Lucas' RadMusic Script Builder

This book documents how to use the RadMusic Script Builder to make custom RadMusic Scripts.

Introduction

What is the RadMusic Script Builder?

Lucas' RadMusic Script Builder is a tool that allows you to convert RMS files to and from an XML format to edit them.

Download

This tool is available from the Donut Team downloads page.

Command Line Arguments

See Command Line Arguments.

XML Format

See XML Format.

Tutorials

See Tutorials.

Version History

1.1.1

This update was released on July 5th, 2018.

- Added error handling when inputting invalid FLAC and Ogg Vorbis files.

Version History

1.1

This update was released on July 2nd, 2018.

- Added support for FLAC and Ogg Vorbis sound files to coincide with the release of the FLAC Support and OGG Vorbis Support hacks for Lucas' Simpsons Hit & Run Mod Launcher.

Version History

1.0

Initial Release on May 29th, 2018.

Command Line Arguments

This is a table of the various command line arguments that the RMS Builder supports.

Command Line Argument	Description	Initial Release
`-inputxml`	Specify an input XML file.	1.0
`-inputrms`	Specify an input RMS file.	1.0
`-outputxml`	Specify an output XML file.	1.0
`-outputrms`	Specify an output RMS file.	1.0
`-outputtypes`	Specify an output text file that will contain RMS types.	1.0
`-rsdpath`	Specify a path or paths to your mod's music files so the tool can get the file size and other information for custom tracks automatically.	1.0

XML Format

Composition

The **Composition** element is used as the root of the file.

It contains **FadeTransition** elements, **StitchTransition** elements, **Event** elements, **State** elements, **RSDFile** elements, **Stream** elements, **Clip** elements and **Region** elements.

```
<Composition SoundMemoryMax="1700000" CacheMemoryMax="2000000" StreamSizeMin="500">
...
</Composition>
```

- **SoundMemoryMax**: Unknown.
- **CacheMemoryMax**: Unknown.
- **StreamSizeMin**: Unknown.

Composition attributes should only be specified on the **Composition** element if it is not the root of a file being included.

Include

An **Include** allows you to load another XML file containing parts of a **Composition**. This is useful for organization.

```
<Include Path="Transitions.xml" />
```

- **Path**: The path to the file to be included.

FadeTransition

Fade Transitions are used to define how to transition from one **Region** to another. They can also contain **Beat** elements.

```
<FadeTransition SourceRegion="M1_main_region" TargetRegion="M1_end_Neg_region" SourceTime="5" SourceSta
<Beat>1</Beat>
<Beat>2</Beat>
<Beat>3</Beat>
<Beat>4</Beat>
</FadeTransition>
```

- **SourceRegion**: The region to handle a transition from. Use "anything" for transitioning from any region.
- **TargetRegion**: The region to handle a transition to. Use "anything" for transitioning to any region.
- **SourceTime**: Unknown.
- **SourceStart**: Unknown.
- **TargetTime**: Unknown.
- **TargetStart**: Unknown.

Beat

```
<Beat>1</Beat>
```

Unknown.

StitchTransition

Unused.

- **SourceRegion**: The region to handle a transition from. Use "anything" for transitioning from any region.
- **TargetRegion**: The region to handle a transition to. Use "anything" for transitioning to any region.
- **TransitionRegion**: Unknown. Optional.

Event

Events are generally used by the game executable and the StageStartMusicEvent command in mission scripts to perform various event actions listed below.

```
<Event Name="M1_start">
	<PlayRegionAction Region="M1_main_region" RegionResumeType="Resume" />
</Event>
```

- **Name**: The name of the event used by the game and via mission scripts.

PlayRegionAction

A **PlayRegionAction** is used to play a **Region** and specify the resume type for it.

```
<PlayRegionAction Region="M1_main_region" RegionResumeType="Resume" />
```

- **Region**: The name of the **Region** to play.
- **RegionResumeType**: Specify if the **Region** should **Restart** or **Resume**.

PushRegionAction

A **PushRegionAction** is used to push a **Region** on top of the stack and play it. It also specifies how to transition to the **Region** being pushed on top and how to transition back when necessary.

```
<PushRegionAction Region="StoneCutters_Tunnel_region" TargetRegionResumeType="Resume" CurrentRegionRes
```

- **Region**: The name of the **Region** to push to the top of the stack and play.
- **TargetRegionResumeType**: Specify if the **Region** being pushed to the top should **Restart** or **Resume**.
- **CurrentRegionResumeType**: Specify if the **Region** currently playing should **Restart** or **Resume** when the **Region** being pushed to the top of the stack is later popped off the top of the stack.

PopRegionAction

A **PopRegionAction** is used to pop a **Region** off the top of the stack.

```
<PopRegionAction Region="StoneCutters_Tunnel_region" />
```

- **Region**: Specify the **Region** to pop off the top of the stack. If the specified region is not currently on top, this action will do nothing. Optional.

