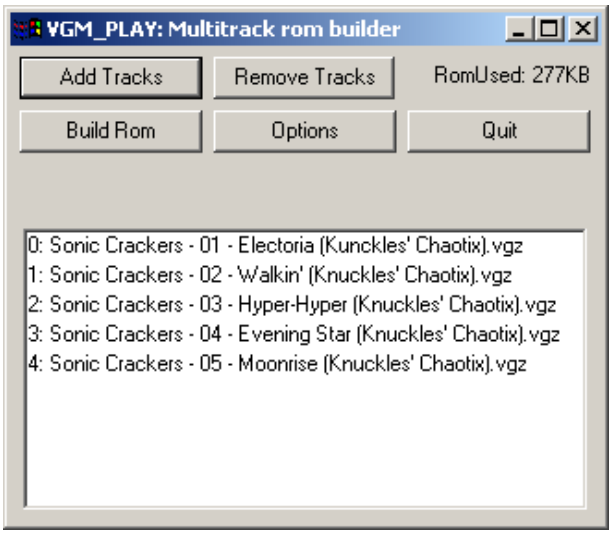
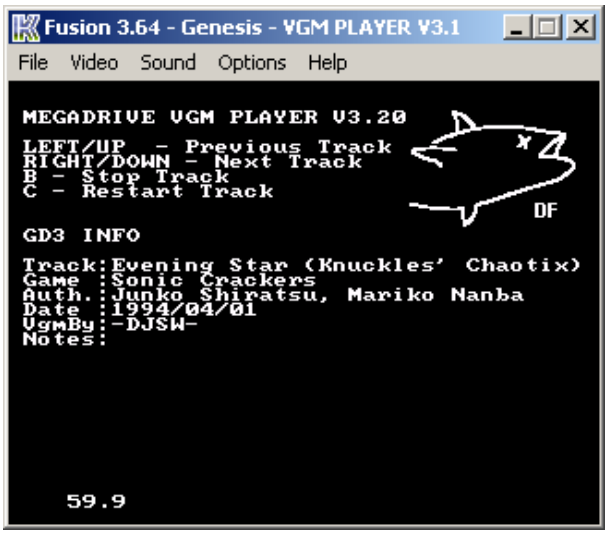
Sega MegaDrive Vgm Player

VGM_PLAY V3.41
VGM_PLAY V3.30
VGM_PLAY V3.11
VGM_PLAY V2.01
V3.20 Example Rom





Features

- Works on real hardware
- Correct playback & Accurate timing
- Track selection & Auto advance
- Displays track GD3 Information
- GUI Rom builder
- Supports Drag/Drop and M3U playlists
- Emulator calling / Killing
- Rom file padding & playback rate scaling
- Duplicate write removal, optimization

Project History

The sega megadrive vgm player is the first project I ever released,
it has been in development since 2007 and has gone through 3 major versions.

Version 1: 2007
This version used both the z80 and 68k, the 68k decoded the vgm data into register writes and the z80 wrote that decoded data to the hardware at a predicatable rate. It could not be ran on the real hardware due to requiring 64kb of shared ram betweent the z80 and 68000, no such programmable cart supplied such sram. This version died very quickly.

Version 2: 2008-2009
This version was 68k only and worked on the real hardware. It managed to play most vgm files without problems and with correct timing. It came with a console UI rombulder where multiple vgm files could be selected and built into the one rom and on playback displayed the gd3 information for the track. Some vgm files had weird fucked up starts with hundreds of redundant registers writes with zero delays between causing an unplesant sound at the start. This version mitigated this

problem by collecting numerous writes together into a single state update preceding the first register write with delay.

Version 3: 2009-
A major problem with Version 2 was the enormous size of vgm dumps, the number of tracks that could be fit into a 4MB rom was severly limited and in some cases a single track was larger than 4MB. Version 3 converted the vgm files into a custom format to save space allowing a rom to contain more tracks. Along with encoding optimizations duplicate register writes were removed and dac writes were averaged to produce run length sample rate pairs. The rombulder was upgraded to win32 GUI supporting drag and drop and also m3u files. For some files the compression ratio was greater than 90%.

Known uses:
DefleMask Tracker
  Default rom for MegaDrive
16Bit AudioPhile Project
  Playback of vgm sets on the real hardware for recording.
  Suplied custom version with playback rate scaling,
  by default the playback speed is slightly wrong (less than 1%)

Duplicate Write Removal (DWR)

The vgm data is optimized to reduced size by eliminating duplicate (redundant) register writes. With some sound engines the majority of register writes are redundant so duplicate write removal can make large savings. The VGM set for the game "HellFire" results in a 54MB rom with DWR dissable, enabled brings this down to ~4MB. In older versions of the vgm player (V3.00-V3.11) DWR was not applied to frequency registers due to the complexity related to the MSB-LSB latch system. In V3.20b DWR was extended to freq regs, it was disscovered later that this change broke real hardware playback (I should have done some tests). Most emulators handle the frequency latch incorrectly, they implement a seperate latch for each channel whilst the real hardware has one single latch for the entire chip, other emulator specific weirdness exists, I was able to construct a vgm file which sounds different on the real hardware, Gens(original) and KegaFusion. I did not want to give of freq reg DWR as it results in massive savings for some VGM sets. The DWR was split into 3 modes, "safe mode", "emulator mode", and "hardware mode". The savings in hardware mode are generally less than with the "emualtor mode" due to the shared freq latch but can still be significant. Eliminating redundant writes on the shared freq latch took some effort, a write to the latch could only be determined to be redundant at some point in the future when the data would either be utilised by a LSB write or overwritten without being used (I nearly went mad during debug).

MSB Frequency Latch Emulation

Many emulators incorrectly emulate the MSB freq latch. The real hardware has a single MSB latch which is shared between all the channels, Fusion emulates the freq latch but incorrectly in that each channel has its own latch. Gens does not emulate the latch behavour at all, any freq write LSB or MSB takes effect imidiatly. Because of these differences it is possible to construct a vgm file that sounds different on each of these systems. An example of this follows.
weird-test.vgm
weird-test.bin

0:00 / 0:17

Fusion3.64

gens2.12

0:00 / 0:17

0:00 / 0:15 mess0155b