AutoMusic Documentation

Thank you for purchasing AutoMusic for Unity! This package is designed to be easy to get set-up with, but it is also a deep application with a lot of capability for those willing to spend some time with it: This documentation will help you focus that time.

A musical background is not required to use this package, but may be of help understanding some of the terminology & customising instruments.

If you do have a musical background. This documentation will still be of use to you as you have probably not made music in this way before.

Contents

Quick Start

The easiest way to set-up a new music system is to begin with one of the included prefabs or scenes, and modify it to your purposes.
The simplest included systems are ‘Arcade’ & ‘LeanTemplate’: You can make these sound like your chosen music genre by switching out a few audioclips & adjusting bpm & similar settings.

If using ‘Audio Sources’ for your instrument output modes, you may also need to raise the ‘Max Real Voices’ number in Project settings > Audio. It is suggested to set this to at least 64.

If you do wish to create a music system from scratch, you simply need an object hosting each of these components:

Master Clock	The central hub of the system, governing tempo & distributing messages to instruments
Harmonic Hub	In charge of the notes/chords that are allowed in your system
Sound Module(s)	The individual instruments that comprise the system. They come in many flavours & it’s surprisingly easy to write your own. You need at least one instrument to hear something, there is no prescribed upper limit to how many a system can contain (please see the performance section later in this document for more information on this). **Each Sound Module (instrument) object must also contain an AudioSource component.**

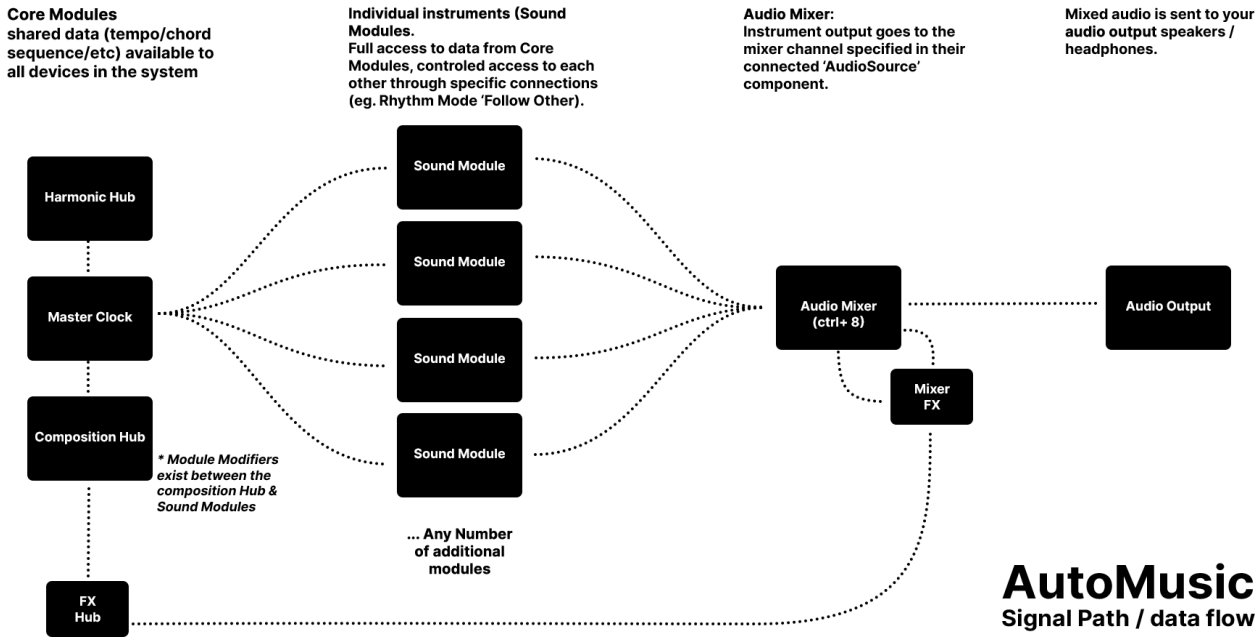
Additionally, there are a set of components that can be added to your system to increase its potential:

Composition Hub	Controls which instruments are active, how frequently to ‘reroll’ what patterns individual instruments are playing, & when to switch settings on per-instrument modifiers. Additionally it communicates with the following component to modulate fx assigned to the AudioMixer channels…
FXHub	The FXHub sends messages between the CompositionHub & AudioMixer to apply automation to mixer channels based on settings in your composition. This can easily range from subtle tweaks to dramatic alterations.

This package contains additional components that perform more supporting roles, they will be outlined in this documentation and they can be seen utilised in the example scenes & prefabs.

Signal Path

This (simplified) overview of how data flows through the AutoMusic system may help you understand the core concepts involved.



Core Component Reference

Master Clock

This is the core of the system, all other modules/components in some way feed off it. The BPM & Swing controls can be modified directly in real-time, **the other Musical Settings only refresh when regenerating patterns**, and they can also be manipulated via the Composition .Hub

If you are intending to expand this system for your own purposes, check inside the .cs file for this object to see some hidden debug parameters that may be of use in understanding what's going on under the hood, or hooking your code into the system.

