

Riding The Wave

Using radare2 – by pancake



Use Cases

R2 is known to be a very versatile tool.

It was born as a forensics tool and evolved into debugging, static analysis, emulation, data and code visualization, scripting, decompilation..

But there's still an area to explore...

DISCLAIMER

I'm not a Musician or an expert on Sound processing.

But I wanted to learn a bit on those topics and used r2 as an excuse.

As always (R)

The Waves

- We all love messing with bytes and hexadecimal values
- Chiptune music trackers and editors user interfaces look like hex editors
- Tweaking r2 a bit we can use it to make music

But why stick to just make music?

- isn't r2 a re framework? Provide tools to find and identify waves
- Parse music binary files, do and reverse stego, etc
- Improve visualization commands for data analysis

Sounds Chips

- Some of them are coprocessors that create different types of waves like sid or a basic pc speaker.
- Others provide a dsp interface, which basically plays what we feed in using a specific binary format to define the waves data.
- Everybody have tried this already
 - \$ cat /dev/urandom > /dev/dsp
 - \$ cat /dev/sda > /dev/dsp

Uses

- Creating music with code
- Demoscene
- Electronic music / Chiptune
- Mostly for artistic purposes



Playing music in DOS

- Use rasm2 to assemble a .COM and use DOSBOX to run the PoC
- <http://www.intel-assembler.it/portale/5/make-sound-from-the-speaker-in-assembly/8255-8255-8284-asm-program-example.asp>



The screenshot shows a DOSBox window running on a Mac OS X desktop. The assembly code in the left pane is as follows:

```
rasm2 -f intel -l intel -o ./music.com -a 0x00000000
[0x00000000]> pd 20
0x00000000 b0f6    mov al, 0xb6 ; 182
0x00000002 e643    out 0x43, al ; 'C'
0x00000004 b8d01e00 mov eax, 0x42e611d0
0x00000009 88e0    mov al, ah
0x0000000b e642    out 0x42, al ; 'B'
0x0000000d e461    in al, 0x61 ; 'a'
0x0000000f e603    or al, 3
0x00000011 e661    out 0x61, al ; 'a'
0x00000013 bb19@b9. mov ebx, 0xffffb90019
0x00000018 ff4975 dec dword [rcx + 0x75]
...
0x0000001c 4675F7 jne 0x16
0x0000001f e661    in al, 0x61 ; 'a'
0x00000021 24fc    and al, 0xfc
0x00000023 e661    out 0x61, al ; 'a'
0x00000025 e661    invalid
0x00000026 e661    invalid
0x00000027 e661    invalid
0x00000028 e661    invalid
0x00000029 e661    invalid
[0x00000000]> o
3 * --- 0x00000025.music.com
[0x00000000]> l
```

The right pane shows the DOSBox file system:

```
DOSBox 0.74, CPU speed: 3000 cycles, Frameskip 0, Program: DOSBOX
./
.
  <DIR> 25-08-2018 13:59
..
  <DIR> 03-11-2018 3:58
COM'1  ENR  <DIR> 21-08-2018 3:58
COM'2  JBL  <DIR> 21-08-2018 3:58
POWERLOG  <DIR> 21-08-2018 3:57
.A  12,646,573 21-08-2018 18:29
.B  1,269,269 21-08-2018 18:29
.C  9,682,328 21-08-2018 18:29
ADB  LOG  339 24-08-2018 15:50
CMS  TXT  4,495 21-08-2018 23:47
DRIVEF'1 501  <DIR> 21-08-2018 3:58
DRIVEF'2 501  <DIR> 21-08-2018 3:58
FIX-RZ'1 PAT  2,095 23-08-2018 17:10
ISPY-M'1 PAT  2,339 22-08-2018 16:55
KEYSTU'1  <DIR> 21-08-2018 3:58
MCHD'1 0  29,632 22-08-2018 17:19
P  2,743 22-08-2018 17:19
RUN  109 24-08-2018 3:40
11 File(s)  23,529,992 Bytes.
7 Dir(s)  262,111,744 Bytes free.
```

