Conserving an Ancient Art of Music: Making SID Tunes Editable

Gerald Friedland, Kristian Jantz, and Lars Knipping

Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Germany
{fland,jantz,knipping}@inf.fu-berlin.de

Abstract. This paper describes our approach for editing PSID files. PSID is a sound format that allows the original Commodore 64 synthesizer music to be played on modern computers and thus conserves a music subculture of the eighties. So far, editing PSID files required working directly with Commodore 64 machine language. The paper gives a small overview of sound synthesis with the Commodore 64, argues why this topic is still interesting for both musicians and computer scientists, and describes our editing approach.

1 Motivation

Many users still remember the Commodore 64 (C64) which took over the market in 1982. It was the best-selling single computer model ever [1]. Jointly responsible for the success were its, for that time revolutionary, music synthesizing capabilities. For creating sounds, the C64 was equipped with a coprocessing synthesizer chip called Sound Interface Device (SID) [2]. Comparable audio quality on IBM compatible PCs has not been offered until years later. Before 1987, when the AdLib soundcard came out, PCs only produced beep tones. As a result, the C64 was used as a standard tool to create sounds for years. Thousands of compositions produced then are still worth listening to even though their sound is quite artificial. Many of todays respectable computer musicians made experiences with the C64[1], like Rob Hubbard[2] who worked from 1988 to 2001 as Audio Technical Director at Electronic Arts. The celebrated Chris Hülsbeck[3], who wrote many titles for C64 as well as Amiga and Atari games, is now composing music for Nintendo Game Cube. SID is a subculture that is still very existing. The High Voltage SID Collection[4], for example, is a large collection of music files. SID music databases are still well maintained, not only because of the unique sound, but also because it reminds people of a culture of their youth [3].

From a computer scientist's point of view, SID files are also interesting because they consist of program code for the C64 that has been extracted from original software. Players emulate the SID and parts of the C64. In contrast to

[1] http://www.c64gg.com/musicians.html
[2] http://www.freenetpages.co.uk/hp/tcworh/
[3] http://www.huelsbeck.com
[4] http://www.hvsc.c64.org

U.K. Wiil (Ed.): CMMR 2003, LNCS 2771, pp. 74–81, 2004.

Table 1. Comparing SID and MIDI

Song	MIDI	SID
Smalltown Boy	51,977	7,601
Das Boot	47,957	4,222
Popcorn	25,486	6,758
Giana Sisters Theme	21,568 (1 song)	23,517 (8 songs)

regular multimedia formats, SID files are computational complete. Converting the format to a modern sampling format or MIDI results in information loss, e.g. endless loops or sounds that depend on random numbers cannot be converted. Editing SID files therefore means automatized editing of a computer program. Modern techniques like flow analysis need to be applied here. The simple architecture of the SID Chip and the C64 makes this kind of music especially suitable to teach the basics of music synthesis and computer programming. The C64 did not have much memory, it came along with only 38911 free bytes of program memory forcing everybody to write economic code[5].

We believe the SID is an important part of computer music history. Our goal is to save this knowledge academically by writing about it and make it accessible for teaching. Giving comfortable editing facilities allows people to experiment with SID sounds.

2 Comparing SID Files with MIDI

Because of the memory restrictions of the Commodore64, SID files are very small. Table 1 gives a comparison of SID and MIDI file sizes in bytes.

Of course the tunes do not sound the same. MIDI files sound less artificial as they are played back with more sophisticated synthesizers. MIDI players also use wave tables to make melodies sound even more natural. A wave table can contain high quality recordings of a real instrument at various pitches. SID was not designed to play sampled sound and wave tables were not used. A composer had to skillfully combine the available waveforms to create the illusion of real music instruments. However, SID creates interesting sound effects and melodies with only a fraction of the size of a MIDI file. Note, that most of the code in SID files shown in the Table has been extracted manually from computer games and may contain graphic control as well as game logic. When playing SID files, this part of the code is redundant and could be eliminated. So the actual code needed to play the sound is even smaller. In both size and quality, SID files are comparable to Yamaha's Synthetic Music Mobile Application Format (SMAF)[6] files used for mobile devices, but SID files are computational complete and can also generate sounds based on calculations. This makes the SID format an interesting choice for storing sound effects. Theoretically it is also possible to create a SID tune that is different every time it is played.

[5] Although experienced programmers knew how to get a few bytes more out of it e.g. by unconventional use of the video memory.

