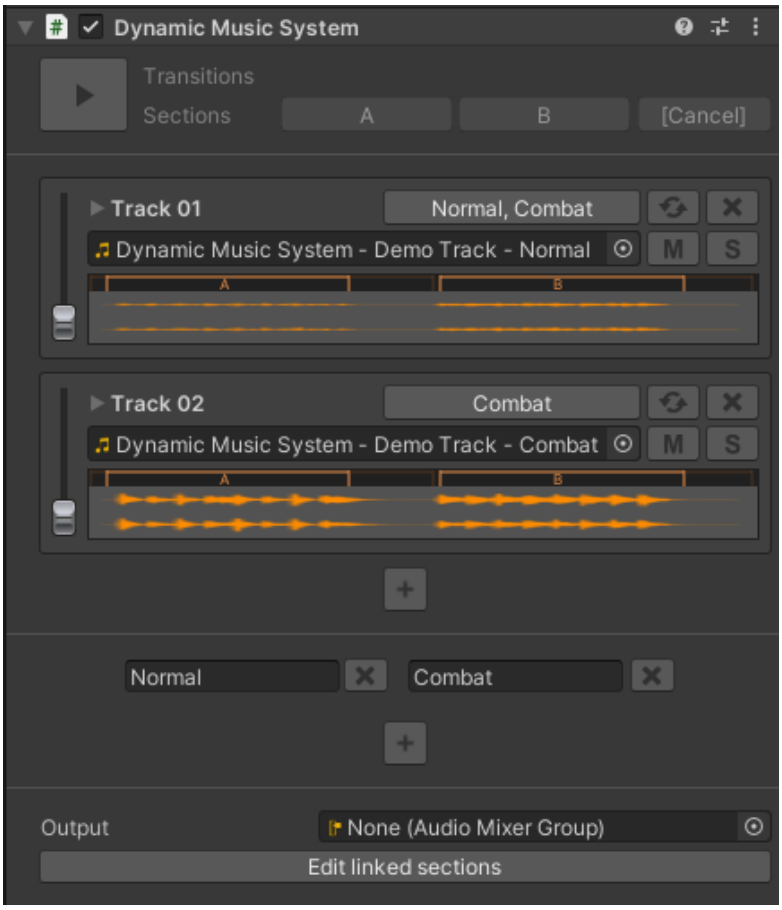
DYNAMIC MUSIC SYSTEM GUIDE

By Mordi Audio



A lightweight and simple, but flexible adaptive music system for Unity.

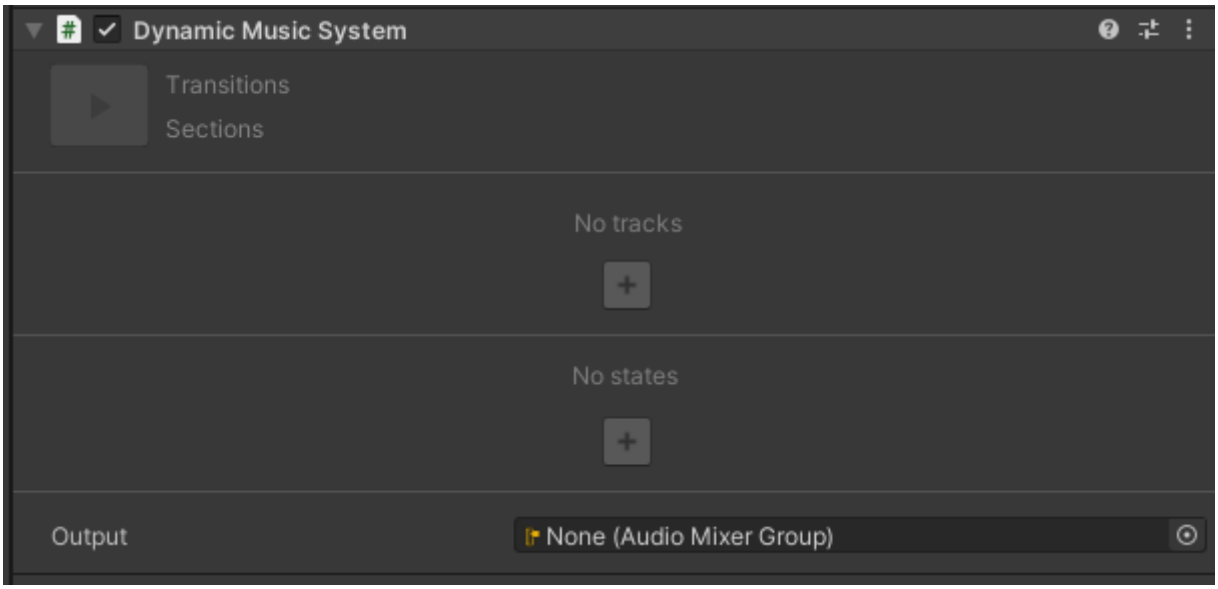
- This asset has been tested in Unity version 2020.3.12f1.
- In order to use all features, you must have access to a DAW or audio editing program that can add cue-markers to wave-files.
- Only wave files are supported, but letting Unity convert to another format is supported.