System Settings	
Global Latency	Applying a small amount of latency can improve system stability. Do not increase further than required as this can harm performance (see Performance section of this document).
Audio Mixer Snapshot	The mixer settings to load into the Unity Audio Mixer

Musical Settings	
BPM	Musical playback speed. Can be modified in real-time.
Swing	The amount of swing/shuffle feel to add to 1/16th notes. Ranges from completely straight to far too much. Can be modified in real-time.
Divisions Per Beat	How many time divisions between each beat… typical values are 4 or 3.
Beats Per Bar	The ‘time signature’ of the sequence
Bars Per Section	Number of bars in each generated sequence.
Repeats Per Section	How many times to repeat the sequence before considering regenerating something new
Fills Every nth Bar	Many of the Sound Modules implement a ‘fills’ control, where every n number of bars rhythmic & pitch variation is added to the sequence.
Velocity Curves	These curves can be assigned (in variable quantities) to the instruments, modulating the strength of notes depending on where they land in musical time. The curve loops every bar.
Force Regen Instant	Force a regeneration of the musical sequence instantly. This button is provided primarily as a debug control, to create the same functionality in code, call RegenDevices(true);
Force Regen On Loop	This suggests for the devices to regenerate at the end of the current sequence. If unconnected to a Composition Hub, or based on Composition Hub properties, this will not regenerate all devices : It is done this way as often it sounds more musical not to have everything jump pattern at once.
Verbose Debug	(Editor only) will cause a lot of information to copy to the editor console, for debugging purposes.

Harmonic Hub

The harmonic hub is responsible for controlling what musical scale sequences are generated in, along with how complex those sequences are & what types of chords they can be comprised of.

This system works by modifying the pitch of samples to create musical sequences, so the actual musical **key of** your sequenc will depend on what note your source samples are. Unless otherwise noted in filenames, all of the pitched audio clips supplied with this package are playing the note ‘C’.

The system will automatically generate the correct minor/major/etc form to fit the current scale.

Use Injected Sequence	Please refer to information detailed in the 'Using Injectables' section of this document.
Scale	The type of musical scale (mode) to use when generating sequences
Transpose	How many semitones of pitch offset to shift the sequence by. Useful for key changes.
Start On Root	Force the generated chord sequences to begin with the root note of the given scale
Chord Count	Min & Max number of chords to use in the sequence
Bars In Sequence	Number of bars over which to play chord sequences before looping
Proportion Chords Beat One	Biases chords to change with bars. Chord sequences will become more rhythmically 'chaotic' with this set lower.
Extension Count	Min & Max number of notes to add on top of the root note for each chord in the sequence
Homogenous Extensions	Interacts with the following control. When on, all chords in the sequence will use the same extension stack, when off each chord will independently pick a stack to use
Chord Extension Stacks	Expressed in scale intervals, the notes to add to the generated roots to build full chords. For example, for a typical root-third-fifth chord this stack should have two entries: 3 & 5.

Composition Hub

Although not strictly necessary to make the system work, the Composition Hub is very important for getting the most out of this package. It's the core component in creating movement/evolution/variation within a system, governing the type & amount of change to apply to individual modules, which modules to have active for a given section, and sends messages to the FX Hub to apply further modulation for your music system.

The Composition Hub is what elevates systems from playing single passages to playing ever-evolving pieces.

The core functionality of the Composition Hub is provided through a. Links to Module Modifiers & b. An array of Composition Layouts. Both of these are detailed further on in this document.

Due to its power, it's also an area of the system that requires a bit of attention to get the most out of.

Device Links	
Master Clock	Drag in a reference to your systems Master Clock component
Sound Modules	Drag in references to any sound modules you want to be managed by this device. I*n order to play, those devices will also need to be included in a Composition Layout, as described below.*
Modifiers	Drag in references to any sound module Modifiers that you want to be managed by this device. Periodically (based on the settings in this component & it's Composition Layouts, modifiers will be called to change the settings on their connected devices.
FX Hub	Link in references to any FX Hubs you are using.
Composition Properties	
Force Next Layout Index	The next time a composition layout change is requested by the system (or user), instead of randomly selecting from the CompositionLayouts array, this index in the array will be selected. After triggering that layout, this value will return to it's default '-1' value, meaning the system returns to random layout selection.
Harmonic Stability	Higher value will cause the Harmonic Hub in you system to tend towards sticking to the same sequence for longer durations
Master Step Divisions	An array of any master 'Divisions Per Beat' values you want considered for use by the Master Clock that this Composition Hub serves
Master Step Divisions First Entry Bias	Interacts with the above control : higher values add stability to the system by encouraging it to use the first value in the Master Step Divisions array more frequently than the other given values.
Master Beats Per Bar	An array of any master Beats Per Bar' properties you want to be considered for use by the Master Clock that this Composition Hub serves
Layout Stability	Higher value will cause this component to maintain the current Composition Layout for longer durations
Seed Stability	Higher values will cause the Master Clock to hold onto generated patterns (note data / rhythms etc) for longer durations

Active Modifiable	The modifier system allows any number of disabled devices to have their settings retriggered, but active devices being switched out is limited by this value : the function of this is to reduce jarring amounts of change.
Composition Layouts	An array of Composition Layouts, which are core building blocks defining musical segments of devices enabled/disabled, and fx to assign or modulate on audio mixer channels. This is further detailed in the next section of this document,
Master Channel Properties	*The following properties also rely on settings that can be controlled through the FX Hub*
Master Filter Chance	Percentage chance of a master filter curve preset being applied to this section (if no filter curve in use, the filters will default to fully open)
Master Filters	If the above parameter 'succeeds' one of these presets will be applied to a set of low & high pass filters on the master channel
Master FX Chance	Percentage chance of a master FX curve preset being applied to this section (if no FX curve in use, the FX will default to fully off)
Master FX	If the above parameter 'succeeds' one of these presets will be applied to a delay & reverb fx that all the Sound Modules are run through.

Composition Layout

Your Composition Hub can contain any number of Composition Layouts. They are extremely useful for giving defined moods/atmospheres to different musical packages. By default the Composition Hub will periodically switch between layouts (based on user set rules), but you can also call the Hub to switch to a specified Layout at the end of the current musical sequence.

CompositionLayout is not a component : they are added directly through the 'Composition Layouts' array in the Composition Hub.