The bottom status bar shows the path: /Users/pancake/prg/radare2-extras/keystone

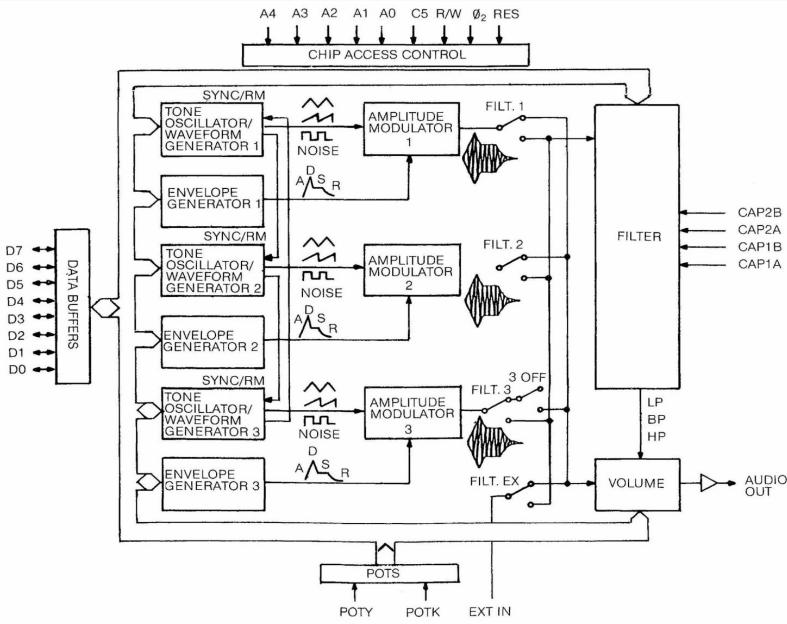
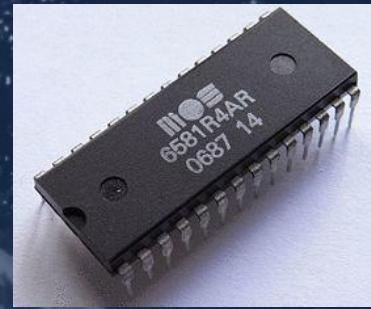
The SID

Probably the most famous chip in
the chiptune scene. C64=

- SID 6581/8580
(3x osc, 4x wave, filter, ADSR, ring)

http://archive.6502.org/datasheets/mos_6581_sid.pdf

<https://chipmusic.org/forums/topic/8189/basic-guide-to-programming-the-sid-with-assembly-language/>



6581 SID BLOCK DIAGRAM

The Gameboy



Music was created
programmatically or implementing
a tracker, but everything was
done internally.

<http://www.chrisantonellis.com/files/gameboy/gb-programming-manual.pdf>

<https://www.youtube.com/watch?v=n3Jqv8t7JU0>

```
RYOGA [Sun\EMULADORES\ROBOT_testSoundMusic.c]
File Search View Document Tools Macros Window Help
C:\RISC - MASTER
void STOP_SONG_FROM_MBC1_BANK_X(){
    gbt_Stop();
}

void PRINT_MESSAGE(int x, int y, char *c){
    gotoxy(x,y);
    printf(c);
}

void CLEAR_SCREEN_LINE ( UBYTE y ){
    UBYTE x;
    for (x = 0 ; x < (y+1)*28 ; x++ ){
        set_bg_data(x,tile_offset,(unsigned char *)TEXT_CEHPTV);
    }
}

void CLEAR_SCREEN_BACKGROUND(void){
    UBYTE x;
    for ( x = 0 ; x < tile.height ; x++ ){
        CLEAR_SCREEN_LINE ( x );
    }
}

void PLOT_BACKGROUND_TILE(int coord_X, int coord_Y, int tan_tile_X, int tan_tile_Y, unsigned char *tile_data, unsigned char *map_data){
    set_bg_data(coord_X, coord_Y, tile_data);
    set_bg_tile(coord_X, coord_Y, tan_tile_X, tan_tile_Y, map_data);
    SHDU_BKG;
    DISPLAY_ON;
}

95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
```

SOUND PROGRAMMING FOR GAMEBOY (BANK SWITCHING) - GBDK: Game Boy C Programming [GAMEBOY HOMEBREW]

1,443 views

LIKE DISLIKE SHARE

LSDJ / NanoLoop

There are ROM cartridges for gameboy (as well as other console) to use them as music instruments. Having a sample tracker interface and interacting with the gamepad controls.

SONG	PUI	PU2	WAV	NOI	PUI
C	20	49	62	71	
D	09	30	61	71	
E	00	30	62	71	
F	21	50	61	71	150
G	22	51	62	71	1E 5
H	23	52	61	71	2E 4
I	24	53	62	71	WF 4
J	25	54	61	71	NC 3
K	26	55	62	71	
L	27	56	61	71	
M	28	57	62	71	
N	09	30	61	71	
O	00	30	62	71	
P	05	35	61	71	P
Q	06	36	62	71	S
R	07	37	61	71	G

