fabric components are the building blocks that allow you to create sophisticated audio designs using any combination of the following available components:

Fabric provides a menu option that allows to automatically create a new component node in the Fabric hiearchy saving time.



Component properties

All components come with a custom inspector UI that allows modifying their properties according to their position in the hierarchy.

Fabric v2.1.7

A component displays different properties according to their position in the hierarchy or if they have a listener attached.

The node info provides an overview of the component internal state and properties such as: the number of instances used/exist, volume and pitch with their randomisation offsets as well as CPU current/max usage.

Propagation of component properties

All of the component properties are propagated through their hierarchy and are either added or multiplied with their parent properties. However, it is possible to override some of the parent properties and use their own values instead.

Volume / Pitch properties:

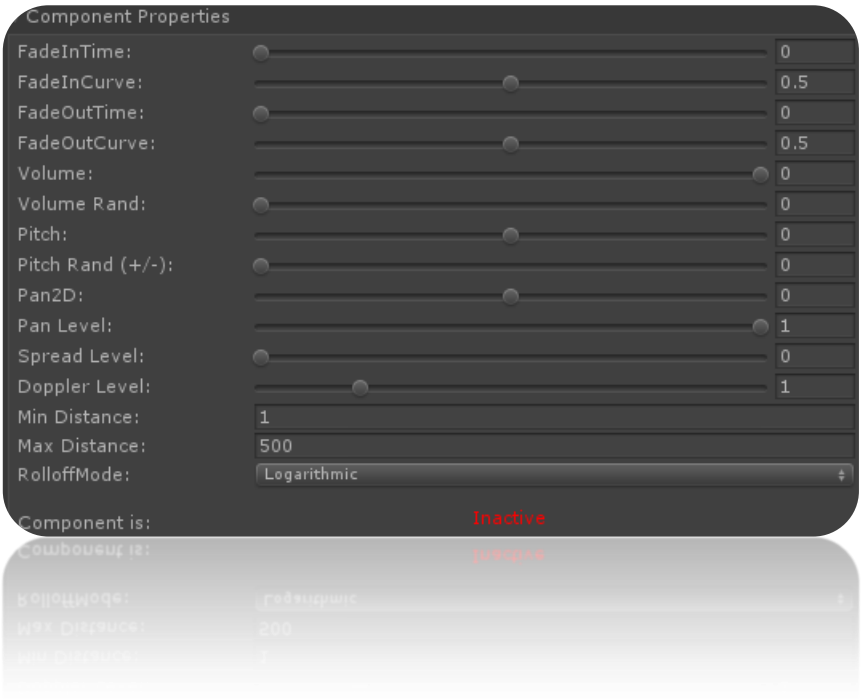
The component volume and pitch properties by default are always multiplied with its parent volume and pitch properties unless the override tick box is selected which then uses the node's properties.

3D/2D Properties:

The default behaviour of these properties is to passed down from their parent unmodified. When the override tick box is selected for either property then they become the properties that are going to be propagated down the hierarchy.

Top node:

All components that are just below the FabricManager are “top nodes” so they don’t provide any override options.

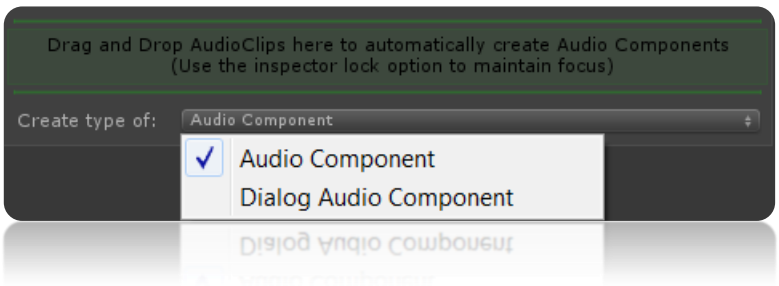


Child node:

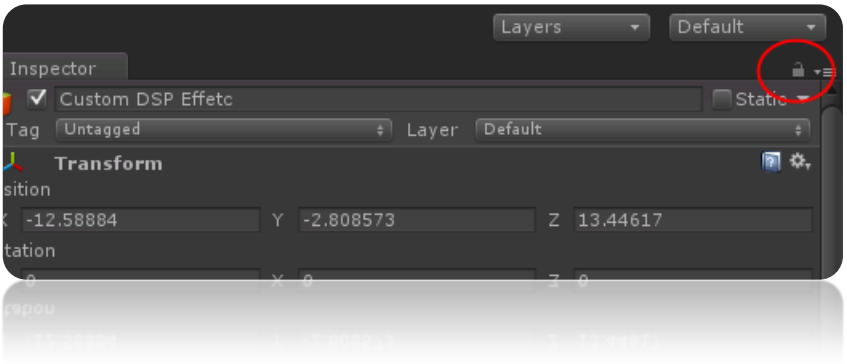
When a component is located as a child of another component it allows to override most of the parent properties.

Import Selected Audio Clips:

Fabric provides a method to quickly import a selection of audio clips as audio components or dialog audio components. This is particularly useful when it is necessary to import a large number of audio such as dialog files into a random component etc.



In order to import multiple audio clips first the inspector UI needs to be locked by pressing the small lock icon located on the top right hand corner (red circle).

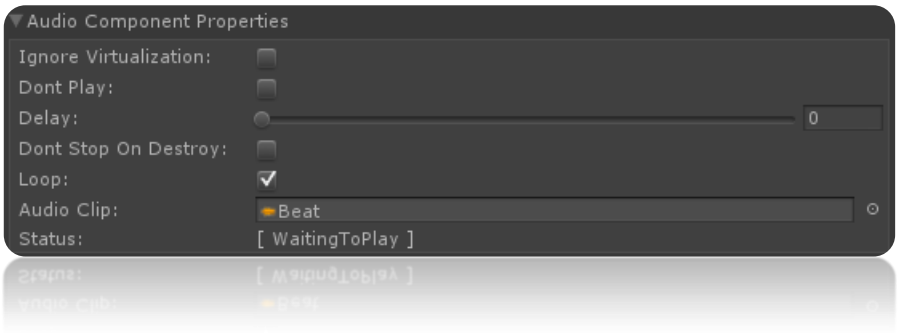


This allows the selection of multiple audio clips without the inspector view loosing focus.

The audio clips are selected in the project tab view can be simply dragged and dropped in the drop box area and for each clip an audio component will be created according to the type selected.

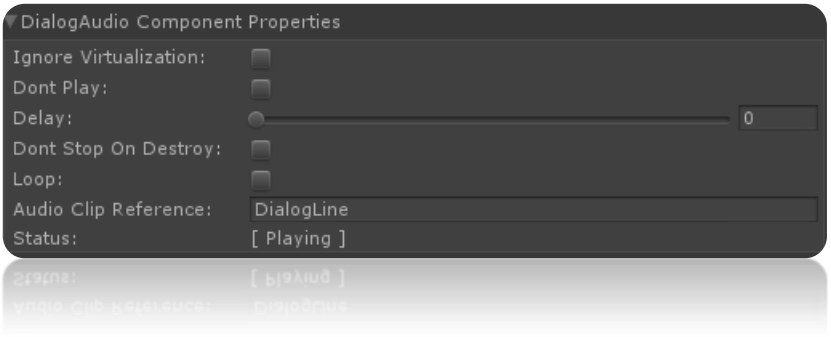
Audio Component

This is the lowest component in a hierarchy and extends Unity's existing audio source component functionality. An audio component cannot have any children components attached.



Dialog Audio Component

Dialog Audio component provides the same properties as the standard Audio Component and its using the Language properties in order to load and play an audio clip on demand. This allows to localise large number of VO in games for multiple languages easily and without having to duplicate Fabric hierarchy etc.



The audio clip will get loaded before the component is about to play and will get unloaded when is finished playing saving audio memory usage.

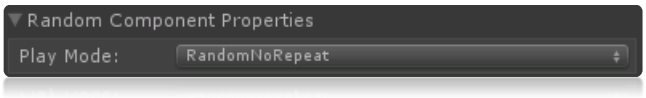
Timeline component

Timeline component provides a powerful multi-track editor functionality that allows layering and crossfading a number of sounds together. It also provides parameters driven by the game that can be linked with the timeline and volume/pitch curves.

A more detailed overview of the timeline features and how it works can be found in the timeline window section.

Random Component

This component triggers a series of child components in a random order or in a way that avoids repetition.

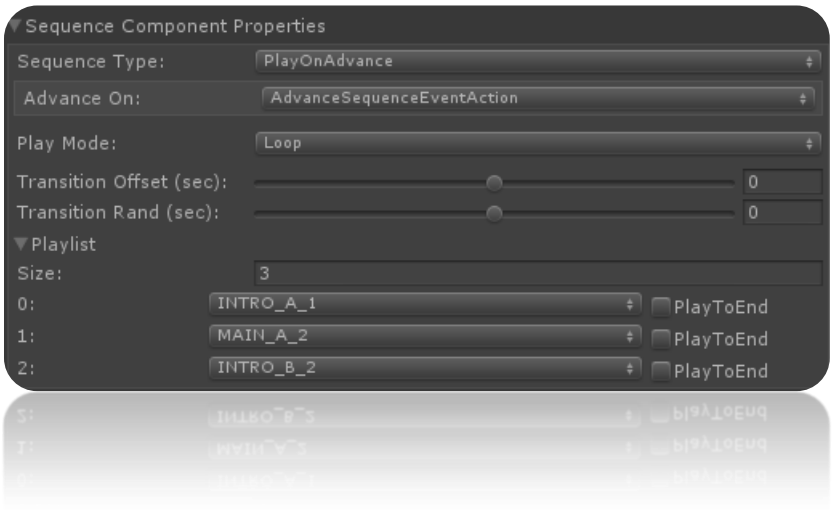


In random mode it is possible to set the weights for each child component:

Sequence Component

The sequence component triggers a series of child components defined by a playlist. It continuously advances (PlayContinuous) on the playlist in a sample-accurate way and depending on the play mode selected it will either loops back to the beginning or stop.

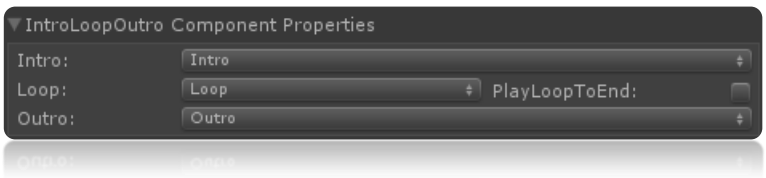
Furthermore in advance mode it is possible to configure the sequence to advance to the next entry when an “AdvanceSequence” event action is received (PlayAdvance) or on a PlaySound event action.



Finally, it is possible to set and randomize the Transition Offset to occur either before or after the end of the current track. This allows you to overlap two entries or introduce delays etc.

