Because software fun
doesn't have to
end with `make install`

BINARY EXPLORATION

Ensuring compatibility

- Many popular file formats go undocumented. This can be accidental or intentional.
- Often, the value of your software depends on its ability to interact with a popular format.

Accessing data

- Whether for reasons legal or otherwise, sometimes you really need to get at the information in a file.
- If the software to open it is gone, disassembly is impossible. Using an RE approach is your only hope.

Customization

- Replace graphics, text and levels in old games to give them new life.
- Sometimes design decisions (like hotkeys) may just bug the shit out of you.
  - Why not just “fix” them?


This is not how you reverse engineer
BEN AFFLECK AARON ECKHART UMA THURMAN
PAYCHECK
12.25.03

Necessary skills

- Logical reasoning
- Ability to form and test hypotheses
- Pattern recognition
- Literate in binary and hexadecimal

Optional skill:

- Fluent in assembly on target platform

Common types of data

- Resource packages
  - Used to contain many files in one stream
- String containers
- STRUCT data
- Graphical data
  - Bitmap and bitplane
- Network streams

Resource packages

- Things we will need to store:
  - Magic bytes – used to identify
  - File count – what's in the package?
  - Name – name of contained file, if named
  - Offset – where the file begins
  - Size – how many bytes used
  - Data – file contents

Data in raw chunks

```
0000h:  00080000 00C80000 00280300 00600400
0010h:  00700600 00100900 00200B00 00C00D00
0020h:  00401000 00F01200 00C81500 00881800
0030h:  00F81A00 00981E00 00502100 00382500
…
0210h:  00586A01 00A86D01 00C87001 00F87201
0220h:  00F07601 00F87601 00407901 00487901
0230h:  00C07901 00187D01 00000000 00000000
…
07E0h:  00000000 00000000 00000000 00000000
07F0h:  00000000 00000000 00000000 00000000
0800h:  40000000 00002500 30000000 34000000
0810h:  34000000 38000000 00000000 00000000
```

File position + size

```
0000h:  43000000 45502E44 41540000 00000000  C...EP.DAT......
0010h:  00000000 4C060000 15160000 455F3031  ....L.......E_01
0020h:  4A2E4441 54000000 00000000 611C0000  J.DAT.......a...
0030h:  77000000 455F3031 502E4441 54000000  w...E_01P.DAT...
0040h:  00000000 D81C0000 FD010000 455F3032  .... ... ...E_02
…
0630h:  69090000 5343454E 4D32322E 44415400  i...SCENM22.DAT.
0640h:  00000000 124A0100 1C0B0000 6F514C50  .....J......oQLP
0650h:  184C505D 185A594C 4C545D18 5D565C5D  .LP].ZYLLT].]V\]
0660h:  5C141850 5D184F59 4B185948 48575156  \..P].OYK.YHHWQV
```

String containers

- Things we need to store
  - Offset – where string starts
  - Length – how long the string is, if not terminated
  - Handle – alternative link to string
  - Text – text in the string