NES: NTRQ

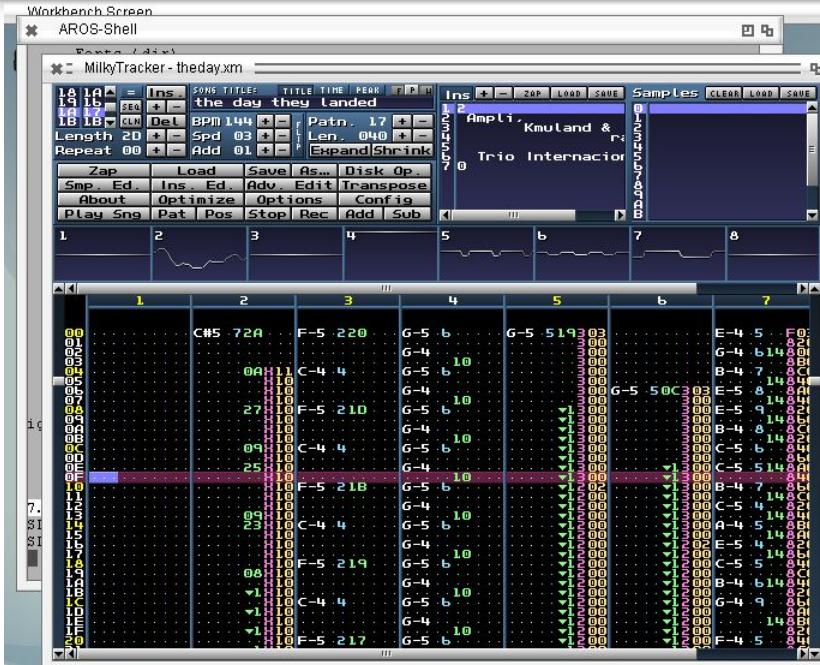
There are also chiptune trackers for other console platforms.

- This is for NES/Famicom

*NTRQ***		A	B	C	D	E	HT	SPD
SNG	02							01 0 000
SPD	20							02 0 000
PLN	20							03 0 000
PTS	50							04 0 000
00	F 6 03	F 6 03	F 5 03	1 8 03	3 8 03			
01	G 6 - -	G 6 - -	G 5 - -	1 A - -	3 A - -			
02	G 8 - -	G 8 - -	G 7 - -	1 2 - -	5 2 - -			
03	A 6 - -	A 6 - -	A 5 - -	1 C - -	3 C - -			
04	A#4 - -	A#4 - -	A#3 - -	0 5 - -	2 5 - -			
05	B 1 - -	B 1 - -	B 0 - -	0 2 - -	0 2 - -			
06	C 3 - -	C 3 - -	C 2 - -	0 F - -	0 F - -			
07	C 4 - -	C 4 - -	C 3 - -	1 B - -	1 B - -			
08	C#5 - -	C#5 - -	C#4 - -	0 8 - -	2 8 - -			
09	B 4 03	B 4 03	B 3 03	0 6 03	2 6 03			
0A	- - 02	- - 02	- - 02	- - 02	- - 02			
INSTRMNT : SR								
00	0 1 8 8 0 0 0 0 0 0	0 0 FFF	0 0 0 0 0 0					
01	0 1 8 8 0 0 0 0 0 0	0 1 0 0 F 0	0 1 0 0 0 0					
02	0 1 8 8 0 0 0 0 0 0	0 2 0 0 F 0	0 2 0 0 0 0					
03	0 0 0 2 8 0 0 0 0 0 0	0 3 0 0 F 0	0 3 0 0 0 0					
04	0 0 0 2 8 0 0 0 0 0 0	0 4 0 0 F 0	0 4 0 0 0 0					

MilkyTracker

That's a nice multi-platform,
open-source music tracker.





Encoding

Encoding it in data

- Configure the audio device
 - Signed 16bit integers
 - Mono / Stereo
 - 4000hz resolution XXX

Audio wave is encoded as an array of shorts (16bit mono). In the case of being stereo, it is composed by a pair of 16bit signed values.

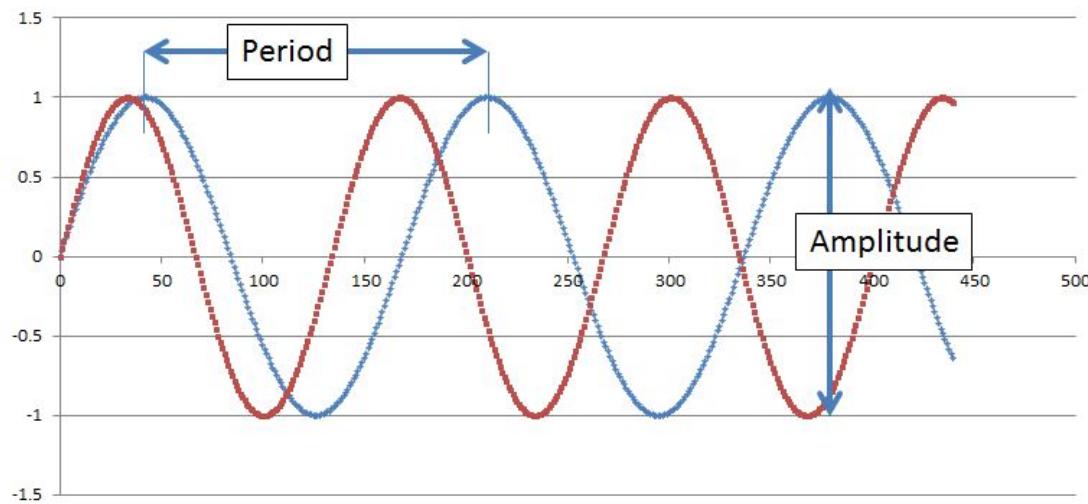
Easy!