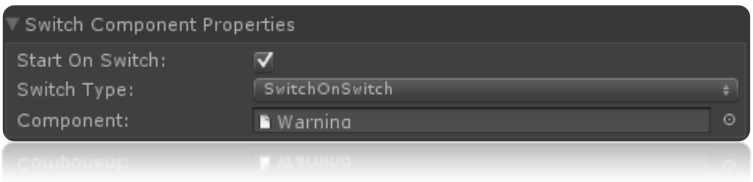
IntroLoopOutro Component

IntroLoopOutro component provides common sequence behaviour. The Intro will play first followed by the loop. When the component is stopped it will play the Outro section before is stopped



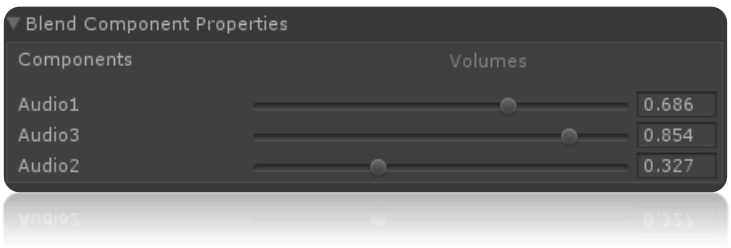
Switch Component

The switch component selects which child component to trigger from an option that is set by the game. The option is **ALWAYS** the name of the component that is required to trigger.



Blend Component

The Blend component is very similar to the group component, it triggers all its child components at the same time but it doesn't appear on the mixer view. Also through its inspector UI it is possible to set the children volumes, saving time from having to go to each node.

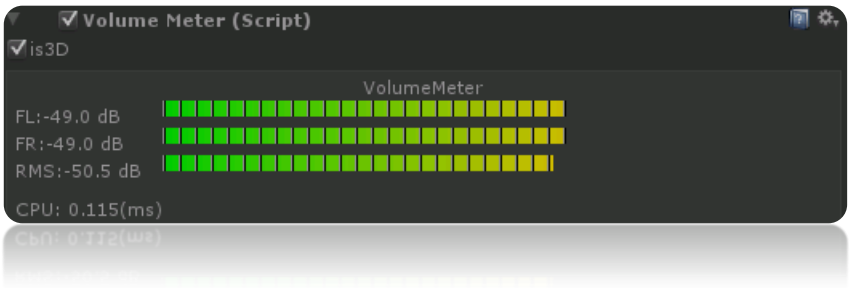


Group Component

The group component controls the volume and pitch properties of all the sounds in its hierarchy.

Volume Meter Component

The volume meter component displays the audio levels of the node in which the component is added. It will try and collect a mixed down version of ALL audio sources that exists below the node.

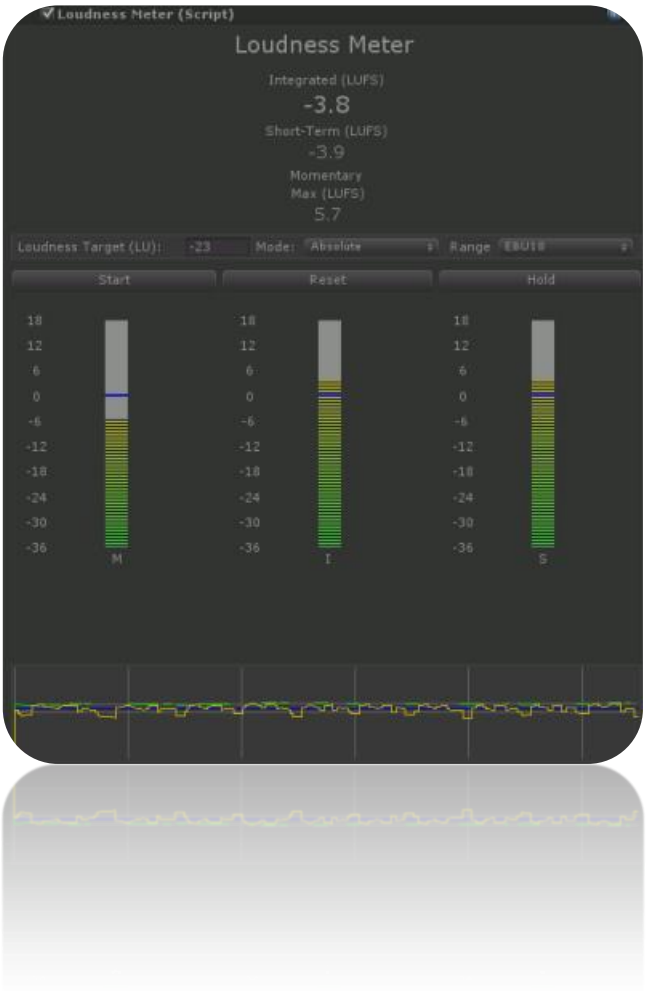


Loudness Meter Component

Loudness meter offers EBU R 128-compliant loudness metering which measures the loudness levels that listeners will perceive.

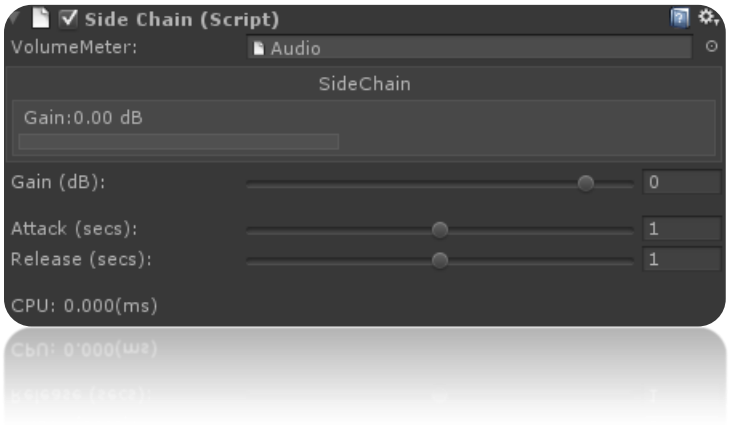
There are three standard types of loudness metering available:

- Momentary (M) loudness with an integration time of 400ms.
- Short-term (S) loudness with an integration time of 3s.
- Integrated (I) loudness which is a long-term metering with a gating function.



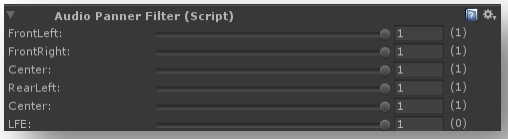
Side chain Component

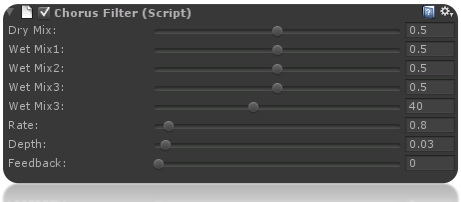
This component uses the volume meter RMS property in order to reduce the volume of the component in which is added. A typical method using a side chain is reducing the music levels during speech.

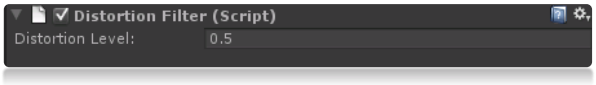


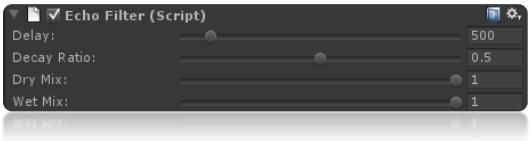
DSP Components

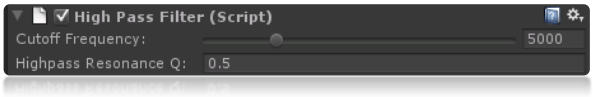
Fabric provides the same number of DSP components that Unity supports. A DSP component can be inserted anywhere in the Fabric hierarchy. At runtime it will collect all audio sources underneath that node and will add the effect on each source. This way it's possible to set an effect property at the same time for all instances either through the component inspector UI, Fabric's event system or in code using the API.

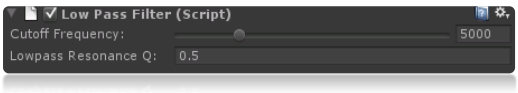


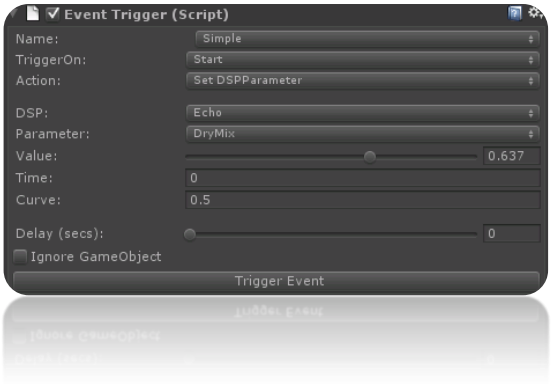








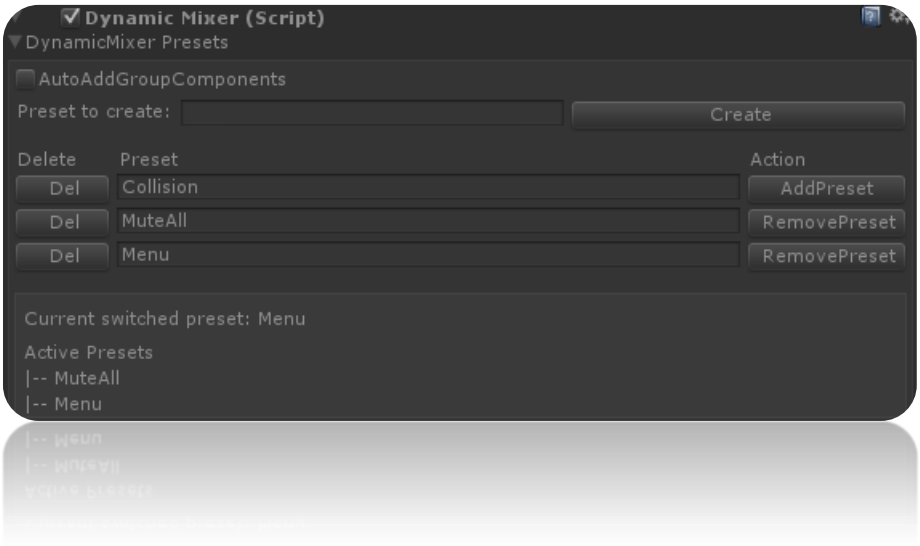



Event Trigger (Script)
Name:
Simple
TriggerOn:
Start
Action:
Set DSPParameter
DSP:
Echo
Parameter:
DryMix
Value:
0.637
Time:
0
Curve:
0.5
Delay (secs):
0
Ignore GameObject
Trigger Event

Dynamic Mixer

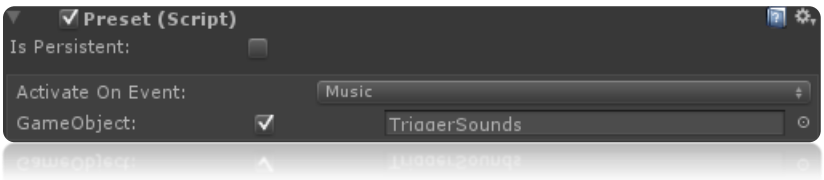
The dynamic mixer component changes the audio balance of the game at runtime. Its strength lies in the ability to add multiple presets together, each changing the volume or pitch properties of a group component. The properties from each active preset are added together to produce the final volume and pitch values that are then passed to the group components. It is also possible to switch from one preset to another thereby giving a nice, smooth transition of the parameters.

