ChiptuneSAK

CRX 2020

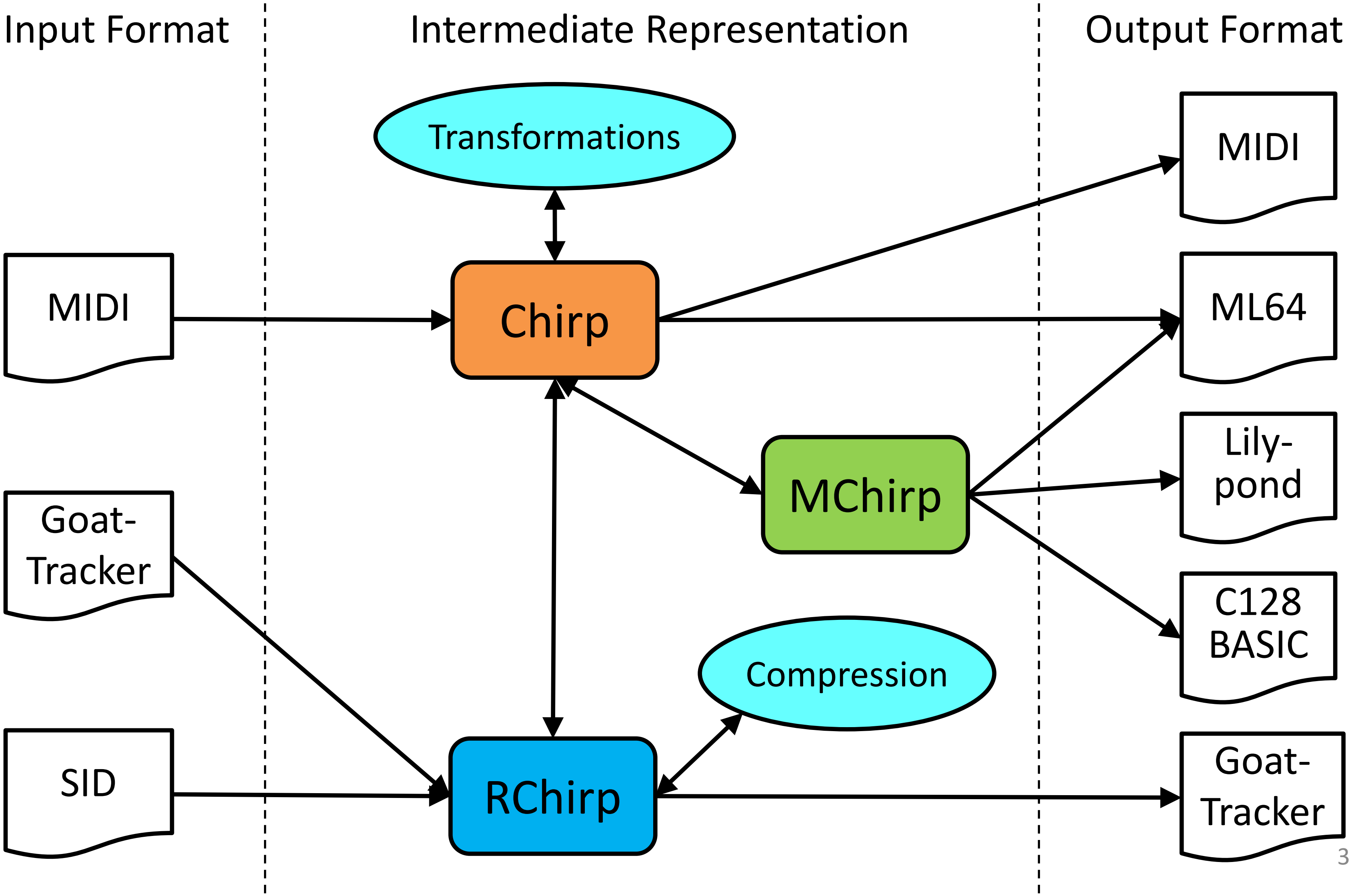
David Youd
David Knapp

Slide Deck V1.0

Introducing ChiptuneSAK

- ChiptuneSAK (*S*wiss *A*rmy *K*nife) is a music processing toolset for note data
  - Goal: Take some of the tedium out of creating chiptunes
  - Python, open source
- Typical Workflow:
  1. Import note data from a music format
  2. Data converted into Chirp (*CH*iptunesak *I*ntermediate *R*e*P*resentation), which can be processed and transformed in many ways
  3. Export note data to a (potentially different) music format
- Initial focus is Commodore music, but can be extended to other “chiptune platforms”

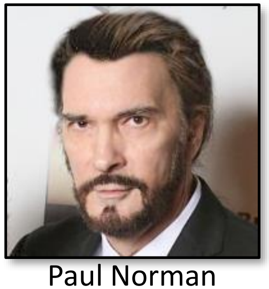
Workflow: Input, Transform, Output



C64 SID Music

Early / American SIDs

Kenneth Arnold



Paul Norman

Dave Warhol

Contemporary / European SIDs*

Reyn Ouwehand

Johan Åstrand

Glenn Gallefoss

Emphasis on melodic content – looks even better when written as sheet music	Timbre and mood can be more important than melodic content
Play engines mostly serve up a stream of notes	Play engines push the very limits of the SID chip
Music you can play on the piano	Complex sounds you can eurodance to

- ChiptuneSAK does much of the heavy lifting needed for tasks on **the left side**
- As for **the right side**, making well-chosen notes sound good is still up to you ;)

* Parallels to Dr. Kenneth McAlpine's comparison of Koji Kondo's Nintendo music with Rob Hubbard's Commodore music, see https://www.lacedrecords.com/blogs/news/ode-to-joysticks-excerpts-part-1-the-early-pioneers-of-video-game-music

Today’s Talk

- 4 demonstrations:
  1. MS-DOS captured MIDI to (C64) SID and to sheet music
  2. MS-DOS captured MIDI to stereo SID
  3. SID to MIDI and to sheet music with triplet handling
  4. MIDI to C128 BASIC music commands

Workflow Example 1

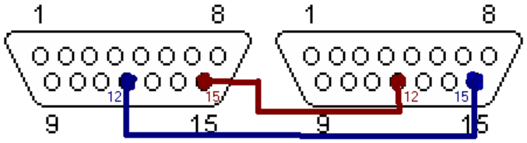
Betrayal at Krondor (MS-DOS) -> MIDI and GoatTracker

What is MIDI?

- MIDI (*M*usical *I*nstrument *D*igital *I*nterface) transmits and stores music note data
- MIDI as a protocol
  - Stream of events on serial bus
    - note_on, note_off, change_program (instrument), controllers
    - Music devices addressed by channel number (1-16) in messages

- MIDI as a file
  - Events plus metadata
    - Metadata: Time signature, key signature, song name, etc.
  - Stored in delta-time format
    - Time since last event

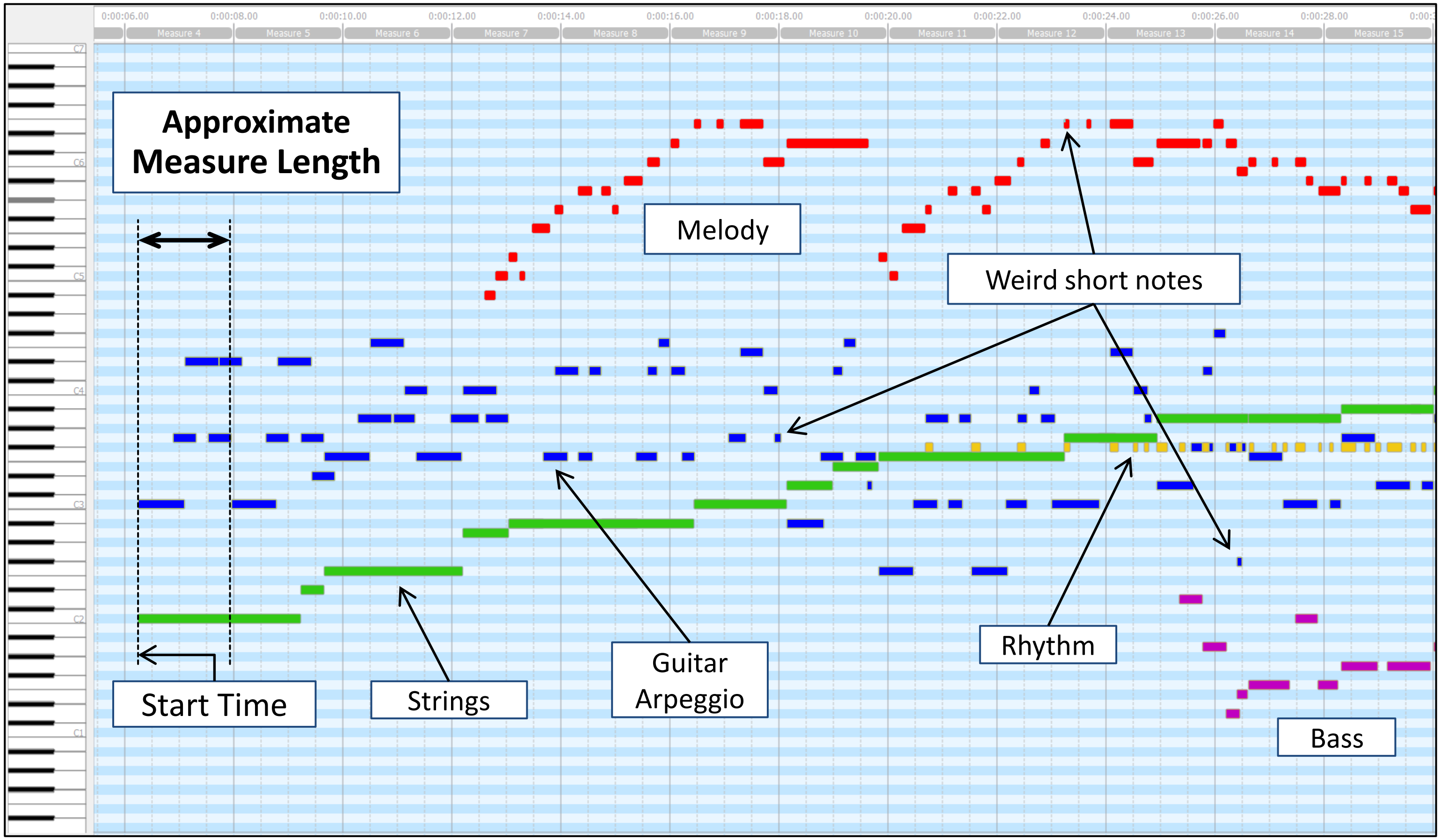
MIDI Game Music Capture



- MS-DOS MIDI game music can be captured as MIDI events from the soundcard
  - – Cable connected from the soundcard's MIDI OUT
  - – While the game was playing, another computer receives and saves the MIDI messages
- Captured MIDI (from the wire) is merely a stream of unscaled events
  - – From a listening perspective, the note data is fine
    - But from a music-processing perspective, the note data is messy
- ChiptuneSAK has useful MIDI cleanup tools