Section Name	The name of this layout… purely for user reference of recording the 'intention' of this particular layout
Primary Modules	All of these Sound Modules will be enabled while this layout is active
Secondary Modules	A random selection of these Sound Modules will be enabled while this layout is active
Repeat Only Modules	A random selection of these Sound Modules will be enabled if this layout continues to play beyond a single loop.
No Repeat	Disallow this layout from repeating beyond a single loop

No Repeat Allowed Followers	If ‘No Repeat’ is enabled, after the initial loop a random entry from array will be used as the following layout. This is an array of integers that correspond to the full array of Composition Layouts loaded in the Composition Hub
Secondary Start Active Chance	The system will roll against this value to determine whether a secondary module will be activated.
Repeats Start Active Chance	The system will roll against this value to determine whether a repeat-only module will be activated.
Build Up Stages	If using the ‘Build Up’ Section Modular, the appropriate modules for this layout will progressively come in over the course of the sequence in this many stages.
Strip Down Factor	If using the ‘Second Half Strip Down’ Section Modular, at the half way point of the sequence this ratio of the active devices will be disabled.
Section Modulators	This is a bank of utilities for adding structural movement within a sequence, by enabling/disabling or volume fading devices. A random entry from this array will be used when playing the sequence. The available options are: - No Change - Add Secondary - Remove Secondary - Second Half Strip Down - Build Up - Fade In Secondary
FX Modulators	An FX Modular is a bank of curves that correspond to a target. The curve is a standard Unity AnimationCurve, the target is a string that should correspond to a parameter that has been exposed in the AudioMixer. Please see Unity documentation for information on how Mixer parameters work.
FX Modulator Chance	The chance of one of the above fx modulators being triggered. Only a single FX Modular can be played at once, but an FX Modular can contain any number of curves/targets.

FX Hub

Much of the FX Hub functionality takes place beyond the scenes, as it serves as a middleman between the Composition Hub & the Audio Mixer.

It’s user-facing parameters are principally concerned with controlling the settings for the main delay send effect in the mixer.

Mixer	A link to the Unity Audio Mixer
Delay BPM Sync	Enable to have the delay effect speed be based on the current BPM
Delay Time Steps	If BPM sync enabled, the delay duration will be set to this many 1/16 notes
Delay Time MS	If BPM sync disabled, the delay duration will be this many milliseconds in duration
Delay Decay	Feedback setting for the delay effect
DIrect Sound Overdrive	Adds an overdrive effect to any modules using the DirectSOund output type : Low values will provide a subtle saturation, high values will completely trash your audio
Direct Sound Spatialisation	Direct Sound / Synth modules will use their AudioSource 3D sound settings if this tickbox is enabled
Sidechain Compression Settings	The following settings all control the 'Sidechain Compression' feature , which ducks the volume of (DirectSound) modules based on the inputs configured here
Sidechain Src Modules	Modules in this array will trigger the compressor, triggering volume envelopes
Sidechain Curve	The slope/curve of the volume envelope triggered by the compressor. Time is on the X axis, volume on Y. Values on both axis should be within 0-1
Sidechain Curve Duration	The amount of time (in musical beats) that the curve will progress over
Sidechain Min Trigger Velocity	This slider can trim out low velocity notes, stopping them from triggering the envelope

Filter Interaction

This is a utility component designed to improve mixing between instruments that could potentially clash. For example, having a high-pass filter filter out low-end from the synth channel if the kick or bass instruments are playing.

Check the 'FXHub' object in the 'SoundSet_ElectroHard' scene for an example of this component being used.

Mixer	A link to the Unity Audio Mixer
Controlling	Check the active state of these modules to determine whether to enable

Modules	this filter
Target effect	String value connecting to an exposed parameter in the Audio Mixer (Please see Unity documentation for information on how Mixer parameters work);
Controller Off On Values	If none of the controlling modules are active, the target effect parameter will be set to the X (off) value of this vector2… if all of the controlling modules are active, that parameter will be set to the Y (on) value. An intermediate value will be used if some but not all controlling modules are active.
Smoothing	Applies a softening value to the transition time between the off/on values, to remove jarring changes or clicks.

Crossfader

Crossfades between two instruments for smooth modulations between sounds. When the switch is completed the modifiers associated with the instruments will request a preset switch on the instrument being mixed away from.

This device also applies filters to the instruments being crossfaded to make the transition cleaner.

By default, this device will track playhead position from the Master Clock, but you can also switch it to a ‘manual’ mode and control the crossfade manually.

The crossfader relies on AnimationCurves to work… points in these curves should have time keys spanning 0-1, value keys of 0 & 1 represent the A & B decks respectively, with in-between values lerping between. Unless in manual mode, the time value is stretched over the full section length as tracked by the Master Clock.

Note: This device sometimes triggers an editor warning in the console, this warning seems to be safe to ignore but the issue will be looked into for a future release.

A Modifier	The Module Modifier connected to the instrument representing the ‘A deck’
B Modifier	The Module Modifier connected to the instrument representing the ‘B deck’
CrossFade Pow	Volume curve applied to the crossfader, for fine-tuning the effect of the crossfader when it’s between 0-1.
Cross Fader Modifier Influence	Chance of modifier being switched to a different preset once the crossfade is complete
Default Curve	For setting the attributes of the curve used when the device ‘fails’ the diff curve chance check. By default a generic curve will be placed in this slot that solos the ‘A’

	instrument, muting the ‘B’ instrument.
Diff Curve Chance	Chance of using one of the curves in the ‘curves’ array instead of the default curve.
Curves	Library of curves that the crossfader can switch to (either manually or automatically). These curves can be simple fades over duration of a section, or complicated back/forths with many keys creating complex shapes.
Manual Control	Don’t automatically update the curve position etc, instead allow for direct script control. The value to modify to update the crossfader position is ‘pointInCurve’, which is a float parameter expecting a value between 0 & 1. This value does not move the crossfader directly, rather it controls the time at which the current curve is to be evaluated.

