SETUP

Initial Setup

1. Ensure you've only imported the streaming assets for the platforms you intend to use. By default Unity will pull in Linux, Windows, Mac, Android, iOS. It's a lot of data.
2. Move the StreamingAssets folder out of the MusicGenerator folder and into your main Assets path (merge if needed, all files are in a sub-directory of that so should not cause conflicts). We'll be loading the needed music generator assets from here.
3. Drag the prefab /MusicGenerator/Assets/GeneratorPrefab/MusicGenerator prefab into your scene hierarchy (if your game or app has multiple scenes, only use one of these prefabs in your parent, or first loaded scene. It will persist and be available for all scenes).
4. An audio mixer called "GeneratorMixer" will be created the first time this scene is run.
5. You will most likely need to change your Audio project settings. Especially with generator configurations that play many notes/instruments, particularly with a fast tempo, you may encounter issues.

If you hear notes not playing, or cutting off, popping, increase the values at:

- Edit->ProjectSettings->Audio->MaxVirtualVoices
- Edit->ProjectSettings->Audio->MaxRealVoices

I haven't needed to, but it's possible you'll need to adjust the DSP buffer size or other settings if you're still encountering issues.

6. To note: If you're needing to edit the mixer properties while the program is running in the Unity Editor, the MusicGenerator member boolean 'mEnableLiveMixerEditing' needs to be set to true. Otherwise set false all other times or edit the mixer while not running the generator. There is an FMOD error caused by unloading the audio clip bundle assets, and the boolean will force the bundles to not be unloaded.

7. Additionally, after editing the mixer, you'll need to regenerate the asset bundles to save the mixer to the bundles. Assets>MusicGenerator>Build And Clean All Music Assets.

To create new song configurations

1. Open the scene at MusicGenerator/Assets/EditorScene/MusicGenerator.unity
2. In your game tab, set the aspect ratio to 16:9 (or some variation of that) and make sure the zoom level is set to 1.
3. Though you're free to run this through the editor, it's a bit more roomy and performant to go ahead a build the executable and run that as a standalone app. Either way, configs will export fine.
4. Use the interface to create a song to your liking. See the "Tips" chapter for assistance. Hold down the 'Shift' key over ui elements for a tooltip about what it does.
5. When you're done, enter a name for this song in the Export box under the staff player and hit export.
6. Please see the scene demo or Tips documentation for additional help or useful functions.

To use in your project:

See the demo scene for a sample usage scenario. Most of the music variables are editable on the fly in your game for whatever reasons you need.

After creating songs to your liking, you're able to load and adjust them on the fly through the public MusicGenerator class functions. The generator prefab has a default configuration it will load, set this to whatever your startup music will be, and use the public load functions in the MusicGenerator class to load new ones on the fly.

When building your project:

Before building your project, a couple things need to be done:

1. Copy your instrument saves from your persistentDataPath https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html and into your streamingAssets/MusicGenerator/InstrumentSaves/directory so they can be included in your builds.

2. Remove the streaming assets for any platform you're not building for (if even temporarily to build). You don't want to be bloating your linux builds with windows assets and vice versa.

3. For mobile builds, a couple recommendations:

    - in order to minimize build size, it's strongly recommended to remove any instruments you're not using in your configurations. It's about 1MB per instrument.

    - Due to the slower load times, and needing to load the asset bundles individually, it's also recommended to use configurations that use fewer instruments (i.e. it takes twice as long to load 10 instruments as it does 5).

If you're encountering crashes/errors

If you're encountering an error on line 55 of the AssetBundleBuildClass.cs, it's likely you're on a slightly older version of Unity, and as such, they've changed the BuildTargets a bit.

Change the offending line to:

BuildAssetBundles(Application.streamingAssetsPath + "/MusicGenerator/Mac", BuildTarget.StandaloneOSXIntel64);

For all other errors/crashes, generally it's been due to assetBundles not behaving nicely with different versions of Unity. I've included a script to rebuild them. If you drag the /MusicGenerator/Editor/ folder into your project's main /Assets folder and then in the Unity Editor menu, run Assets>MusicGenerator>Build And Clean All Music Assets.

If after rebuilding them you're still having issues, please contact me at stickandbindlegames@protonmail.com. I'm happy to help!

IMPORTING CUSTOM SAMPLES

To include custom samples, I've tried to streamline the process a little, though some restrictions apply:

1. New instruments must be 36 notes long (3 octaves) starting on C and ascending 3 octaves (finishing on a B 36 notes higher).
2. Files must be named 1.mp3, 2.mp3 et cetera, through 36.mp3.
3. If you haven't already, ensure that you've moved/ merged the streaming assets folder from MusicGenerator/ to your main Assets directory (Application.datapath).
4. Place these 36 files inside a folder with your instrument name and place this folder in /MusicGenerator/Assets/Resources/Music/ (take care it doesn't conflict with an existing instrument in streaming assets otherwise it will overwrite it)
5. Repeat for additional instruments.
6. Move the folder /MusicGenerator/Editor to your main Application.Database path (the main /Assets/ folder of your project). Additionally, make sure you've already moved the Streaming assets folder into the main Application.Database path.
7. New Unity menu options should appear at the top of unity under Assets/MusicGenerator/:

BuildAndCleanAllMusicAssets : One button does all. Will build new asset bundles for these instruments and place in StreamingAssets/MusicGenerator/ for linux, mac, and windows, and clean up any files used in the creation process.

If you want to build for a single platform, you need to run:

Menu>Assets>"Build PrefabsFromPaths",

then:

"Build Linux Assets" (for example),

then:

"Clean Up Music Prefabs".

in order to build the prefab, save it to an asset bundle, then clean up the temporary prefabs that were created.

In your MusicGenerator object in your scene hierarchy, under mBaseInstrumentPaths, add the name of the instrument folders you added.

When creating instruments, the generator supports mixed instruments, where several versions of the notes are imported, and when playing, randomly chooses among them. This helps avoid a mechanical feeling if the same note is repeatedly played (for instance, the ViolinShortMix instrument included in the asset has two samples

included. One has short and choppy sounding notes, the other short but smooth notes and when it plays, any given note is random between them. Compare this instrument against the normal ViolinShort instrument to see the difference).

To import a mixed instrument, you'll want each instrument's folder name to include a "_1", "_2", "_3" etc.

For example, the violinShortMix folder structure before importing looked like

ViolinShortMix_1

ViolinShortMix_2

ViolinShortMix_3

Do not nest the files under a parent instrument directory, they should be located as:

MusicGenerator/Assets/Resources/Music/ViolinShort_1,

MusicGenerator/Assets/Resources/Music/ViolinShort_2,

etc.

its name in the MusicGenerator's mBaseInstrumentsPath remains simply "ViolinShortMix" and the generator will detect the variations.

To import percussion or SFX (things with a single note), prefix the name with "P_" i.e. "P_Snare". The same rules for variation apply (i.e. P_Snare_1, P_Snare_2 if you have multiples).