Text with pointers

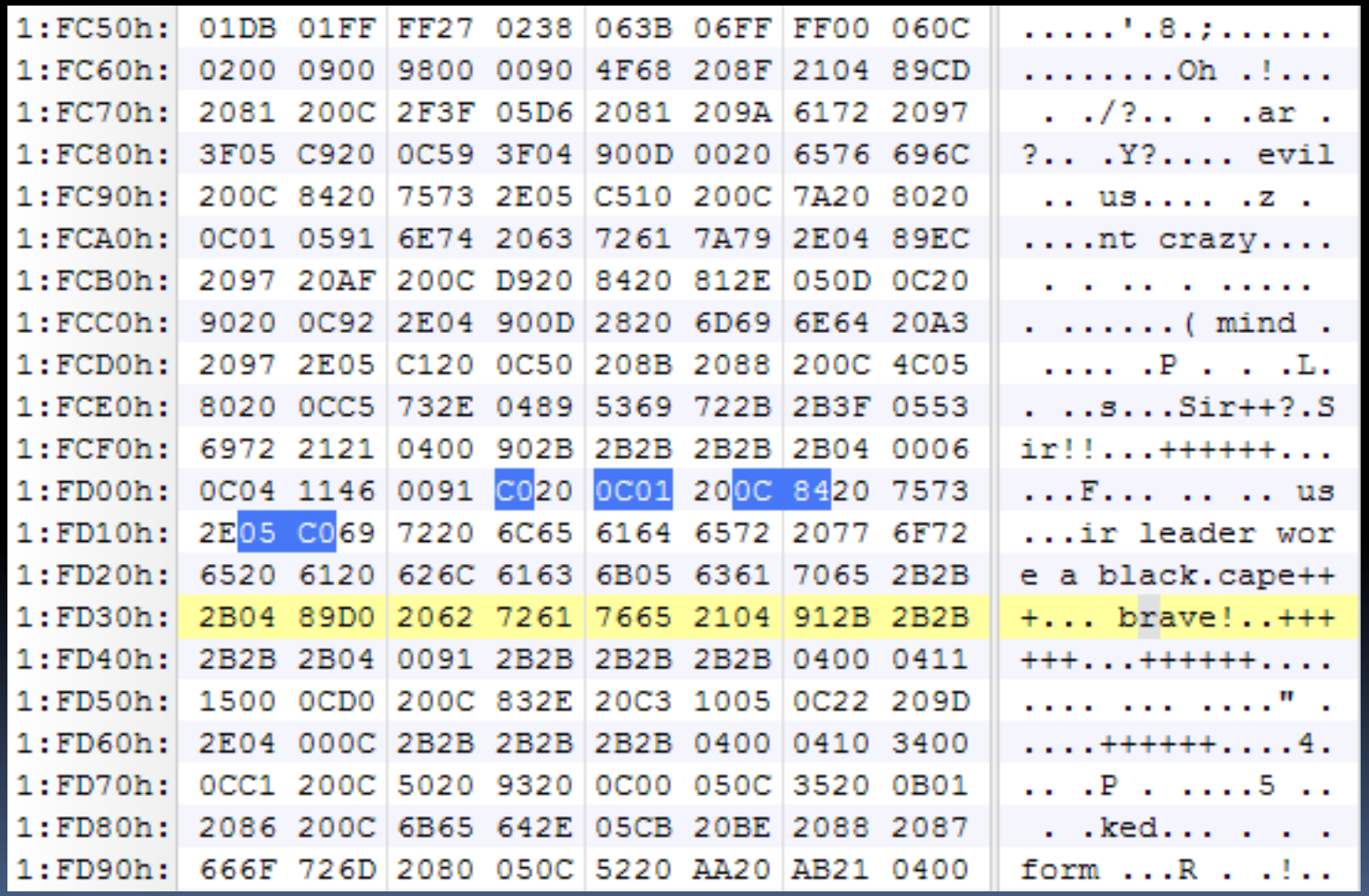
```
0000h:  2900 03CD CDCD 0000 0000 0500 0000 A672  )..迂.........字
0010h:  A6EA 3000 0000 0000 0000 0000 0000 03CD  串0.............
0020h:  CDCD 0500 0000 1100 0000 A672 A6EA 3100  迂.........字串1.
0030h:  0000 0000 0000 0000 0000 03CD CDCD 1600  ............迂..
0040h:  0000 0800 0000 A672 A6EA 3200 0000 0000  ......字串2.....
0050h:  0000 0000 0000 03CD CDCD 1E00 0000 0A00  .......迂.......
0060h:  0000 A672 A6EA 3300 0000 0000 0000 0000  ..字串3.........

0460h:  0000 03CD CDCD 9C01 0000 0A00 0000 A672  ...迂.........字
0470h:  A6EA 3430 0000 0000 0000 0000 0000 0000  串40............
0480h:  A5EC BAB8 0039 3837 7332 3033 6131 402E  伊爾.987s203a1@.
0490h:  4150 532C 3100 2A2E 4150 532C 3100 3531  APS,1.*.APS,1.51
04A0h:  312C 3332 372C 3000 3133 3539 2C34 3833  1,327,0.1359,483
```

