Chiptune Player Documentation (v1.2.2)

by Mick @ GamePhase

(For Gamemaker Studio)

Introduction

Chiptune Player is an extension that uses game_music_emu (GME) and audio buffers in GMS to play chiptune music in several formats including NSF (NES, Nintendo Entertainment System), GBS (Nintendo Game Boy) and SPC (Super Nintendo). Chiptune Player currently supports Windows and Android. Individual voices can be muted and tempo can be changed in real time. An example of usage is included in the extension.

License

game_music_emu is released under LGPL (GNU Lesser General Public License), so you are allowed to use it in commercial games (since the library is dynamically linked to your project). To comply with LGPL, the source code for the shared library need to be included in your game project (the source code I used to build the library is included with this asset and is only 154KB, the archive also includes instructions for building the library on the different platforms). You should also mention in your game credits that you are using the game_music_emu library. The game_music_emu project can be found here: https://bitbucket.org/mpyne/game-music-emu.

Quick Start

To use Chiptune Player in your project, you need to add the Chiptune Player extension to your project (from Marketplace / My Library in Gamemaker Studio). The object *obj_gme_demo* is an example (for testing) and can safely be deleted from your project (it's recommended that you take a look at it first).

Once the extension is added to your project you can simply use the code below to load and play a song. You don't need to add obj_gme to your rooms, it's a persistent object and an instance of it will be added automatically the first time the script GME_LoadSong(...) is called.

```
sound_index = GME_LoadSong("test.nsf"); // Load a song
if(sound_index != noone) { // Check if song has been loaded
  GME_StartTrack(0); // Start the first track (subtune)
  GME_Play(); // Play the track
}
```

For more advanced use, check *obj_gme_demo* and the documentation of scripts below.

Important notes for pre-gradle GMS1.4

If you have a version of GMS1.4 older than 1.4.1675 (pre-gradle), you need to remove the *libgme.jar* file from the following folder in your project: "yourproject.gmx\extensions\GameMusicEmu\AndroidSource\libs". If you don't delete that file, you will be unable to compile the project for Android.

Scripts

GME_LoadSong(filename)

Arguments: filename (string)
Returns: The index of the sound queue on success, *noone* otherwise.
Description: Load a song file with the given filename. If the file can't be found or another problem is encountered, the value noone will be returned, otherwise the index of the sound queue is returned. The returned sound queue index can be used with the normal audio_ functions (audio_sound_gain etc.)

GME_NumTracks()

Arguments: none
Returns: The number of tracks of the loaded song (integer)
Description: This script will return the number of tracks (or subtunes) in the loaded song, use with GME_StartTrack() explained below.

GME_StartTrack(track_number)

Arguments: track_number (integer)
Returns: 1 on success, 0 otherwise
Description: You need to set the track (subtune) before calling GME_Play() or any other scripts below. You can get the number of tracks in the song with the script above.

GME_Play()

Arguments: none
Returns: nothing
Description: Play the loaded song / track.

GME_Pause(pause)

Arguments: pause (bool)
Returns: nothing
Description: Pause / unpause the loaded song / track. Calling GME_Play() after this will resume playback at the paused position.

GME_Stop()

Arguments: none
Returns: nothing
Description: Stop the loaded song / track. Calling GME_Play() after this will restart the track from the beginning.

GME_SetTempo(tempo)

Arguments: tempo (float)
Returns: nothing
Description: Set the tempo of the song / track. Setting this to 1.0 will result in the normal tempo for the track, 0.5 will set tempo to half, 2.0 will be double tempo etc.

GME_NumVoices()

Arguments: none
Returns: The number of voices (integer)
Description: Returns the number of voices (or channels) in the song / track.

GME_MuteVoice(voice, mute)

Arguments: voice (integer), mute (bool)
Returns: nothing
Description: Mute or unmute a voice (or channel) of currently loaded song / track. Setting the mute argument to true (or 1+) will mute the voice, setting it to false (or 0) will unmute the voice. The voice argument is zero based, meaning the first voice is numbered 0 and so on.

GME_MuteVoices(mask)

Arguments: mask (bitmask)
Returns: nothing
Description: With this script you can mute / unmute multiple voices at once. This is achieved using a bitmask. The least significant bit is the first voice. The value 5 (B00000101 in binary), will mute voices 0 and 2 and unmute all other voices. If you unused to binary values, you can use a loop with the GME_MuteVoice function instead.

GME_GetTrackLength()

Arguments: none
Returns: Track length in milliseconds
Description: Get the track length in milliseconds. Track length is not available for all supported music formats. If the length is not available, -1 is returned.

GME_GetPosition()
Arguments: none
Returns: Current track position in milliseconds

GME_SetPosition(position)
Arguments: position in milliseconds (integer)
Returns: 1 on success, 0 otherwise
Description: Due to the nature of chiptunes, a seek is required to set a position, this can be slow.

GME_GetName()
Arguments: none
Returns: The song name
Description: An empty string will be returned if the song name is unavailable

GME_GetAuthor()
Arguments: mask (bitmask)
Returns: nothing
Description: An empty string will be returned if the author is unavailable

GME_GetComment()
Arguments: mask (bitmask)
Returns: nothing
Description: An empty string will be returned if a comment is unavailable

GME_GetCopyright()
Arguments: mask (bitmask)
Returns: nothing
Description: An empty string will be returned if no copyright info is available

GME_Free()
Arguments: mask (bitmask)
Returns: 1 on success, 0 otherwise
Description: Free the music player object

Troubleshooting

Do you experience noise in music playback? If you have a room / game speed of lower than 60, you may need to increase the number of buffers used by the Chiptune Player extension. You can do this by increasing the value of the variable *buffer_count* to 20 (or even higher) in the create event of *obj_gme*.