The background of the image is a dark, deep blue color with a highly textured, wavy pattern. This texture is reminiscent of ocean water or turbulent air, with numerous small, white-capped crests and deep, dark troughs. The lighting is somewhat dim, creating a moody and dramatic atmosphere.

Waves

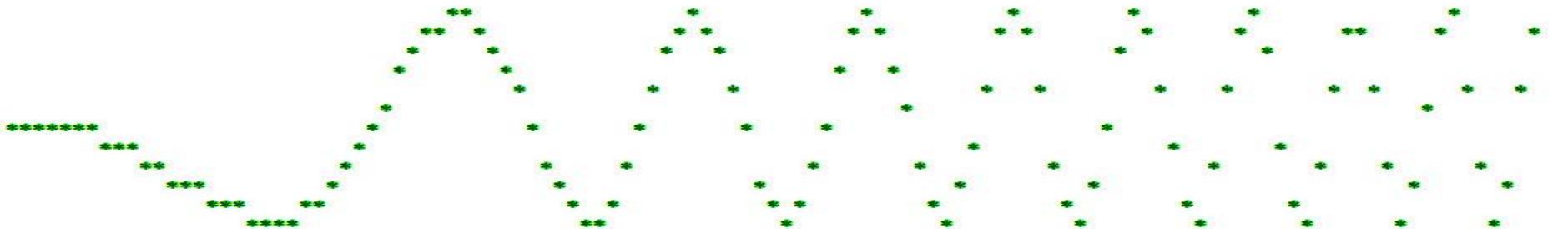
Amplitude

Aka volume. We can change the amplitude by dividing or multiplying each short value provided



Frequency

- Bass, Treble
- Defines the pitch
- Measured in Hz
- How many cycles the sinus wave will perform in



Arpeggio

The action of playing multiple relative tones starting at a base one in a short period of time.

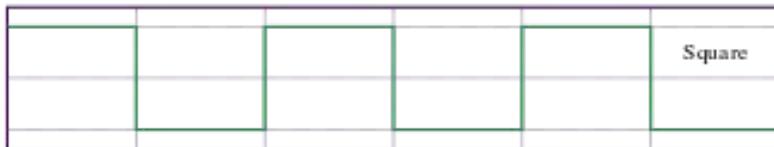
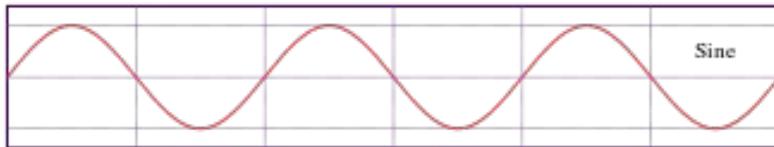
- See the auE command

Some musicians use arpeggio to simulate voices and chirps.

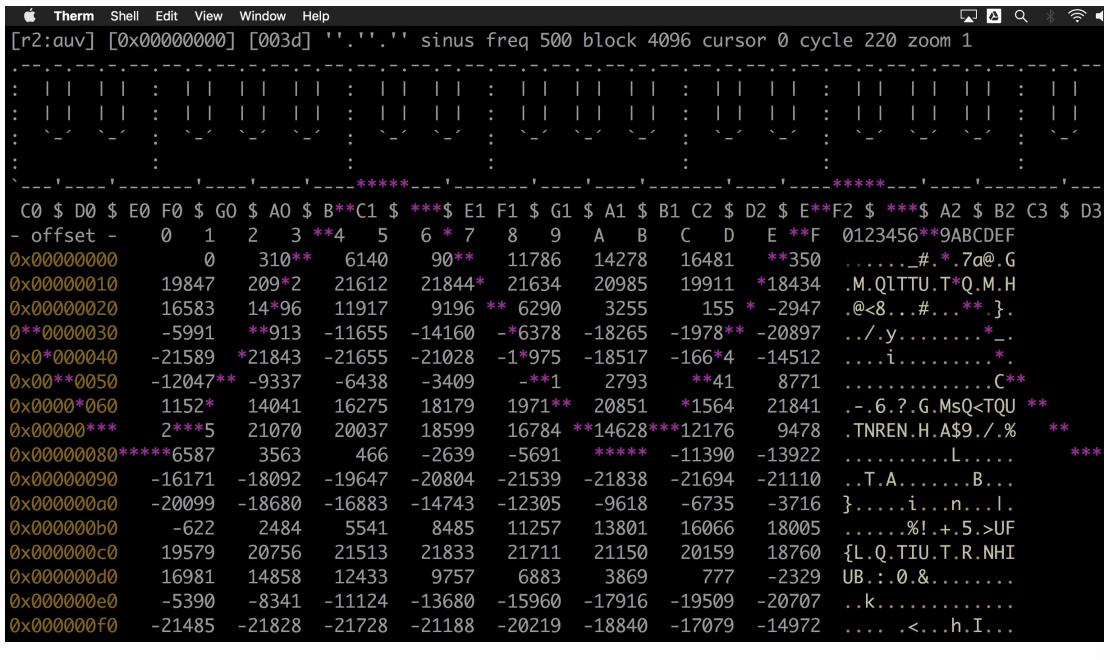
- Well known blip sound from chiptune music.