Raw, unterminated strings

A5D0h:	A4B0	BBF2	A148	A7F5	B370	BBBB	A147	B9B3	什麼？李逍遙：像
A5E0h:	A741	B36F	BAD8	C65A	BEEE	AABA	A448	A141	你這種蠻橫的人，
A5F0h:	A4A3	B5B9	A741	A440	C249	B1D0	B056	ABE7	不給你一點教訓怎
A600h:	A6E6	A148	C5FD	A970	A45D	B9C1	B9C1	B351	行？讓妳也嘗嘗被
A610h:	A651	A662	BEF0	A457	AABA	B4FE	A8FD	A149	吊在樹上的滋味！
A620h:	AA4C	A4EB	A670	A147	A741	A149	A149	A7D6	林月如：你！！快
A630h:	A7E2	A7DA	A9F1	A446	A149	A4A3	B54D	A7DA	把我放了！不然我
A640h:	A65E	A568	A440	A977	A573	A7DA	AF52	ACA3	回去一定叫我爹派
A650h:	A448	A7E2	A741	ADCC	B371	B371	A7EC	B05F	人把你們通通抓起
A660h:	A8D3	A141	A5B4	C25F	A741	ADCC	AABA	BB4C	來，打斷你們的腿
A670h:	A149	BBC8	AAE1	A147	A4BD	A46C	A149	B17A	！銀花：公子！您
A680h:	A9F1	A446	A470	A96A	A761	A149	AC4F	A5A3	放了小姐吧！是奴
A690h:	B141	B9EF	A4A3	B05F	A470	A96A	A141	A470	婢對不起小姐，小
A6A0h:	A96A	A575	AC4F	A662	AEF0	C059	A457	A141	姐只是在氣頭上，
A6B0h:	A8C3	A4A3	AC4F	AF75	AABA	AD6E	B1FE	AE60	並不是真的要殺害
A6C0h:	A7DA	ADCC	AA4C	A4EB	A670	A147	BBC8	AAE1	我們林月如：銀花
A6D0h:	A149	C1D9	A4A3	B4C0	A7DA	C350	B86A	A149	！還不替我鬆綁！
A6E0h:	A65E	C059	ACDD	A7DA	ABE7	BBF2	BEE3	AA76	回頭看我怎麼整治
A6F0h:	A741	B36F	BDE2	A448	A7F5	B370	BBBB	A147	你這賤人李逍遙：
A700h:	A4A3	A6A8	A4A3	A6A8	A149	A741	ADCC	AD59	不成不成！你們若
A710h:	AC4F	AF75	A4DF	B751	A662	A440	B05F	A141	是真心想在一起，

- When a string table has no pointers and the strings have no terminators, there are two possibilities:
  - It is directly accessed by code
    - You will need to hack and track each call from the executable
  - It is mapped by a separate files