Main features

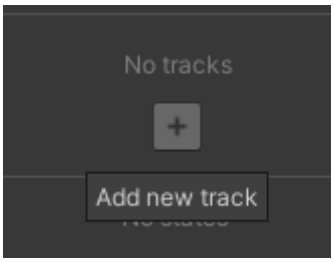
- Horizontal resequencing
- Vertical layering
- Both of the above combined
- Test your configuration directly in the Unity editor
- Read cue markers directly from wave files

Getting started

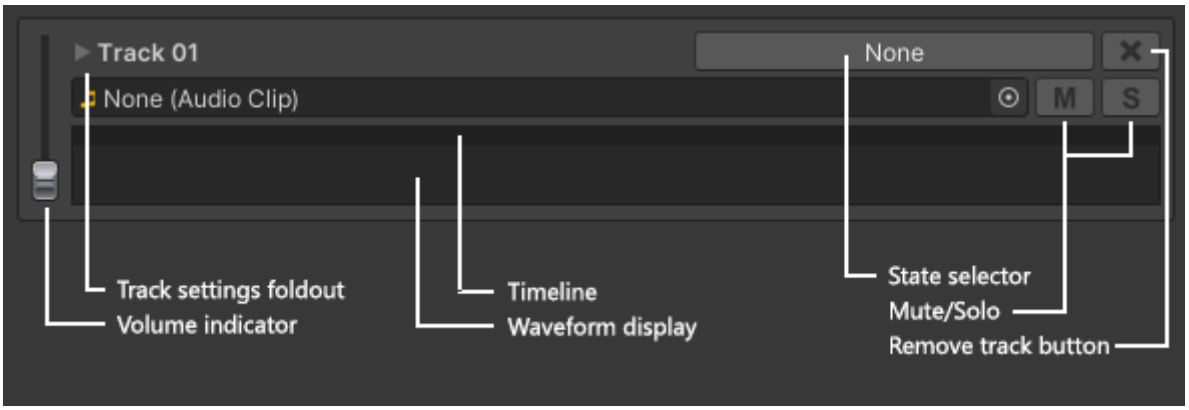
Add the component "Dynamic Music System" to your game object.