MIDI Capture Example: *Betrayal At Krondor*

(Sierra On-Line, 1993),
Mercantile Theme,
by Jan Moorhead



Approximate Measure Length

Melody

Weird short notes

Start Time

Strings

Guitar Arpeggio

Rhythm

Bass

MidiEditor (https://www.midieditor.org, free for win/linux) for viewing

ChiptuneSAK Finds the Scaling Factors To Fit Notes to the Measures

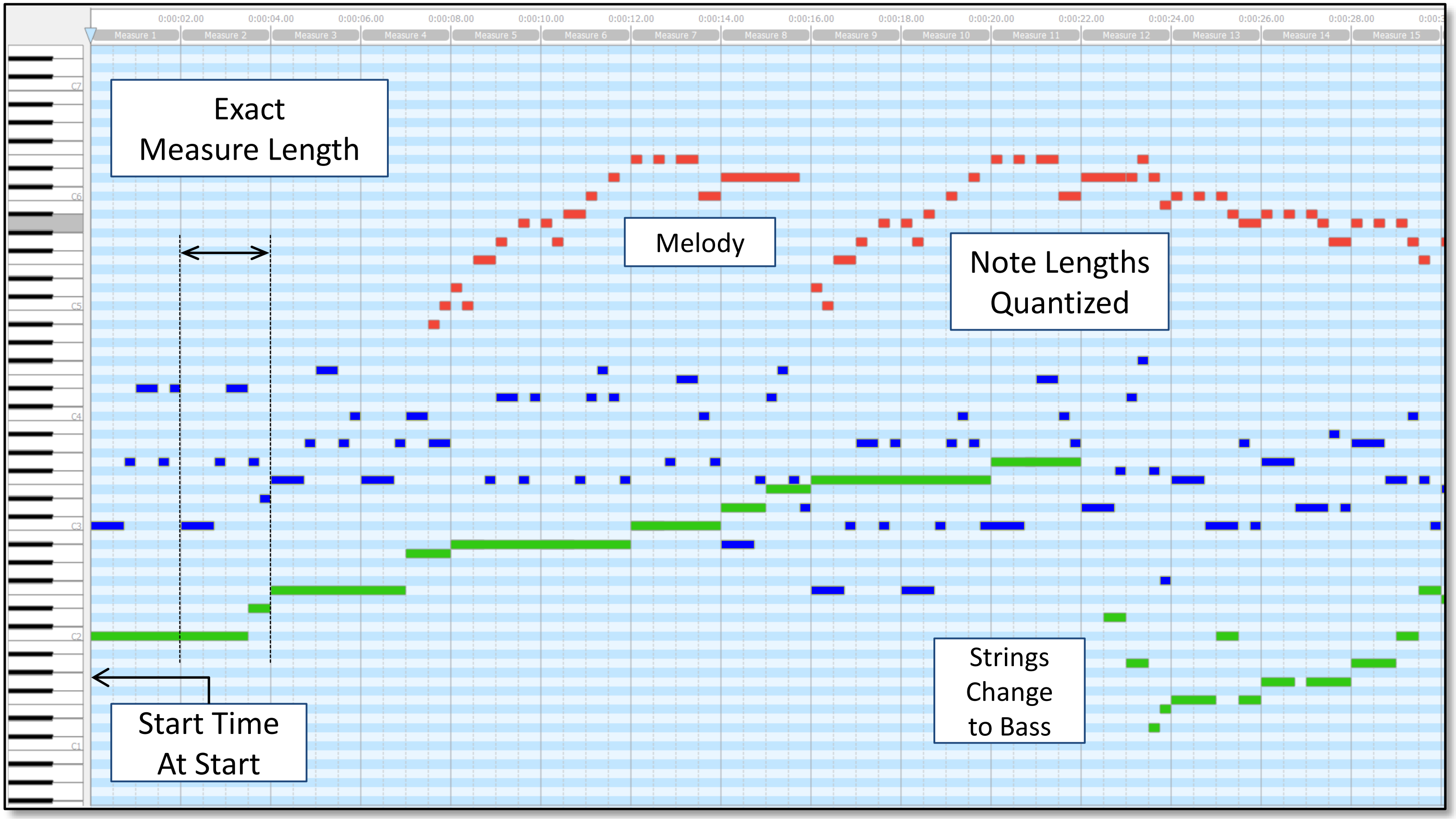
- Start by giving ChiptuneSAK a rough estimate; estimate scale and offset corrections (by eye)
  - Original MIDI file had PPQ (pulses per quarter note) = 192 ticks
    - We prefer PPQ = 960, so a factor of ~5
  - Desired 4 quarter notes per measure (QPM), was about 3.5
    - Estimate more like x5 PPQ factor * 4 QPM / 3.5 QPM observed = 5.7
- Our FitPPQ algorithm finds best-fit scale factor with initial estimate of 5.7
  - scale_factor = 5.89, offset = 2398
- Let’s do a quick sanity check
  - Look at last measure (measure 60) in output MIDI
    - First note of the last measure at tick 226,588
      - 59 * 4 * 960 = 226,560
    - Only off by 28 ticks!
- This auto-scaling ensures that the quantization to follow will work well, and that the notes will respect the time signature

Processing the Mercantile Theme Note Data

- In the upcoming demo, we're going to show how to convert this song to sheet music and to a C64 SID
- We'll show how easy it is to perform the following transformations:
  - ChiptuneSAK note timing cleanup
    - Apply scale and offset to data
    - Use automatic quantization algorithm…
      - Finds 240-tick granularity for durations (= 16th note)
      - Finds 480-tick granularity for note start times (= 8th note)
    - Transform short note durations
      - Set minimum note length to 8th note (480 ticks)
        - In this case, sounds better and looks better on sheet music
  - ChiptuneSAK track changes to target a C64
    - Merged two tracks into track 3
      - Removed notes to preserve track's single-note polyphony
    - Switched instruments mid-way through track 3
      - Once bass comes in, it replaces the strings track
    - Truncated song to only play through once

Mercantile Theme
After ChiptuneSAK Processing

Scaling, quantization, and minimum note length



Exact Measure Length

Melody

Note Lengths Quantized

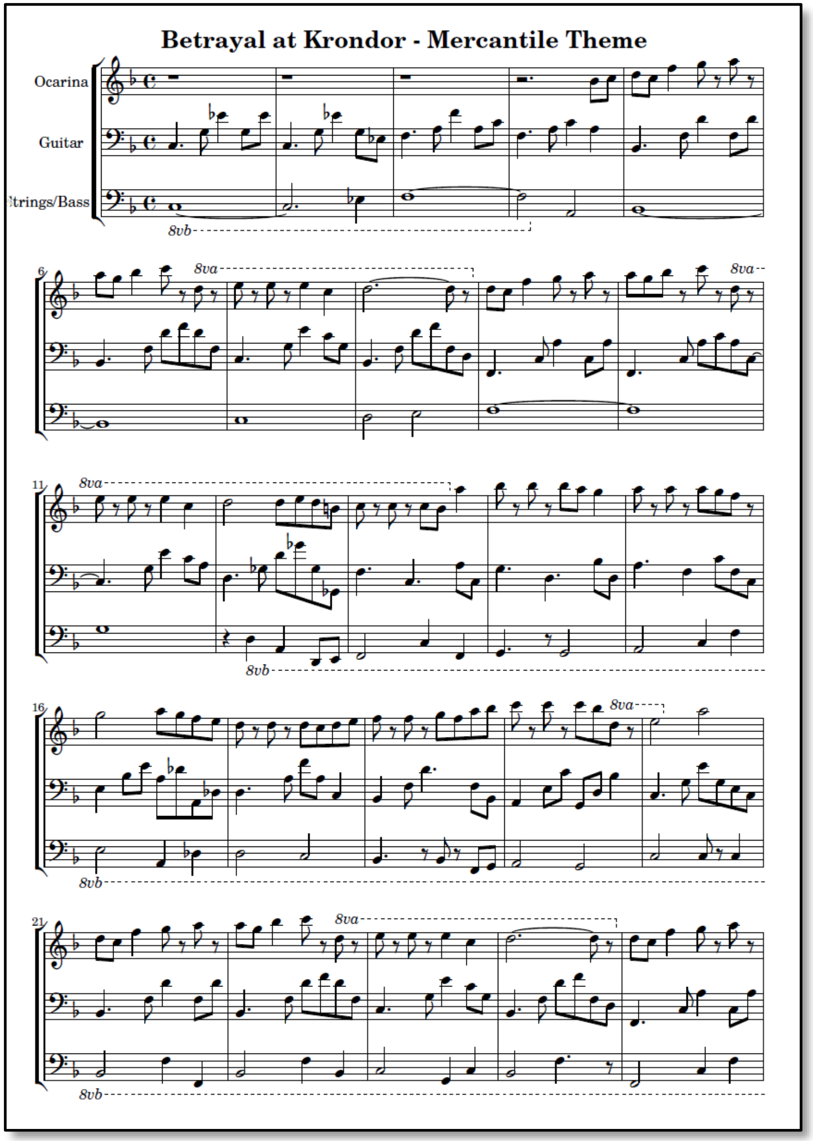
Start Time At Start

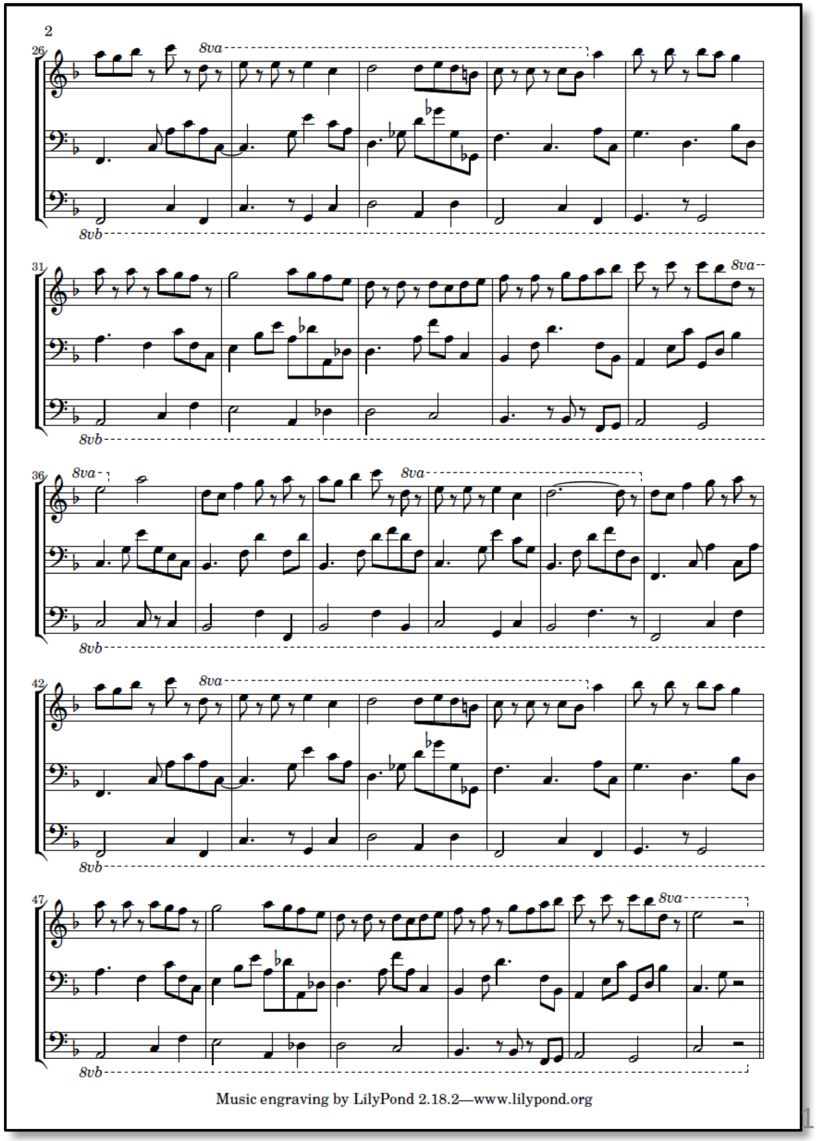
Strings Change to Bass

Output Format: LilyPond Sheetmusic

- ChiptuneSAK exports to LilyPond markup
  - LilyPond (https://lilypond.org) is **free** sheet music software
    - Windows, MacOS, Linux
    - Markup is a bit like LaTeX, but more friendly (human readable)
- Our LilyPond exporter handles:
  - Setting clefs as needed (including mid-song)
    - Creating 8va and 8vb sections
  - Complex triplet scenarios (shown later)