StartLayerAction

Unused.

- **LayerName**: The name of a **Layer**.

StopLayerAction

Unused.

- **LayerName**: The name of a **Layer**.

Event (with States)

An **Event** with states is similar to a regular **Event** except it references a **State** element and defines multiple actions inside **StateValue** elements.

```
```

```
<Event Name="M2_start" State="Mission2">
	<StateValue>
		<!--Stage1-->
		<PlayRegionAction Region="M2_S1_start_region" RegionResumeType="Resume" />
	</StateValue>
	<StateValue>
		<!--Stage2-->
		<PlayRegionAction Region="M2_S2_start_region" RegionResumeType="Resume" />
	</StateValue>
</Event>
```

- **Name**: The name of the event used by the game and via mission scripts.
- **State**: The name of the state to use for this event.

StateValue

A **StateValue** should be defined for each state inside the referenced **State**. Each **StateValue** can contain the various event actions listed previously.

```
<StateValue>
	<!--Stage1-->
	<PlayRegionAction Region="M2_S1_start_region" RegionResumeType="Resume" />
</StateValue>
```

State

A **State** is referenced by an event that has multiple different music states that can be controlled by SetMusicState. It can contain **Value** elements.

```
	<State Name="Mission5">
		<Value>Stage1</Value>
		<Value>Stage2</Value>
```

```
</State>
```

- **Name**: The name of the state referenced by the **Event** and used in the first argument of SetMusicState command.

Value

Used to define the names for the states inside the **State** to be used in the SetMusicState command.

```
<Value>Stage1</Value>
```

NOTE: There is a limit of 6 state values among all **State** definitions in any given music RMS file.

RSDFile

An **RSDFile** element defines information about an RSD File.

```
<RSDFile FileName="Simpsons_Theme" Size="7788348" AudioFormatEncoding="PCM" AudioFormatChannels="2" Au
```

- **FileName**: The name of the file without the `.rsd` file extension. This is relative to the `sound\music` folder.

The following attributes should only be specified if the RSD file specified above is not present in any specified RSD Path(s):

- **Size**: The size of the file in bytes.
- **AudioFormatEncoding**: The type of encoding the file uses. Supported types are **PCM**, **VAG**, **PCMB**, **XADP**, **GADP** and **RADP**.
- **AudioFormatChannels**: The amount of audio channels the file contains.
- **AudioFormatBitResolution**: The bit resolution of the samples in the audio.
- **AudioFormatSamplingRate**: The sample rate of the audio.

Stream

A **Stream** is an object that represents a music track. They reference an **RSDFile** and set an appropriate tempo and time signature for it.

```
<Stream Name="Simpsons_Theme" RSDFile="Simpsons_Theme" TempoTrack="172 4/4" />
```

- **Name**: The name of the **Stream** to be referenced by **Regions** and **Transitions**
- **RSDFile**: The name of the **RSDFile** the **Stream** will use.
- **TempoTrack**: The tempo and time signature of the **RSDFile**.
- **TempoTrackStartBeat**: Unknown.
- **Streamed**: Unknown.

Clip

Unused.

- **Name**: The name of the **Clip**.
- **RSDFile**: The name of the **RSDFile** the **Clip** will use.
- **TempoTrack**: The tempo and time signature of the **RSDFile**.
- **TempoTrackStartBeat**: Unknown.

Region

A **Region** contains one or more **Layer** elements.

```
<Region Name="FE_region">
<Layer Name="l">
```

```
    <LogicRepeatEvent>
        <StreamEvent Name="Simpsons_Theme" />
    </LogicRepeatEvent>
</Layer>
</Region>
```

- **Name**: The name of the **Region**.
- **ExitRegion**: The name of a **Region** to switch to when this region ends. Optional.
- **Volume**: The volume of the region's contents. Optional, defaults to 1.0.

Layer

A **Layer** element can contain various types of sequence event listed below. They can also contain **Beat** elements.

```
<Layer Name="l">
    <LogicOrEvent>
        <StreamEvent Name="Sunday_Drive_End_Sus1" />
        <StreamEvent Name="Sunday_Drive_End_Sus3" />
    </LogicOrEvent>
</Layer>
```

- **Name**: The name of the layer.
- **Constant**: Unknown. Optional, defaults to true.
- **Volume**: Specify the volume of the layers contents. Optional, defaults to 1.0.

LogicAndEvent

A **LogicAndEvent** simply executes every sequence event inside it.

```
<LogicAndEvent>
    <LogicRepeatEvent Times="2">
        ...
```

```
</LogicRepeatEvent>
<LogicOrEvent>
...
</LogicOrEvent>
</LogicAndEvent>
```

LogicRepeatEvent

A **LogicRepeatEvent** will repeat the sequence events inside it forever or for the specified number of times.

```
<LogicRepeatEvent MinTimes="2" MaxTimes="3">
<LogicAndEvent>
<SilenceEvent Time="2000" />
<SilenceEvent Time="3000" />
</LogicAndEvent>
</LogicRepeatEvent>
```

- **Times**: Specify the amount of times to repeat the event.