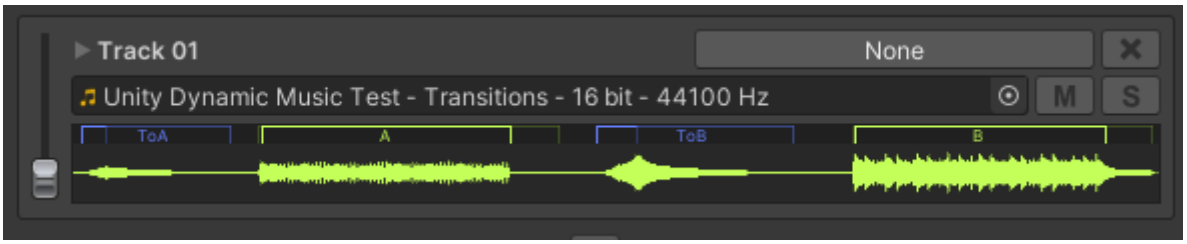
Click the [+]-button below "No tracks" to add a track.



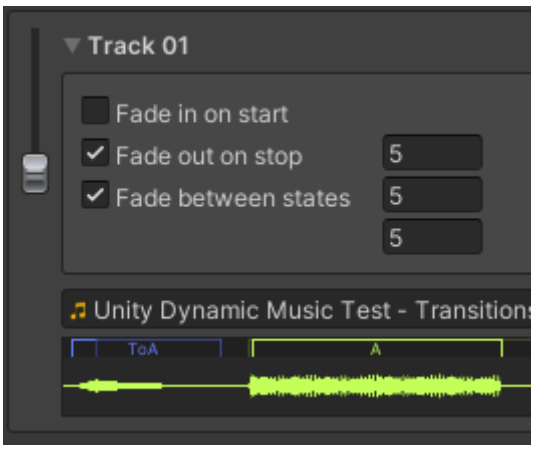
The track control panel appears. Note that the volume fader is only an indicator. It cannot be changed manually.



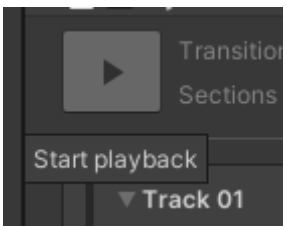
Choose a clip and the waveform will be displayed. Any sections or transitions found in the clip will appear on the timeline. Notice that a few AudioSource components were added to your object. Each track needs two AudioSources to function properly. If any transitions are present, the track needs one additional AudioSource, for a total of three.



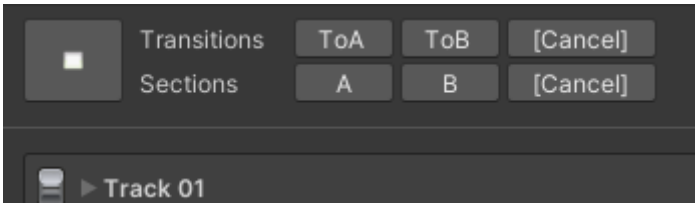
Clicking the track settings foldout lets you change how the track handles certain events. The fade values are given in seconds.



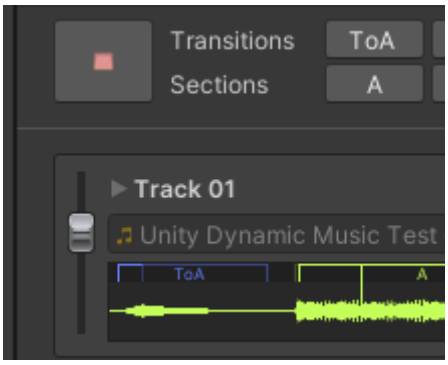
Click the "Start playback" button to audition your clip in the inspector.



While playing, if your clip has sections or transitions, you can queue them by clicking on each corresponding button. Hit "[Cancel]" to clear the queue.



Clicking "Stop" queues a stop-command on all the tracks, meaning that they will either start fading down or continue playing until the end of the last queued section, depending on the track settings. If you click stop again, all tracks will stop playing immediately.