[6] http://www.smaf-yamaha.com

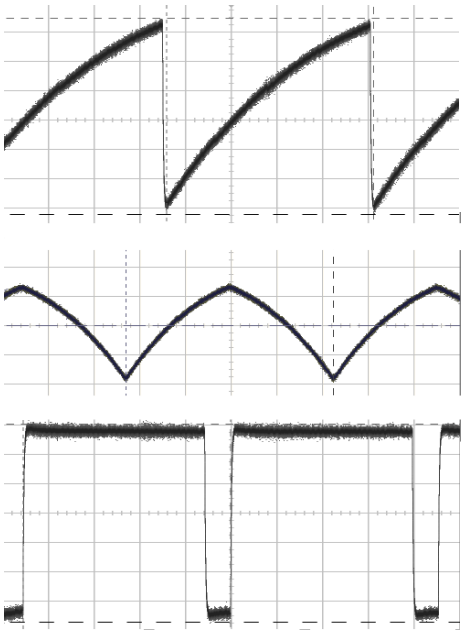

Fig. 1. The SID supported the three shown waveforms and noise

3 Technical View on the SID

The SID chip (MOS 6581) is a synthesizer. It creates tones by overlapping the waveforms sawtooth, triangle, square and noise (Figure 1). The SID is programmed by 28 registers using memory mapped I/O [2]. Music is generated by a regular C64 program that sets values to the registers. A sound is defined by four parameters: pitch, volume, timbre and the volume dynamics. The latter two parameters allow us to differentiate various instruments. The SID chip has three voices that can be manipulated independently. The pitch is controlled by the wave frequency, the volume can be set directly for each voice and the timbre results from the selected waveform. In addition one can create absorbing effects with the help of programmable filters. SID filters have low pass, band pass and high pass filter sections to manipulate frequencies from 30 Hz to 12 kHz. The volume dynamics is controlled by an envelope generator. The generator describes the speed at which the sound rises to his maximal volume and how fast it dies away afterwards. One can also set the volume at which a tone will be held. A musician composing a song was forced to define every single sound on its own to produce the desired song [4]. The C64 User Manual gives tables for looking up the values for waveform and volume dynamics to simulate different instruments as well as frequency tables for different notes. Figure 2 shows an excerpt from a table of frequency values. A composer's task was to put everything together into a program. See Figure 3 for an example. For efficiency reasons, this was usually done in machine code instead of C64 Basic language.

Nr.	Note-Oktave	Frequenz(Hz)	Parameter	Hi-Byte	Lo-Byte
0	C-0	16.4	278	1	22
1	C#-0	17.3	295	1	39
2	D-0	18.4	313	1	57
3	D#-0	19.4	331	1	75
4	E-0	20.6	351	1	95
5	F-0	21.8	372	1	116
6	F#-0	23.1	394	1	138
7	G-0	24.5	417	1	161
8	G#-0	26.0	442	1	186
9	A-0	27.5	468	1	212
10	A#-0	29.1	496	1	240
11	H-0	30.9	526	2	14
12	C-1	32.7	557	2	45
13	C#-1	34.6	590	2	78
14	D-1	36.7	625	2	113
15	D#-1	38.9	662	2	150
16	E-1	41.2	702	2	190
17	F-1	43.7	743	2	231
18	F#-1	46.2	788	3	20
19	G-1	49.0	834	3	66
20	G#-1	51.9	884	3	116
21	A-1	55.0	937	3	169
22	A#-1	58.3	992	3	224
23	H-1	61.7	1051	4	27
24	C-2	65.4	1114	4	90

Fig. 2. Frequency values for certain notes from [4]

```
10 REM MICHAEL ROW THE BOAT ASHORE
20 SI=54272:FL=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 POKE L,15:POKE TH,13:POKE TL,15:POKE A,3*16+16:POKE H,9
40 READ X:READ Y:READ D
50 IF X=-1 THEN END
60 POKE FH,X:POKE FL,Y
70 POKE W,65
80 FOR T=1 TO D:NEXT
90 POKE W,0
100 GOTO 40
110 DATA 17,103,250,21,237,250,26,20,400,21,237,100,26,20,250,29,69,250
120 DATA 26,20,250,0,0,250,21,237,250,26,20,250,29,69,1000,26,20,250,0,0,0,250
130 DATA -1,-1,0
```

Fig. 3. C64 Basic program for the song "Michael row the boat ashore" from [4]

4 Playing SID Files on Modern Computers

SID tunes were originally C64 programs executed by the computer's 6510 CPU. To make them playable on a modern PC, one needs to extract the relevant parts of the code from the program and transfer it to a PC. This process is called ripping and requires knowledge of C64 assembly language. It is hard work for a "ripper" to decide what is player code and what belongs to the game. He or she only sees assembler code. Because of these required skills there is only a handful of people that convert C64 music into SID tunes[7].