Injectable Generators

The ‘InjectableGenerator’ class is a system of components that can create (or import) sequences of BeatInstances & transmit those sequences to SoundModules to be used in place of the built in procedural generation systems built-in to each instrument.

This system is designed to be customisable & expandable. The initial release includes a single implementation of this system, for converting MIDI files to work within the AutoMusic eco-system.

Using Injectables

Sound Modules can be connected to Injectable Generators by enabling the ‘Use Injected Sequence’ at the top of each Sound Module & connecting a valid Injectable component into the revealed Injectable Source slot. It is safe to enable/disable this toggle & switch between connected components in the source slot during playmode.

Injected Sequences will only be used if they contain valid data, else the system will fallback to generated sequences.

Additionally, sequences can be connected to the Harmonic Hub in the same manner: this will use the Injected Sequence to generate chord sequences that will then be used by any module set to ‘Follow Chords’ for it’s pitch logic.

MIDI To Beat Instance

This component takes in standard MIDI files & generates sequences that fit the format used in AutoMusic.

Active	Disable this to bypass this component
Sequence Index	Set to -1 to pick a random index from the SrcMIDI preset array, set to a number within the range of entries in that array to switch to that entry.
Manual Regen Only	Enable to only switch sequences when manually / scripted to do so, else sequences will be selected through the MasterClock / Composition Hub in the same manner as Sound Modules
Master Clock	Connection to your systems Master Clock
Src MIDI	This array can contain connections to any number of MIDI files. MIDI files are pre-converted to AutoMusic system upon entry in the editor to maximise device compatibility. There are a number of settings / filters that can be applied to the incoming MIDI data, please refer to the hover-text tooltips available in editor for details.

Sound Modules

The ‘instruments’ in this system. They all follow the same format of generating rhythmic (and where appropriate, pitch) data to create music, but they are all tuned to serve different compositional roles: For example, drums, bass, or chord sequences. As such, they all have similar but different control layouts. The control layouts have been optimised to provide as much control with as few controls as possible.

Most of the modules require a single audio clip to be loaded, but some (3x3, hi-hats) require multiple clips.

All Sound Modules also require an AudioSource component to be added to the same gameObject. Settings from the AudioSource component are also considered when playing back the audio, so that component should be used to control what mixer channel the sound plays through, and this also means that you can choose per instance whether the sound plays back in 2d or 3d.

Depending on the ‘Output Mode’ of the model, additional AudioSource components may be generated when in play mode.

Settings on sound modules can be overridden by Module Modifiers, see later in this document for details.

Just because a sound module is called something like ‘kick’ or ‘bass’, doesn’t mean it strictly has to be used for that purpose!

Properties shared by all Sound Modules:

Active	Active state of the module. If this module is managed by a Composition Hub, this value is set by the Hub. It is also perfectly safe to use the active state of the gameObject hosting the module to disable any instrument.
Use Injected Sequence	Please refer to information detailed in the ‘Using Injectables’ section of this document.
Output Mode	Most SoundModule types will give the option to select the output mode : the ‘correct’ mode to choose will depend on your intentions with the instrument. Please see the Output Mode section of this document for more information. The two modes are : **Audio Sources** : A number of Audio Source components are added to this object (based on the number set for the instruments ‘voice count’, and these are cycled through sequentially to allow the instrument to schedule & play sounds in musical time. **Direct Sound** : The instrument only uses one Audio Source component & instead loads the required directly into the audio stream during OnAudioFilterRead().
Audio Clip	The audio clip that will be played any time a trigger is requested by the module. The exact timing, pitch, and volume of this clip will be set each time it is played back. *Note: for instruments that have additional audio clip inputs, this property is unused.*
Pitch Multiplier	Multiply the playback pitch determined by note/transpose data by this value to get the final playback pitch. Useful for retuning samples to work together (or simply to get the sound you desire).
Loop Bar Count	Bar count duration per generated loop. The loop is then repeated to fill the full generated section (as defined by the Master Clock). Beyond this loop length, this sequence is still further modified to fit harmonic structure or to account for features such as Fills.
Swing Multiplier	Multiplies the Master Clock swing parameter by this value to get the amount of swing used by this instrument
Velocity Curve	Select the index of the Master Clock velocity curve to use for this instrument
Velocity Influence	Scale the velocity curve by this value to get the final per-note velocity (loudness)
Humanise	Offset the timing of each output note by this many milliseconds (negative or

Offset Milliseconds	positive) to give a more ‘human’ feel if desired. This is only intended to be used for relatively low numbers (roughly between -50 +50) and results may be unexpected outside of that range
Density Trim Inner Loop	This loop runs along the duration of each loop : for points in time where the curve is >= 1, all notes will be let through. Where the curve is <= 0, no notes are let through. Between those values, a corresponding random percentage of the notes will play. An example use case for this is having a 2 bar loop where you assign a curve that only lets notes through for the 1st bar in each loop.
Density Trim Full Sequence	As above, but instead of per-loop, this curve spans the duration of the entire master section. An example for this is a curve that slopes up from 0 to 1, making for an increasingly dense output sequence over the duration of a 16 bar section.

Percussion Modules

Module Ghosts

Takes in note timings from a source device, and creates ‘ghost’ notes based on that: duplicated notes offset either forward or backwards in time, with their velocity progressively reduced as the amount of time they are offset from the ‘real’ note increases.