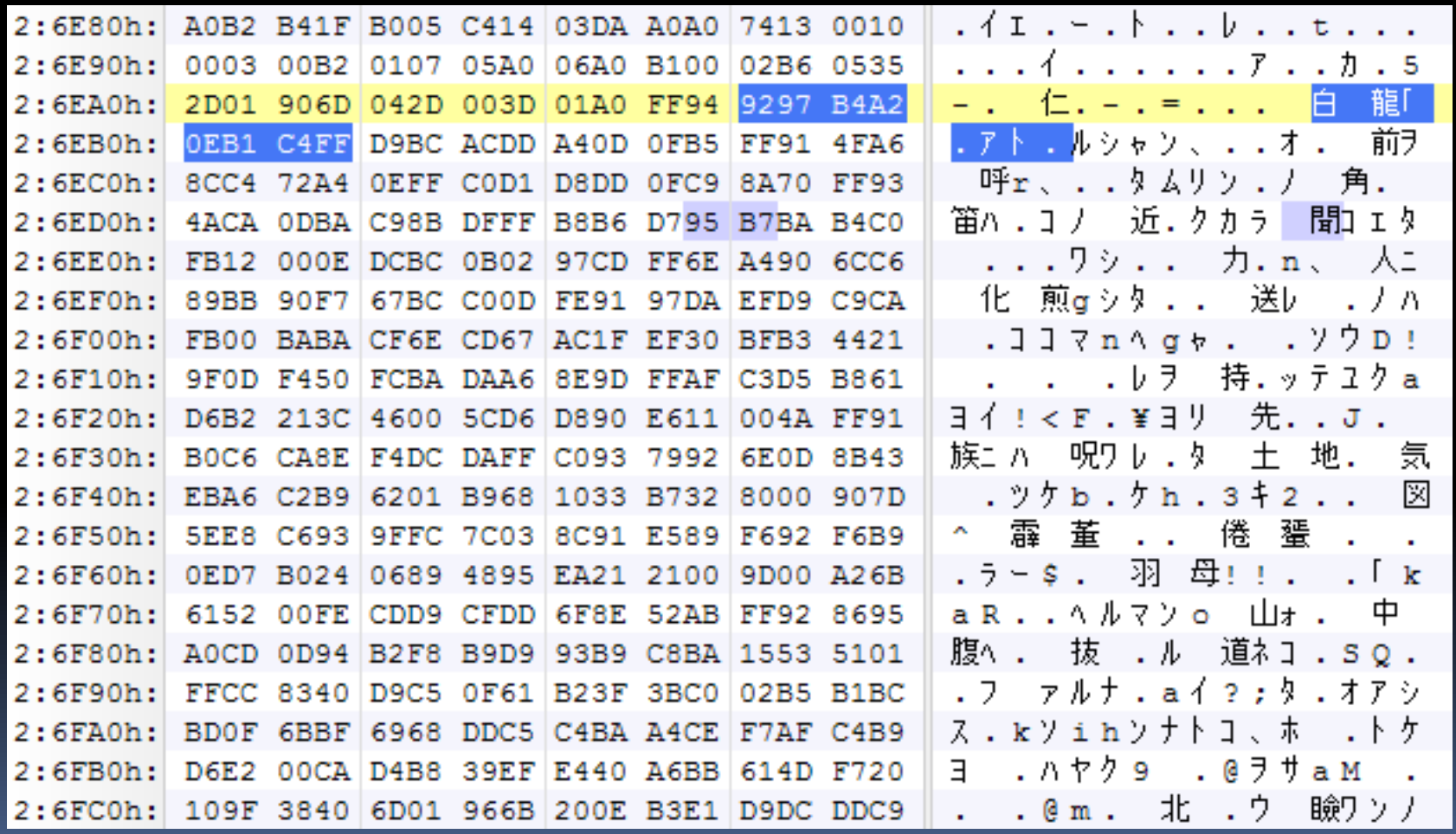
Compressed by dictionary



1:FC50h:	01DB	01FF	FF27	0238	063B	06FF	FF00	060C	.....'.8.;.....
1:FC60h:	0200	0900	9800	0090	4F68	208F	2104	89CD	........Oh .!...
1:FC70h:	2081	200C	2F3F	05D6	2081	209A	6172	2097	. ./?.. . .ar .
1:FC80h:	3F05	C920	0C59	3F04	900D	0020	6576	696C	?.. .Y?.... evil
1:FC90h:	200C	8420	7573	2E05	C510	200C	7A20	8020	.. us.... .z .
1:FCA0h:	0C01	0591	6E74	2063	7261	7A79	2E04	89EC	....nt crazy....
1:FCB0h:	2097	20AF	200C	D920	8420	812E	050D	0C20	. . ..
1:FCC0h:	9020	0C92	2E04	900D	2820	6D69	6E64	20A3	. ...(mind .
1:FCD0h:	2097	2E05	C120	0C50	208B	2088	200C	4C05	.... .P . . .L.
1:FCE0h:	8020	0CC5	732E	0489	5369	722B	2B3F	0553	. ..s...Sir++?.S
1:FCF0h:	6972	2121	0400	902B	2B2B	2B2B	2B04	0006	ir!!...++++++...
1:FD00h:	0C04	1146	0091	C020	0C01	200C	8420	7573	...F... .. .. us
1:FD10h:	2E05	C069	7220	6C65	6164	6572	2077	6F72	...ir leader wor
1:FD20h:	6520	6120	626C	6163	6B05	6361	7065	2B2B	e a black.cape++
1:FD30h:	2B04	89D0	2062	7261	7665	2104	912B	2B2B	+... brave!..+++
1:FD40h:	2B2B	2B04	0091	2B2B	2B2B	2B2B	0400	0411	+++...++++++....
1:FD50h:	1500	0CD0	200C	832E	20C3	1005	0C22	209D	...." .
1:FD60h:	2E04	000C	2B2B	2B2B	2B2B	0400	0410	3400	....++++++....4.
1:FD70h:	0CC1	200C	5020	9320	0C00	050C	3520	0B01	.. .P . ....5 ..
1:FD80h:	2086	200C	6B65	642E	05CB	20BE	2088	2087	. .ked... . . .
1:FD90h:	666F	726D	2080	050C	5220	AA20	AB21	0400	form ...R . .!..

- Replaces groups of bytes with an entry from a dictionary.
- Created by heuristic code which scans to find which groups of bytes could save the most space when replaced.
- Usually two or more bytes: byte 1 specific dictionary, byte 2 specifies an entry in dict.
  - Can be one byte. These mini-dictionaries may fill out unused bytes.