The most popular SID format on the PC is the so called PSID format. It consists of a header containing copyright information, the name of the song, the author, and technical parameters that control different properties like the speed of the tune. The header also defines three memory addresses: the load, init and play address. The load address tells the player where the SID tune is located in the memory of the emulated C64. The init address contains a reference to code executed before playing. The play address is frequently called by the SID file

[7] http://www.geocities.com/SiliconValley/Lakes/5147/sidplay/doc_ripping.html

player to simulate a periodically called interrupt handler which, on the original machine, was used to play the music in the background. If the SID file relies on the init method to play back the music, the call to the play method is omitted. There is a variety of standalone SID players available for example SIDPlay2[8] and SIDPlayer[9]. There are also plug-ins for popular players such as WinAmp[10].

5 The Supporting Framework: Exymen

Exymen [5] is an open source editing tool for arbitrary media formats. It offers an application programming interface (API) allowing to extend the editor for any multimedia format and filtering effect. The extension mechanism is based on an implementation of the OSGi standard [6]. This mechanism, which originally comes from the field of ubiquitous computing, manages loading, updating, and deleting software components from the Internet while the application is running. The concept also permits plug-ins to extend other plug-ins.

Our SID editor is a software component for Exymen, a plug-in that extends Exymen to handle SID files. The SID editing plug-in allows to import the PSID file format into Exymen. Users can edit the files by deleting or creating waveforms, doing cut and paste operations or merging other PSID files. The main advantage of using the Exymen API is that we did not have to implement a new graphical user interface for the PSID editor but can integrate it conveniently into an existing one. Yet, there is a variety of media formats that can be edited with Exymen and hopefully soon to be combined with SID tunes.

6 Editing PSID files

As explained above, PSID files contain two addresses required for playback, the init address and the play address. The playing environment first calls the code located at the init address. It is used to set up the environment by initializing certain memory fields needed later in the program. The init method can also be used to decompress data. Simple decompression algorithms have been used on the C64 already. It is even possible to play back the whole sound in the init method. However, the more popular method to play back SID files is to use the play address. This address is called at some predefined frequency by the SID player. To make editing SID tunes easier, we convert it into a simple event format. The conversion is done emulating the 6510 CPU and listening to the relevant memory addresses that map SID registers. See Table 2 for an overview.

The registers 25 to 28 do not manipulate the sound created by the SID chip. Instead registers 25 and 26 are used to connect input devices and receive their values. Registers 27 and 28 were used by game programmers to get pseudo random values, since they represent the current values of the envelope generator and the oscillator.

[8] http://sidplay2.sourceforge.net/
[9] http://www.uni-mainz.de/~bauec002/SPMain.html
[10] http://www.labyrinth.net.au/~adsoft/sidamp/

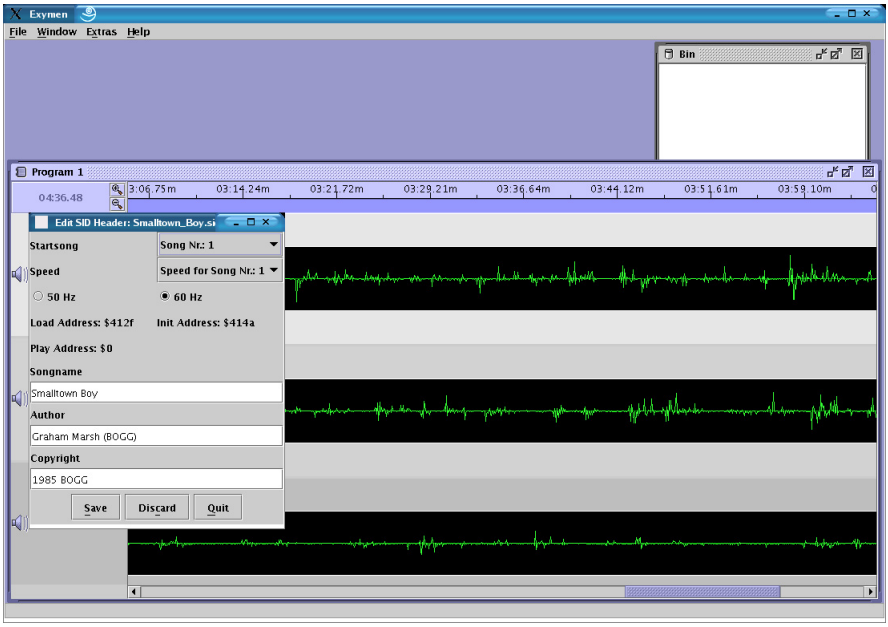

Fig. 4. Screenshot of Exymen editing a SID file

We have decided to use a very simple internal format for a converted SID tune: Every value that has been sent to the SID chip is saved with a time component that measures the distance to the next value and an address. Since the emulator runs faster than the original C64 we use the emulated C64 interrupts to determine the elapsed time between the values. This format allows us to manipulate every single value, since it gives complete control on all data that has been sent to the SID.