TIPS

In general, these are some good rules of thumb, but like anything with music, rules are often broken, so consider these mere suggestions :P Mostly included as programmer / developer types don't always also have any music theory background.

Configuring a new song:

Feel absolutely free to use / improve on the presets included. It's what they're there for :)

To Note: Not every configuration you're able to throw together is going to sound good (it'd be like just handing a bunch of instruments to monkeys who randomly pluck at 'em, but happen to pluck in the right key). It takes a small amount of working with the configuration until it sounds up to what you'd like. The tips below are an attempt to ease this process in creating a new song configuration:

Hold shift in the UI editor for Tooltips (this is really useful for seeing what each does, and any relevant notes regarding the options).

1. Use the new Instrument / new Percussion to add instruments.
2. Use the red 'X' button on the side instrument panel to remove them.
3. It's helpful to make good use of the 'group' dropdown on the instrument panel. Each group has a random chance of playing each measure (odds of playing for these are set on the GlobalSettings). This helps a lot toward adding a sense of dynamics to a song as instruments will come in and out for different measures (I prefer to add my bigger, louder things on a group with a lower chance of playing to create a sort of 'crescendo' effect, for example). Additionally, under the advanced settings tab, there's a 'Dynamics Style' dropdown with linear and random options. If linear is chosen, the dynamics will move up and down in increments and play all groups under the highest selected (i.e. it will move from group 1

to 2 to 3, maybe back to 2, etc. But, if 3 is playing, 1 and 2 also are. With the random option, each group is determined independently.

4. The stereo pan slider on the instrument panel configures how the sound is routed through stereo (left / right). If things sound muddy, it's best to arrange instruments in different ways (there are a lot of resources and tutorials online regarding panning and mix placement for audio recording).
5. In general, you will want at least one 'rhythm' instrument. This will help establish the rhythm for the piece. These will always play on their time step.
6. Adding too many lead instruments will likely sound a bit confusing, or even dischordant (while the lead instrument takes care to avoid dischordant, it does not take into account other lead instruments).
7. For lead instruments, you'll generally want them on a 1/16 or 1/8 timestep. the 'shortMix' variation of the instruments feel a bit more natural and less robotic and will help smooth out the piece.
8. Instruments set to 'melody' will only play chordal notes, but have options for their odds of playing.
9. The 'strum' options for rhythm can be used in a few ways, whether strumming a harp, or having a snare or other instrument play faster than the 1/16 timestep (I like using it on percussion quite a bit).
10. in the Advanced panel, there are some chord progression options. You can set the odds of tonic, dominant, subdominant chords, or even select particular chord steps to never play (say you wanted to force the song to a particular progression, or avoid a downer chord in a song you're trying to make happy).
11. Modes and scales are fun. :) I suggest playing with their dropdowns in the Global Settings tab if you're unfamiliar with music theory in general. They will change the 'feel' of your song quite a lot.
12. instrument and global effects are fun as well. But don't overdo it (I usually do and then have to scale it back :P).
13. It's easy to get things muddy and crummy sounding pretty quick. I recommend starting slow. Add a rhythm instrument, add a lead or melody, and get each sounding okay before moving on. Adding a lot of unconfigured instruments will decidedly sound confusing :P Sometimes less is more.
14. If things sound too 'random' make use of the 'UsePattern' dropdown, especially on instruments with a quicker timestep. It will add some repetition.
15. for lead and melodic instruments, it helps to use the multiplier and lower the odds of playing to 'group up' the lead notes that are played.

Usage tips:

There are a lot of public variables and ways to use this player. In addition to just generating music, it allows you to change things on the fly. Mute and unmute instruments or speed up tempo based on in-game events, change the settings to give things a darker tone, etc. I've taken care to expose useful variables and functions for on the fly editing. Anything that is changed in the standalone UI editor is available for editing (though, not all are advisable to change on the fly).

Additionally, I've added a 'Measure Editor' tab that lets you set specific instruments and notes to export, load and play on the fly (think SFX when opening a chest, or starting a level, etc). See the included demo scene for an example of playing a pre-made clip.

README UPDATE

Update 1.1:

There were some major classes refactored for this update.

This update will require the user to replace the MusicGenerator object in their scene with the update one.

Care was taken to hopefully avoid many conflicts with older versions. All the previous public functionality should remain intact, so barring instances where the user was editing

the music generator classes directly, it should ideally not conflict. The exception being the event listeners mentioned below.

Several variables were moved around to other classes, so if they were accessed directly, may no longer exist.

Feel free to contact me at stickandbindlegames@gmail.com with any questions or help sorting out conflicts.

- The data structures were changed around a bit. They should update on their own the first time they're played.
- An arpeggio feature was added to rhythm and melodic instruments. This works similarly to strum, but will play the notes in a random order.
- For Groups: groups will now roll over from 4 to 1 in linear mode, rather than only go up and down.
- The previous event listeners were replaced with UnityEvents. My apologies for changing that after the fact. It will require changing your subscriptions to the classes.

The added benefit is that you can use the UnityEditor to add functions to fire with them.

Update 1.2:

There were some major classes refactored for this update.

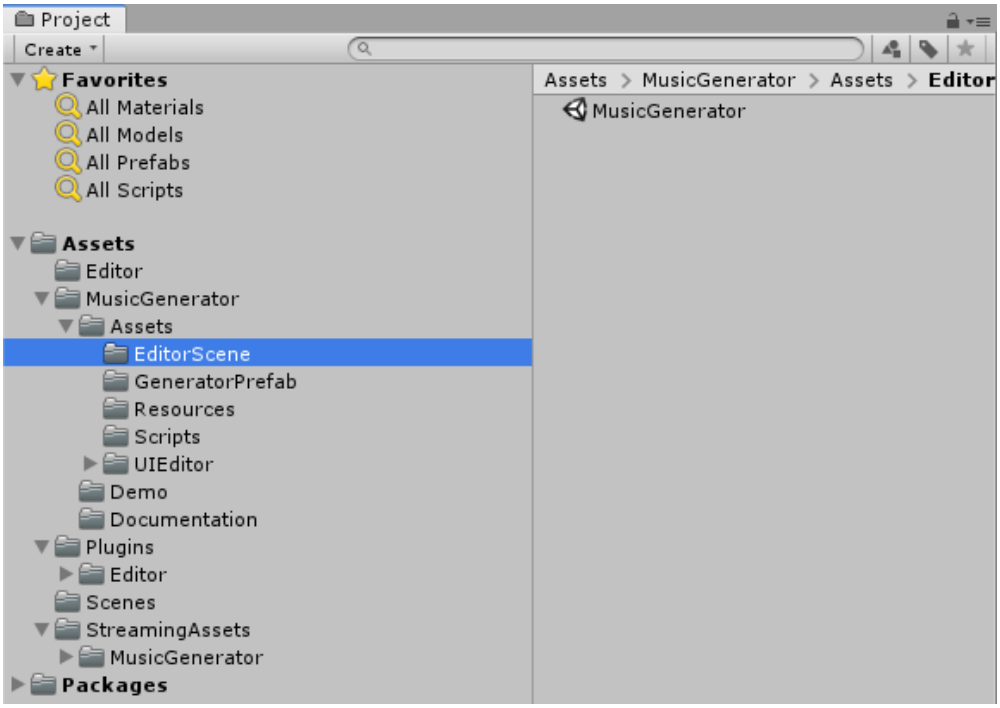
- Mobile support was added for Android and iOS. While the editor scene will run on mobile, the UI is not redone for mobile, so unless you have small fingers, a stylus, or a tablet for a mobile device, usage may be difficult.
- Pentatonic scales were added, you can select specific notes to avoid, or use the button on the left to enable the 'default' pentatonic scales for major/minor.
- Garbage generation during play was reduced to near zero for the update methods. Other general improvements made as well.
- MusicGenerator::PlayAudioClip() has had its signature changed. An event for 'OnNotePlayed()' has been added. This is to enable the use of MIDI events. Subscribe to this event to use it for MIDI output, and return false if you want to suppress the playing of the clip internally (for example, if you're using the MIDI event to fire off your own audio file).
- New Color Scheme :P
- Updated the project to the latest stable build (currently 2018.3.9f1). This is preparation for the 2018 LTS. Generally I'll try to stay on the LTS, but since the update to 2018 seems imminent, I've gone ahead and adapted most of the code to support it in advance.
- A couple new presets added "Sneaky" and "Silly March".