In the dynamic mixer inspector it is possible to create or delete presets that can be called up at any time either from the event trigger component or programmatically using the API.



Preset

Presets is simply a collection of group presets with a unique name. Presets can be activated at any moment either from the event trigger component or through the API.

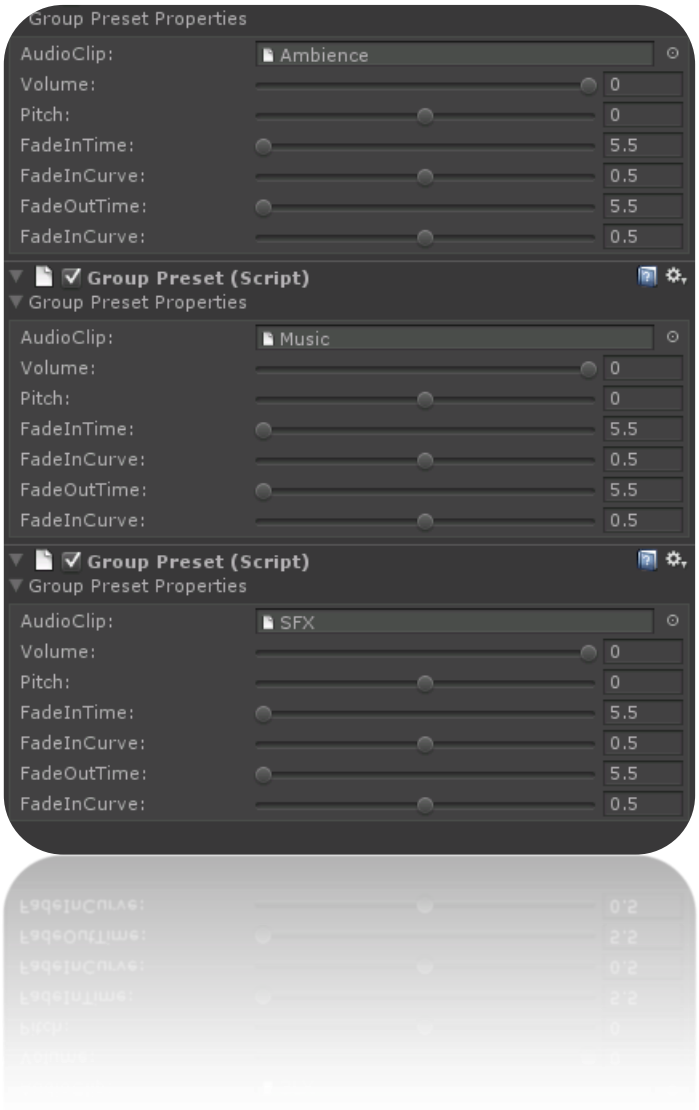


A preset can be persistent which allows it to remain active even when the dynamic mixer receives a reset event action. It can also be activated automatically when an event is posted and remain active for as long as the event is playing. When the event is no longer active the preset will be deactivated.

This is a low CPU usage ducking technique which could be used to replace side-chain when performance is an issue.

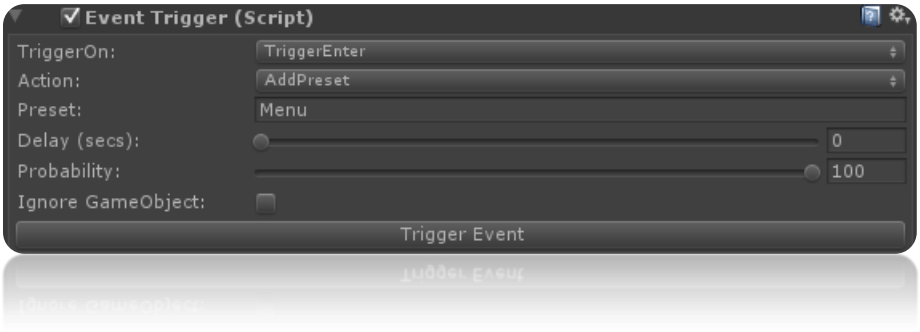
Group preset

A group preset defines properties that will make a change to an assigned group component. It is possible to change the volume, pitch and specify the duration and curve type (i.e. linear, log, exp) for each property change.



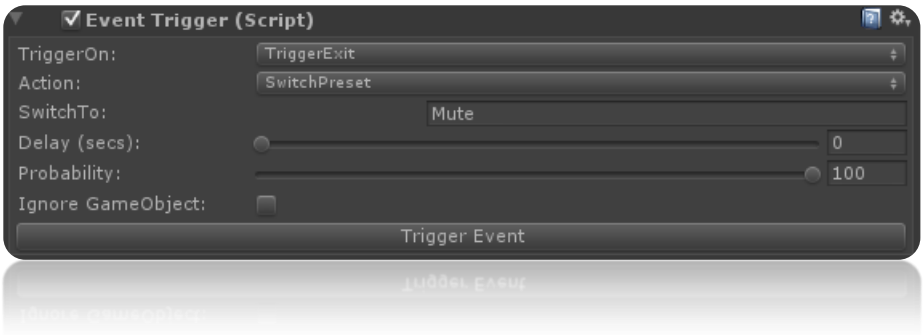
Adding presets at runtime

Adding a preset at runtime could be done either through the event trigger component or the API.



Switch preset at runtime

Sending a “SwitchPreset” event action will cause the mixer to switch from the currently active preset to the target preset.



Events

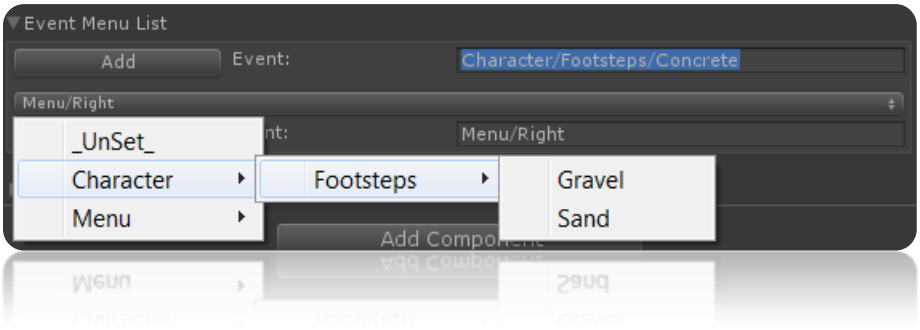
Fabric's event based system allows events to be used by the game in order to drive the audio. Each event can be set to perform an action chosen from a list such as: play, stop, set volume, set pitch, set parameters. Events can be implemented very easily in a game either in the editor or by code even before any audio authoring is done. This method allows creating and re-iterating on the audio content of the game quickly and efficiently.

EventManager

The event manager component is responsible for managing the list of events as well as the flow of events between the game and the fabric components.

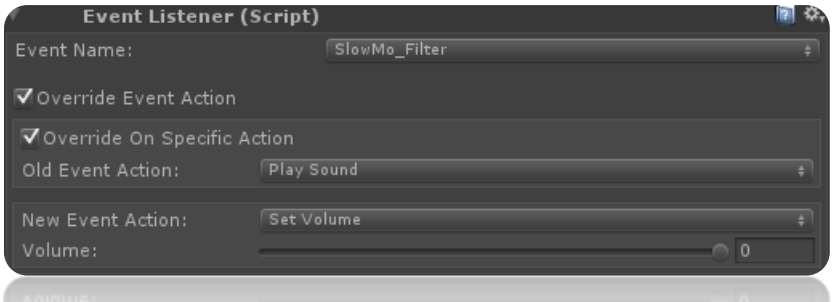
NOTE: EventManager component needs to be located in the same game object as the Fabric Manager component. Fabric will automatically add it at runtime if one is not available but it will not be possible to store the event list.

The EventManager allows the creation of event sub-menu hierarchy by using the "/" when typing an event path. So for example the event with the name "Character/Footsteps/Gravel" will be shown as a sub-menu for both the event listener and trigger menu options.



Listener

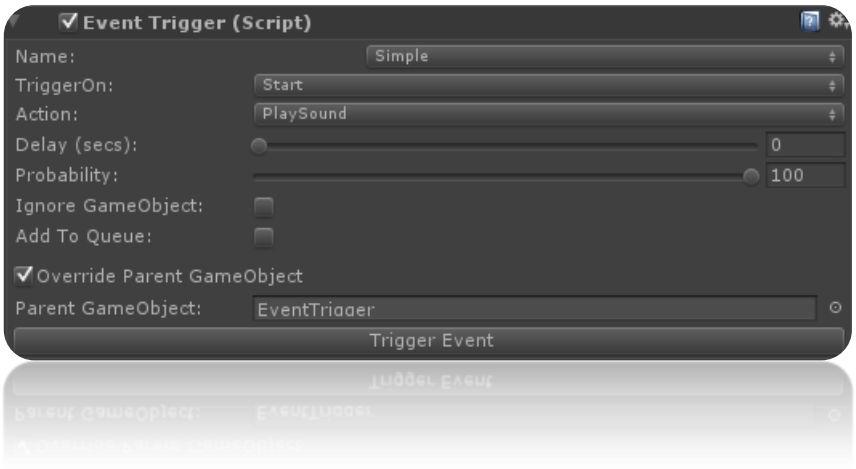
The listener simply waits and listens for a specific event name and only when it receives that name it will respond and perform an action. Furthermore, a listener can accept and set parameters if they are supported by the component.



It is possible to override any or a specific event action from the incoming event and replace it with a new one. This can be very useful since it is possible with a single event to perform multiple actions.

Trigger

The Event trigger component is responsible for sending an event into Fabric with a specific action (i.e. Play, Stop, Set Volume/Pitch) and parameter values. Every component that is listening for that event within the Fabric hierarchy will respond accordingly.



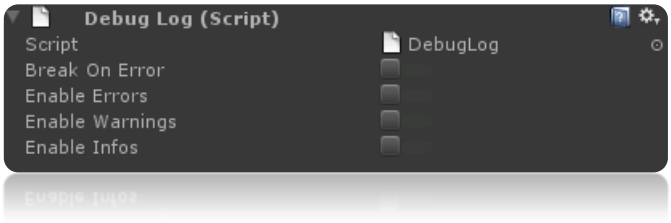
An even trigger can be set to trigger when certain game object function is called or by code through the event manager.