Rhythm Src	The instrument from which to derive note timings
Back Ghosts	Duration of time (in steps) that notes can be offset backwards in order to generate ghosts backwards
Forward Ghosts	Duration of time (in steps) that notes can be offset forwards in order to generate ghosts forwards
Rhythmic Intensity Backwards	The ratio of input notes for which to generate ghosts that play before the input note
Rhythmic Intensity Forwards	The ratio of input notes for which to generate ghosts that play after the input note
Iterations	The number of times to re-run the generated ghosts through this algorithm (generating ghosts of ghosts of ghosts…)
Roll Chance	The chance of a generated ghost being converted in to a rapid ‘roll’
Roll Step Divisor Max	The maximum speed of a roll (in divisions of 16th notes… for example very fast 64th notes)

Harmonic Hub	Optionally attach a Harmonic Hub component, to enable arpeggiation feature
Arpeggiation	Amount of arpeggiation to apply to the incoming note data (percentage of notes to play using a varied chord tone)

Module Hats

Has slots for 2 audioClips, with the intention of these being assigned clips of a closed & open hi-hat. By default, this setup is used to have open hats tend to the off-beat timings, if you instead want your open hats on the beats, simply load the audio clips in reverse slot order.

Base Rhythmic Distortion	How much the played notes will have a tendency to play on off-beat steps
Base Rhythmic Density	The min & max number of notes to play for each beat. Each beat will choose a random number of steps from this range on which to play notes
Closed Hat	Audio clip representation the closed hi-hat
Open Hat	Audio clip representation the open hi-hat
Offs Are Open Chance	Percentage of off-beat notes that will trigger the open hat sample. All other notes will use the closed hat sample.

Module Kick

Percussive instrument that generates patterns suitable to represent a kick drum.

In addition to the publicly exposed controls, this device will always play a note on the first step of a each loop.

Tlp: If you want to generate a simple '4 on the floor' kick pattern,enter (0) as the Base Rhythmic Distortion & enter (1,1) as the Base Rhythmic Density.

Rhythm Logic	Rhythm algorithm to use for this instrument, Options are: -Density Disorder : User sets a min/max number of active steps per beat & amount of offbeat variation to apply. *This is the same as the algorithm titled 'Percish' on other instruments, naming convention will be unified in a future release* -Euclid Variations : *This is the same as the algorithm titled 'Snarish' on other instruments, naming convention will be unified in a future release*.
Fill Intensity	Beats that the Master Clock declares as suiting a fill will be reprocessed by a factor of this slider (slider at 0 == no change on fills)
Rhythm Props : Density Disorder	*These settings are only active when Rhythm Logic is set to* Density Disorder

Base Rhythmic Distortion	How much the played notes will have a tendency to play on off-beat steps
Base Rhythmic Density	The min & max number of notes to play for each beat. Each beat will choose a random number of steps from this range on which to play notes
Rhythm Props : Euclid Variation	*These settings are only active when Rhythm Logic is set to Euclid Variations*
Repeat Stack	One of these numbers will be selected each time the instrument needs to choose how many steps to wait before playing the next note
Repeat Stack Power Function	This power curve is applied when selecting an entry from the repeat stack. Higher values will make numbers at the start of the stack more likely to be used.

Module Perc

Intended to play patterns of percussion instruments that don't fit into simple kick/snare/hat paradigms.

Takes in an array of audio clips (so a pattern will contain multiple different sounds), but will not play more than a single clip per step.

Clips	Array of clips, will pick a random one for each 16th note step that the rhythm algorithm decides to activate.
Base Rhythmic Distortion	How much the played notes will have a tendency to play on off-beat steps
Base Rhythmic Density	The min & max number of notes to play for each beat. Each beat will choose a random number of steps from this range on which to play notes
Fill Intensity	Beats that the Master Clock declares as suiting a fill will be reprocessed by a factor of this slider (slider at 0 == no change on fills)

ModuleRandomBeatDrop

Very simple instrument that plays a sound at the onset of every musical section, as declared by the Master Clock section length setting.

An example of this in use can be scene in the ElectroRetro sample scene.

First Step	Step on which to play the sound : Sequences start on step '0'
Clips	Array of clips from which to randomly select a single random one to play

Module Snare

Percussion module that specialises in playing patterns suitable for snare drums, or hand claps.

Beats Until Start	Wait this many beats before playing the first note. This is not limited to full beats, for example to play the first note on the offbeat directly after a sequence starts, set this value to 0.5
Fill Intensity	Beats that the Master Clock declares as suiting a fill will be reprocessed by a factor of this slider (slider at 0 == no change on fills)
Half Time	When generating the pattern, double all step counts so as to create a half-time feel rhythm
Snare Repeat Stack	One of these numbers will be selected each time the instrument needs to choose how many steps to wait before playing the next note
Repeat Stack Power	This power curve is applied when selecting an entry from the repeat stack. Higher values will make numbers at the start of the stack more likely to be used.

Pitched Modules

Module Bass

Outputs pitched notes, playing monophonically (one at a time). The options for pitch & rhythm are principally geared towards playing basslines, but it will of course sound like whatever audioclips you play through it.