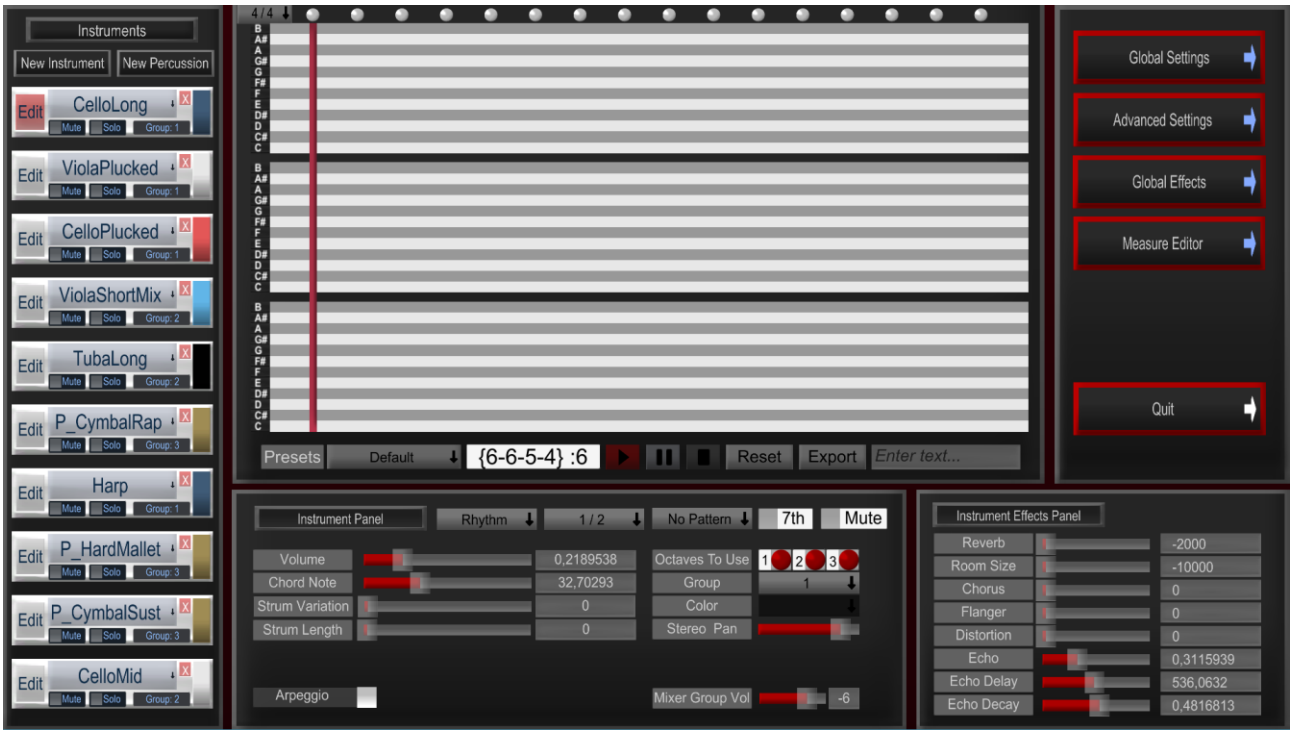
CONTACT

Feel free to contact me (Tyran) at stickandbindlegames@protonmail.com with any questions, comments, bugs, suggestions, or anything really. :)