NOTE: It is very important to remember that when a game object is used when posting an event it is associated with a component instance. The same game object **MUST** be used in order to control the instance.

Debug log

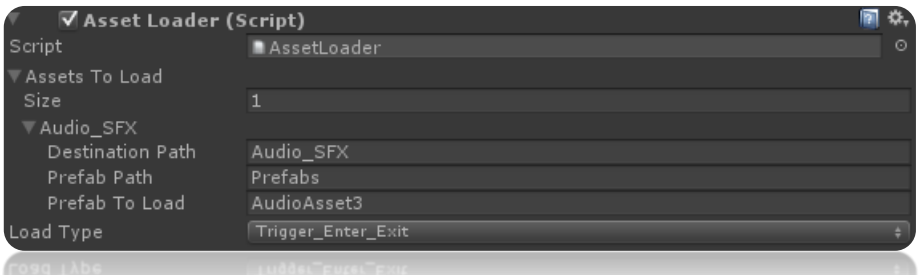
DebugLog provides the option to choose which type of messages will be displayed in the console output therefore reduces the amount of information displayed.

NOTE: The DebugLog component needs to be located in the same game object as the Fabric Manager component. Fabric will automatically add it at runtime if one is not available but it will not be possible to store it's properties.



AssetLoader

AssetLoader is a component that loads and unloads prefab assets that contain Fabric components. It is possible to define when the asset will load the list of its prefabs such as: Start/Destroy, Enable/Disable, Trigger Enter/Exit, Collision Enter/Exit.



NOTE: On the destination path each node name MUST be separated with the "_" character.

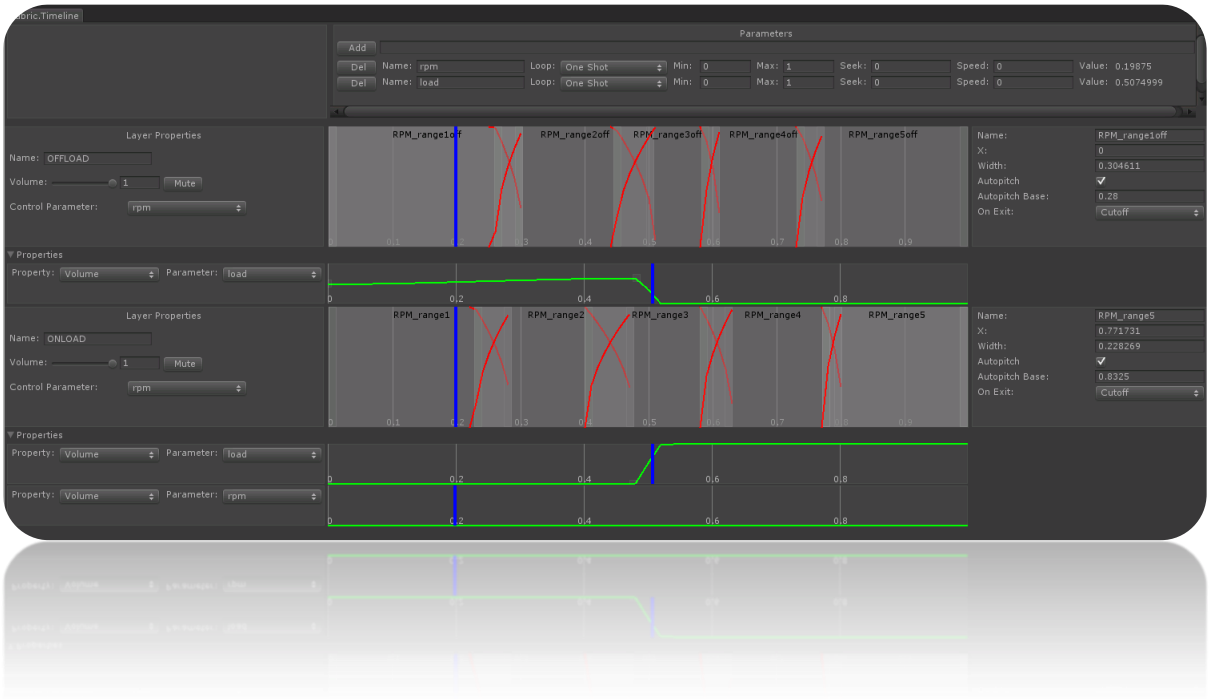
Fabric windows

Fabric provides a number of custom windows that allows quickly iterating and focusing on the audio functionality that is important.

Timeline

The timeline component is a powerful multi-track component that allows layering and crossfading a number of sounds together, linking the timeline and volume/pitch curves with game parameters.

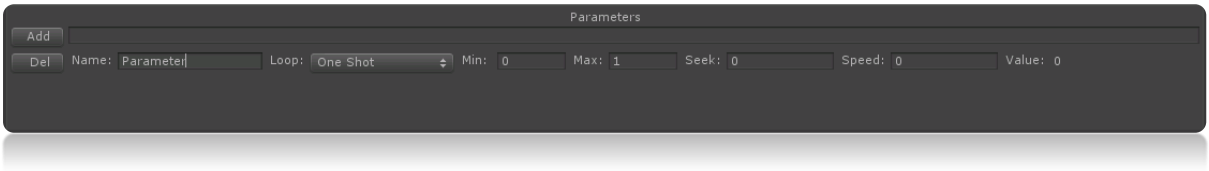
A timeline component can be added in the hierarchy by selecting the Fabric->Components->Timeline menu option.



Parameters

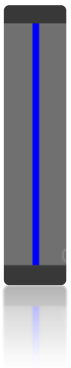
A timelime parameter provides the link between the game and the timeline internal functionality. With a parameter it is possible to trigger regions that are placed in a layer as well as control property/parameter graphs.

A timeline parameter can be controlled using the event trigger component or through the API.



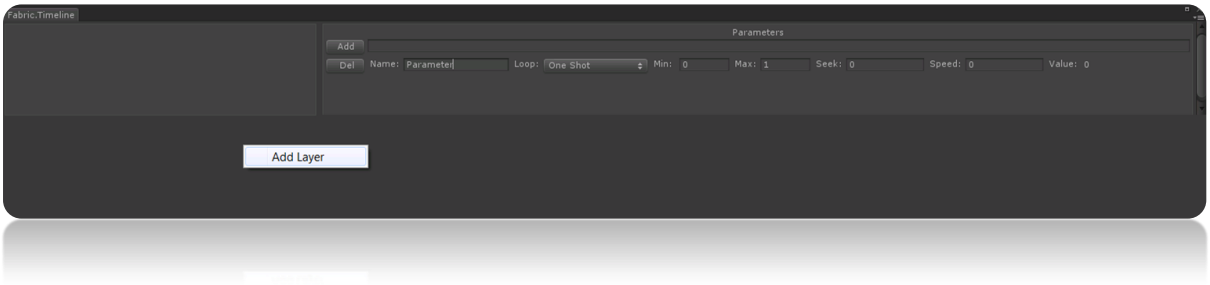
When a layer has a controlled parameter selected it displays a blue vertical bar in the timeline view. The length of the timeline is mapped to the range of the parameter.

It is possible to select the parameter bar and drag it left or right or use the slider displayed in the timeline inspector view.

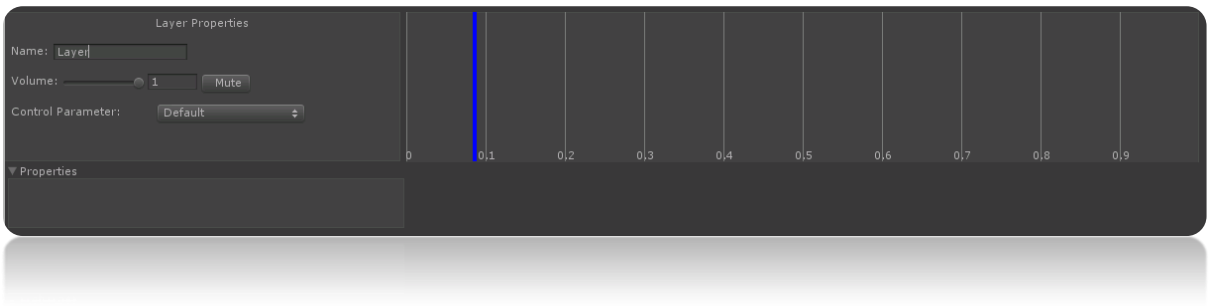


Layer

A layer is added into the timeline by right clicking into an empty region and selecting the “Add Layer” option.



A layer provides a timeline in which multiple regions can be inserted. When two or more regions overlap a crossfade is applied.

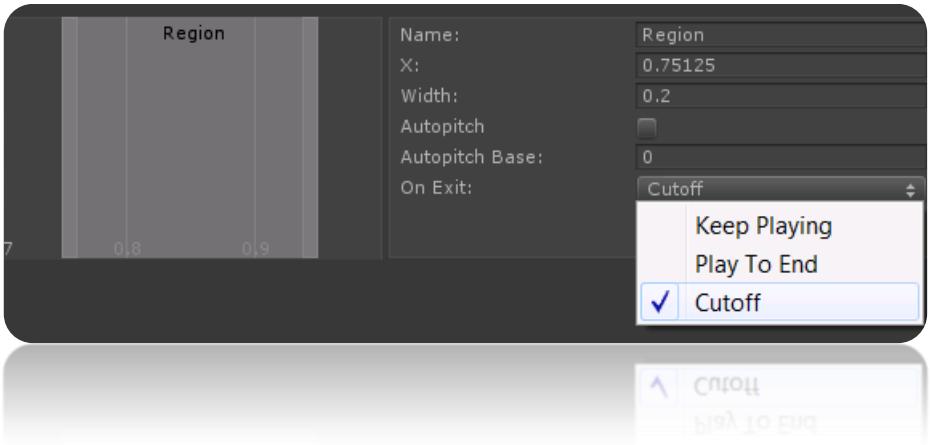


Region

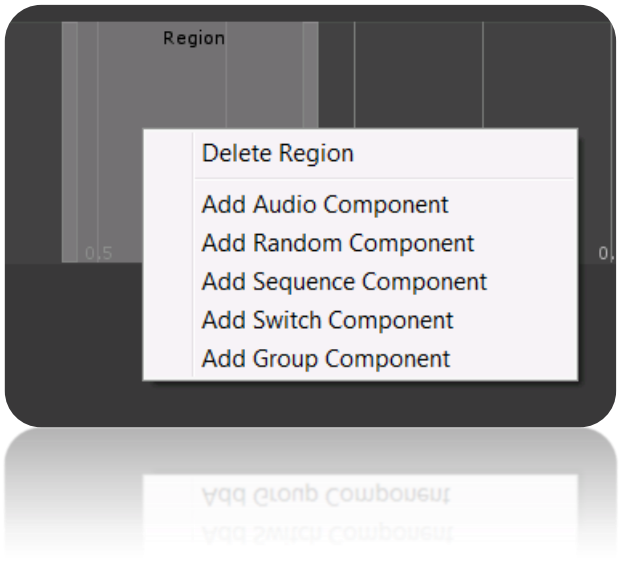
A region defines an area in the timeline in which any component will be activated when the parameter cursor is entered, and stopped, according to the “On Exit” behaviour, when is exited.