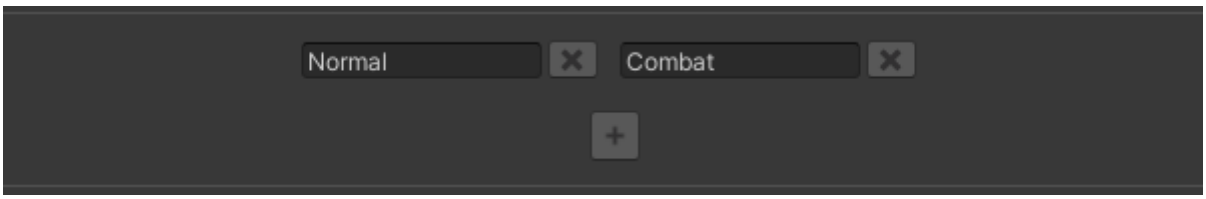


States

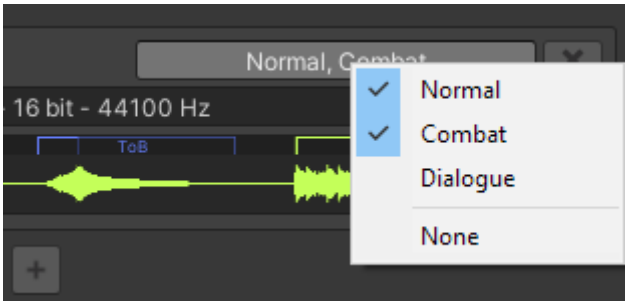
States are used to determine when a track should be heard. This can be used to control the vertical layering of the tracks.

How to use states

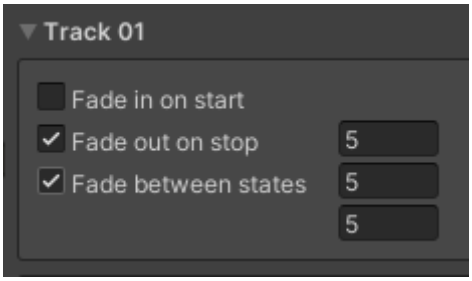
Click the [+]-button to add a state. Give it a short but descriptive name. Keep in mind that the first state will be entered by default when playback starts. Clicking the [X]-button will remove the corresponding state.



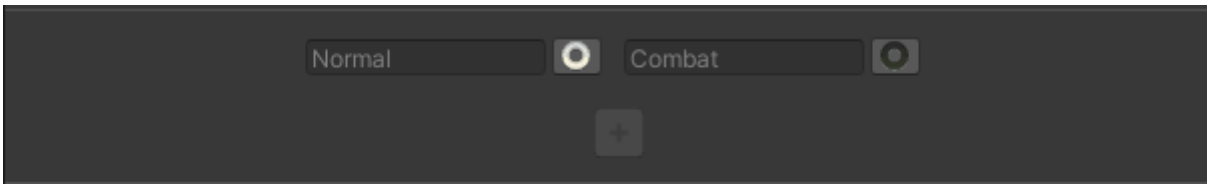
Using the state selector in the track control panel you can add or remove states from that track. Each track can be tagged with multiple states. If no states are selected, the track will play in all states.



Use the track settings to edit how the track enters or exits its states. If "Fade between states" is disabled, the track will play until it hits a "Stop" cue. If enabled, it will fade in or out using the durations entered (in seconds).



During playback, you can enter states by clicking the circle button next to each state. The white circle denotes the currently active state.


Normal
Combat

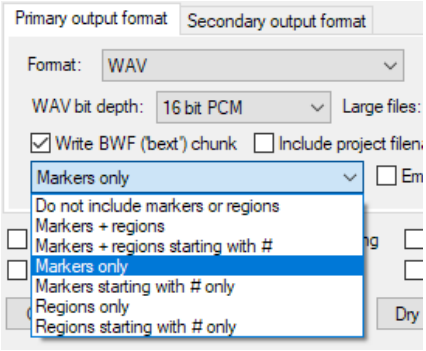
Sections

A section is essentially a vertical "slice" of an audio file, used for horizontal resequencing. By default, sections are looped.

How to add sections

Sections are defined by adding cues (also known as markers) to the wave-file. When a clip is selected, the system reads the file and extracts the cues.