The size of a SID file is limited by the small C64 main memory and since a regular SID player depends on this when playing back a SID tune, we are not allowed to exceed this boundary.

Often SID tunes play endlessly because they loop back. The whole SID tune is unrolled by our technique and hence takes more memory to be stored. On the other hand we eliminate redundant code resulting from unclean ripping of demos or games. At the time being we do not detect loops, so we stop converting when 38kB of memory are reached.

The user is now able to edit certain values of the SID file, for example one can copy passages from the SID tune to another thus creating a mix. Another possibility is to modify the time components, altering the speed of the SID tune. One can increase pitch and volume dynamics. Of course, the user is also able to edit all header data (except for load, init and play address) enabling him to modify name and title information.

After editing, the data can be exported to a new SID tune containing a player method written by us.

Table 2. SID Registers

Registers: Voice 1, 2, 3	Description
0, 7, 14	frequency, lo-byte (0..255)
1, 8, 15	frequency, hi-byte (0..255)
2, 9, 16	duty cycle, squarewave only, lo-byte (0..255)
3, 10, 17	duty cycle, squarewave only, hi-byte (0..15)
4, 11, 18	waveform: noise = 129, square = 65, sawtooth = 33, triangle = 17
5, 12, 19	attack: 0*16 .. 15*16, decay: 0 .. 15 (hard to soft)
6, 13, 20	sustain: 0*16 .. 15*16 (silent to loud) release: 0 .. 15 (fast to slow)
21	filter cutoff frequency, lo-byte (0..7)
22	filter cutoff frequency, hi-byte (0..255)
23	resonance: 0 .. 15*16 (none to strong) filter: extern = 8, voice 3 = 4, voice 2 = 2, voice 1 = 1
24	bitmask: high-pass = 64, band-pass = 32, low-pass = 16 volume: 0 .. 15 (silent to loud)
25	paddle X
26	paddle Y
27	oscillator
28	envelope

7 Ongoing Work

The system described here is just the first attempt to make direct and comfortable editing of SID files possible. In order to provide full functionality, SID tunes must also be played back in Exymen. We also want to be able to merge SID music with modern media formats. Another essential goal is to preserve computations in exported tunes.

We believe that we can detect loops with flow analysis to avoid storing data repeatedly. The simplicity of the Commodore 64 is a benefit in this case: Our idea is to keep track of all states the C64 passes while playing back a SID file to notice repetitions. Tracking all states is possible with a modern machine, since C64 programs are small and one state of the 6510 can be stored in 7 bytes[11].

We also plan to use a simple compression in the init method that decompresses the music data before they are played back.

At this moment we are limited to SID files that do without calling the routine at the play address, but produce sound only through a call to the init method. We must adapt our emulator to provide exact timing when calling the play method. This will be done by measuring the CPU cycles between packets.

[11] Of course, using this analysis relies on SID tunes not using self modifying code.

8 Summary

With the system presented here, an interesting piece of computer music history is conserved for teaching purposes. The experiences with the SID, the knowledge of C64 and the basics of music synthesis is of unpayable value. The SID technology introduced many of todays computer music professionals into their field. Music on the C64 is still an active sub culture. Not only are there still many fans of SID music, there is even a company that sells the SID chip as a PCI card for modern PCs[12].

The topic is interesting for teaching, because it offers opportunities to use modern techniques like compression and flow analysis on a simple but real architecture. The size of the tunes compared to their quality is really remarkable. We think the technical and artistical achievements of former times should not be forgotten.

The project started in a multimedia course at Freie Universität Berlin and has been released open source[13].

References

1. G. Friedland, Commodore 64, in R. Rojas (ed.) Encyclopedia of Computers and Computer History, Fitzroy Deaborn, New York 2001.
2. PATENT NO.: 4,677,890, ISSUED: July 07, 1987 (19870707), INVENTOR(s): Yannes, Robert J., Media, PA (Pennsylvania), US (United States of America), (APPL. NO.: 6-455,974 FILED: February 27, 1983 (19830227).
3. Mathias Mertens, Tobias O. Meißner, Wir waren Space Invaders, Eichborn, Frankfurt 2002.
4. Commodore Business Machines, Commodore 64 User Manual, 1982.
5. G. Friedland, Towards a Generic Cross Platform Media Editor: An Editing Tool for E-Chalk, Proceedings of the Informatiktage 2002, Gesellschaft für Informatik e.V., Bad Schussenried 2002.
6. OSGi Service Platform Release 2 (2000), edited by OSGi, IOS Press, Amsterdam 2002.

[12] http://www.hardsid.com
[13] See http://www.exymen.org