A region can be added by selecting anywhere in the layer timeline and right clicking.

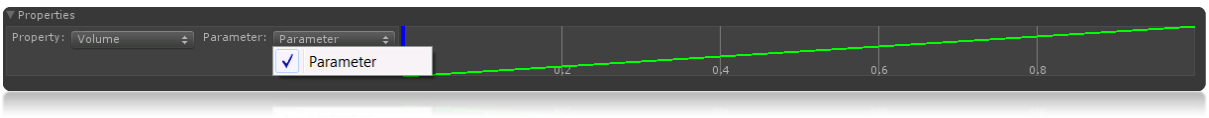
A region provides a number of properties. For a detailed description of each property please refer to the reference manual.



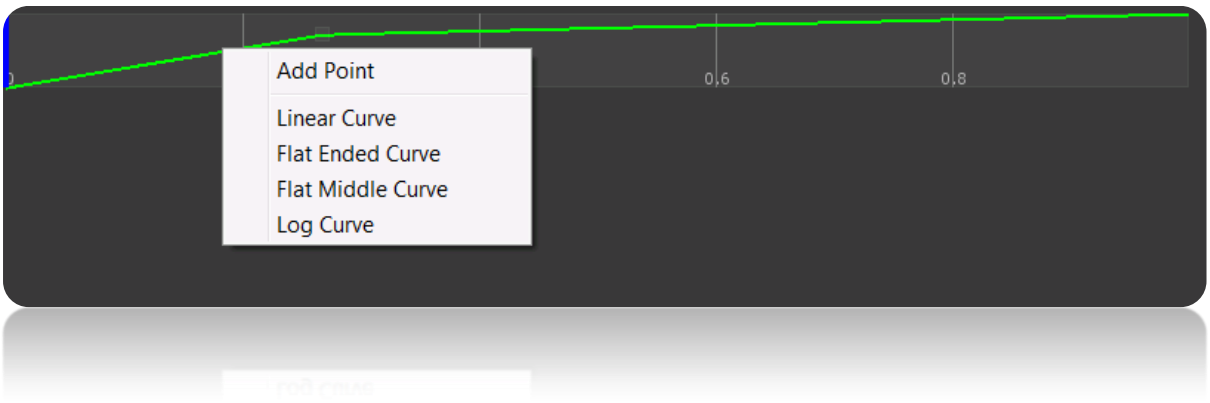
When a region is created it is possible to right click on it and add any of the Fabric components shown in the context menu.



On each layer it is possible to add property/parameter graphs that allow to link and control a layer property from a timeline parameter such as volume and pitch.



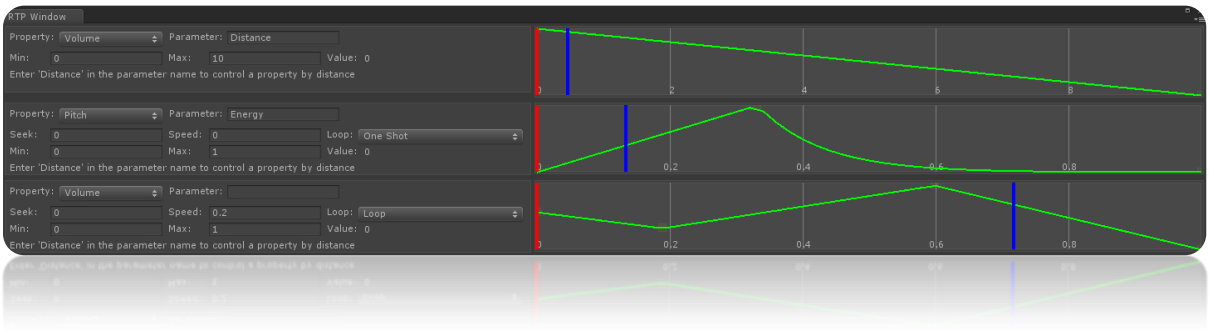
It is possible to add points in the graph by right click on the line. It is also possible to define the type of curve for the line between two points.



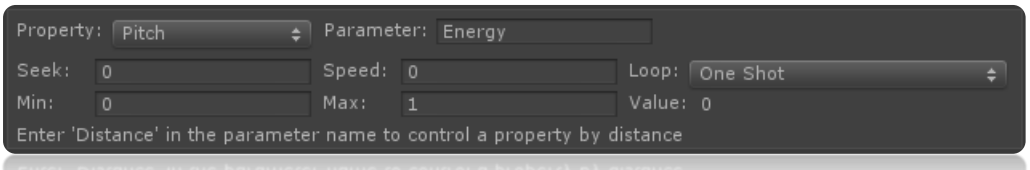
A point can be deleted by selecting on it and pressing the mouse right click.

Runtime parameter (RTP) Window

The runtime parameter (RTP) window allows you to link parameters from the game with internal component properties.

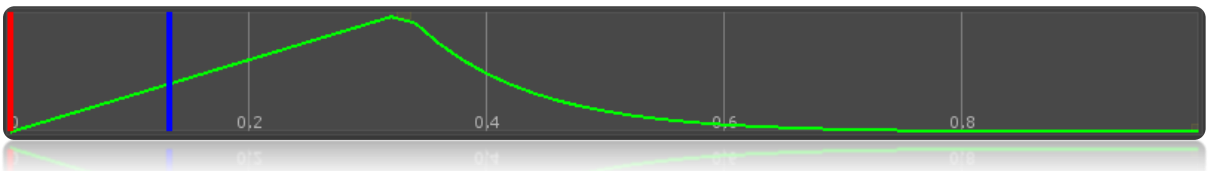


The RTP parameter area allows setting which property a parameter will control. It is possible for a parameter to have a seek value that defines the amount of time it will take to reach the target value, speed which moves it continuously in one direction and loop type that determines if the parameter that has speed will loop when it reaches the end.



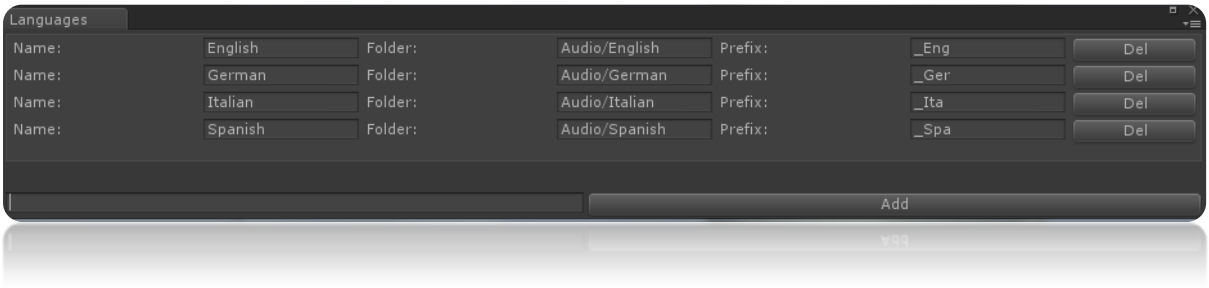
It is possible to get access to the distance value between the component and the camera if you type “Distance” in the parameter name. This way it is possible to control any of the component properties using the distance parameters.

The graph defines the mapping between the input parameter and the property that will be changed. Just like the timeline it is possible to add/delete points in a graph, change the curve line between points (i.e. linear, log, exp etc)



Languages

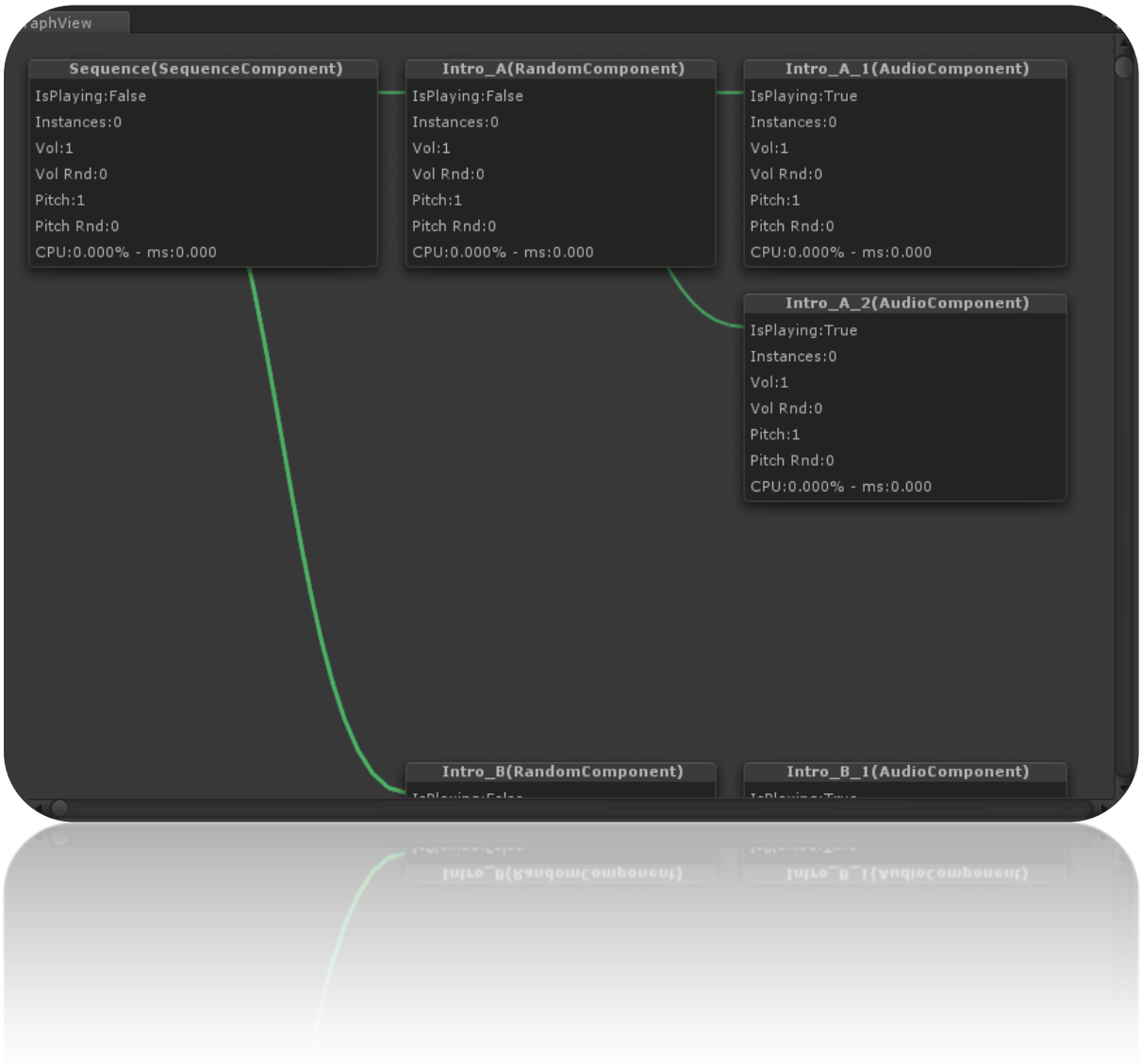
It is possible to provide localization for hundreds of dialog files by creating languages that are used by the Dialog Audio Component. The component will automatically add at the start of the audio clip reference name the currently active language folder and append at the end the language prefix.