Harmonic Hub	Connect a reference to your systems Harmonic Hub (required for the device to know what notes to play)
Pitch Logic	The algorithm used for picking note pitches. Options are: -Chord Roots : Pitches will be based on the chord currently active in the harmonic hub -Spam157 : Pitches will be based around the 1st, 5th, and 7th notes of the current Harmonic Hub scale -PentatonicSpam : Pitches will be based around the pentatonic scale derived from the current Harmonic Hub scale.
Arpeggiation	Governs how much pitch variation to use, following the rules as selected by the Pitch Logic setting. For example, if Pitch Logic is set to ‘Chord

	Roots' & Arpeggiation is set to 50, 50% of the notes played will be the root note of the current chord, the remaining notes will be higher pitches within that chord.
Rhythm Logic	The algorithm used for picking note timings. The options are: -Percish : Essentially this selects random 16ths on which to play a note. -Follow Other : Copies rhythm from another instrument (typically used for having a bassline follow a kick drum) -Snarish : Defines a step on which to play the first note, then picks a random number of steps (selected from the 'snare repeat stack') to wait before next playing a note. -Mark Changes : Plays a note every time the Harmonic Hub switches to a new chord.
Loop Sound	Loop the audio clip played by this instrument, to give long sustained notes. Notes will be automatically cut-off at the end of sequences.
Rhythm Props: Follow Other	*These settings are only active when Rhythm Logic is set to Follow Other*
Rhythm Source	The instrument from which to derive the rhythm.
Deletions From Rhythm Source	Ratio of notes to remove from the source rhythm
Additional Inbetweens From Rhythm Source	Ratio of additional notes to dd between those played by the source rhythm. These notes will be played at reduced velocity.
Additional Arp On Inbetweens	Inbetween notes will be more likely to play as arpeggiated (pitched away from the chord/scale root) as a factor of this control
Rhythm Props : Percish	*These settings are only active when Rhythm Logic is set to Percish*
Base Rhythmic Distortion	How much the played notes will have a tendency to play on off-beat steps
Base Rhythmic Density	The min & max number of notes to play for each beat. Each beat will choose a random number of steps from this range on which to play notes
Rhythm Props : Snarish	*These settings are only active when Rhythm Logic is set to Snarish*
Beats Until Start	Wait this many beats before playing the first note. This is not limited to full beats, for example to play the first note on the offbeat directly after a sequence starts, set this value to 0.5
Snare Repeat Stack	One of these numbers will be selected each time the instrument needs to choose how many steps to wait before playing the next note

Repeat Stack Power	This power curve is applied when selecting an entry from the repeat stack. Higher values will make numbers at the start of the stack more likely to be used.

Module 3x3

Aimed at emulating the format of a well known acid-device, with the ability to play both tied & accented notes in a monophonic fashion.

This device can takes in multiple AudioClips that the instrument can then make choices on which particular note to play for a given step. By default, the device accesses the ‘Clips Bass’ AudioClips array; if the instrument opts to ‘Octave’ a given note, the instrument will instead access the ‘Clips Octave’ array. Additionally, there should be two audioClips loaded into each of the Bass/Octave arrays: The instrument defaults playing the [0] entry, if a note is selected to be accented the instrument will instead play the [1] entry.

Although inspired by the acid machine, it is not limited to this purpose & the multiple clips loaded do not need to follow the suggested convention.. For example you could load in audio clips sourced from 4 different sources to get some wild variation out of this instrument.

Clips Bass	Two audio clips for use by bass notes, entered into the array as non-accented followed by accented
Clips Octave	Two audio clips for use by Octaved notes, entered into the array as non-accented followed by accented
Density	The ratio of 16th notes to activate
Accents	The ratio of active steps to accent
Octaves	The ration of active steps to instead playback as the octave samples
Ties	The ratio of notes to play back as ‘ties’ (legato): Instead of triggering new notes, the previous note ie pitchbent to the required pitch
Arpeggiation	The ratio of active notes ot play as non chord/scale root notes
Fill Mode	Can be set to ‘None’ or ‘Reroll’ : Reroll simply generates a new rhythm/pitch configuration for beats that the Master Clock declares as being appropriate for fills.
Rhythm Mode	The rhythm logic to use for this instrument. Can be: -Sixteenth Scatters : Plays ‘random’ 16th notes -Snarish : *See description for Bass Module*
Pitch Logic	*See description for Bass Module*

Fold Hi-Pitches Target	If a note is set to be pitched higher than this multiplier, the instrument will instead pitch the clip to playback an octave lower
Snare Repeat Stack	*See description for Bass Module*
Harmonic Hub	Connect a reference to your systems Harmonic Hub (required for the device to know what notes to play)

Module Chords

Designed to play notes that tightly conform to the chord sequence as defined in your Harmonic Hub.

Although called ‘Chords’, the loaded audioClips should be single note… the instrument will playback multiple instances of this clip at varying pitches to generate chords procedurally.

Harmonic Hub	Connect a reference to your systems Harmonic Hub (required for the device to know what notes to play)
Chord Extension Range	The number of chord-tones above the root from which to allow played notes to be played
Rhythm Mode	Algorithm to use when generating rhythms. Options are: -Mark Changes : Play a chord every time the Harmonic Hub makes a chord change -Stagger Chord Up : Instead of playing all of the chord notes at once, play them sequentially, playing from lowest to highest -Stagger Chord Down : As above, but start with the highest note & work down -Stagger Chord Bi : As above, but play the notes in a random order -Follow Other : Inherit rhythm from another instrument
Stagger Steps	When using one of the ‘stagger’ modes, wait a number of steps between each note based on a random entry from this array (essentially the same logic as the ‘snarish’ rhythm mode).
Stagger Repeats	Once staggered, repeat the stagger sequence this many times. WIll be cut off and retriggered by a chord change
Rhythm Props : Follow Other	*These settings are only active when Rhythm Logic is set to Follow Other*
Rhythm Source	The device from which to inherit the rhythm
Deletions From	Ratio of notes to trim from the source rhythm

Rhythm Source	
Additional Delays From Rhythm Source	Works as for the ‘stagger repeats’ property detailed above
Loop Sound	Loop the source audio clip to produce sustained notes.

Module Loop Player

This module differs from the others in that it does not generate pitch/rhythm content, rather it takes in pre-made loops & plays them back in sync with the Master Clock. These loops can then be further modified by Composition/FX systems in the same way as any other module.

The loops can be any length, but keep in mind you need to enter the correct length of the loop in order for it to play back in time.