Compressed by LZ (DEFLATE)



2:6E80h:	A0B2	B41F	B005	C414	03DA	A0A0	7413	0010
2:6E90h:	0003	00B2	0107	05A0	06A0	B100	02B6	0535
2:6EA0h:	2D01	906D	042D	003D	01A0	FF94	9297	B4A2
2:6EB0h:	0EB1	C4FF	D9BC	ACDD	A40D	0FB5	FF91	4FA6
2:6EC0h:	8CC4	72A4	0EFF	C0D1	D8DD	0FC9	8A70	FF93
2:6ED0h:	4ACA	0DBA	C98B	DFFF	B8B6	D795	B7BA	B4C0
2:6EE0h:	FB12	000E	DCBC	0B02	97CD	FF6E	A490	6CC6
2:6EF0h:	89BB	90F7	67BC	C00D	FE91	97DA	EFD9	C9CA
2:6F00h:	FB00	BABA	CF6E	CD67	AC1F	EF30	BFB3	4421
2:6F10h:	9F0D	F450	FCBA	DAA6	8E9D	FFAF	C3D5	B861
2:6F20h:	D6B2	213C	4600	5CD6	D890	E611	004A	FF91
2:6F30h:	B0C6	CA8E	F4DC	DAFF	C093	7992	6E0D	8B43
2:6F40h:	EBA6	C2B9	6201	B968	1033	B732	8000	907D
2:6F50h:	5EE8	C693	9FFC	7C03	8C91	E589	F692	F6B9
2:6F60h:	0ED7	B024	0689	4895	EA21	2100	9D00	A26B
2:6F70h:	6152	00FE	CDD9	CFDD	6F8E	52AB	FF92	8695
2:6F80h:	A0CD	0D94	B2F8	B9D9	93B9	C8BA	1553	5101
2:6F90h:	FFCC	8340	D9C5	0F61	B23F	3BC0	02B5	B1BC
2:6FA0h:	BD0F	6BBF	6968	DDC5	C4BA	A4CE	F7AF	C4B9
2:6FB0h:	D6E2	00CA	D4B8	39EF	E440	A6BB	614D	F720
2:6FC0h:	109F	3840	6D01	966B	200E	B3E1	D9DC	DDC9

- Lempel-Ziv compression uses links back to earlier data to save space.
- Because of this, the start of an LZ block is usually less-compressed than the tail.
- LZ uses groups of 8 bytes. A block with no compression usually begins with an oxFF.
  - This is because oxFF == 11111111 in base2.
  - 1's indicated uncompressed bytes while 0's link back to bytes already in the window.

STRUCT data storage

```
3:6460h:  00 00 00 00 00 01 02 05 02 00 02 01 32 01 00 03 07 02 00 79 00 00 FF FF FF FF
3:6474h:  00 00 00 00 00 01 02 05 02 01 01 01 32 01 00 03 06 02 0B 79 00 00 FF FF FF FF
3:648Eh:  00 00 00 00 00 01 02 05 02 00 02 01 32 01 00 03 06 02 0B 7A 00 00 FF FF FF FF
3:64A8h:  00 00 00 00 00 01 02 05 02 01 02 01 32 01 00 03 06 02 0B 7B 00 00 FF FF FF FF
3:64C2h:  00 00 00 06 00 02 01 06 02 01 01 04 2D 00 00 03 06 02 0B B4 00 00 FF FF FF FF
3:64DCh:  00 00 00 0E 00 00 00 05 00 00 00 00 50 00 00 03 06 07 00 00 00 00 FF FF FF FF
3:64F6h:  00 00 00 03 00 02 01 08 02 02 00 03 2D 01 00 03 06 02 0B 81 00 00 FF FF FF FF
```

" "	Soldier	AT+1	DF+2	Move=5	Reward=20P
"Fighter"	Soldier	AT+1	DF+2	Move=5	Reward=20P
"Fighter"	Soldier	AT+1	DF+2	Move=5	Reward=20P
"Fighter"	Soldier	AT+1	DF+2	Move=5	Reward=20P
"Gladiator"	Bandit	AT+2	DF+1	Move=5	Reward=20P
"Vampire"	Undead	AT+0	DF+0	Move=5	Reward=70P
"Knight"	Cavalry	AT+2	DF+1	Move=5	Reward=20P

- STRUCT data consists of fields in fixed lengths.
  - If you're written on in C, it looks exactly like that.
- Usually you can use known values to locate a STRUCT. Consider searching for bytes you know, then seeing if they land in a pattern.
  - If you play an RPG and it shows STR, DEF, AGL and MDF in that order, it's reasonable to assume the programmers stored them in that order.