Shape

- Triangle
- Square (Pulse)
- Sinus (Same as cosinus)
- Saw
- Noise
- Silence
- ...



Tones



Notes

The notes defined in the Piano keys
are always the same frequency.

Those can be exported flags.

MIDI number	Note name	Keyboard	Frequency Hz	Period ms
21	A0		27.500	36.36
23	B0		30.868	32.40
24	C1		32.703	30.58
26	D1		36.708	27.24
28	E1		41.203	25.71
29	F1		43.654	22.91
31	G1		48.999	20.41
33	A1		55.000	19.26
35	B1		61.735	18.18
36	C2		65.406	17.16
38	D2		73.416	16.20
40	E2		82.407	15.29
41	F2		87.307	14.29
43	G2		97.999	12.13
45	A2		110.00	10.81
46	B2		123.47	9.091
47	C3		130.81	8.581
48	D3		146.83	7.645
50	E3		164.81	7.216
52	F3		174.61	6.068
53	G3		196.00	5.727
55	A3		220.00	5.102
57	B3		246.94	4.816
59	C4		261.63	4.545
60	D4		293.67	4.290
62	E4		329.63	3.822
64	F4		349.23	3.608
65	G4		392.00	3.214
67	A4		440.00	2.863
69	B4		493.88	2.551
71	C5		523.25	2.273
72	D5		587.33	2.025
74	E5		659.26	1.804
76	F5		698.46	1.607
77	G5		783.99	1.432
79	A5		880.00	1.351
81	B5		987.77	1.204
83	C6		1046.5	1.073
84	D6		1174.7	1.012
86	E6		1318.5	0.9556
88	F6		1396.9	0.8513
89	G6		1568.0	0.8034
91	A6		1760.0	0.7159
93	B6		1975.5	0.6757
95	C7		2093.0	0.5682
96	D7		2349.3	0.5363
98	E7		2637.0	0.4778
100	F7		2793.0	0.4257
101	G7		3136.0	0.4018
103	A7		3520.0	0.3580
105	B7		3951.1	0.3010
107	C8		4186.0	0.2681
108			J. Wolfe, UWSW	0.2389



Noise

- Brown
- White
- Pink

But there are more! Choose it with the `aun` command.

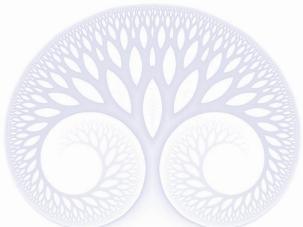
- Red, Green, Blue, Violet, Grey, Black, ...

https://en.wikipedia.org/wiki/Colors_of_noise

Binaurals!

- Mixing different frequencies and patterns of noises it is possible to stimulate parts of the brain to help on concentration, sleeping, meditation...

Now r2 can help you make your baby sleep!



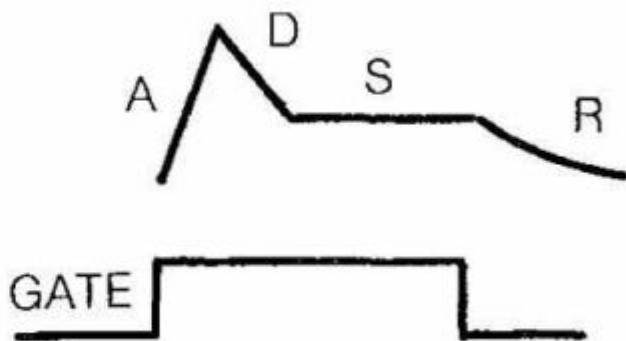
Effects

- Fade in/Fade out
- Pitch Glitching
- Frequency change over time
- Mirroring with delay (echo)
- Volume level