Virtually all DAWs support this feature, but the process is slightly different for each DAW. Usually, there is a way to add markers or cues in the playlist/arrangement window. In Reaper, there is a dropdown setting in the “Render” dialog that lets you embed the markers into the wave file as cues.



A section consists of up to four cues, each of which will need to be specifically named.

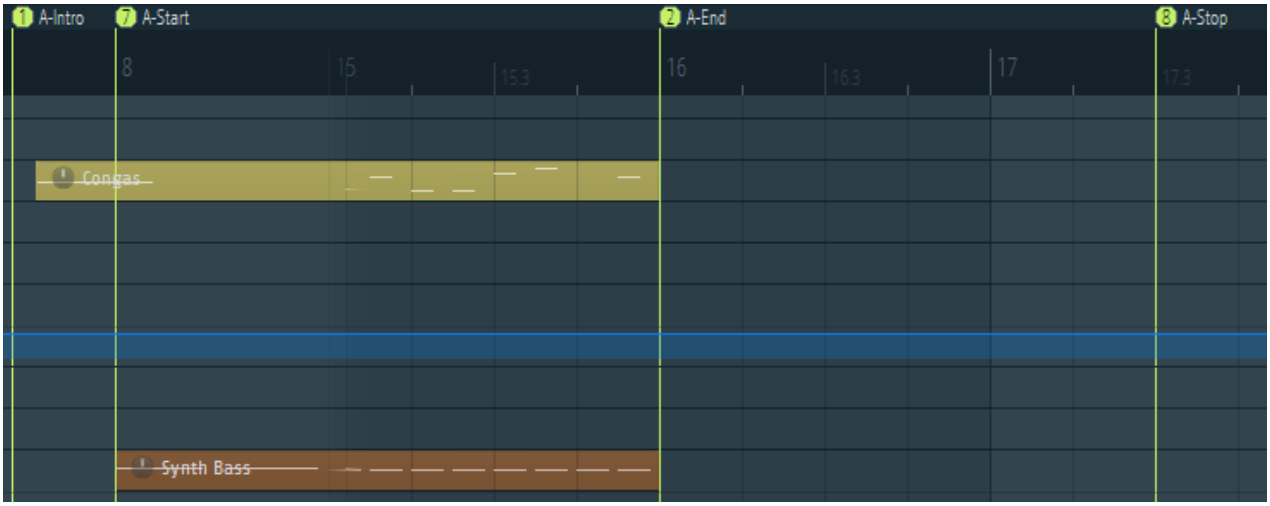
- SectionName-Intro
  - Denotes intro to section. Usually at a silent point right before any sound occurs.
  - The intro cue is optional.
- SectionName-Start
  - Denotes the start of the section.
- SectionName-End
  - Denotes the end of the section.
- SectionName-End-GotoSectionName
  - Same as a regular end cue, but tells the system to enter a specific section. This is called a “linked section”, which can be edited later in Unity.
- SectionName-Stop
  - The point at which the clip will stop playing. Should be placed right after all instruments have ringed out.
  - The stop cue is optional.

"SectionName" can be anything, as long as dashes (-) are not used. I recommend keeping the name short, such as "A" or "Main".

Tips

- Every track plays independently of each other. This can be used to great effect by having sections that are out of sync with each other, for example to create varying ambient musical soundscapes or even ambience sound effects.
- A section can be used as a transition into another section by queuing two sections together using the `QueueSection()` method, or by using the “Goto” keyword in the End cue.