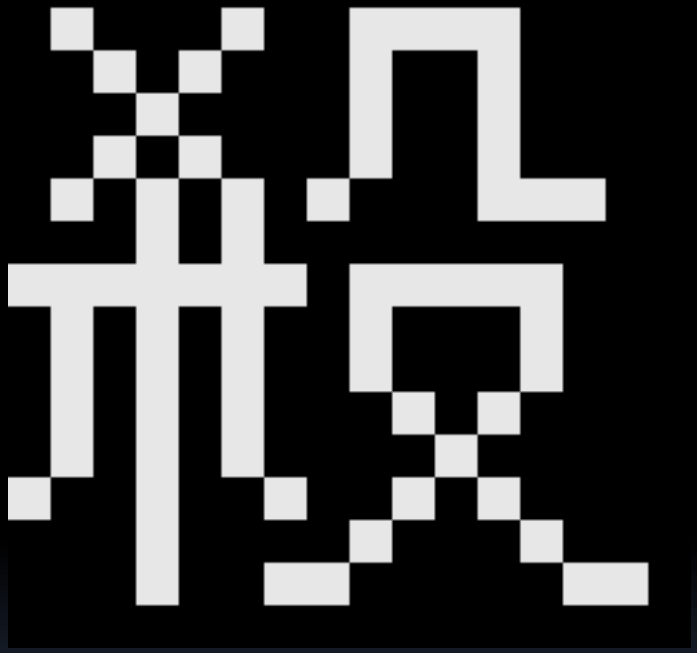
A few tips for STRUCTs

- Consider the size of the value. Can it go above 255? Above 65535?
  - Programmers usually won't use an int32 to store a value that never exceeds 16 bits.
- Check what the executable links to. It may reference an API whose documentation will tell you what the STRUCT contains.
  - Don't forget about endianness! Value order will be backwards in compiled binary.

Graphical data

- Things we need to store:
  - Coordinates
  - Color palette
  - Pixel colors
  - Tile map (optional)

How binary becomes graphic

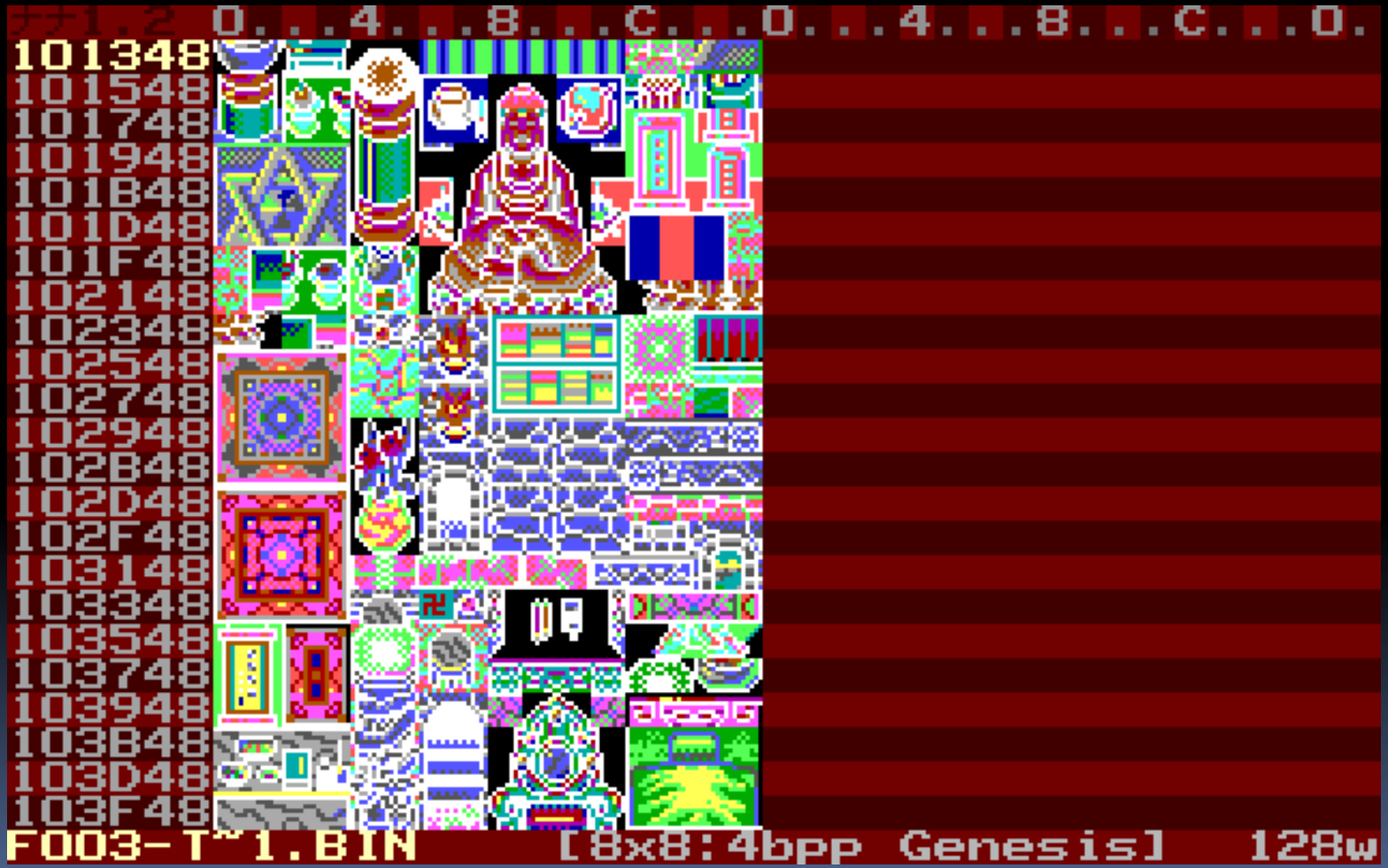


```
0000000000000000 = 0x0000
0100010011110000 = 0x44F0
0010100010010000 = 0x2890
0001000010010000 = 0x1090
0010100010010000 = 0x2890
0101010100011100 = 0x551C
0001010000000000 = 0x1400
1111111011111000 = 0xFEF8
0101010010001000 = 0x5488
0101010001010000 = 0x5450
0101010000100000 = 0x5420
1001001001010000 = 0x9250
0001000010001000 = 0x1088
0001001100000110 = 0x1306
0000000000000000 = 0x0000
```