Attack, Decay, Sustain, Release

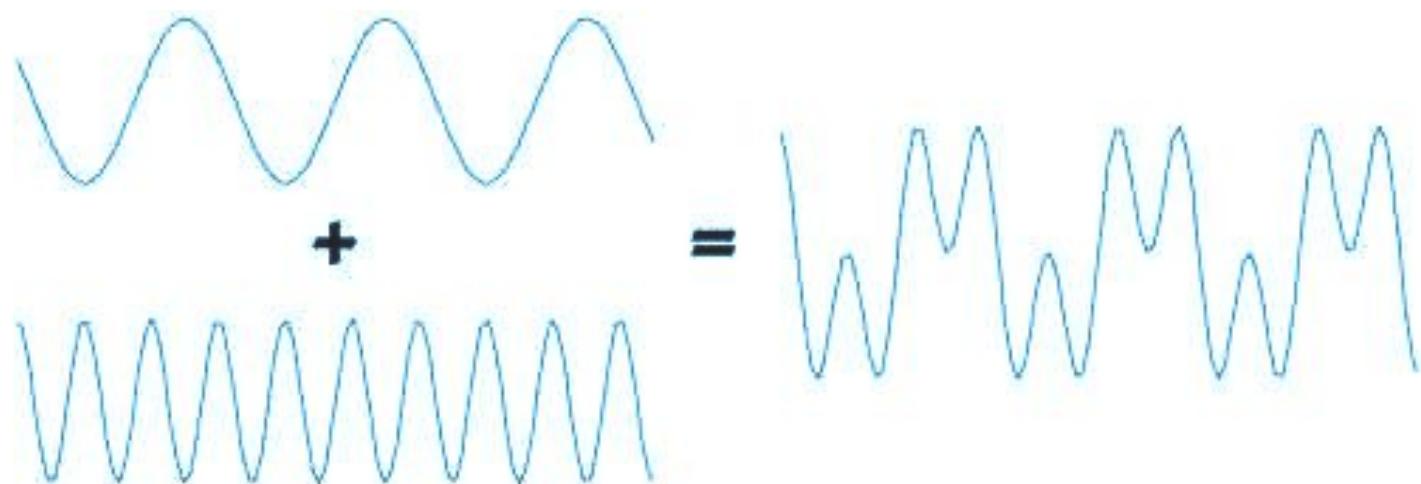
- Must be applied as post-processing filters

ATTACK:	10 (\$A)	500 mS
DECAY:	8	300 mS
SUSTAIN:	10 (\$A)	
RELEASE:	9	750 mS



Mixing waves

Use the aum command to sub the short values.





Designing our Audio CPU

Extending Existing Arch

We can easily extend r2 to handle syscall interruptions when emulating any architecture.

- Write assembly for mips, arm, x86, z80, ...
- Interact with the audio processor module using syscalls
- Hook e esil.hook.intr=
- We could emulate gameboy audio chip

But this is not that fun...

The Outside World

- Mapping hardware registers
- Making them available right from the register set
- Using IO ports?
- Using syscalls?
- Background mainloop thread?

Designing a new Audio CPU

- Register size
- Number of registers
- Instruction encodings
- Endianness
- How to make it sound?

ESIL

One of the best features of ESIL is the flexibility and extensibility of the language itself. Allows us to have our own custom instructions, hook memory operations, implement syscalls... and now, you can even run r2 commands!

- ahe !?E Hello world
- aes

The Plugin

r2au

- r2pm -i r2au

The dependencies to build this plugin are the following:

- C compiler
- pkg-config
- r2 installed
- Libao installed. (Linux, Mac), should be easy to build on Windows.

libao: How it works?

- `ao_initialize()`
 - `Driver = ao_default_driver_id()`
 - `Dev = ao_open_live (Driver, rate, bts, channels, endian)`
 - `ao_play (buffer, length)`
 - `ao_close(Dev)`
- `ao_shutdown()`

Commands in r2au

```
[0x00000000]> au
```

```
Usage: au[.abfeEimopvw] [args]
```

```
au. - play current block (au.& in bg)
```

```
aua - analyze wave in current block
```

```
aub - audio blocksize
```

```
auf - flags per freqs associated with keys
```

```
aye - audio effects (arpeggio, percent, ...)
```

```
auE [arpeggio] - arpeggio chords to use
```

```
aui - init audio
```

```
...
```

Commands in r2au (2)

...

aum - mix from given address into current with bsize
aun [noisetype] - select [white, pink, brown]
auN [amplitudes] - space separated volume changes 100 = 1
auo - apply operation with immediate
aup - print wave (aupi print piano)
auv - visual wave mode
auw - write wave (see auw?)

Commands in r2au (auw?)

Write waves following the specific shape, frequency, amplitude and arpeggio settings.

- See aun, auE and auN

```
[0x00000000]> auw?
```

Usage: auw[type] [args]

fill current block with wave

default wave shape type can be changed like this:

> 'auws;auw 400' is the same as > 'auws 400'

args: frequence

types:

(s)in .'.':.'

(c)os :..:..

(z)aw /|/|/|/

(Z)aw \|/\|/\|

(p)ulse |_|'_|_|

(v)pulse '___'_

(n)oise /.:/.:.

(k)ross.. ><><>

(t)ri.. /_/_/_

(-)silen _____

(i)nc _.-.-'`

(d)ec '`--...`

```
[0x00000000]>
```

Background Loop

TheStranger implemented a pretty nice tasking model inside r2 that can be used to run code in background threads by sharing time and blocking when running a command. This allows us to run a play command repeated N times in background while we modify the buffer in real-time using the visual audio mode.