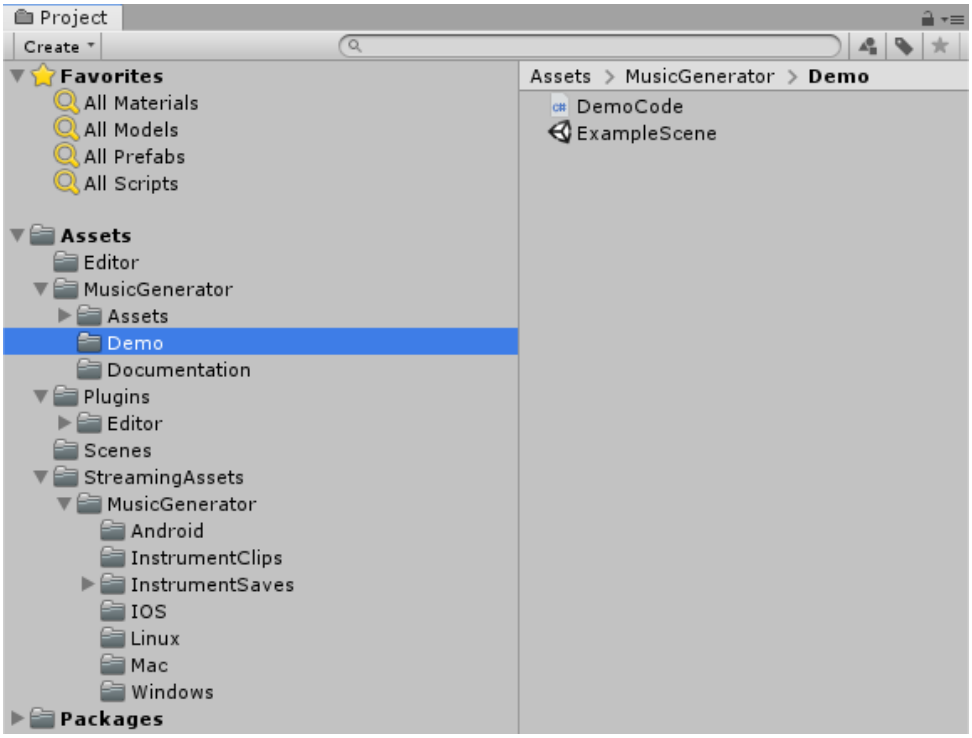
APPENDICES

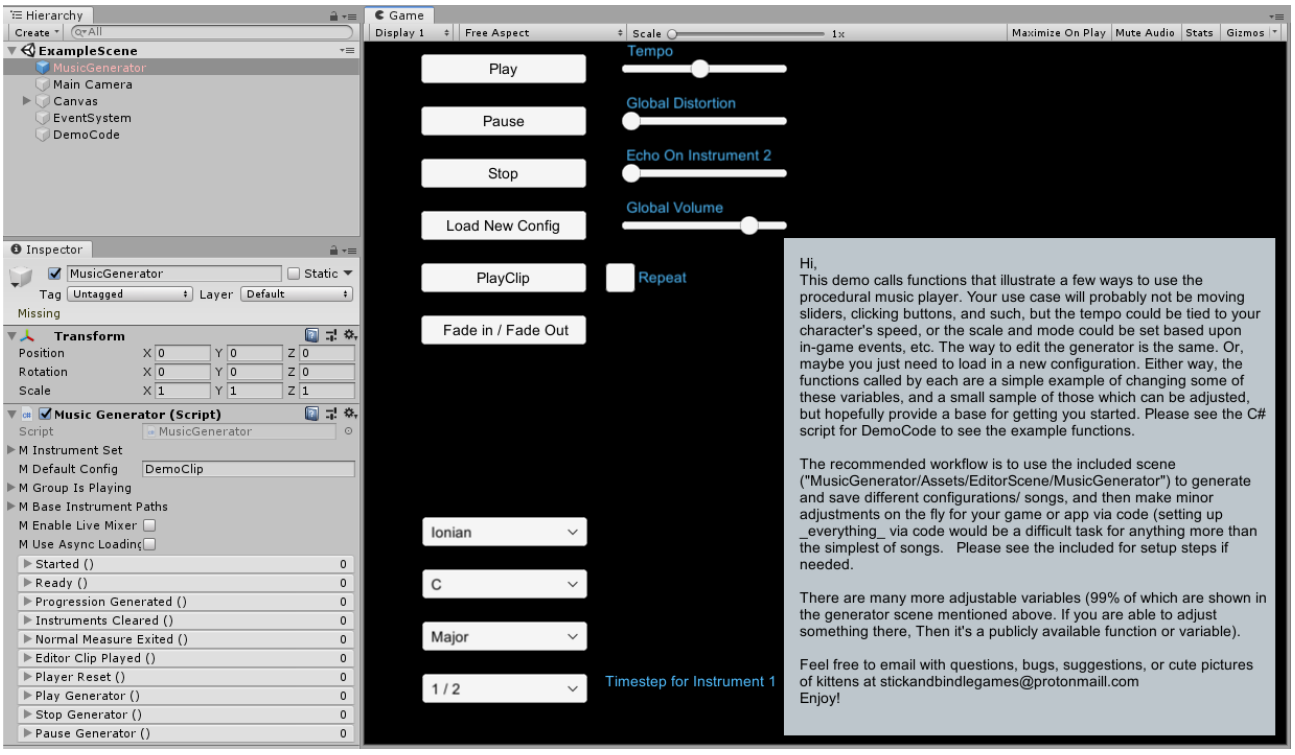
EditorScene - Music Generator





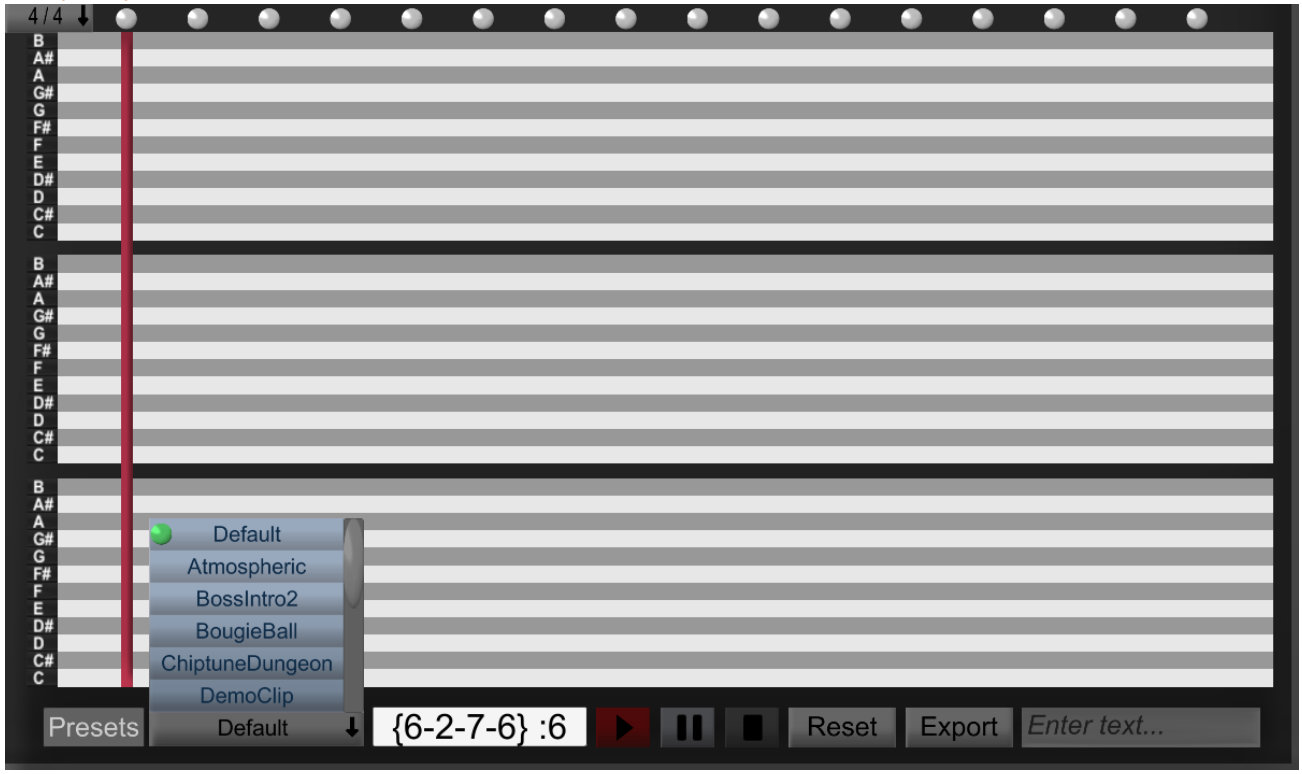
Demo - ExampleScene



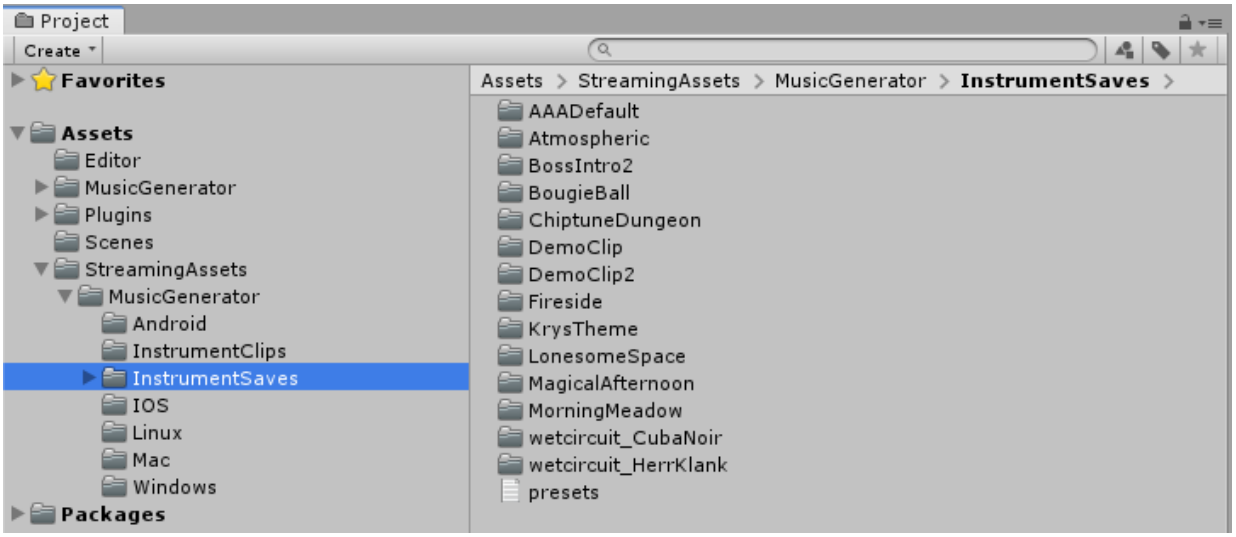


Music Generator - ToolTips index

Player panels

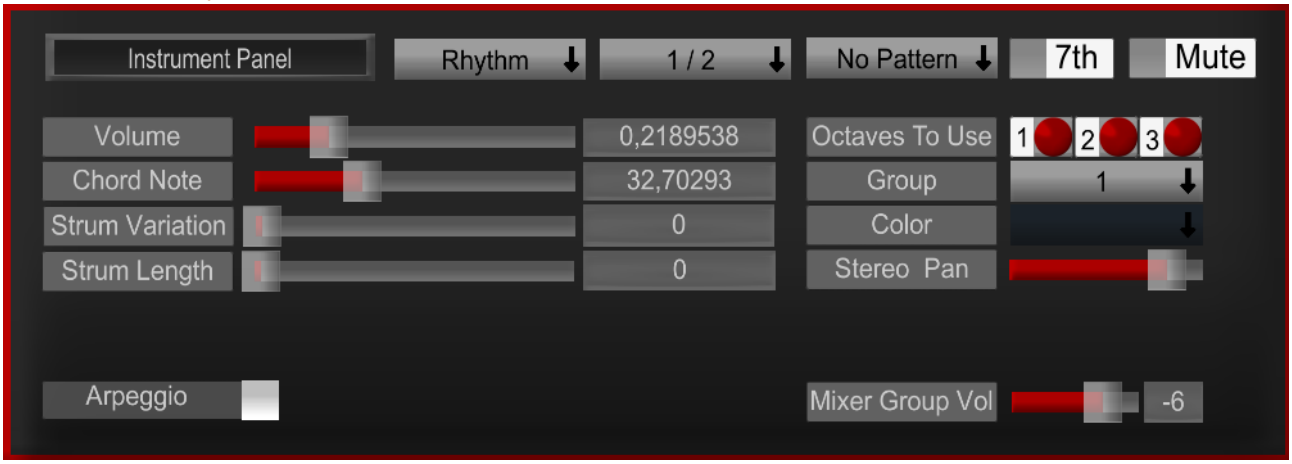


- **Presets**. Loads a preset configuration.
- **ChordProgressionDisplay**. This is the current chord progression and current chord step. So {1,2,3,4}, 2 is playing a chord progression I_II_III_IV and is currently playing the second chord in that progression.
- **Export.** Exports the configuration to file with the name set in the input field.



- **NewInstrument**. Adds a new instrument to the arrangement. While this can be done in real time, it's generally best not to. Use the groups and group rate values if you'd like to have instruments come in and out for different measures.