Generated Sheet Music





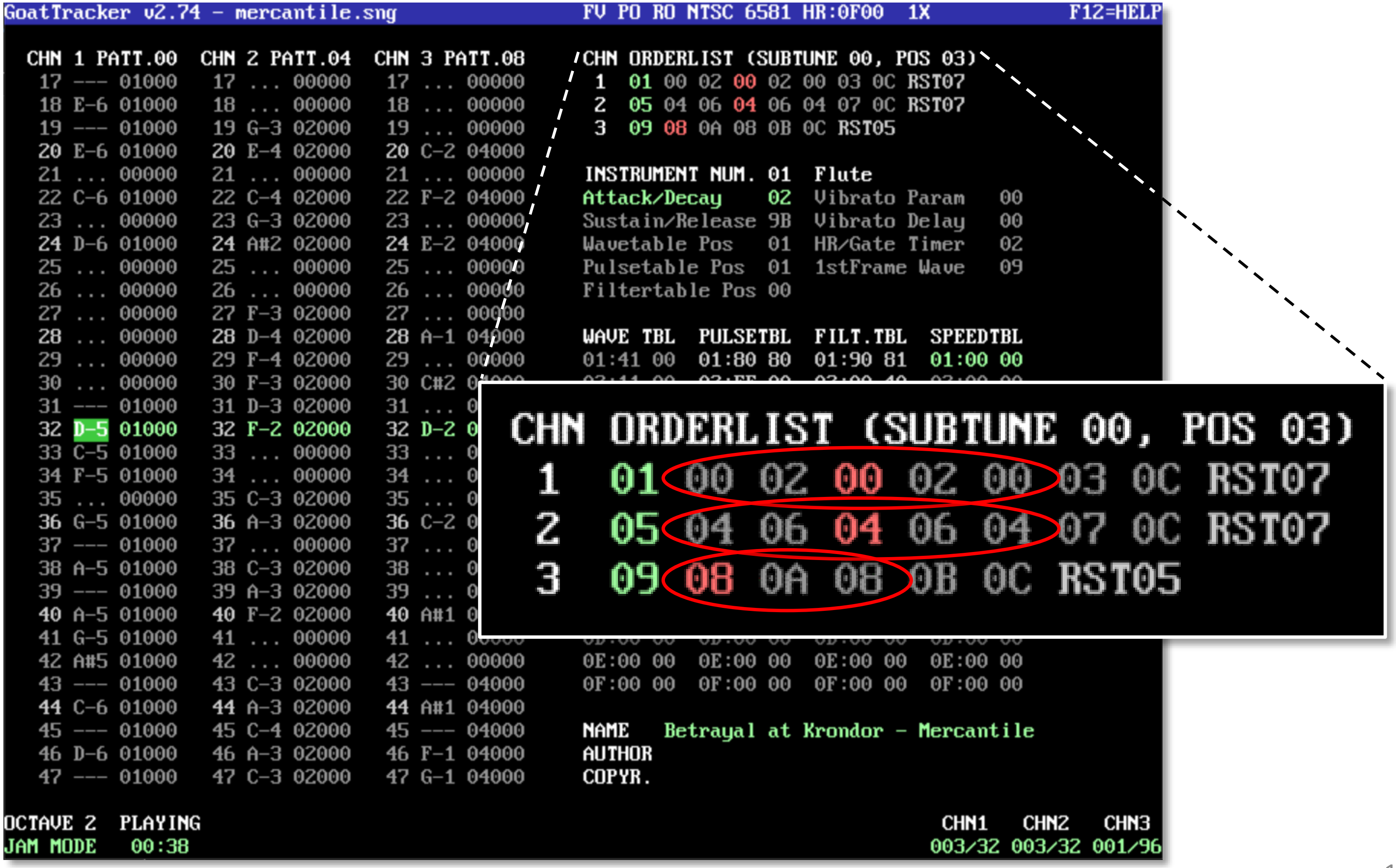
Trackers: Overview

- A goal of ChiptuneSAK is to export to tracker music file formats
- Tracker background:
  - Trackers are music creation programs that
    1. can fully utilize the sound capabilities of the machine
    2. minimize music player/data sizes and playback CPU cost
  - Features of trackers:
    - Voices organized into columns, with notes spanning rows
    - Music organized into re-playable patterns, which can be used in any order on any voice
      - Patterns can be played back transposed, at different tempos, etc.

GoatTracker

- GoatTracker is a tracker for creating C64 SID files
  - Runs on modern platforms: Windows, MacOS, Linux
  - Project lead: Lasse Öörni
- We selected GoatTracker to be the initial tracker supported by ChiptuneSAK
  - For both import and export
- ChiptuneSAK automatically finds pattern opportunities and uses these to “compress” the note data when exporting

Automatic Pattern Detection/Creation (Music Sequence "Compression")



```
GoatTracker v2.74 - mercantile.sng           FV PO RO NTSC 6581 HR:0F00  1X          F12=HELP

 CHN 1 PATT.00  CHN 2 PATT.04  CHN 3 PATT.08  CHN ORDERLIST (SUBTUNE 00, POS 03)
  17 --- 01000   17 ... 00000   17 ... 00000   1  01 00 02 00 02 00 03 0C RST07
  18 E-6 01000   18 ... 00000   18 ... 00000   2  05 04 06 04 06 04 07 0C RST07
  19 --- 01000   19 G-3 02000   19 ... 00000   3  09 08 0A 08 0B 0C RST05
  20 E-6 01000   20 E-4 02000   20 C-2 04000
  21 ... 00000   21 ... 00000   21 ... 00000  INSTRUMENT NUM. 01  Flute
  22 C-6 01000   22 C-4 02000   22 F-2 04000  Attack/Decay    02  Vibrato Param   00
  23 ... 00000   23 G-3 02000   23 ... 00000  Sustain/Release 9B  Vibrato Delay   00
  24 D-6 01000   24 A#2 02000   24 E-2 04000  Wavetable Pos   01  HR/Gate Timer   02
  25 ... 00000   25 ... 00000   25 ... 00000  Pulsetable Pos  01  1stFrame Wave   09
  26 ... 00000   26 ... 00000   26 ... 00000  Filtertable Pos 00
  27 ... 00000   27 F-3 02000   27 ... 00000
  28 ... 00000   28 D-4 02000   28 A-1 04000  WAVE TBL  PULSETBL  FILT.TBL  SPEEDTBL
  29 ... 00000   29 F-4 02000   29 ... 00000  01:41 00  01:80 80  01:90 81  01:00 00
  30 ... 00000   30 F-3 02000   30 C#2 0
  31 --- 01000   31 D-3 02000   31 ... 0
  32 D-5 01000   32 F-2 02000   32 D-2 0
  33 C-5 01000   33 ... 00000   33 ... 0
  34 F-5 01000   34 ... 00000   34 ... 0
  35 ... 00000   35 C-3 02000   35 ... 0
  36 G-5 01000   36 A-3 02000   36 C-2 0
  37 --- 01000   37 ... 00000   37 ... 0
  38 A-5 01000   38 C-3 02000   38 ... 0
  39 --- 01000   39 A-3 02000   39 ... 0
  40 A-5 01000   40 F-2 02000   40 A#1 0
  41 G-5 01000   41 ... 00000   41 ... 0
  42 A#5 01000   42 ... 00000   42 ... 00000  0E:00 00  0E:00 00  0E:00 00  0E:00 00
  43 --- 01000   43 C-3 02000   43 --- 04000  0F:00 00  0F:00 00  0F:00 00  0F:00 00
  44 C-6 01000   44 A-3 02000   44 A#1 04000
  45 --- 01000   45 C-4 02000   45 --- 04000  NAME    Betrayal at Krondor - Mercantile
  46 D-6 01000   46 A-3 02000   46 F-1 04000  AUTHOR
  47 --- 01000   47 C-3 02000   47 G-1 04000  COPYR.

OCTAVE 2  PLAYING                                                  CHN1   CHN2   CHN3
JAM MODE   00:38                                                 003/32 003/32 001/96
```

```
CHN ORDERLIST (SUBTUNE 00, POS 03)
 1  01 00 02 00 02 00 03 0C RST07
 2  05 04 06 04 06 04 07 0C RST07
 3  09 08 0A 08 0B 0C RST05
```

Demo 1 / Code Walkthrough

Workflow Example 2

The Secret of Monkey Island (MS-DOS) -> Stereo GoatTracker

Targeting Multiple SID Chips

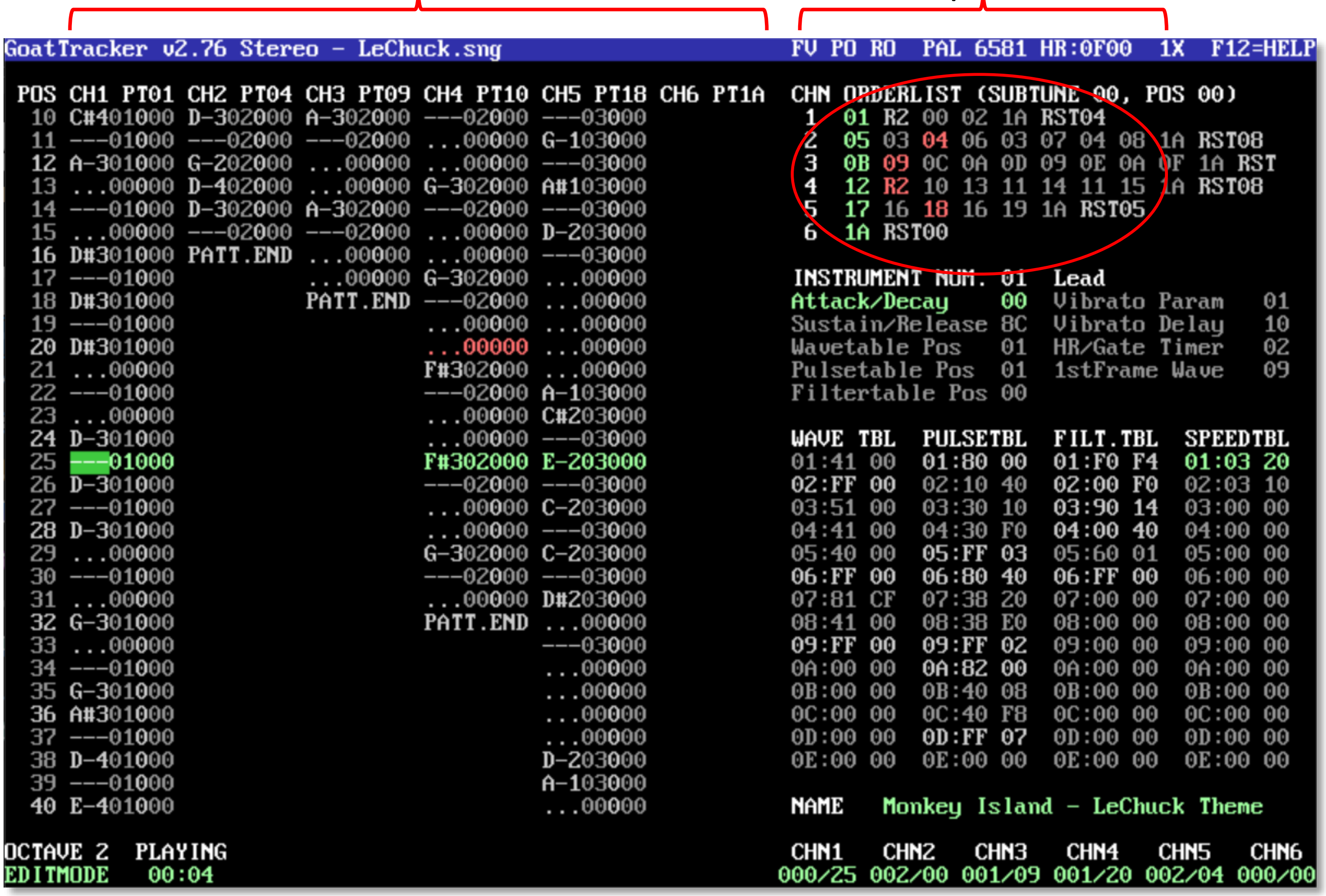
- From *The Secret of Monkey Island* (Lucasfilm Games, 1990) we decided to create a "stereo SID" from *LeChuck's Theme*, by Michael Land

- In the previous example we merged two tracks to target a single 3-voice SID
  - In this stereo example, we split ("explode") a track containing three-note chords into three separate voices