The background of the image is a dark, moody texture that looks like turbulent water or a cloudy sky. It features various shades of blue and grey, with bright white highlights that create a sense of depth and movement. The overall atmosphere is somber and dramatic.

Visual

auv: visual audio mode

Scroll around the memory viewing the:

- Waveform
- HexDump
- Write keybindings
- Jam mode

Interactions

- Similar to r2 using hjkl to move around
- +/- to increase/decrease
- p/P to change the print mode
- Press '?' for help

GamePad Support

If we want r2 embedded in
consoles we need a gamepad
designed interface.

- make -C gamepad
- make -C gamepad install
- r2 -c “god mode on” /bin/ls



DEMO

```
$ r2 -cauv malloc://4M
```

Show the following actions

- Make a sin wave
- Play it
- Create a wav file
- Enter audio visual mode

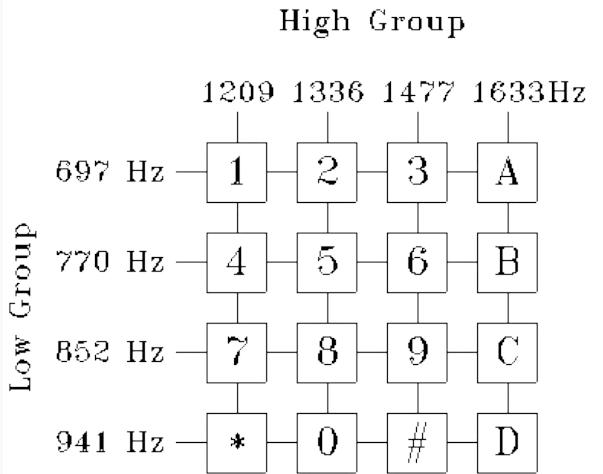
Bonus Demos

Telephone

Taking two sounds and mixing them together.

To make each number and * # sound like a key.

- auv!



```
uv] [0x00000000] r2p
-
.-'-'-----.
| ==. |
| .-----.| 
|| dtmf: ||
|| 662772121|
|`-----'|
| [ ( ) . ] |
| .-----|
| | [1][2][3] |
| | [4][5][6] |
| | [7][8][9] |
| | [*][0][#] |
| -|
`-----'
```

JAM

Map your keyboard to
different notes and play
in real time

[r2:auv] [0x00000000] [0053] ..'...'.. cos.. freq 261 block 4096 cursor 0 cycle 220 zoom 1

	C4 \$ D4 \$ E4 F4 \$ G4 \$ A4 \$ B4 C5 \$ D5 \$ E5 F5 \$ G5 \$ A5 \$ B5 C6 \$ D6 \$ E6 F6 \$ G6 \$ A6	
0x00000000	21845	83% [#####]
0x00000006	19705	80% [#####]
0x0000000c	13707	70% [#####]
0x00000012	5024	57% [#####]
0x00000018	-4642	42% [#####]
0x0000001e	-13400	29% [#####]
0x00000024	-19533	20% [#####]
0x0000002a	-21841	16% [#####]
0x00000030	-19871	19% [#####]
0x00000036	-14010	28% [#####]
0x0000003c	-5405	41% [#####]
0x00000042	4258	56% [#####]
0x00000048	13088	69% [#####]
0x0000004e	19354	79% [#####]
0x00000054	21830	83% [#####]
0x0000005a	20031	80% [#####]
0x00000060	14309	71% [#####]
0x00000066	5784	58% [#####]
0x0000006c	-3873	44% [#####]
0x00000072	-12772	30% [#####]
0x00000078	-19169	20% [#####]
0x0000007e	-21813	16% [#####]
0x00000084	-20184	19% [#####]
0x0000008a	-14602	22% [#####]