Example

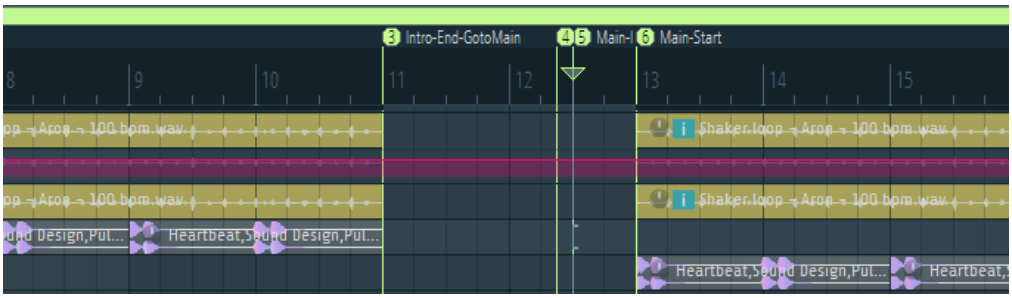


This section has a short intro containing a conga fill. I placed the "Intro" cue right before the fill, and the "Start" cue right on the downbeat. The "End" cue is placed right where the "Start" cue of the next section should happen. The "Stop" cue is placed a few seconds after the "End" cue, which lets the reverb of the conga drum fade to silence while the next section starts.

Linked sections

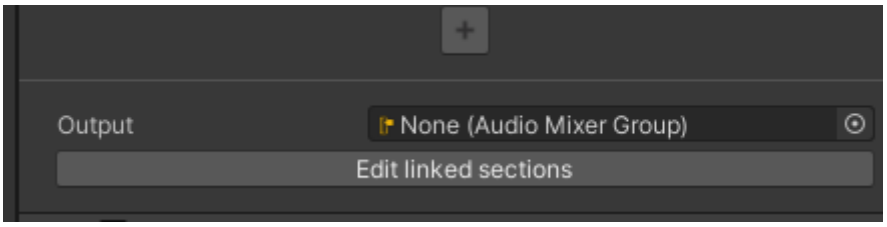
The special “Goto” keyword is used to link sections together. This is useful in several cases.

For example; you have a bespoke intro section in your track that should only play once, and then lead into the main looping section. In this case, you would put “Goto-Main” in the End-cue of your intro.

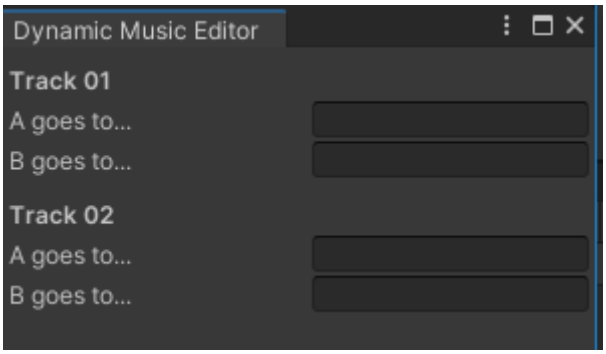


Another example; you want to make it so the track responds more quickly to events in the game, but you still want to have a long, varied piece of the track that loops. To do this, you would break this piece of the track into shorter sections that are linked together. At each section's "End" cue, you would link the next section. This means that the track can switch to any other section at each end-cue, which makes the music feel more dynamic.

Section linking can also be configured using the "Edit linked sections" button at the bottom of the inspector view.



When the button is clicked, a dialog opens up. Here, you can enter the name of any section you want a given section to be linked to. Keep in mind that you need to enter this information for each track. If the tracks do not use the same linked sections, they will eventually play out of sync.



Transitions

Transitions are used to smooth out the shift from section to another. Typically, a transition is a cymbal swell, cymbal crash or a drum fill.

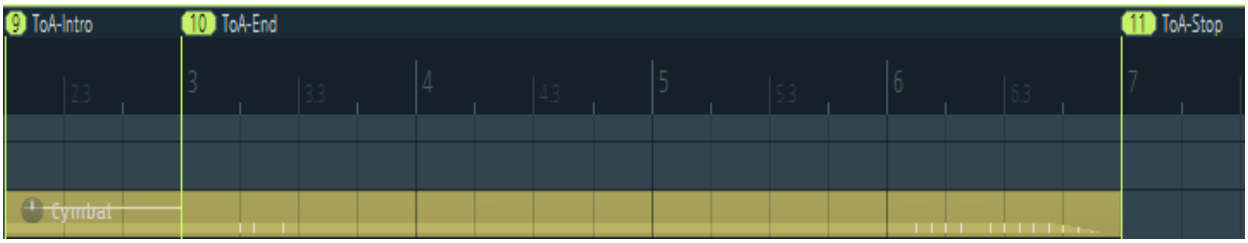
How to add transitions

Transitions are added in the same way as sections, except that a transition does not have the "Start" cue.