The language currently active can be set in the Fabric manager or by code.

Graph View

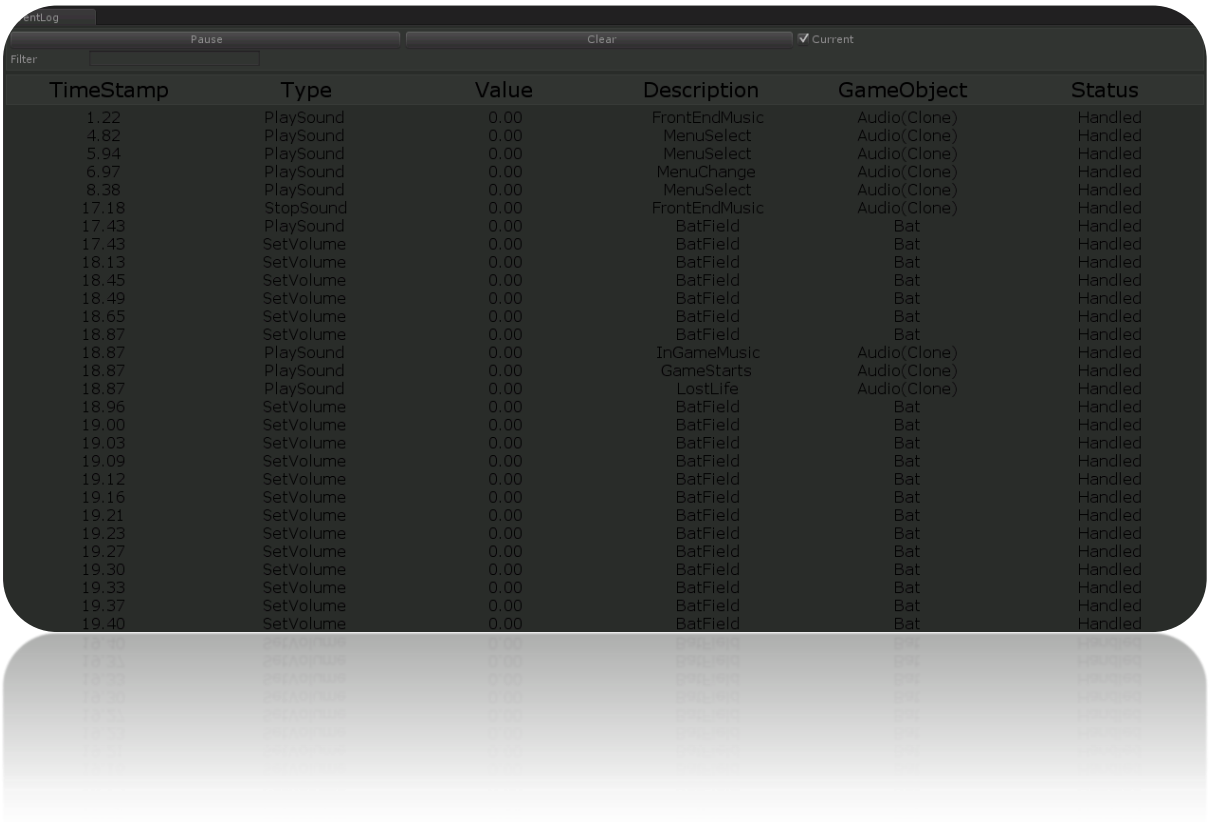
Provides a flat view of the whole component hierarchy with runtime information of their state.



Event Log

Event log view allows monitoring the flow of events and identifying potential problems or missing events.

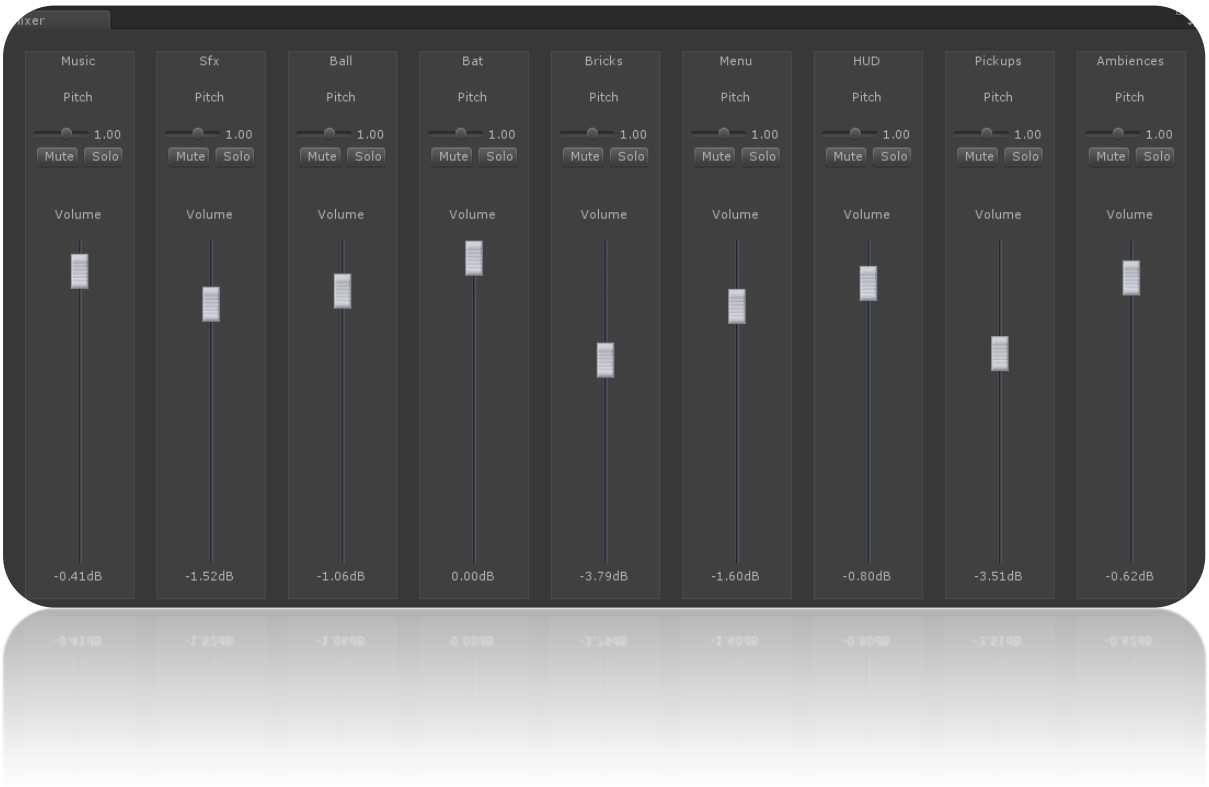
The event log view allows pausing or clearing the event log list, selecting to follow the current flow of events as well as filter event according to their description name.



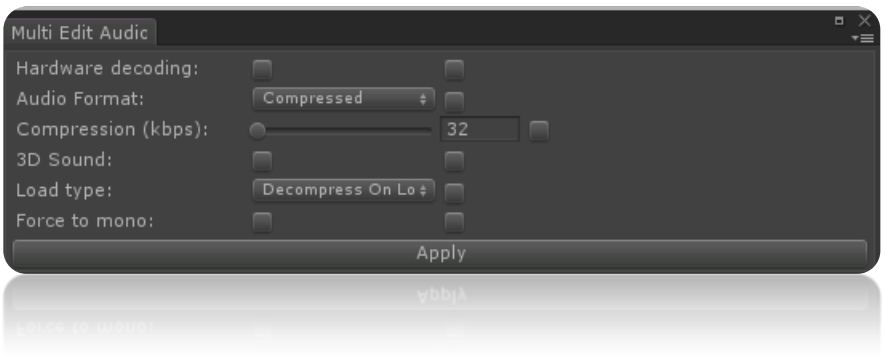
Mixer

The mixer window displays the group components in the hierarchy. From this view it is possible to change the volume or pitch properties of individual components.

The mixers Mute and Solo functionality allows to easily to mix and balance the audio levels in a game.



Multi edit Audio



The multi edit audio window allows selecting any number of audio clips together and editing their properties collective with a touch of a button.

Previewer

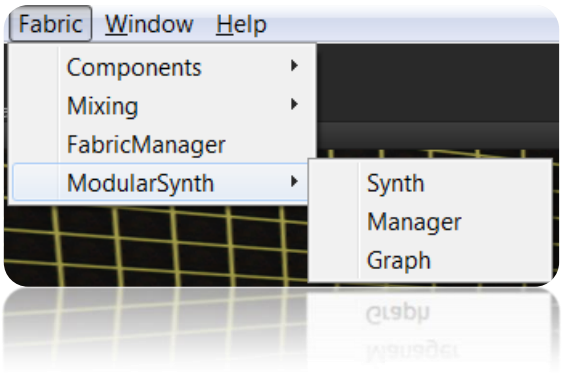
If a component has an EventListener attached then the previewer will display the Play and Stop buttons allowing a preview version of the component in the editor.

ConvertSamplesToSec

This utility script is used to convert all component properties that have been set as samples in projects created before Unity 4.1 version. Running this script once will be enough to convert the values to seconds.

Modular Synth Extension

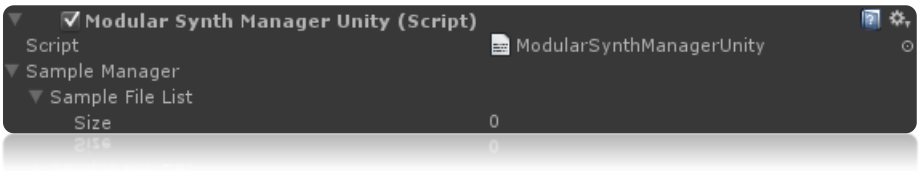
Fabric provides a modular synth which can link audio modules together in order to create complex audio effects or generators.