- **Edit**. Opens the 'Instrument Panel' below, where you're able to edit properties of this instrument.
- **Mute**. Mutes the instrument. Notes / themes / patterns and such are still generated, but not played.
- **Solo**. Mutes all other instruments but this one.
- **Delete**. Deletes the instrument from the arrangement.
- **InstrumentDropdown**. Selects the 'type' of instrument (i.e. violin, cello, etc). To import new instruments, please see the documentation.

Instrument panels



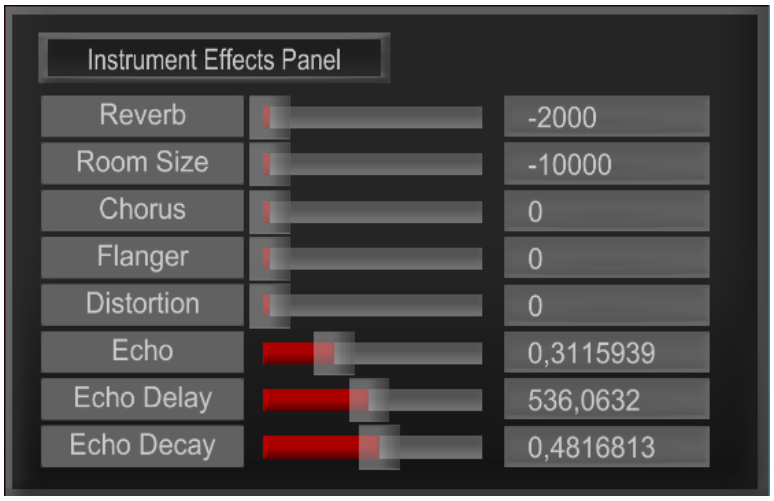
- **Succession**. Sets whether this instrument is a melody, lead, or rhythm. (rhythm plays 100% of the timesteps to which it is set. Selecting melody will play a random chord note and enable other options for 'odds of playing', Lead is similiar to melody, but is not limited to chord notes and can play most notes in the scale. If melody has 100% chord chance, it will always play a full chord.