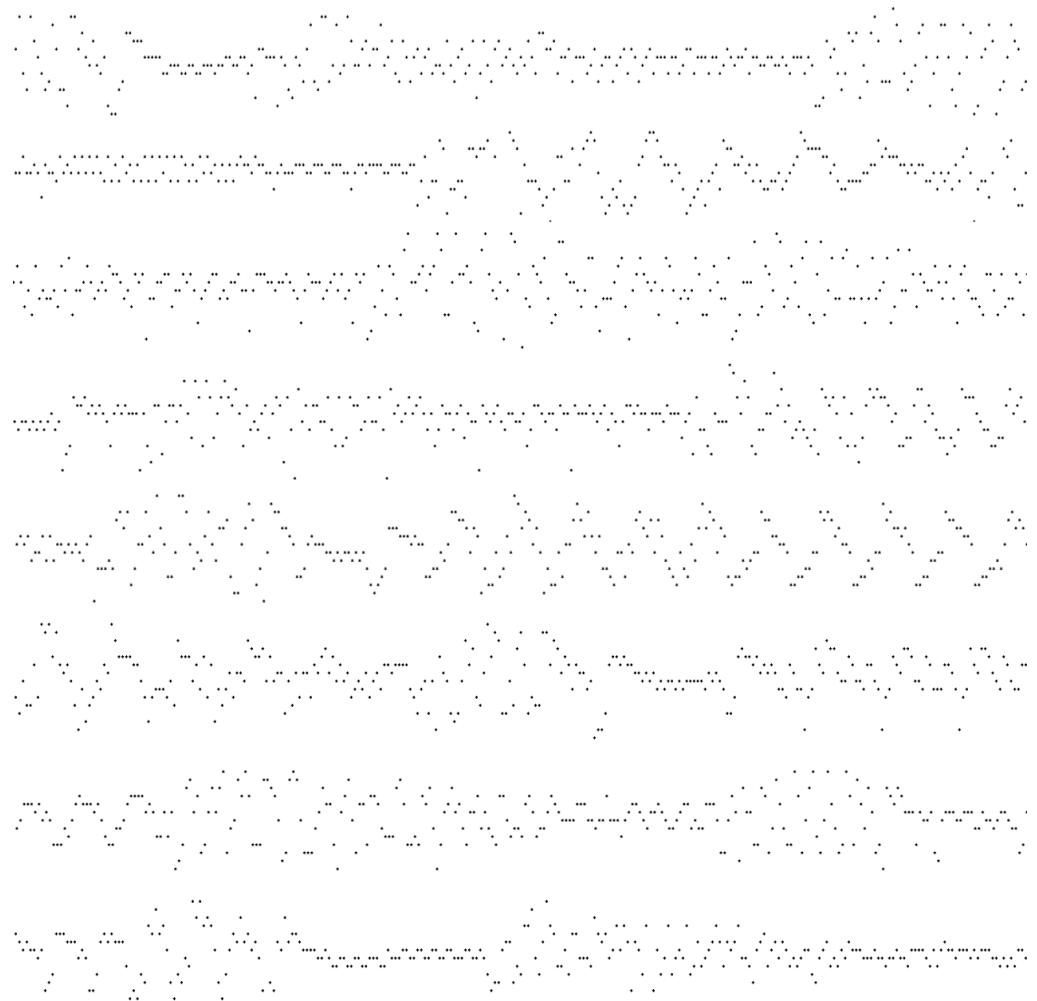
Samples

Importing a sample from
an mp3/wav

```
[0x000000000 0% 230 malloc://1M]> pd $r
0x0000000000000000 0000fc17 nop
0x0000000000000004 0a2e .short 11786
0x0000000000000006 6140 .short 16481
0x0000000000000008 874d .short 19847
0x000000000000000a 6c54 .short 21612
0x000000000000000c 8254 .short 21634
0x000000000000000e c74d .short 19911
0x0000000000000010 c740 .short 16583
0x0000000000000012 8d2e .short 11917
0x0000000000000014 9218 .short 6290
0x0000000000000016 9b00 .short 155
0x0000000000000018 99e8 .short -5991
0x000000000000001a 79d2 .short -11655
0x000000000000001c 06c0bbb2 wait 48050
0x0000000000000020 abab .short -21589
0x0000000000000022 69ab .short -21655
0x0000000000000024 f9b1 .short -19975
0x0000000000000026 d4be .short -16684
0x0000000000000028 f1d0 .short -12047
0x000000000000002a dae6 .short -6438
0x000000000000002c c9fe .short -311
0x000000000000002e d116 .short 5841
0x0000000000000030 032d933f mov r13, r32
0x0000000000000034 034d3c54 mov r13, r64
0x0000000000000038 ab54 .short 21675
0x000000000000003a 454e .short 20037
0x000000000000003c 9041 .short 16784
0x000000000000003e 902f .short 12176
0x0000000000000040 bb19 .short 6587
0x0000000000000042 d201 .short 466
0x0000000000000044 c5e9 .short -5691
0x0000000000000046 82d3 .short -11390
0x0000000000000048 d5c0 .short -16171
0x000000000000004a 41b3 .short -19647
0x000000000000004c ddab .short -21539
0x000000000000004e 42ab .short -21694
0x0000000000000050 7db1 .short -20099
0x0000000000000052 0dbe .short -16883
0x0000000000000054 efcf .short -12305
0x0000000000000056 b1e5 .short -6735
0x0000000000000058 92fd .short -622
0x000000000000005a a515 .short 5541
```

Play Song

Make a song like in the
90's with Nokia



Sleep Fix

Try brown noise with your children!

\$ r2 -c'aui;auN brown;'

-c 'aub 1M;auwn 400;au.'

malloc://1M



AntiMosquito

We can finally be safe
from mosquitos with
radare2! (or not)





Questions?