LeChuck Export To Stereo GoatTracker

Uses 5 of the 6 available voices

Music sequence compression

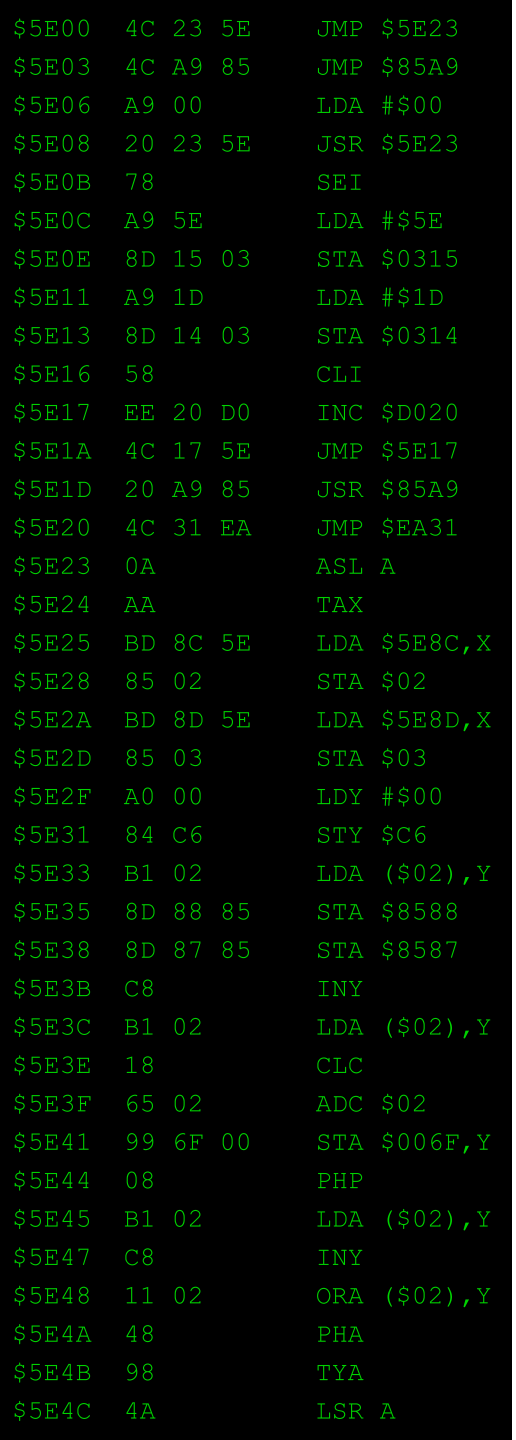


```
GoatTracker v2.76 Stereo - LeChuck.sng                      FV PO RO  PAL 6581 HR:0F00  1X  F12=HELP

 POS CH1 PT01 CH2 PT04 CH3 PT09 CH4 PT10 CH5 PT18 CH6 PT1A  CHN ORDERLIST (SUBTUNE 00, POS 00)
  10 C#401000 D-302000 A-302000 ---02000 ---03000            1  01 R2 00 02 1A RST04
  11 ---01000 ---02000 ---02000 ...00000 G-103000            2  05 03 04 06 03 07 04 08 1A RST08
  12 A-301000 G-202000 ...00000 ...00000 ---03000            3  0B 09 0C 0A 0D 09 0E 0A 0F 1A RST
  13 ...00000 D-402000 ...00000 G-302000 A#103000            4  12 R2 10 13 11 14 11 15 1A RST08
  14 ---01000 D-302000 A-302000 ---02000 ---03000            5  17 16 18 16 19 1A RST05
  15 ...00000 ---02000 ---02000 ...00000 D-203000            6  1A RST00
  16 D#301000 PATT.END ...00000 ...00000 ---03000
  17 ---01000          ...00000 G-302000 ...00000           INSTRUMENT NUM. 01  Lead
  18 D#301000          PATT.END ---02000 ...00000           Attack/Decay    00  Vibrato Param   01
  19 ---01000                   ...00000 ...00000           Sustain/Release 8C  Vibrato Delay   10
  20 D#301000                   ...00000 ...00000           Wavetable Pos   01  HR/Gate Timer   02
  21 ...00000                   F#302000 ...00000           Pulsetable Pos  01  1stFrame Wave   09
  22 ---01000                   ---02000 A-103000           Filtertable Pos 00
  23 ...00000                   ...00000 C#203000
  24 D-301000                   ...00000 ---03000           WAVE TBL  PULSETBL  FILT.TBL  SPEEDTBL
  25 ---01000                   F#302000 E-203000           01:41 00  01:80 00  01:F0 F4  01:03 20
  26 D-301000                   ---02000 ---03000           02:FF 00  02:10 40  02:00 F0  02:03 10
  27 ---01000                   ...00000 C-203000           03:51 00  03:30 10  03:90 14  03:00 00
  28 D-301000                   ...00000 ---03000           04:41 00  04:30 F0  04:00 40  04:00 00
  29 ...00000                   G-302000 C-203000           05:40 00  05:FF 03  05:60 01  05:00 00
  30 ---01000                   ---02000 ---03000           06:FF 00  06:80 40  06:FF 00  06:00 00
  31 ...00000                   ...00000 D#203000           07:81 CF  07:38 20  07:00 00  07:00 00
  32 G-301000                   PATT.END ...00000           08:41 00  08:38 E0  08:00 00  08:00 00
  33 ...00000                            ---03000           09:FF 00  09:FF 02  09:00 00  09:00 00
  34 ---01000                            ...00000           0A:00 00  0A:82 00  0A:00 00  0A:00 00
  35 G-301000                            ...00000           0B:00 00  0B:40 08  0B:00 00  0B:00 00
  36 A#301000                            ...00000           0C:00 00  0C:40 F8  0C:00 00  0C:00 00
  37 ---01000                            ...00000           0D:00 00  0D:FF 07  0D:00 00  0D:00 00
  38 D-401000                            D-203000           0E:00 00  0E:00 00  0E:00 00  0E:00 00
  39 ---01000                            A-103000
  40 E-401000                            ...00000           NAME   Monkey Island - LeChuck Theme

OCTAVE 2  PLAYING                                           CHN1   CHN2   CHN3   CHN4   CHN5   CHN6
EDITMODE   00:04                                           000/25 002/00 001/09 001/20 002/04 000/00
```

Demo 2 / Code Walkthrough

Workflow
Example 3

Skyfox SID ->
Lilypond



```
$5E00   4C 23 5E    JMP $5E23
$5E03   4C A9 85    JMP $85A9
$5E06   A9 00       LDA #$00
$5E08   20 23 5E    JSR $5E23
$5E0B   78          SEI
$5E0C   A9 5E       LDA #$5E
$5E0E   8D 15 03    STA $0315
$5E11   A9 1D       LDA #$1D
$5E13   8D 14 03    STA $0314
$5E16   58          CLI
$5E17   EE 20 D0    INC $D020
$5E1A   4C 17 5E    JMP $5E17
$5E1D   20 A9 85    JSR $85A9
$5E20   4C 31 EA    JMP $EA31
$5E23   0A          ASL A
$5E24   AA          TAX
$5E25   BD 8C 5E    LDA $5E8C,X
$5E28   85 02       STA $02
$5E2A   BD 8D 5E    LDA $5E8D,X
$5E2D   85 03       STA $03
$5E2F   A0 00       LDY #$00
$5E31   84 C6       STY $C6
$5E33   B1 02       LDA ($02),Y
$5E35   8D 88 85    STA $8588
$5E38   8D 87 85    STA $8587
$5E3B   C8          INY
$5E3C   B1 02       LDA ($02),Y
$5E3E   18          CLC
$5E3F   65 02       ADC $02
$5E41   99 6F 00    STA $006F,Y
$5E44   08          PHP
$5E45   B1 02       LDA ($02),Y
$5E47   C8          INY
$5E48   11 02       ORA ($02),Y
$5E4A   48          PHA
$5E4B   98          TYA
$5E4C   4A          LSR A
```

SID Importing

- A SID file contains Commodore-native code that plays music, along with headers that describe how to execute the code
- We've created code to import SID files into Chirp, process them, and export to various output formats
- Meant to be an alternative to Michael Schwendt's SID2MIDI tool
  - A great tool, but a few limitations:
    - Closed source (not updated since 2007)
    - Windows only
    - Won't process RSIDs (SID files that require higher-fidelity emulation)

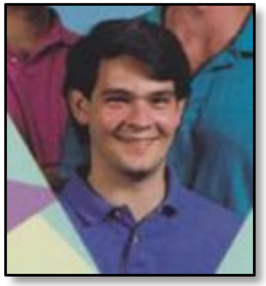
SID Importing (cont…)

- We wrote an all-python 6502/6510 emulator
  - Required to execute SID music code
  - We studied a C language reference implementation
    - Lasse Öörni 's (Cadaver, of GoatTracker fame) and Stein Pedersen's excellent SIDDump tool (v1.08, 2020)
- Our emulator has been thoroughly tested:
  - Boots a C64 as part of our testing suite
    - Checks virtual screen for “38911 BASIC BYTES FREE” ☺
  - Passes all relevant Wolfgang Lorenz C64 tests
    - Several hundred assembly language tests that take an hour to run
  - Passes the Klaus Dormann BCD test suite

SID Importing (cont…)

- Wrote Commodore 64 layer to capture how emulated SIDs interact with the SID chip
  - Can handle PSIDs as well as many RSIDs (ignoring digi)
  - Supports multispeed (>1 play calls per screen update)
  - Makes it easy for the python-fluent to inspect SID behavior
    - e.g., describes interrupt drivers, all zeropage usage, etc.

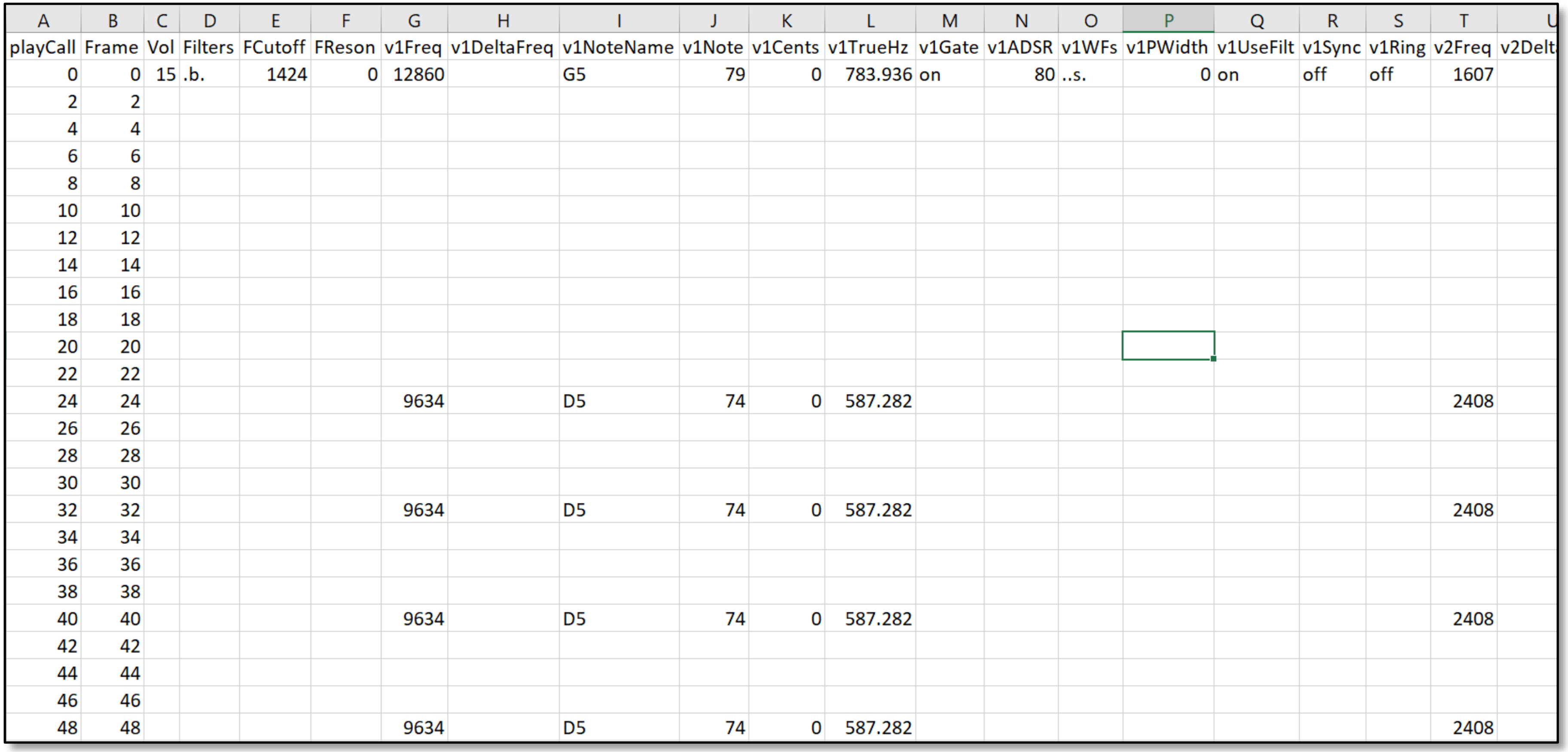
- To demonstrate our SID to sheet music capabilities, we'll be using Douglas Fulton's *Skyfox* (EA, 1984) game music*



Douglas Fulton, 1988

* A great composition, but obscured by poor timbre choices, and a play engine that barely separates repeated notes (2 millisecond attack pulse separation via a gate on->off->on that happened in mere microseconds).
But definitely worthy of a remix on remix.kwed.org. Any volunteers?