There are three key modular synth components

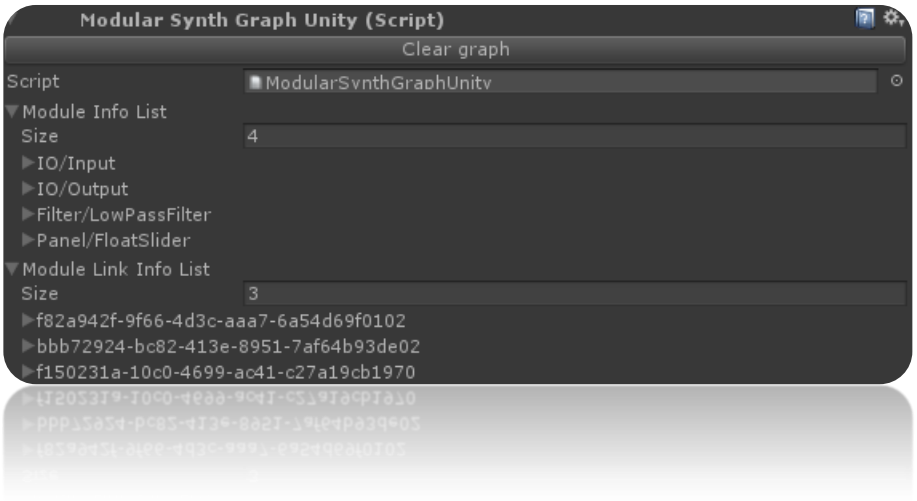
Synth manager

The manager is responsible for managing all the modular synths and also the samples used by the sample player module.



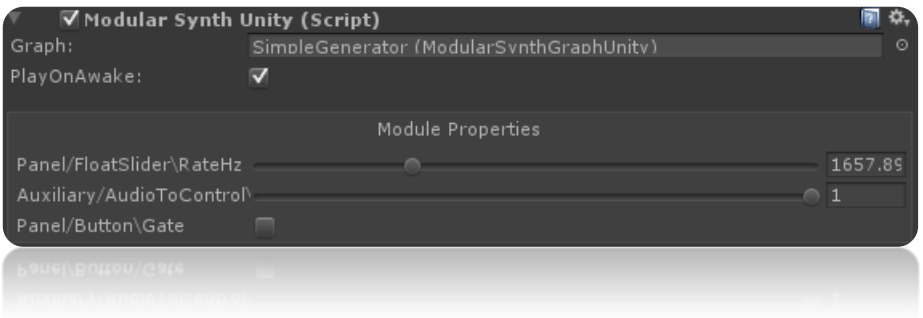
Synth Graph

The synth graph stores all the modules and their connections



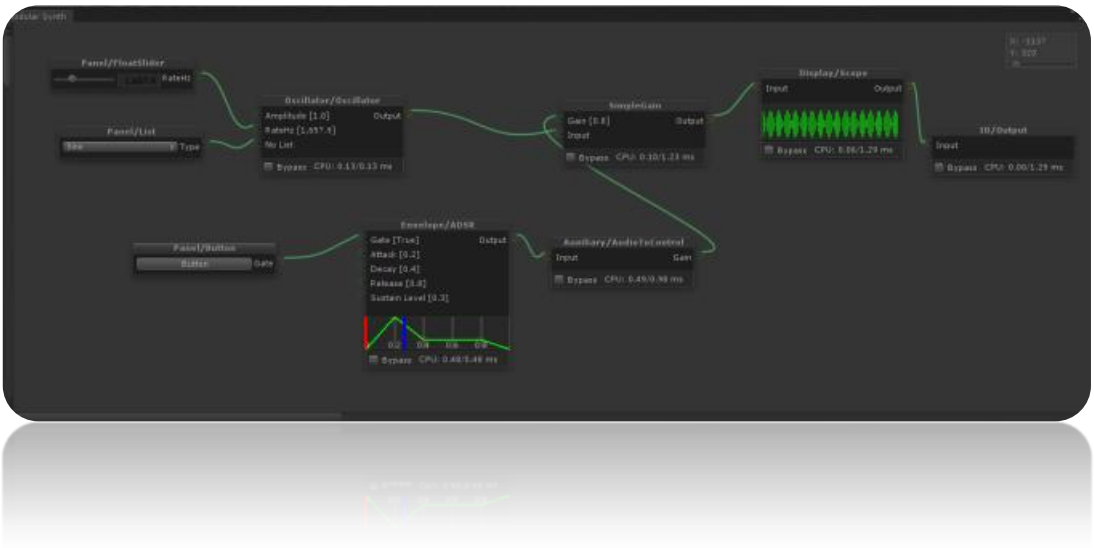
Synth

The synth component can accept any modular synth graph for processing. It also displays any panel module properties.



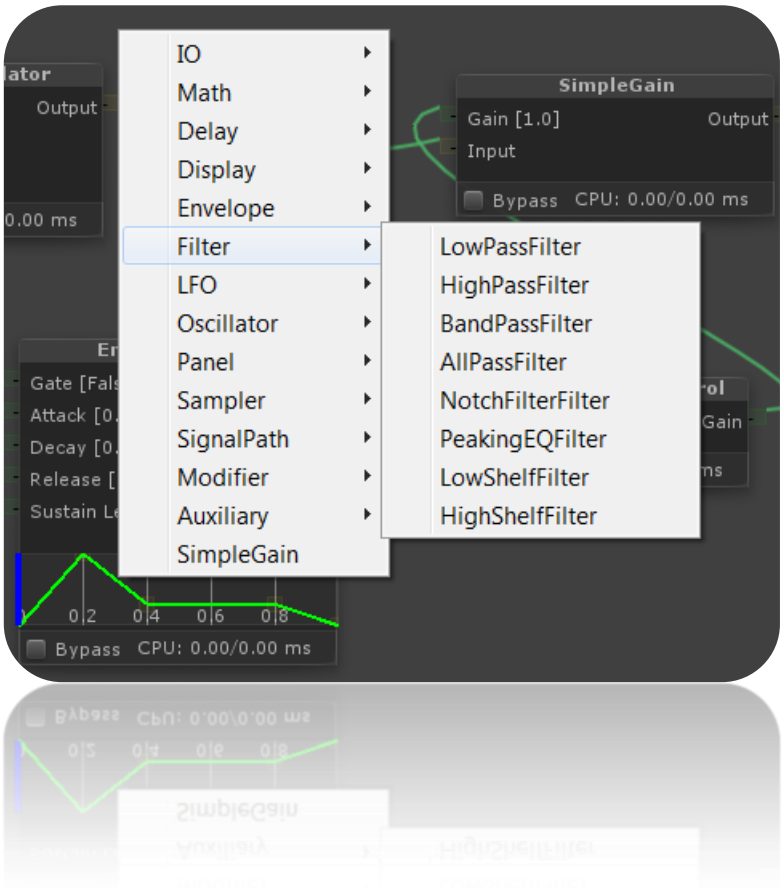
Synth window

This is the main window that displays all the modules in a graph as well as allowing the addition of new ones, removing or linking them together



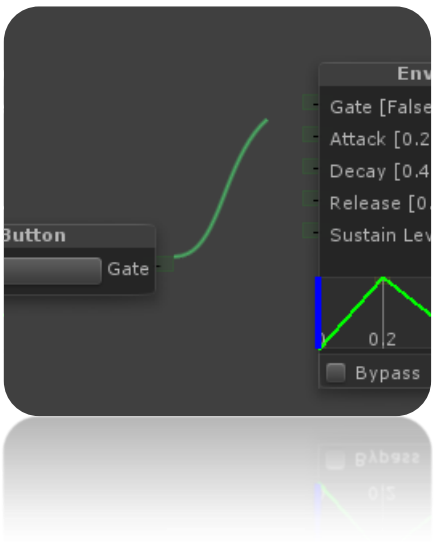
Adding modules to the graph

It is possible to add a new module by right clicking on the module window, this will show a drop down menu with the different modules available.



Linking module pins

Modules can be linked together by selecting an output pin and holding the left mouse button down; this will show a green line that can be dragged around into an input pin of a similar type on another module.

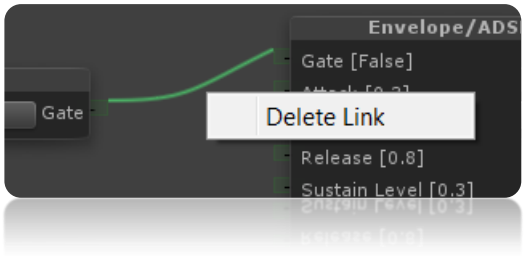


There are two different types of input/output pins:

Pins	Description
Audio input	Receives audio data from audio output pin
Audio output	Outputs audio data from module
Control input	Receives control input data (float, int, bool, string, list)
Control output	Outputs control data (float, int, bool, string, list)

Deleting module pins

Links can be deleted by hovering on top of a link line and clicking on the right mouse button.



Modules

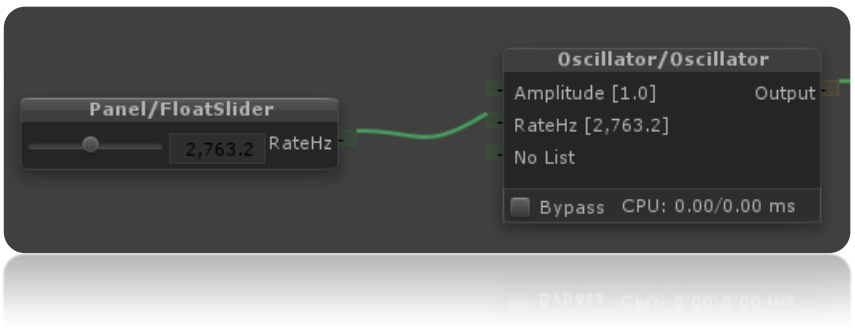
Modular synth provides two different types of module:

Audio processing modules

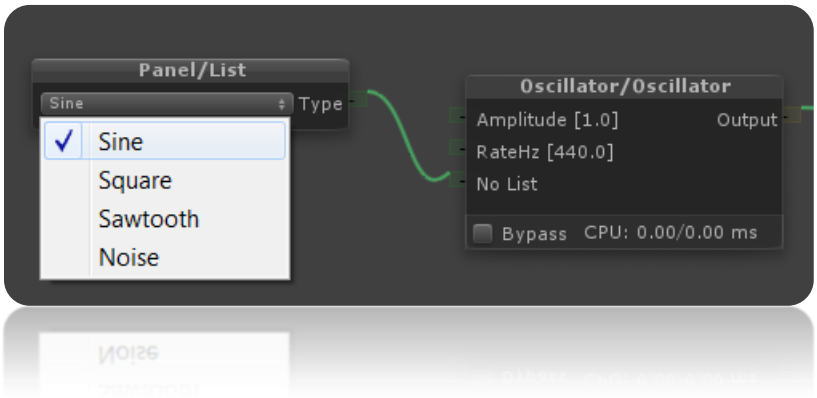
These are the processing modules that usually have input and/or output audio or control pins.