The system does make effort to resync the speed of these loops to fit your current tempo, but for technical reasons BPM adjustments on loops is not as flexible as for other modules. See parameter table below for information on this.

As this module is not generative like the percussion/pitched modules, some of the default parameters do not make sense/function for this module.

Loop Bar Count	This parameter tells the system how many bars the loop should be played over. Ensure the stated value here corresponds to the loop's true length.
Time Slice sync Per steps	The number of steps to wait between re-syncing the audioclip. This is a trade off between accurately conforming to BPM changes, and potential for small ‘clicks’ to appear between slices. If you are planning to play your loop back at a tempo different from what it was recorded at, this value should be set to the smallest unit of time represented in the clip.
Pitch	Pitch of the sample, as a raw multiplier (this is not semitones). 1 makes the clip playback at default pitch.

Output Mode

Most Sound Modules offer a choice of OutputMode. Where there is one logically ‘correct’ OutputMode for a Sound Module, that module is locked to that output.

For most situations it is recommended you use DirectSound output mode : It has all the features plus more of the Audio Source output mode, with more predictable performance cost. The Audio Source output mode will likely be deprecated in a future release (instruments using that mode will be automatically switched to DirectSound instruments).

It is perfectly safe to use a combination of the output modes across your instruments : you do not need to have all instruments set to the same mode.

It is not possible to change the output mode of an instrument during playback, this choice must be mode in edit mode.

Output modes rely on settings from a single Audio Source component that should be on the same GameObject as the module and set-up by the user with the desired settings (2d/3d, mixer channel etc).

Audio Sources	+Reacts to reverb zones -Can consume a lot of audio voices, potentially leading to other gameplay sounds being dropped by Unity
DirectSound	+Requires far fewer audio voices +Best option for including many instruments in your system +Allows access to additional sound design tools that can be triggered/modulated per note : envelopes & filters -Doesn't react to reverb zones (reverb can be applied via mixer effects)
Synth	+Extensive sound design possibilities +Opportunities for real-time modulation / sound changes that sample-based systems don't afford -Performance can suffer with multiple instruments -Currently in an experimental state of development

There are a selection of settings that are only valid for each Output Mode. Also, Sound Modules will only display the settings from this group that make sense for that module.

Audio Sources	
Voice Count	The instrument will add and play through this number of Audio Source components sequentially in a 'round-robin' fashion in order to achieve multiple simultaneous sounds. The default setting of 12 will be plenty for most devices : you may need to raise this number if your instrument plays either many rapid notes, or several overlapping notes. Instruments playing only occasional notes could have this number reduced. Setting too low = missed notes. Setting too high = potential performance issues
Direct Sound	
Use Filter	Enable to attach a low-pass filter to the module. The cutoff value can then be controlled by the following 'Filter Freq' control, and will be modulated by the Velocity Influence control on the Sound Module. Filter resonance can also be controlled via the 'Filter Res' control.
Use Amp Envelope	Enable to add an ADSR (attack/decay/sustain/release) envelope to the instrument, adding the ability to shape the amplitude of the instrument

	on a per-note basis. If ‘Use Filter’ is also enabled, this envelope can also be routed to the filter for further modulation.
Use FIlter Curve	This enables a set of curves that modulate the filter frequency and resonance values over the duration of a musical section (as set in the Master Clock). These curves can also be repeated a given number of times per-section for faster/loopier results
Sidechain Compression	Raising this slider will ‘duck the instruments volume based on note triggers, the deeper settings for this are on the FXHub component
Synth	
Dependent on instrument	The included synth instruments are constructed in a modular fashion, with architecture that will be familiar to anyone who has used typical subtractive synths : oscillators, envelopes, filters, and LFOs

Module Modifiers

There is a ‘Module Modifier’ component included for each instrument listed above.

Module Modifiers are a support class that pass messages from the Composition Hub to individual instruments, these messages can be passed along any time the Composition Hub ‘rerolls’ the current layout.

When triggered, the modifier will select an element from the presets array. A preset consists of a table of settings that can be used to override any number of properties on the instrument, for example switching what rhythm logic it uses, or having it load a different audioclip (or set of audioclips).

Creating new ModuleModifier components is very simple, if an existing Module Modifier doesn’t have the set of overrides your project needs it may be simpler to just make a new modifier than make-do with the given one. Duplicate an existing modifier .cs file and rename it: editing it to your needs should be quite straightforward based on careful reading of an existing one

Send Chosen Index To Modules	Sends the index of the selected preset to the device(s) in this array. This is useful for situations such as wanting ‘ghost’ instruments to have settings that suitable correspond to the settings on this instrument.
Random Mode	‘Random’ mode is true-random, rerolling for a fresh preset index every time it’s requested to do so. ‘Pooled Sequence’ Shuffles all of the available presets and plays through them all before rerolling a new sequence. ‘Pooled Sequence’ is technically les random but is often perceptually more random seeming, as it avoids situations where random-chance keeps picking from a smaller subset of presets
Specify Preset	If this is set to a value >= 0, force the instrument to be generated using

	that preset index, else select random index
Target Module	The instrument onto which to assign the current set of overrides
Presets	Array of presets from which to choose.
	Module Modifiers also have a 'hidden' control accessed by right-clicking the components title bar:
Create New Preset From Current Settings	This collects all of the settings governed by the modifier from the connected Sound Module & saves them as a new preset. Keep in mind: -Not all controls are saved in the preset (you can of course easily add controls to the script) -If you create your own module/modifier, you'll need to implement this in your script for the control to work.

Performance Optimisation

This package has gone through multiple rounds of optimisation, but nothing comes for free. **If your system only uses a low number (< 10) of instruments, you shouldn't need to worry about it affecting your framerate.**
If constructing more complex systems, there are essentially 2 phases to keep in mind when constructing your music system.