And then four-color planar …

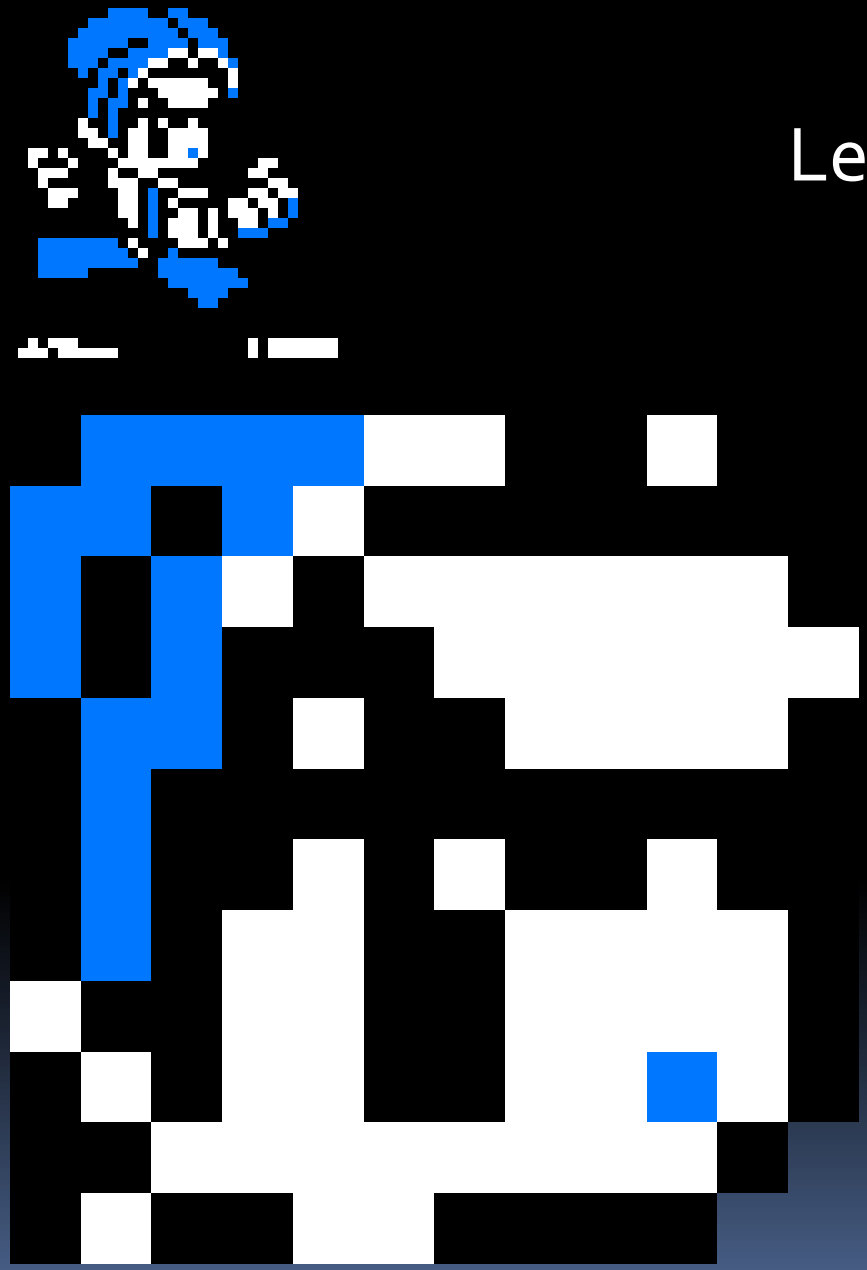
Least Significant (a)	Most Signficant (b)	(ba)
0000001111100000	0000000000000000	0000001111100000
0000111111000000	0000000001000000	0000111111200000
0001111110000000	0000000011000000	0001111112200000
0001111111111100	0000000000000000	0001111111111100
0001110010000000	0001111111110000	0003332232220000
0010010011000000	0011111111111100	0032232233222200
0010011000000000	0011111111111110	0032233222222220
0110011000100000	0111111111111110	0332233222322220
0110000011111100	0111111111111100	0332222233333300
0111000001111100	0111111111111100	0333222223333300
0001100000000000	0001111111111000	0003322222222000
0000011100000000	0000011111000000	0000033322000000
0000111111100000	0000000011000010	0000111133100020
0001111111110000	0001111001100111	0003333113310222
0011111111111000	0011111100101111	0033333311313222
0111111111111000	0111111100110111	0333333311331222