Panel modules:

These types of modules only have an output pin and are responsible for outputting parameter data such as float, integer, etc. They are automatically assigned a name and range when they are connected with a similar type input pin on another module.



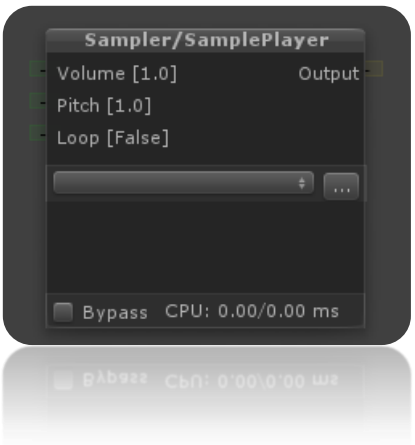
The list panel module by default is an empty list but when it is linked with an input pin it will automatically populate and provide the available options of that input pin. Below is an example of the panel list connected with the Oscillator input type.



Sample manager

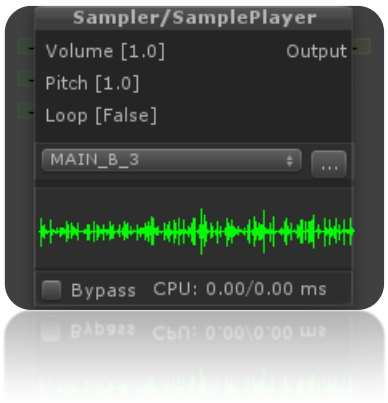
SamplerPlayer module imports audio clips into the modular synth.

NOTE: The sample player can load wavfiles with markers.



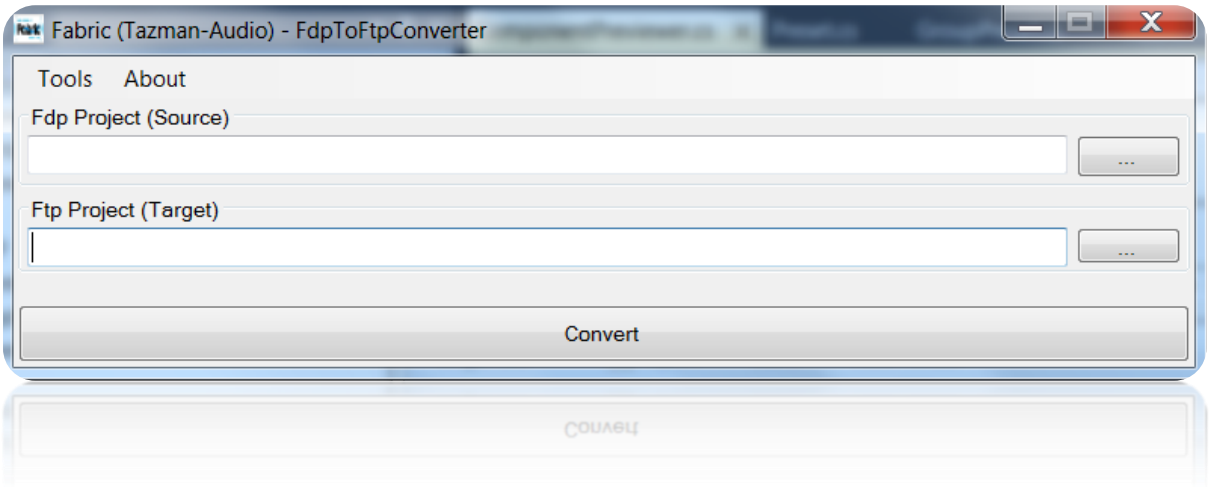
In order for a sample player to see the audio clips they have to be added first to the sample manager. By selecting on the "…" button the manager view window will be displayed allowing addition or removal of audio clips.

Once added to the SamplerManager, they will be available in all the SamplePlayer dropdown menu options.



FdpToFtpConverter Tool

This is a standalone tool that is capable of reading fdp project files and converting them into Fabric's timeline file format (*.ftp). It can also be run as a command line tool passing the projet name, project directory and build directory.



The following must be taken into consideration in order for the conversion to work.

- The audio source directory field must be empty.
- All wavfiles used by the project MUST be in the same folder as the fdp file.

Add Post-Build Command

This command is responsible for automatically adding a post build command into the project (PC configuration only) so that it is possible to automatically convert the project when it is built.

NOTE: The location of the FdpToFtpConverter executable must be added to the environment path.

Supported features

The tool supports a subset of the event properties:

- Max playbacks
- Volume
- Volume randomisation
- Pitch
- Pitch Randomisation
- Fade In Time
- Fade Out Time
- Priority
- Max Playbacks
- Max Playbacks Behaviour
- 3D Roll off
- 3D Min Distance
- 3D Max Distance
- Sound definitions
- Layers
- Parameters supporting seek speed, velocity and loop
- Effect properties: Volume and Pitch
- Effect curves

NOTE: Sound definition is supported as a Random Component so it provides limited support.

Not supported features

- Categories
- Sound definition folders
- DSP effects
- Music
- Sound Banks

Quick walkthrough

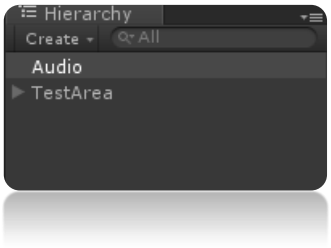
This is a quick walkthrough explaining how to create and trigger a simple audio component in Fabric.

Setting up the scene

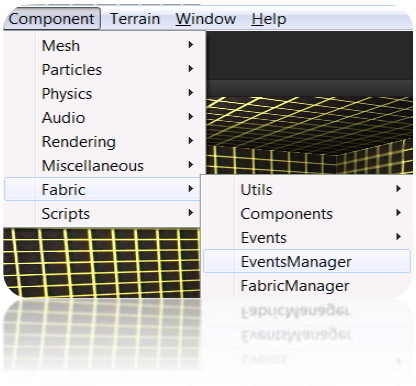
First we are going to start with a new empty Unity project in which we are going to place our audio assets, scene and Fabric hierarchy.

Creating the audio hierarchy

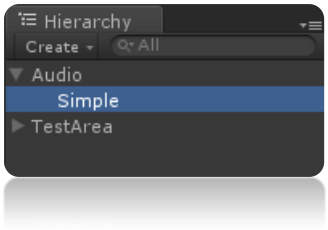
- Create a game object which is going to be the root of Fabric and give it a meaningful name (i.e. Audio, GameAudio etc.)



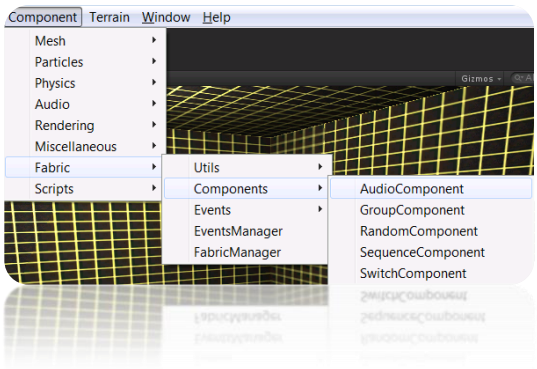
- Highlight the game object and from the Components->Fabric menu add FabricManager and EventManager components.



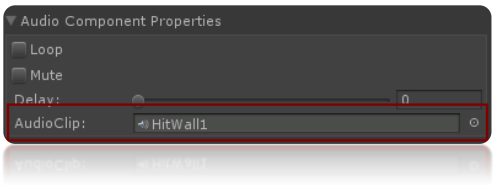
- Create another game object below the root and call it Simple.



- From the Components->Fabric->Components menu selection add the AudioComponent component.



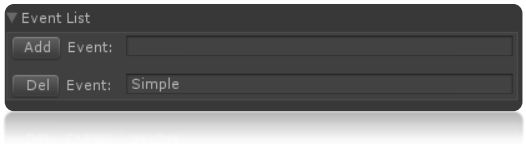
- Next step is to assign the audio clip property of the audio, this could be done either by dragging and dropping the audio file in the AudioClip property or by clicking on it.



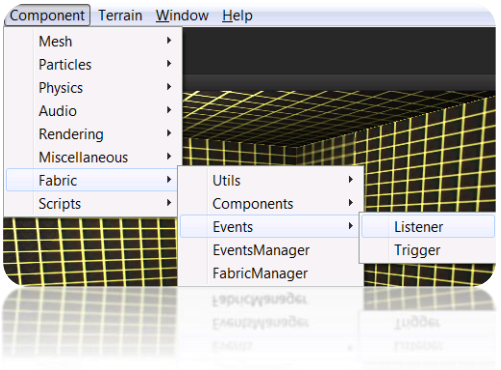
Creating the events

Each component needs to have a listener with a specific event name that it will listen to and respond accordingly.

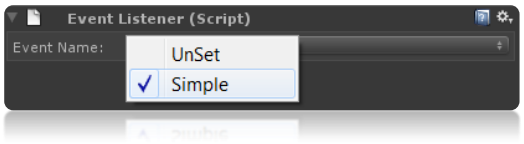
- In the EventManager enter the name of the event and click on the Add button to add it into the list of available events.



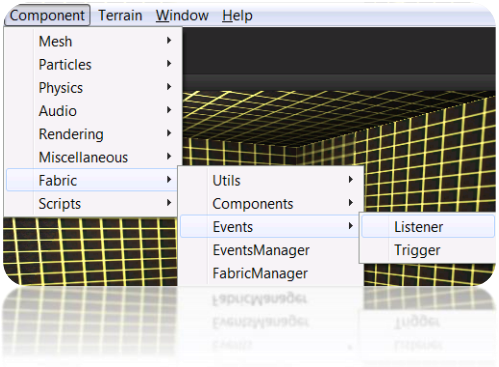
- Highlight the Simple game object and from the Components->Fabric->Events add the EventListener component.



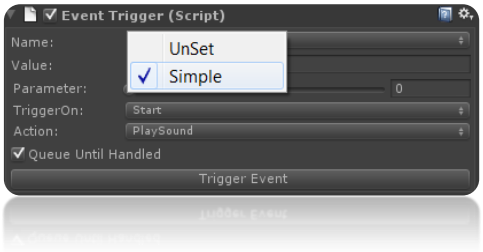
- Next step is to select the event name from the drop down menu.



- Now create another empty game object, outside of the audio hierarchy, and from the Components->Fabric->Events menu selection add an EventTrigger.



- In the event name drop down menu select the same event name as the listener.



- And that's it, when you run the game it will automatically trigger the audio component and you should be able to hear a sound. It is also possible to trigger the audio again by clicking on the trigger button.