Skyfox: SID -> CSV File



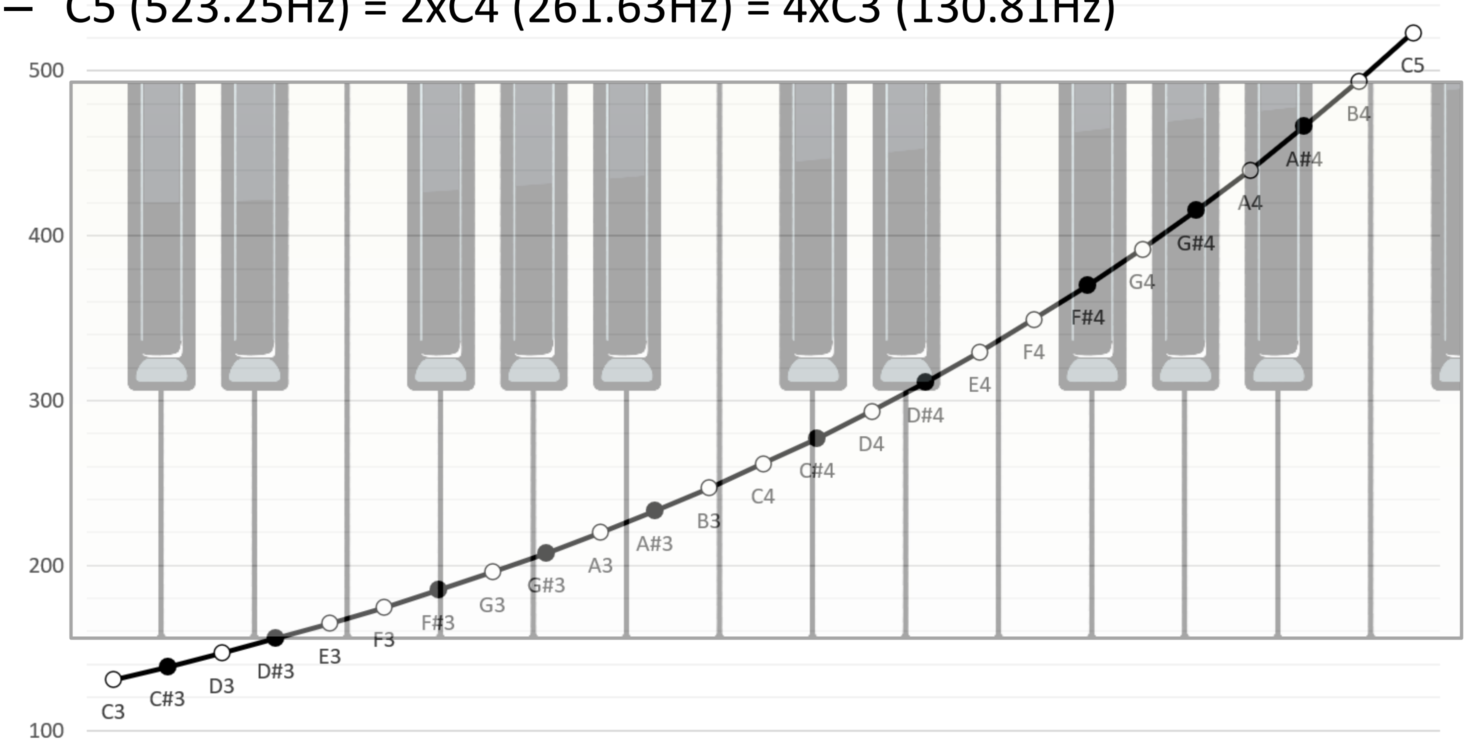
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
playCall	Frame	Vol	Filters	FCutoff	FReson	v1Freq	v1DeltaFreq	v1NoteName	v1Note	v1Cents	v1TrueHz	v1Gate	v1ADSR	v1WFs	v1PWidth	v1UseFilt	v1Sync	v1Ring	v2Freq	v2Delt
0	0	15	.b.	1424	0	12860		G5	79	0	783.936	on	80	..s.	0	on	off	off	1607	
2	2																			
4	4																			
6	6																			
8	8																			
10	10																			
12	12																			
14	14																			
16	16																			
18	18																			
20	20																			
22	22																			
24	24					9634		D5	74	0	587.282								2408	
26	26																			
28	28																			
30	30																			
32	32					9634		D5	74	0	587.282								2408	
34	34																			
36	36																			
38	38																			
40	40					9634		D5	74	0	587.282								2408	
42	42																			
44	44																			
46	46																			
48	48					9634		D5	74	0	587.282								2408	

CSV file (showing voice 1)

- ChiptuneSAK creates CSV files, and will reduce output rows if possible (e.g., removed every other frame above)
- Gives insights into register settings, use of gates, etc.
  - If familiar with piece, can see that if 4/4, a quarter note is 24 frames long

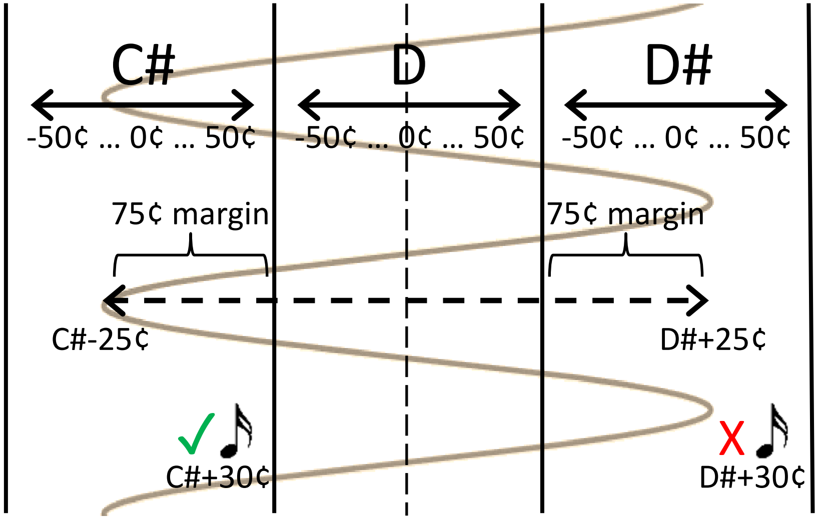
SID Import Vibrato Handling

- We implemented functionality to handle notes with wide vibratos
  - But first, some background…
- Going up an octave doubles the frequency
  - C5 (523.25Hz) = 2xC4 (261.63Hz) = 4xC3 (130.81Hz)



- Pitch intervals can be easily compared using a logarithmic unit called *cents*

SID Import Vibrato Handling (cont…)

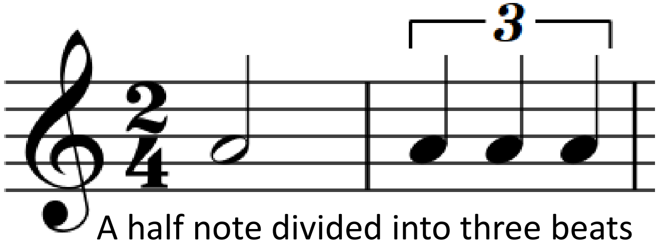


- Each semitone spans 100¢
  - An octave is 1200¢
- Difference between two frequencies: ¢ = 1200 × $\log_2(f1/f2)$

- Handling a D note with a wide vibrato
  - If new candidate note is semitone-adjacent to previous note, and within a given margin (e.g., 75¢), snap to previous note value
  - Example: Previous note is D(+0¢), margin set at 75¢
    - A candidate C#(+30¢) snaps back to a D(+0¢)
    - A candidate D#(+30¢) becomes a D#(+0¢)

ChiptuneSAK Triplet Handling

Dividing a duration into three equal parts usually requires grouping notes into a triplet

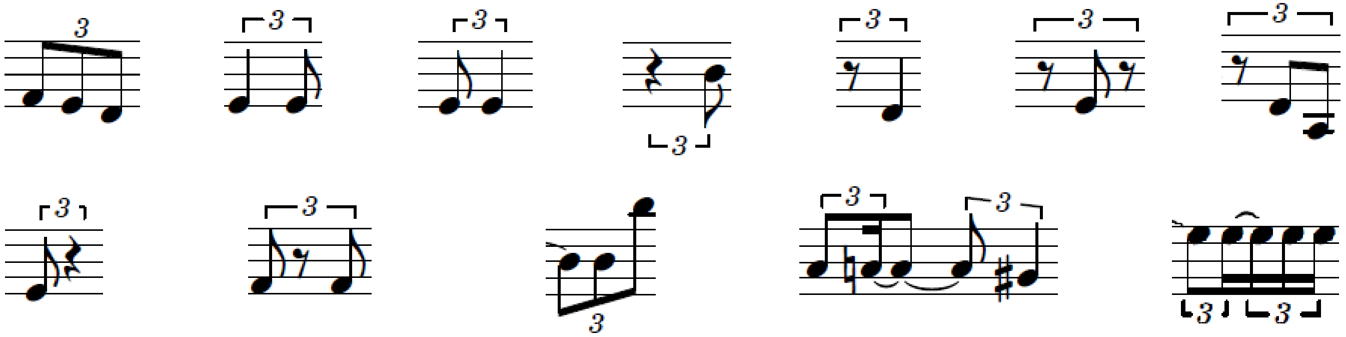


A half note divided into three beats

- You should get reasonable triplet interpretations when importing MIDI into commercial score-writing software
  - i.e., Sibelius, Dorico, Finale, MuseScore, etc.
- ChiptuneSAK needs to interpret triplets as well, in order to:
  - reason about them
  - export them as triplet markup to LilyPond

Triplet Parsing from MIDI is Hard

- A few of the triplet scenarios we've encountered:

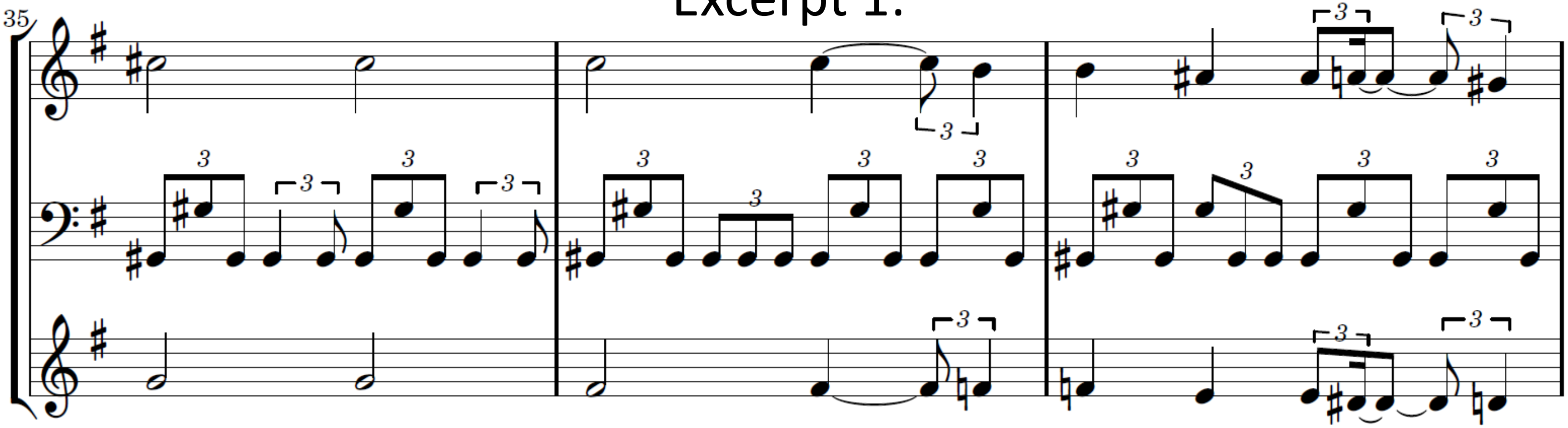


- Many ways triplets present themselves
  - Some of the above could have been rewritten in different but equivalent ways
- ChiptuneSAK MChirp correctly interprets most triplet patterns
  - Our recursive algorithm places all triplet-like notes into triplet objects
  - Our algorithm's assumptions:
    - Note durations must be quantized for triplet durations
    - Triplets do not cross measure boundaries (no one should be doing this)
    - At least one note of the potential triplet has the property that its duration divided by the PPQ has a denominator that is a multiple of 3

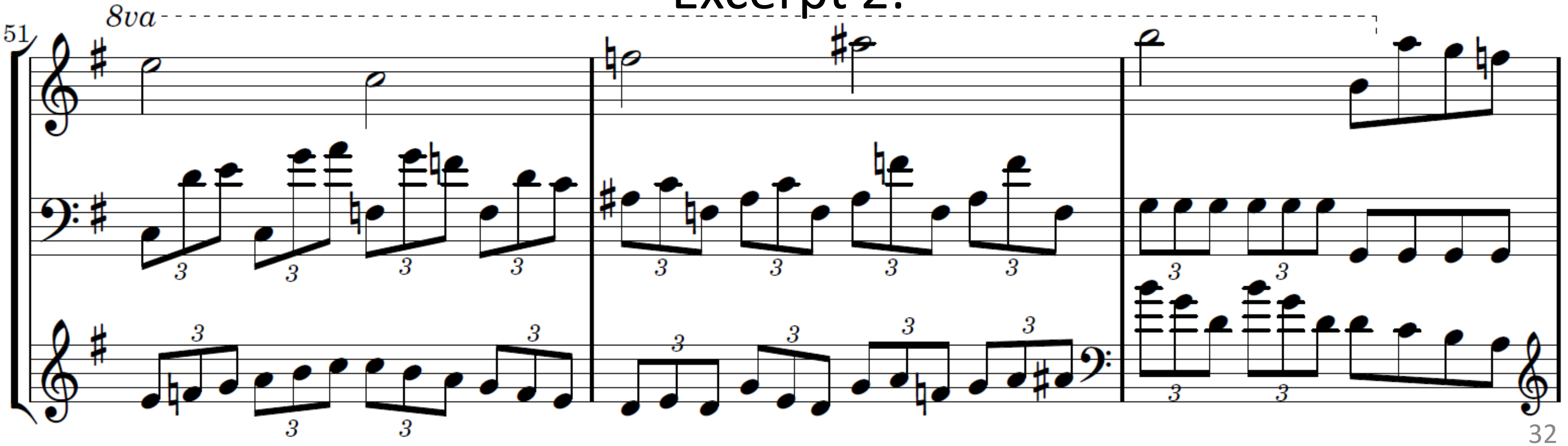
Result: Skyfox Exported to LilyPond

(As 4/4 time, to Showcase Triplet Handling)

Excerpt 1:



Excerpt 2:

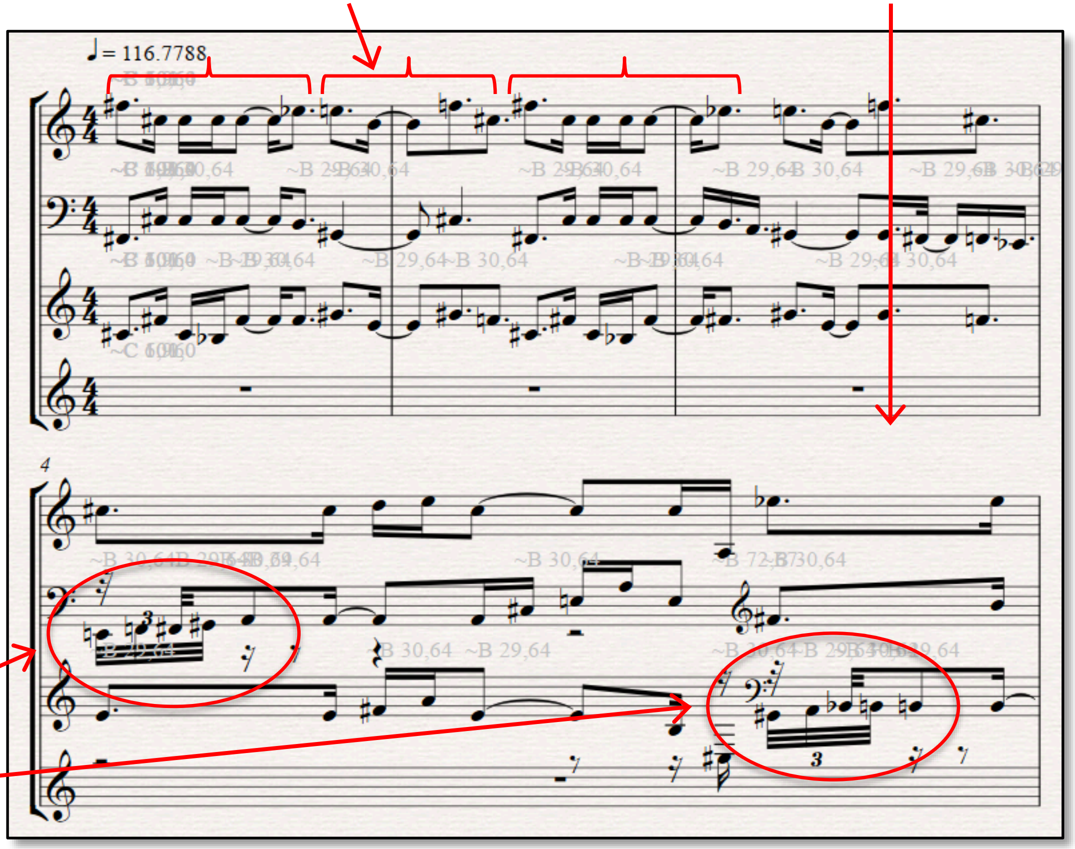


Skyfox SID -> *SID2MIDI* -> Sibelius

- Sibelius is commercial score-writing software
- Manual clean up required on *SID2MIDI* tool's output, due to:
  - MIDI quantizer not being meter-aware
  - Type 0 MIDI export (single track) leads to Sibelius making poor interpretations

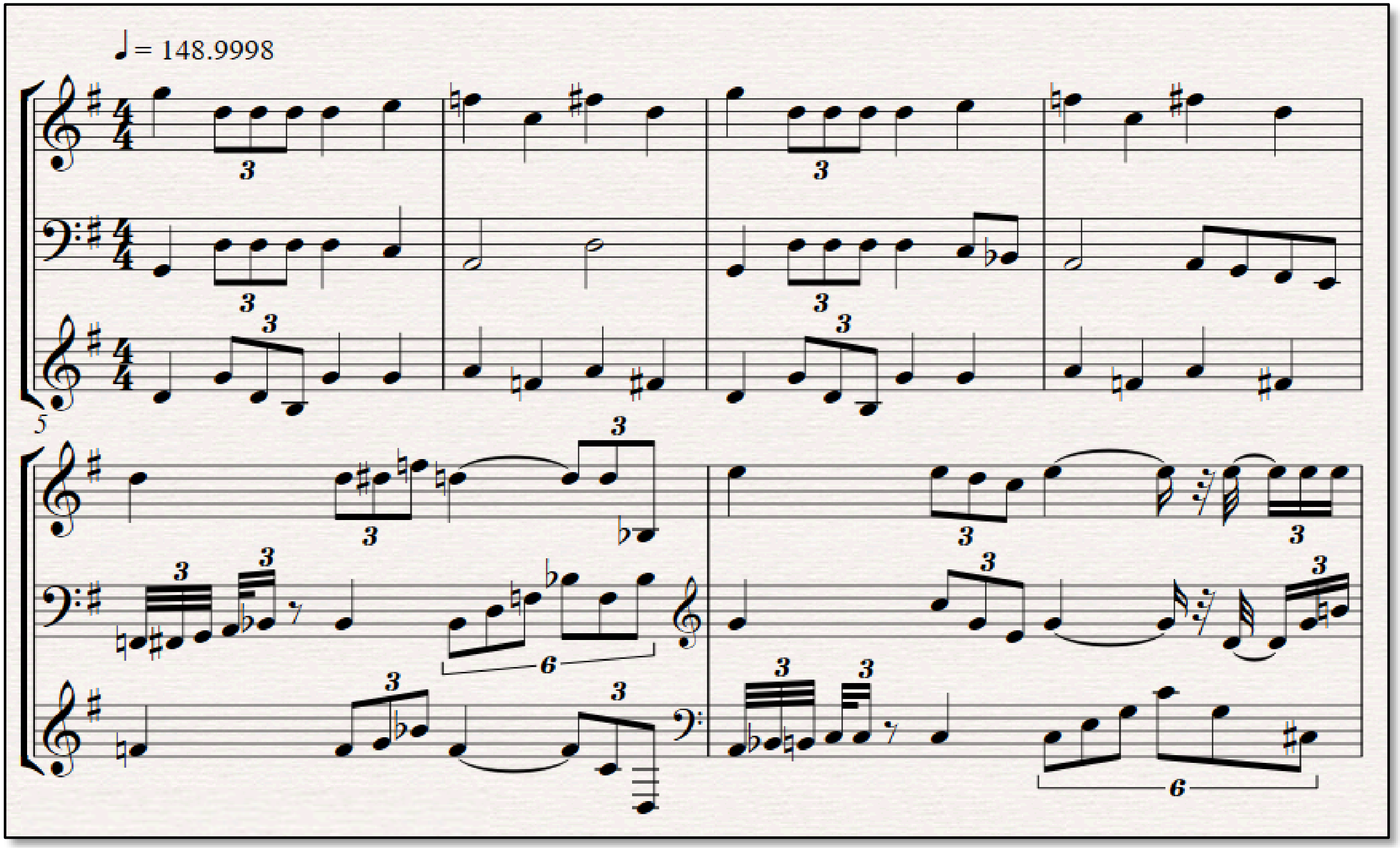
Note durations not scaled to expected measure divisions (in red)

Type 0 export led to a stave per channel



Sibelius adds undesired overlapping polyphony by interpreting multiple voices within a stave

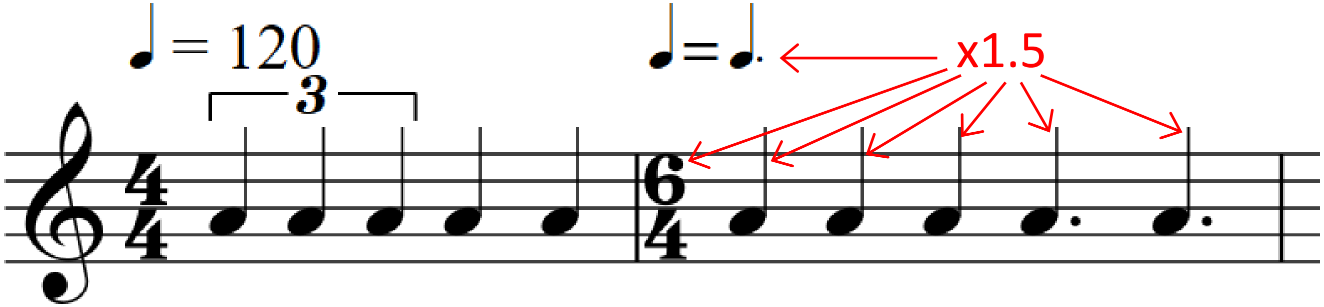
Skyfox SID -> ChiptuneSAK -> Sibelius Requires Less Cleanup



- Music content divided correctly into measures
- Sibelius tuplet guessing (3-tuple, 6-tuple) is much improved
  - But these tuplets are a hint we should be using a compound meter…

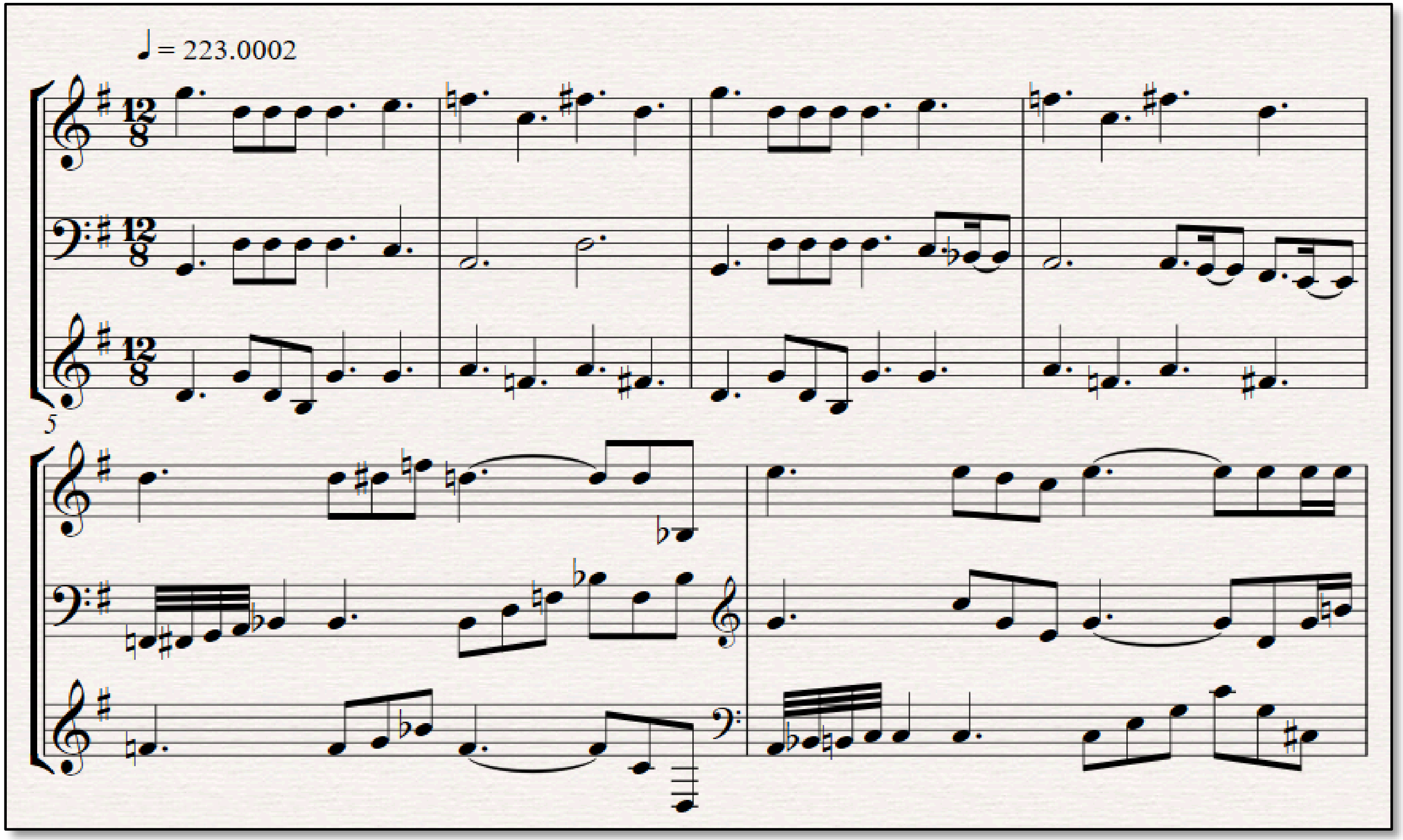
Handling Triplets: Metric Modulation

- Sometimes, explicit triplets get in the way
  - Many chiptune music systems do not support triplets, or they expect only divisions-of-two durations when importing from MIDI (e.g., SID-Wizard)
  - Sometimes removing triplets makes the music more readable
- In this example, we remove explicit triplets by multiplying the time signature, the BPM, and all note durations by 3/2



  - The 2nd measure has the **same rhythm and the same perceived tempo** as the 1st measure
    - Notation indicates 2nd measure performed at 120*1.5 = 180 BPM
- Surprisingly, commercial software is bad at this
  - Easy to do with ChiptuneSAK