- **Timestep**. Based on 16 note measures, this is the rate this instrument with try to play. I.e. 1/16 will play 16 times per line. 1/2 will play twice.
- **Pattern**. Uses a (random) pattern to play notes every 4th sixteenth step. Only applies to notes with 1/8 and 1/6 timesteps.
- **UseSevenths**. Enable Seventh chords instead of only triads. This...is only kinda working right now.
- **Mute**. Mutes the instrument. Notes / themes / patterns and such are still generated, but not played.
- **Volume**. The volume of the selected instrument. This is separate from the global volume and only affects this instrument.
- **ChordNote**. The odds that this instrument will try to play additional chordal notes. If this is a lead instrument, the notes are guaranteed to be chord notes (if set to 100% the instrument kinda ceases to be a lead, as it's always playing chord notes.
- **StrumLength**. Length of time between chordal notes. Plays a 'strum' effect. Semi useful for percussion as well.
- **StrumVariation**. Length of randomness between strums. If used carefully, adds a bit of human error to the strums.
- **OctavesToUse**. The available octaves for this instrument. I.e, if 1 and 3 are selected. It will play notes in the first and third octaves for this instrument.
- **Group**. A very useful option. Groups have a variable (set in the master panel) for the odds that the instruments in this group will play during any given measure. Decided at the beginning of the measure, this is useful to add dynamics to the overall feel of things. All the instruments set to a group will either play, or not play, for any given measure (random dice roll each measure, based on the value set in the master panel). See also the Group Style dropdown in the advanced settings panel.
- **Color**. The color this instrument will display in the instrument list panel and staff player.
- **StereoPan**. Distribution of the sound signal to the output speakers (left/right).
- **Arpeggio**. Similar to the strum feature. This will pluck out the chord notes individually in a randomized pattern each step.
- **AudioGroupVolume**. This is the volume slider for the audio mixer group that this instrument is assigned to. Use as a last resort if the instrument volume and master volume aren't able to get the instrument within an acceptable range.

- **PatternLength**. Length of the pattern (in 16th notes). I.e. If your timestep is 1/16 and the pattern length is 4, it will repeat a pattern of the first four notes that play. If it's set to 1/8, it will repeat a pattern of the first two notes that play. It's the highest possible length of notes that could play.

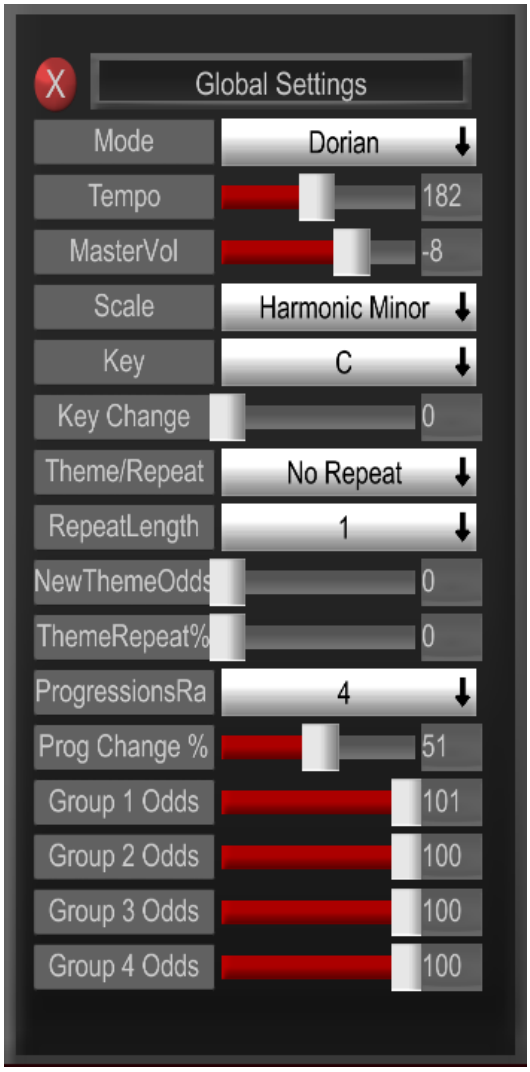
- **PatternRelease**. Number of steps from the end of the measure we stop playing the pattern. Used to create a sense of callback to the pattern, or add resolution.

- **Lead**. Only available for melodic instruments. Breaks the possible notes out of the current chord progression and give a much more distinct melody. Will play any note in the generator's selected scale. Will (hopefully) avoid dissonance. Use caution if adding multiple leads.
- **OddsOfPlaying**. The Odds of this instrument playing. A random roll against this value each timestep determines whether the note will play.
- **MultiplierOdds.** If a note successfully plays, the odds the next note is played are multiplied against this value. (helpful for bunching up groups of notes and reducing how the melody feels. If a note doesn't play, the multiplier doesn't affect the instrument's chance of playing again until a note is successfully played).
- **LeadMaxSteps**. This value is the greatest scale-step a lead instrument can take in a single step.
- **LeadVariation**. This value determines whether the instrument's melody is likely to continue ascending or descending. A value of 0 will randomly ascend/descend while a value of 100 will continue whatever pattern the first two notes create (so if the first two ascend, it will always ascend.
- **PentatonicLead**. This toggles whether the lead instrument will use the pentatonic scale. See: https://en.wikipedia.org/wiki/Pentatonic_scale.
- **LeadAvoids**. This allows you to avoid specific notes of the scale for this instrument to set custom pentatonic scales.



- **Reverb**. How much reverb is added to an instrument.
- **RoomSize**. How large the room is for the reverb variable.
- **Chorus**. Adds a chorus effect.
- **Flanger**. Adds a flanger effect.
- **Distortion**. yup.
- **Echo**. The effect strength of the following two values below.
- **EchoDelay**. Time in between echos for the Echo effect.
- **EchoDecay**. Falloff rate of the echo effect.

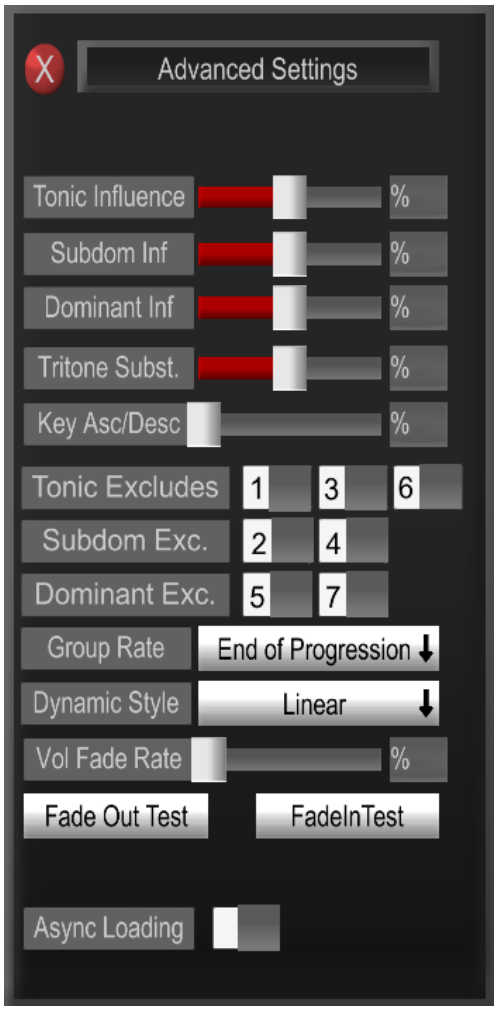
Global settings



- **Mode**. Very useful. Sets the mode of all instruments. Gives a much different overall 'feel'. It's the pitch relationship of the scale/root note.
- **Tempo**. The rate of notes (1/16 notes) per minute. I.e. a value of 100 will play 100 1/16 notes per minute.
- **MasterVol.** The Global volume. Affects all instruments/effects.
- **Scale**. Sets scale for all instruments. To note: Melodic Minor applies to both ascending and descending melodies.
- **Key**. The key all instruments play. Sets the root note of the scale.