OR

- **MinTimes**: Specify the minimum amount of times to repeat the event.
- **MaxTimes**: Specify the maximum amount of times to repeat the event.

OR

Specify no attributes to loop the sequence events forever.

LogicOrEvent

A **LogicOrEvent** will execute one of the sequence events inside it at random.

```
<LogicOrEvent>
```

```
<StreamEvent Name="Tuba_001" />
<StreamEvent Name="Tuba_002" />
<StreamEvent Name="Tuba_003" />
...
<StreamEvent Name="Tuba_065" />
<StreamEvent Name="Tuba_068" />
<StreamEvent Name="Tuba_069" />
</LogicOrEvent>
```

SilenceEvent

A **SilenceEvent** will play silence for the specified amount of time.

```
<SilenceEvent Time="2000" />
<SilenceEvent MinTime="2000" Maxtime="3000" />
```

- **Time**: Specify the amount of time.

OR

- **MinTime**: Specify the minimum amount of time.
- **MaxTime**: Specify the maximum amount of time.

StreamEvent

A **StreamEvent** will play the specified **Stream**.

```
<StreamEvent Name="Simpsons_Theme" />
```

- **Name**: The name of the **Stream** to play.

ClipEvent

Unused.

- **Name**: The name of the **Clip** to play.

VarVolumeEvent

Unused.

- **Volume**: Unknown.

VarPitchEvent

Unused.

- **Pitch**: Unknown.

VarVolumeRandMinEvent

Unused.

- **VolumeRandMin**: Unknown.

VarVolumeRandMaxEvent

Unused.

- **VolumeRandMax**: Unknown.

VarPitchRandMinEvent

Unused.

- **PitchRandMin**: Unknown.

VarPitchRandMaxEvent

Unused.

- **PitchRandMax**: Unknown.

VarAuxGainEvent

Unused.

- **AuxNumber**: Unknown.
- **AuxGain**: Unknown.

VarPositionalEvent

Unused.

- **Positional**: Unknown.

VarPosFallOffEvent

Unused.

- **PosFallOff**: Unknown.

VarPosDistMinEvent

Unused.

- **PosDistMin**: Unknown.

VarPosDistMaxEvent

Unused.

- **PosDistMax**: Unknown.

CallbackEvent

Unused.

- **CallbackName**: Unknown.

Beat

```
<Beat>1</Beat>
```

Unknown.

Notes

The names of elements and attributes in this XML format are derived from Radical's official names as the RMS format stores all of the RadMusic class and field names within it.

Some other design decisions were derived from the plain text RMS format used in other Radical games such as *The Incredible Hulk: Ultimate Destruction*.

Items in this documentation that say "Unknown." are used in Hit & Run but their functionality is not understood.

Items in this documentation that say "Unused." are not used in Hit & Run and their functionality is not understood.

Tutorials

Tutorials

Converting RadMusic Scripts to XML

1. Extracting RMS Files

In the base game, RMS files are located within `ambience.rcf` and `music01.rcf`. These files can be opened with Lucas' RCF Explorer or extracted with Lucas' Radcore Cement Library Builder.

`ambience.rms` can be found in `ambience.rcf`.

`l1_music.rms` to `l7_music.rms` can be found in `music01.rcf`.

2. Converting to XML

Once you have an RMS file extracted, the simplest way to convert it to XML is to drag it onto the RMS Builder's executable.

This will create an XML file of the same name next to the RMS file.

Tutorials

Building RadMusic Scripts from XML

Method 1 (Simple)

The simplest way to build a RadMusic Script from an XML file is to drag the XML file onto the RMS Builder's executable. This will build an RMS file of the same name next to your XML file.

NOTE 1: This method requires your XML files to be in your Mod's `sound\music` folder if you don't want to manually move the RMS file there each time you build it.

NOTE 2: This method also does not cover mods that have custom music since you would need to specify an RSD path with the `-rsdpath` command line argument for the tool to get information about the file.

Method 2 (Complex)

You may want to keep your XML files somewhere other than your Mod's `sound\music` folder. You can do this using command line arguments to tell the RMS builder where to build the output RMS file.

We recommend doing this with a batch file. You'll need to use the `-inputxml`, `-outputrms` and `-rsdpath` command line arguments. You'll need to adjust these paths to your specific setup but here's an example:

```
@"C:\path\to\LRMSB.exe" -inputxml "%~dp0Build.xml" -outputrms "C:\path\to\YourMod\CustomFiles\sound\mus
```