Skyfox MIDI After ChiptuneSAK Metric Modulation



- No triplets in compound meter
  - Again, perceived playback is unchanged

DEMO 3 / Code Walkthrough

Workflow
Example 4

C128 BASIC Music
Commands

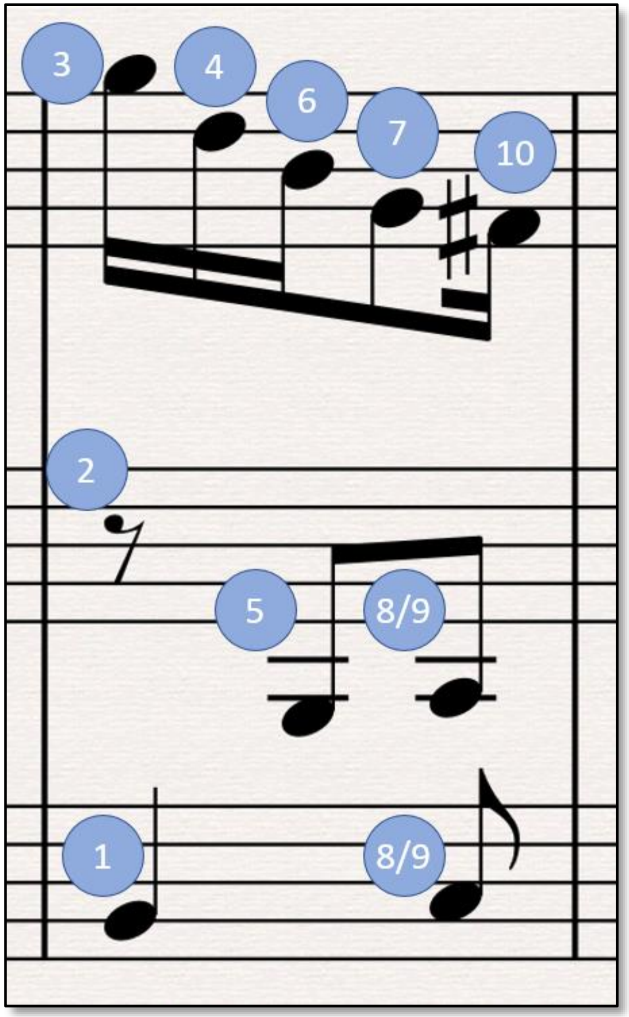
Export Format: C128 BASIC

- The Commodore 128’s BASIC v7.0 includes commands for creating music
  - So far, this has been completely neglected in the scene, so we decided to support it
- HVSC contains hundreds of BASIC SIDs, and at least one C128-only SID (*Ultima V*, Origin Systems, 1988).
  - Can any SID player handle C128 BASIC SIDs?

Commodore BASIC v7.0 Music Syntax

- Uses strings to represent music
  - e.g., `PLAY"CDEFGAB"`
  - ”V” sets voice, “O” sets octave, “#” and “$” for accidentals
- 10 instruments (a fixed ADSR, WF, and PW only)
  - FILTER command control filters
- Supports durations “W”, “H”, “Q”, “I”(eighth), and “S”, with “R” for rest, and “.” for dotted values.
  - No support for ties or triplets
  - Repeats performed by multiple PLAY calls to same string variable
- TEMPO command controls playback speed
  - 1 = slowest, 255 = fastest
  - We did a bit of C128 BASIC ROM reversing to determine that tempo = quarter_notes_per_minute / 60 / 4 * 1152 / frame_rate_Hz
- Voices tend to desync, due to tempo-related roundoff errors
  - Inserting an “M” in the PLAY string will force all voices to sync after currently playing notes end

PLAY Command is *Very* Particular About Note Ordering



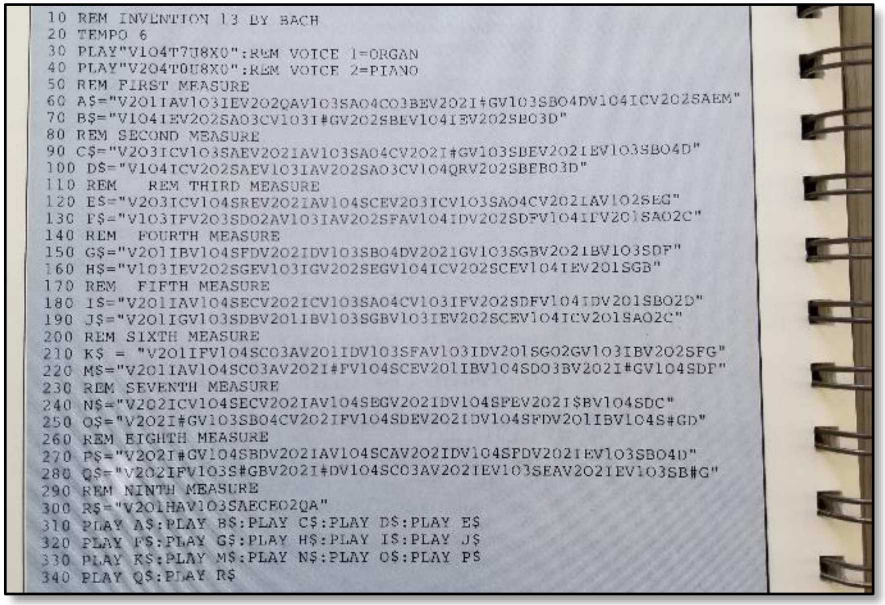
Measure from BWV 799 ($\frac{3}{8}$, 𝄞, 𝄞, 𝄢)

- Longer notes must be included prior to any shorter notes that they overlap with.
- Example: Data excerpt from Bach 3-part invention in Am, with pitches/rests highlighted:

  -
    ```
    PLAY"V3O2QBV2IRV1O5
    SGDV2O3IGV1O4SBIGV2
    O3AV3CV1O4S#FM"
    ```
    - Octave, duration, and voice settings persist until changed
    - Only the order of the two notes labeled 8/9 could be swapped
- #hellatedious

Commodore’s “CODING A SONG FROM SHEET MUSIC” Instructions

- This is a section in both *The Commodore 128 System Guide* (p154-157) and *The Commodore 128 Programmer’s Reference Guide* (p341-343)
  - Type-in code: Bach’s Two Part Invention #13 in A minor
    - A song Commodore used in its TV advertisements