- **KeyChangeOdds**. At the end of the measure, this is the odds that a new key is chosen (just goes around circle of fifths in varying directions. See the Key Ascd/Dscd slider in the advanced settings panel to adjust odds of which direction).
- **Theme/Repeat**. Repeat all measures, will repeat every measure (length set in 'RepeatLength' option).'Use Theme', will select a measure to use as the 'theme', and callback theme occasionally, according to the 'Theme Repeat' variable.
- **RepeatLength**. The lenth of measure to repeat (if repeat is selected). A measure is what's visible in the display panel (16 notes).
- **NewThemeOdds**. How often a new theme is selected (based on measures. So 10% will give 1/10 odds at the beginning of each measure to select the last measure as the new 'theme'.
- **ThemeRepea**t. How often a theme will repeat (based on measures. 10% will repeat the theme roughly every ten measures.
- **ProgressionRate**. How quickly the instruments move through chord progressions, A value of 4 will move to the next chord progression every 4 1/16 beats. A value of 16 will play the entire measure (16 notes) before moving to the next chord in the progression. This changes the overall feel of things quite a bit.
- **ProgressionChangeOdds**. The odds that a new chord progression is chosen next measure.
- **GroupOdds**. How often the instruments set to this group (set in the individual instrument panels) will take part and play during any given Measure. Decided at the beginning of the measure. See also: the Group Style dropdown in the advanced settings panel.

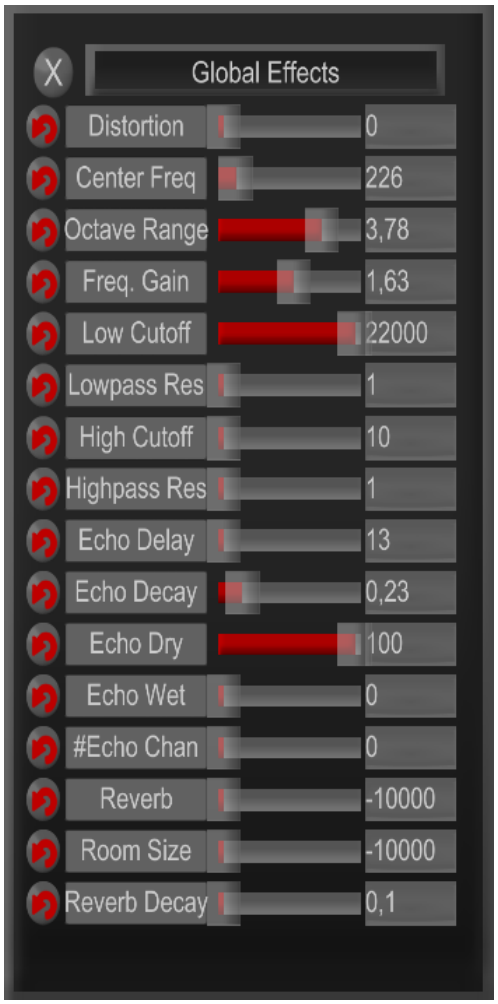
Advanced settings



- **TonicInfluence**. The odds that a chord progression will play a tonic chord (1, 3, 6) for the first or second chord in the progression.
- **SubdominantInfluence**. The odds that a chord progression will play a subdominant chord (2,4) for the second or third chord in the progression. For the second, it will fall back to the Tonic if this random roll fails. For the third, it will fall to the available dominant chords.
- **DominantInfluence**. The odds that a chord progression will play a dominant chord (5,7) for the 3rd or 4th chord in the progression. This will first roll against a subdominant chord if the initial random roll fails, and default to the tonic if both fail.
- **TritoneSubstitution**. Odds a dominant chord will play a flat five substitution instead (augmented sixth. Will add 'tension' to the progression).
- **KeyAscendDescend**. This value is the odds of whether key changes are more likely to ascend or descend (clockwise or counter-clockwise around the circle of fifths).
- **TonicSubdominantDominantExcludes**. If selected, chord progressions will not play that particular chord step during a chord progression. TO NOTE, at least one in each category is required to be unselected. If you want to exclude the entirety of a category, just lower its odds of playing to zero instead. Also, when changing keys this is disregarded as the chord may be needed to smoothly transition to the new key.
- **GroupRate**. Whether new groups are selected at the end of a measure, or the end of a chord progression.

- **DynamicStyle.** Whether the groups are rolled randomly or progress from 1 - 4 in a linear up/down fashion. I.e. In a linear fashion in order for group 4 to play, it would have had to progress through the other groups first and all 4 groups will be playing. Random may have group 1 and 4 playing without 2 and 3. for example.
- **VolumeFadeRate**. The rate at which we fade the volume when FadeOut()/FadeIN() is called on the music generator. Test buttons are included below.
- **AsyncLoading**. Whether we'll load assets asynchronously. If you plan on loading new configurations mid-level and have smaller ram requirements, this is the smoother option for framerates but will take longer to load. Otherwise, load the config at level or app start (if you have plenty of memory overhead, feel free to pre-load instruments for multiple configurations by using MusicGenerator.LoadBaseClips() for each instrument you'll need during the level. This drastically cuts down the time to switch songs, at the expense of longer load times to load in the music assets and the memory needed for them. This is used more for testing in the music editor here than ensuring this config will load asynchronously. Set via code before loading if you want async loading in your app.