- TransitionName-Intro
  - Denotes intro to transition. Usually at a silent point right before any sound occurs.
- TransitionName-End

    - Denotes the "apex" of the transition. This point will by synced to the "Start" cue of the section being transitioned into.
    - Transitions cannot use the “Goto” keyword.
- TransitionName-Stop
    - The point at which the clip will stop playing. Usually at a silent point right after all instruments have ringed out.

Example



This transition consists of a cymbal swell. I added the "Intro" cue right before the cymbal starts swelling. The "End" cue is added right at the point where the queued section should start. The "Stop" cue is located just after the cymbal has stopped making sound. The transition was named "ToA" because it is intended to transition into the section named "A".

How to use during runtime

```
StartPlaybackAllTracks()
```

Used to initialize playback. Takes an optional section name and an optional transition name. If no section is given, it will try to use the first section (in timeline order).

```
GoToSectionAllTracks(sectionName)
```

Tells the system to go to the provided section. Any previously queued sections will be removed from the queue. Takes an optional transition name. This method will start playback if playback has not already been started.

```
QueueSectionAllTracks(sectionName)
```

The provided section will be added to the section queue on all tracks.

```
ClearSectionQueueAllTracks()
```

Removes all queued sections on all tracks.

`SetTransitionAllTracks(transitionName)`

Sets the next transition to be used.

`GoToState(stateName)`

Tells the system to enter the provided state. By default, when playback starts, the first state in the list of states will be entered.

`QueueStopAllTracks()`

Tells each track to stop after the last queued section. If you need it to stop after current section, run `ClearSectionQueueAllTracks()` and then `QueueStop()`. Note that tracks set to "Fade on stop" will not fade when using this method. To fade out, use `StopPlaybackAllTracks()` instead.

`StopPlaybackAllTracks()`

Similar to `QueueStopAllTracks()`, except it will also start fading out tracks that are set to fade on stop.

`StopPlaybackAllTracksImmediate()`

Forces all tracks to stop playing immediately.

Usage examples

Example for a system that has one or more tracks with two sections; "A" and "B", where the B-section should be used for combat and the A-section for non-combat.

```
public DynamicMusicSystem musicSystem;

void OnGameStarted() {
    musicSystem.StartPlayback("A");
}

void OnEnteredCombat() {
    musicSystem.GoToSectionAllTracks("B");
}

void OnExitedCombat() {
    musicSystem.GoToSectionAllTracks("A");
}
```

Example for a system that has only one section, but several tracks. Two states; "Normal" and "Combat".

```
public DynamicMusicSystem musicSystem;

void OnGameStarted() {
    // The first state is used by default (in this case; normal)
    musicSystem.StartPlayback();
}

void OnEnteredCombat() {
    musicSystem.GoToState("Combat");
}

void OnExitedCombat() {
    musicSystem.GoToState("Normal");
}
```

NOTES

- If the Unity Editor loses focus while auditioning in the inspector, the system might not be able to handle sections or transitions correctly. This is not an issue in-game.
- If the time between "Intro" and "Start" is longer than the time between "Start" and "End", the system is probably going to have trouble handle your transitions and sections properly. Internally, the system prepares the next transition or section one second before the intro-time of either (depending on which intro is the longest).
- The component will take control over any existing AudioSource components in the object as it sees fit.
- The system will automatically stop playback if all tracks are either stopped or at zero volume.