```
10 REM INVENTION 13 BY BACH
20 TEMPO 6
30 PLAY"V1O4T7U8X0":REM VOICE 1=ORGAN
40 PLAY"V2O4T0U8X0":REM VOICE 2=PIANO
50 REM FIRST MEASURE
60 A$="V2O1IAV1O3IEV2O2QAV1O3SAO4CO3BEV2O2I#GV1O3SBO4DV1O4ICV2O2SAEM"
70 B$="V1O4IEV2O2SAO3CV1O3I#GV2O2SBEV1O4IEV2O2SBO3D"
80 REM SECOND MEASURE
90 C$="V2O3ICV1O3SAEV2O2IAV1O3SAO4CV2O2I#GV1O3SBEV2O2IEV1O3SBO4D"
100 D$="V1O4ICV2O2SAEV1O3IAV2O2SAO3CV1O4QRV2O2SBEBO3D"
110 REM   REM THIRD MEASURE
120 E$="V2O3ICV1O4SREV2O2IAV1O4SCEV2O3ICV1O3SAO4CV2O2IAV1O2SEG"
130 F$="V1O3IFV2O3SDO2AV1O3IAV2O2SFAV1O4IDV2O2SDFV1O4IFV2O1SAO2C"
140 REM  FOURTH MEASURE
150 G$="V2O1IBV1O4SFDV2O2IDV1O3SBO4DV2O2IGV1O3SGBV2O2IBV1O3SDF"
160 H$="V1O3IEV2O2SGEV1O3IGV2O2SEGV1O4ICV2O2SCEV1O4IEV2O1SGB"
170 REM  FIFTH MEASURE
180 I$="V2O1IAV1O4SECV2O2ICV1O3SAO4CV1O3IFV2O2SDFV1O4IDV2O1SBO2D"
190 J$="V2O1IGV1O3SDBV2O1IBV1O3SGBV1O3IEV2O2SCEV1O4ICV2O1SAO2C"
200 REM SIXTH MEASURE
210 K$ = "V2O1IFV1O4SCO3AV2O1IDV1O3SFAV1O3IDV2O1SGO2GV1O3IBV2O2SFG"
220 M$="V2O1IAV1O4SCO3AV2O2I#FV1O4SCEV2O1IBV1O4SDO3BV2O2I#GV1O4SDF"
230 REM SEVENTH MEASURE
240 N$="V2O2ICV1O4SECV2O2IAV1O4SEGV2O2IDV1O4SFEV2O2I$BV1O4SDC"
250 O$="V2O2I#GV1O3SBO4CV2O2IFV1O4SDEV2O2IDV1O4SFDV2O1IBV1O4S#GD"
260 REM EIGHTH MEASURE
270 P$="V2O2I#GV1O4SBDV2O2IAV1O4SCAV2O2IDV1O4SFDV2O2IEV1O3SBO4D"
280 Q$="V2O2IFV1O3S#GBV2O2I#DV1O4SCO3AV2O2IEV1O3SEAV2O2IEV1O3SB#G"
290 REM NINTH MEASURE
300 R$="V2O1HAV1O3SAECEO2QA"
310 PLAY A$:PLAY B$:PLAY C$:PLAY D$:PLAY E$
320 PLAY F$:PLAY G$:PLAY H$:PLAY I$:PLAY J$
330 PLAY K$:PLAY M$:PLAY N$:PLAY O$:PLAY P$
340 PLAY Q$:PLAY R$
```

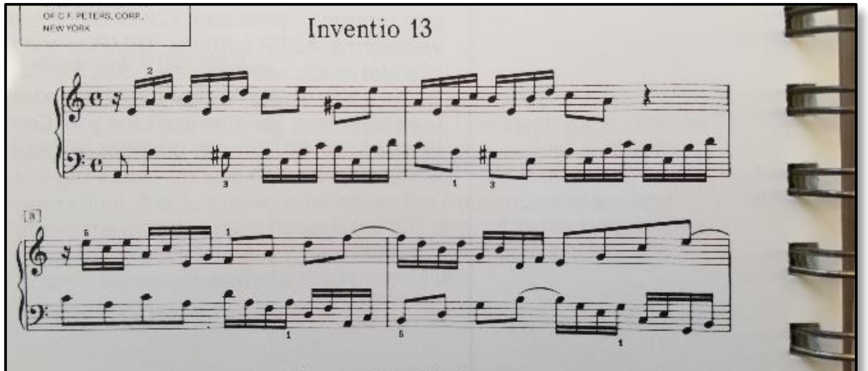
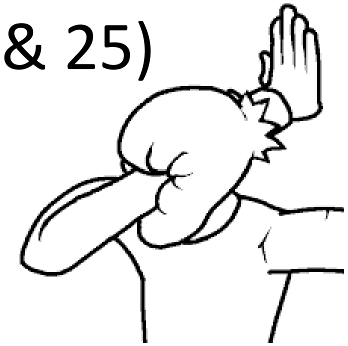


Figure 7-12. Part of Bach's *Invention 13*

So Painful, Even Commodore Staff Couldn’t Get It Right

- Both texts only included 8½ measures (1 to 6.5 and 23 to 25), *but so… many… errors*…
  - Note on wrong beat (measure 1)
  - Notes in wrong octave (measures 3, 5, 24 & 25)
  - Reasserted tied note (measures 4, 5, & 6)
  - Wrong pitch (measure 4)
  - An unnecessary note (measure 4)
  - Horrible voice desynchronization toward the end
- In the demo we’ll play their version, and then how it should have been done
  - Then we’ll show a more complex example, Bach’s **Three** Part Invention #13 in A minor (BWV799)



Print 'n Stick in Your C128 Manual:

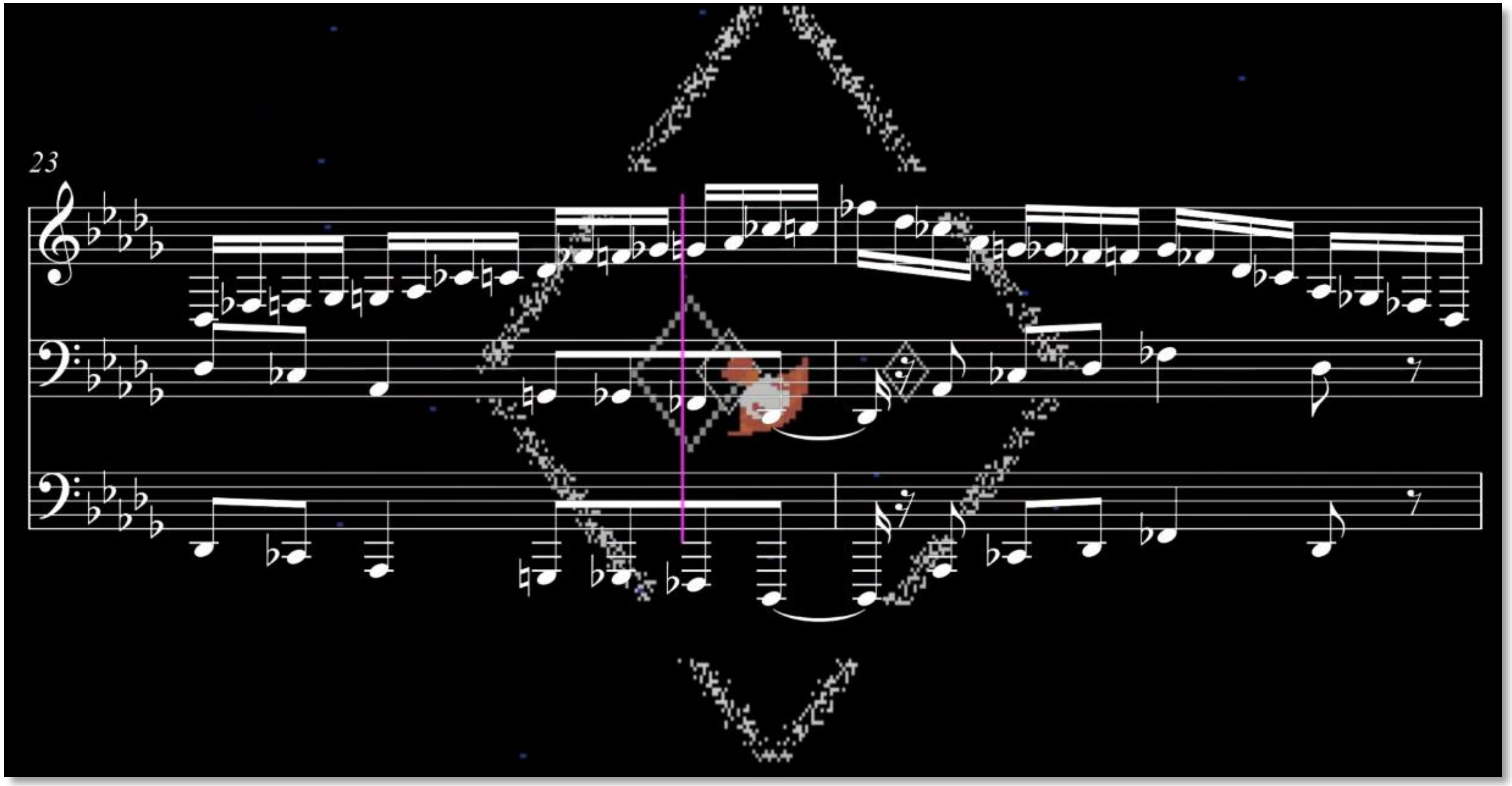
```
10 REM ERRATA FOR C128 SYSTEM GUIDE P156 OR C128 PROGRAMMER'S REF GUIDE P343
20 TEMPO 14
100 AA$="V2O2QA V1IRO4E V2O3HA V1O4IAO5CO4BE V2O3Q#G V1O4IBO5D M"
110 AB$="V1QC V2O3IAE V1O5QE V2O3IAO4C V1Q#G V2O3IBE V1O5QE V2O3IBO4D M"
120 AC$="V2QC V1IAE V2O3QA V1O4IAO5C V2O3Q#G V1O4IBE V2O3QE V1O4IBO5D M"
130 AD$="V1QC V2O3IAE V1O4QA V2O3IAO4C V1HR V2O3IBEBO4D M"
140 AE$="V2QC V1IRO5E V2O3QA V1O5ICE V2O4QC V1IAO5C V2O3QA V1O4IEG M"
150 AF$="V1QF V2IDO3A V1O4QA V2O3IFA V1O5QD V2O3IDF V1O5QF V2O2IAO3C M"
160 AG$="V2O2QB V1IRO5D V2O3QD V1O4IBO5D V2O3QG V1O4IGB V2O3QB V1O4IDF M"
170 AH$="V1QE V2IRO3G V1O4QG V2O3IEG V1O5QC V2O3ICE V1O5QE V2O2IGB M"
180 AI$="V2QA V1IRO5C V2O3QC V1O4IAO5CO4QF V2O3IDF V1O5QD V2O2IBO3D M"
190 AJ$="V2O2QG V1IRO4B V2O2QB V1O4IGBQE V2O3ICE V1O5QC V2O2IAO3C M"
200 AK$="V2O2QF V1IRO4A V2O2QD V1O4IFAQD V2O2IGO3G V1O4QB V2O3IFG M"
210 AL$="V1O5QC V2O3ICG V1H.R V2O4ICEDO3GO4DF M"
220 AM$="V2QE V1IRG V2QC V1O5ICE V2O3QB V1O5IDO4G V2O3QG V1O5IDF M"
230 AN$="V1QE V2O4ICO3G V1O5QG V2O4ICE V1QB V2IDO3G V1O5QG V2O4IDF M"
240 AO$="V2QE V1O5ICO4G V2QC V1O5ICE V2HR V1IDO4GO5DF M"
250 AP$="V1QE V2IRO4G V1O5QC V2O4IEG V1O5QG V2O4ICE V1O5QE V2O3IGB M"
260 AQ$="V2QA V1O6ICO5A V2O4QC V1O5IEA V2O4QE V1O5ICE V2O4QG V1IAO5C M"
270 AR$="V1QD V2O4I#FA V1O5Q#F V2O4ID#F V1O5QA V2O3IAO4D V1O6QC V2O3I#FA M"
280 AS$="V2QG V1O5IBG V2O3QB V1O5IDG V2O4QD V1IBO5D V2O4Q#F V1IGB M"
290 AT$="V1O5QC V2O4IEG V1O5QE V2O4ICE V1O5QG V2O3IGO4C V1O5QB V2O3IEG M"
300 AU$="V2Q#F V1O5IA#F V2O3QA V1O5I#D#F V2O3QB V1O4IBO5#D V2O4Q#D V1I#FA M"
310 AV$="V1QG V2IRE V1O5Q.G V2O4ICEO3A V1O5E V2O4C V1O5C V2O4E V1O5E V2O4G M"
320 AW$="V1QA V2I#FD V1O5Q.#F V2O3IBO4DO3G V1O5D V2O3B V1O4B V2D V1O5D V2O4#F M"
330 AX$="V1QG V2IEC V1O5Q.E V2O3IAO4CO3#F V1O5C V2O3AO4QC V1IAO5C M"
340 AY$="V1O4#F V2R V1O5G V2O3B V1O5#F V2O4C V1O5E V2O3AQB V1O5I#D#F V2O2QB
V1O4IBO5#D M"
350 AZ$="V1QE V2O3IEO4E V1H.R V2O3IBGEO2BGB M"
360 BA$="V2QE V1IRO5G V2O3QE V1O5I#AG V2O3QG V1O5IEG V2O3Q#A V1O5I#CE M"
370 BB$="V2O3Q#C V1O5IGE V2Q.R V1I#CEO4AQ.R V2IGFE M"
380 BC$="V2QD V1IRO5F V2O3QD V1O5IAF V2O3QF V1O5IDF V2O3Q#G V1O4IBO5D M"
390 BD$="V2O2QB V1O5IFD V2Q.R V1O4IBO5DO4GQ.R V2IFED M"
400 BE$="V2QC V1IRO5E V2O3QC V1O5IGE V2O3QE V1O5ICE V2O3Q#F V1O4IAO5C M"
410 BF$="V2O2QA V1O5I#DC V2Q.R V1O4IAO5CO4#FQ.R V2IE#D#C M"
420 BG$="V2O3QB V1IRO5D V2O2QB V1O5IFD V2O3QD V1O4IBO5D V2O3QF V1O4I#GB M"
430 BH$="V2O2Q#G V1O5IDO4B V2Q.R V1I#GBEQ.R V2IDCO3B M"
440 BI$="V2O4QC V1IRE V2O3QA V1O4IAO5C V2O3Q#G V1O4IBE V2O3QE V1O4IBO5D M"
450 BJ$="V1QC V2O3IAE V1O4QA V2O3IAO4C V1Q#G V2O3IBE V1O4QE V2O3IBO4D M"
460 BK$="V1A V2C V1O5C V2O4E V1O5E V2O4A V1O5C V2O4E V1A V2C V1O5C V2O4E V1#F
V2O3A V1O4A V2C M"
470 BL$="V1O5C V2O3#F V1O4A V2O3A V1O4#F V2C V1A V2O3A V1O4#D V2O3#F V1O5C
V2O3A V1O4B V2O3#D V1O4A V2O3#F M"
480 BM$="V2QE V1O4I#GB V2O3Q#G V1O5IDO4B V2O3QB V1O4I#GB V2O3Q#G V1O4IDF M"
490 BN$="V2O3QE V1O4I#GF V2O2QB V1O4IDF V2O2Q#G V1O3IBO4F V2O2QE V1O4IED M"
500 BO$="V2O2QA V1O4ICE V2O3QC V1O4IAE V2O3QE V1O4ICE V2O3QC V1IAO4C M"
510 BP$="V2O2QA V1O4I#DC V2O3QC V1IAO4C V2O2Q#D V1O3I#FO4C V2QR V1O3IBA M"
520 BQ$="V1Q#G V2IRB V1O4QB V2O3I#GE V1O4Q#G V2O3IDB V1O4QE V2O3I#GD M"
530 BR$="V2QC V1IRO4E V2O3QE V1O4IAO5C V2O2Q#G V1O4IBE V2O3QE V1O4IBO5D M"
540 BS$="V2O2QA V1O5ICO4A V2O3Q#F V1O5ICE V2O2QB V1O5IDO4B V2O3Q#G V1O5IDF M"
550 BT$="V2O3QC V1O5IEC V2O3QA V1O5IEG V2O3QD V1O5IFE V2O3Q#A V1O5IDC M"
560 BU$="V2O3Q#G V1O4IBO5C V2O3QF V1O5IDE V2O3QD V1O5IFD V2O2QB V1O5I#GD M"
570 BV$="V2O2Q#G V1O5IBD V2O2QA V1O5ICA V2O2QD V1O5IFD V2O2QE V1O4IBO5D M"
580 BW$="V2O2QF V1O4I#GB V2O2Q#D V1O5ICO4A V2O2QE V1O4IEA V2O3QE V1O4IB#G M"
590 BX$="V2O2WA V1O4IAECEO3HA M"
7000 PLAY"U9V1T5V2T5V3T5":REM INIT INSTRUMENTS
7010 PLAY AA$:PLAY AB$:PLAY AC$:PLAY AD$:PLAY AE$:PLAY AF$:PLAY AG$:PLAY AH$
7020 PLAY AI$:PLAY AJ$:PLAY AK$:PLAY AL$:PLAY AM$:PLAY AN$:PLAY AO$:PLAY AP$
7030 PLAY AQ$:PLAY AR$:PLAY AS$:PLAY AT$:PLAY AU$:PLAY AV$:PLAY AW$:PLAY AX$
7040 PLAY AY$:PLAY AZ$:PLAY BA$:PLAY BB$:PLAY BC$:PLAY BD$:PLAY BE$:PLAY BF$
7050 PLAY BG$:PLAY BH$:PLAY BI$:PLAY BJ$:PLAY BK$:PLAY BL$:PLAY BM$:PLAY BN$
7060 PLAY BO$:PLAY BP$:PLAY BQ$:PLAY BR$:PLAY BS$:PLAY BT$:PLAY BU$:PLAY BV$
7070 PLAY BW$:PLAY BX$
```

DEMO 4 / Code Walkthrough

ChiptuneSAK Now Available

- https://github.com/c64cryptoboy/ChiptuneSAK
  - Docs: https://chiptunesak.readthedocs.io/en/latest/
- Free, open source software
  - fork it, sell it, put it on t-shirts, do whatever you want
- Some importers and/or exporters that could be fun to add:
  - MusicXML, MOD, NSF (Nintendo Sound Format, 6502), SAP (Slight Atari Player, 6502), various VGM formats, a 3SID tracker (e.g., SID-Wizard 1.8), Minecraft Note Blocks, Mario Paint Composer, ABC notation, etc.

My First Use of ChiptuneSAK



- Russell Lieblich’s Score from *Master of the Lamps* (Activision, 1985)
  - Made a YouTube video where the sheet music follows the gameplay
- With ChiptuneSAK, I could finally play around with this RSID

youtu.be/HH9sVayG0oQ

We are particularly grateful to:

Ian Lee

- Python practices consulting. Currently helping to get our code as a project on PyPI.

Markus Brenner

- Tried out our framework by coding up an Ultima music importer from Kenneth Arnold's Apple][Mockingboard player code

Hasse Axəlsson-Svala

- Offered up much-needed GoatTracker and Stereo GoatTracker.sng files as test data

Questions?

Suggestions?

Thanks!