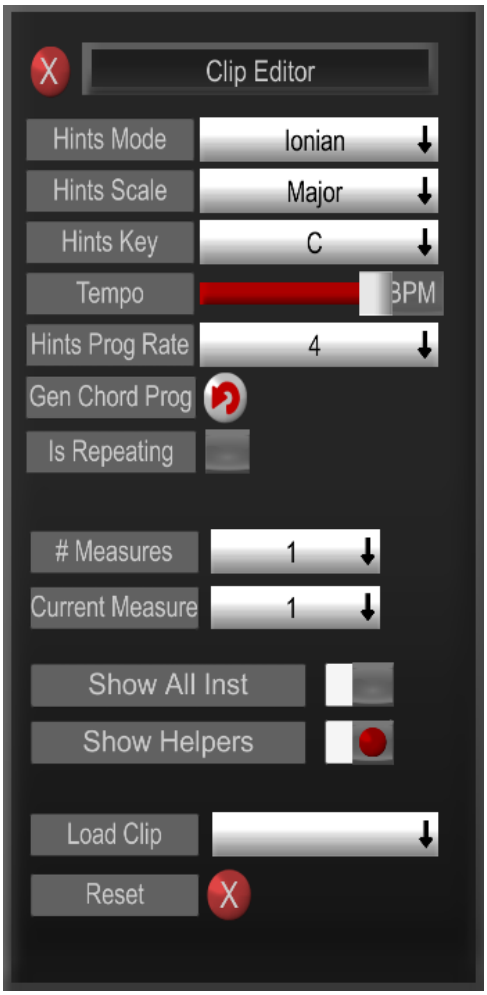
Global Effects



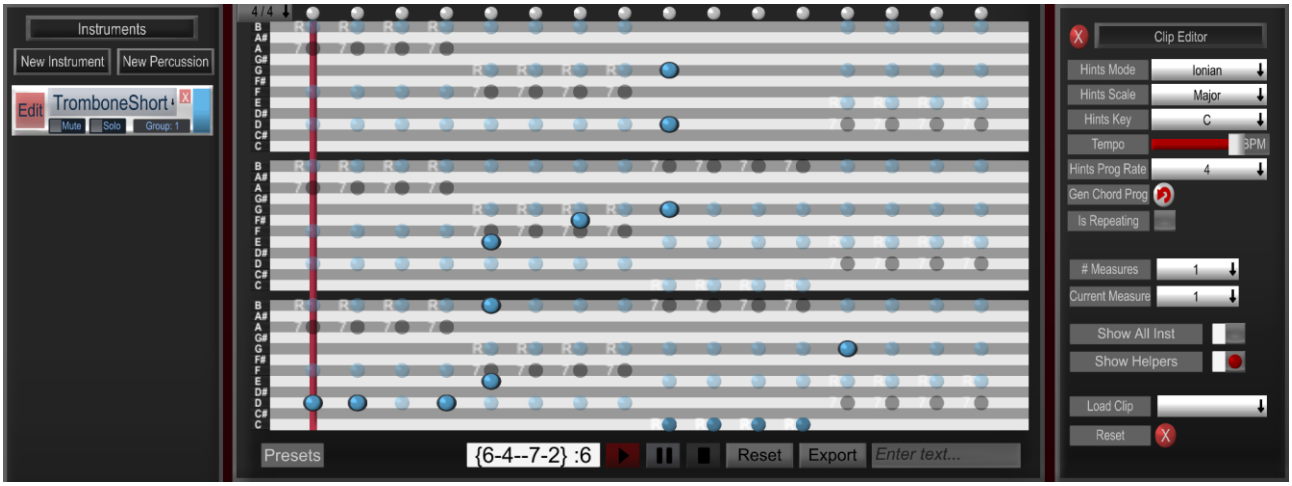
- **Distortion**. Global distortion value.
- **CenterFrequency**. Global ParamEQ center frequency value.
- **OctaveRange**. Global ParamEQ octave range value.
- **FrequencyGain**. Global ParamEQ frequency gain value.

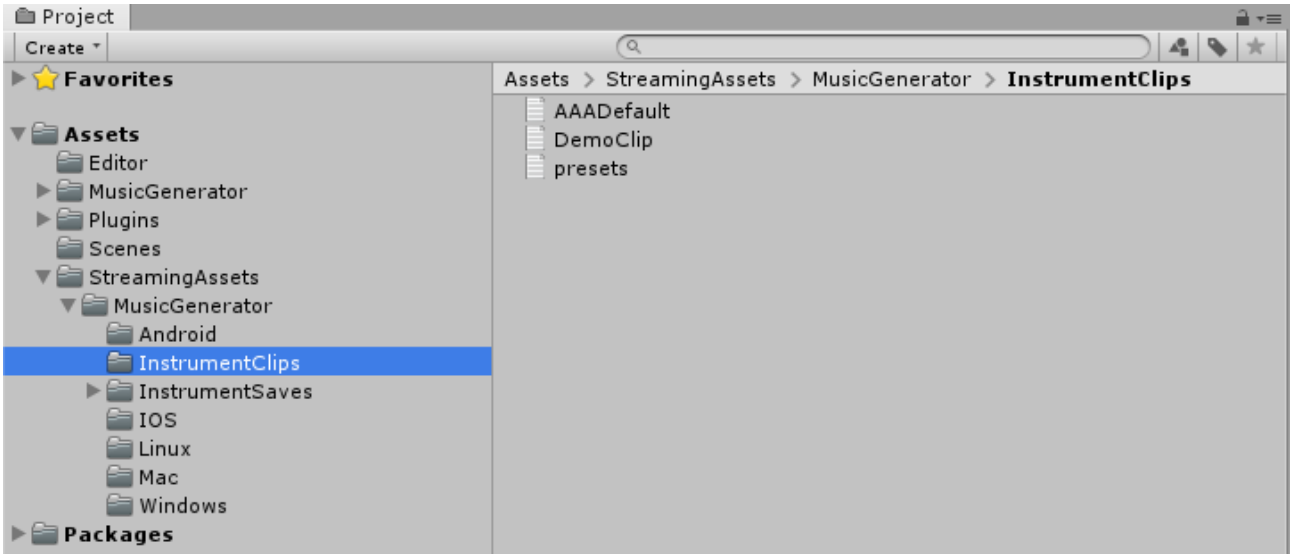
- **LowpassCutoffFreq**. Cutoff frequency for the global lowpass effect plugin.
- **LowpassResonance**. Resonance value for the global lowpass effect plugin.
- **HighpassCutoffFreq**. Cutoff frequency for the global highpass effect plugin.
- **HighpassResonance**. Resonance value for the global highpass effect plugin.
- **EchoDelay**. Global Echo delay value.
- **EchoDecay**. Global Echo decay value.
- **EchoDry.** Dry value for the global echo effect.
- **EchoWet**. Wet value for the global echo effect.
- **NumEchoChannels**. Number of channels for the global echo effect.
- **Reverb**. Global reverb value.
- **RoomSize**. Global room size value for reverb.
- **ReverbDecay**. Global reverb decay value.

Clip Editor



- **ClipRepeat**. Whether or not this clip will export as a repeating or single clip.
- **NumberOfMeasures**. Sets how many measures to enable for this clip.
- **CurrentEditorMeasure**. The currently displayed measure to edit notes/clips for exporting.
- **PlayClipDropdown**. Plays the selected clip.
- **ShowAllInstruments**. Shows the clip notes for all editor instruments.
- **ShowEditorHints**. Shows recommended note options for the editor.


Instruments
New Instrument
New Percussion
Edit
TromboneShort
Mute
Solo
Group: 1
Clip Editor
Hints Mode
Ionian
Hints Scale
Major
Hints Key
C
Tempo
Hints Prog Rate
4
Gen Chord Prog
Is Repeating
Measures
1
Current Measure
1
Show All Inst
Show Helpers
Load Clip
Reset
Presets
{6-4--7-2} :6
Reset
Export
Enter text...


Project
Create
Favorites
Assets
Editor
MusicGenerator
Plugins
Scenes
StreamingAssets
MusicGenerator
Android
InstrumentClips
InstrumentSaves
IOS
Linux
Mac
Windows
Packages
Assets > StreamingAssets > MusicGenerator > InstrumentClips
AAADefault
DemoClip
presets