Generation Phase

This happens at Start and then every time the sequence is regenerated. This is unlikely to cause performance issues unless you are using many (multiple dozens) of instruments. Potentially if you are using a lot of ghost modules with intense settings the amount of note timings being iterated over could cause an issue in this stage sooner, or if many of your instruments have long loop lengths.

Performance issues from this stage will express themselves as brief pauses/stutters in the final beats of a sequence (which is when the next system is being generated). The solution is to reduce the number of instruments you have, with the complexity/length of a pattern generated by an instrument being the main factor in how expensive it is.

Note Playback Phase

Notes are distributed to instruments on a per-beat basis, with the beat-data being sent to their instruments one beat before they are required. This is done in a chunked fashion so as to give the instruments type to conduct any preprocessing of the sounds, and to ensure well-timed playback. The distribution of the notes is unlikely to cause an issue itself (unless many very complicated patterns are being played simultaneously), but of course the sounds then do need to be played back.

Specific performance issues possible from the note playback phase will depend largely on which Output Mode your instrument is set to, further information on this is in the Output Mode section of this document.

In most situations DirectSOund will be the most performant output option, but if you are using very long (30 seconds +) audio clips, Unity may be better at managing that memory footprint via the Audio Sources method.

Output Mode Performance Specifics

Audio Sources.

Instruments using the ‘Audio Sources’ Output Mode rely on using multiple Audio Sources as Unity only allows one scheduled sound per Audio Source component. Also, Unity allocates voices for Sources even before the sound begins to play.

An instrument running out of voices will result in notes not being played; perhaps seemingly random notes, or if your patterns are attempting to playback many rapid notes then none of those notes will make it through (as the voices get stuck in a cycle of continually cutting themselves off before playing). The solution to this is simply to raise the Voice Count for that instrument.

Avoid raising instrument Voice Count values higher than required, as that will consume more CPU resources than required.

The default value of ‘12’ will be safe in most situations, but experiment with lower voice counts on instruments playing simple patterns.

Unity has a global ‘Max Real Voices’ value that limits how many AudioSources can play simultaneously: if using many instruments set to ‘Audio Sources’ output mode this value may need to be raised or Unity may begin to mute other sounds in your game.

Direct Sound

The Direct Sound output mode does not have the same voice count issues, but as it’s less tightly linked with Unity’s inner systems it could potentially not manage its memory consumption quite as efficiently. Also, if using many of the ‘extra’ features enabled by this mode (filters and envelopes) CPU usage should be monitored. The DirectSound mode does have the advantage of performing most processing on Unity’s audio thread instead of the main thread.

Synth

The current implementation of the synth output is significantly harder on performance than the other output modes & it’s only recommended to use this mode for instruments that specifically need the extra sound design features.

Garbage Collection

It is recommended to have Unity’s incremental garbage collection feature enabled.

Audio Clip Settings

- The more/longer audio clips your system uses, the larger their memory footprint.
- Make sure to trim silence from the ends of your audio clips : Audio only ‘costs when it’s actually emitting sound.
- Convert sounds to mono where appropriate
- Clips that don’t have high frequency content can often have their sample rates optimised without adverse affect to their sound
- Follow standard Unity best practises in regard to compression settings (ideal settings can depend on your target platform & the length of the audio clip in question).
- The ‘Preload Audio Data’ option on audio clips can help them not to stutter when first being loaded

Adding Custom Sounds

This system uses standard Unity AudioClips, that can be imported to Unity with any typical format (.wv / .aiff /etc), no custom formats or conversions to .asset files are required.

Sounds imported for the loop module should be imported with correctly trimmed start/end points (ie, they loop correctly).

All other sounds should be single hits (a single snare hit, a single synth stab, etc), with no unnecessary silence at the start or end of the clip: silence at the start will make them trigger out of time, silence at the end will make them consume more memory than required.

You can import your pitched sounds at whatever root note that works for you, but if you want them to play in tune with the included sounds you should play them in the note C.

https://free-sample-packs.com/ has a huge collection of quality sounds for free.

Troubleshooting

No Audio?

- Check your audio isn’t muted in the normal ways to ensure it’s not a wider unity/computer issue.
- Check your current scene contains an Audio Listener component
- Check your instruments & their gameObjects are active
- Check your AudioMixer channels are not muted
- Check that mixer channels do not have filters that have become stuck in ‘closed’ positions (for example low pass filter set to very high frequency)
- Check Audio Sources on instruments do not have volumes set to 0.

- Check Unity console for red text : perhaps it is giving a helpful hint regarding a setting being on an inappropriate value?

Single instrument not playing?

- Check instrument is enabled/active
- Perhaps you have a Composition Hub running but this instrument has not been included in the modules / composition layout
- Does your instrument have a suitable 'Voice Count' setting? *(see Performance Optimisation section of this document).*
- Test another Output Mode (cannot change during play mode).
- Check that instruments mixer output (as described for the general 'No Audio' advice).
- Check Unity console for red text : perhaps it is giving a helpful hint regarding a setting being on an inappropriate value

Instrument 'skipping' notes?

- Does your instrument have a suitable 'Voice Count' setting? *(see Performance Optimisation section of this document).*
- Are the notes being skipped the first time(s) the system tries to play them? If so this is likely related to sounds not being loaded in time *see Performance Optimisation section of this document.*
- Check Unity console for red text : perhaps it is giving a helpful hint regarding a setting being on an inappropriate value?

Instrument (or entire music system) disabling itself?

- Check Unity console for red text : perhaps it is giving a helpful hint regarding a setting being on an inappropriate value?

Audio stuttering?

- Generally too many simultaneous instruments for your machine to process within a given frame: synth-based instruments most likely culprit. Setting Project Settings > Audio > DSP Buffer SIze to 'Best Performance' gives the most overhead