And do it again for 16 …



Linear graphics

- Rather than layer two monochrome images, these group bits linearly to map colors.
- This means that one row of 8 4-color pixels would be formed by 16 bytes in a row.
  - A colored row of 01233210 would be represented by the binary 00 01 10 11 11 10 01 00, or 0x1BE4



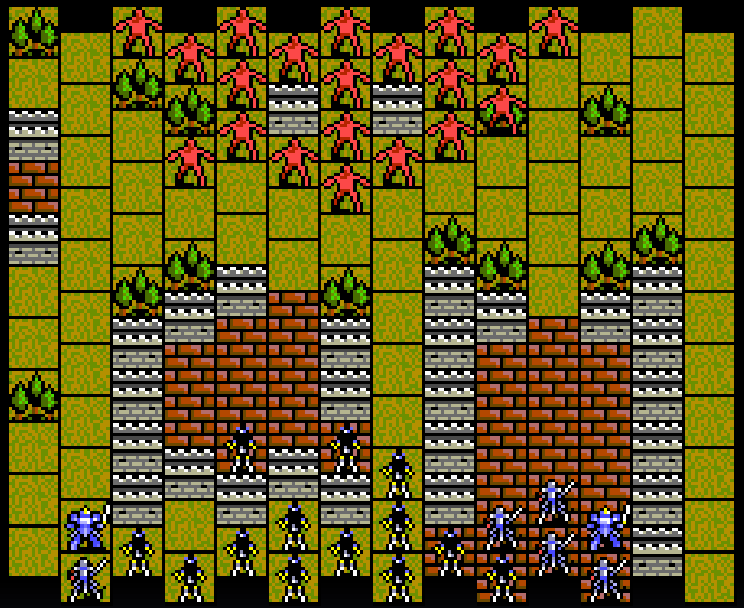
Let's zoom in on the face to see how it works

```
11101010100101111011111
10101110011111111111111
10111001110101010101011
10111011111101010101011
11101011011110101010111
11101111111111111111111
11101111011101111011111
11101101011110101010111
01111101011110101010111
11011101011110101100111
11110101010101010111000
11011110101111111110000
```

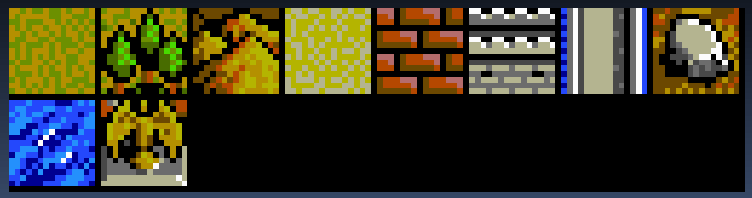
Map formats

- Many graphics are assembled from small slices, or “tiles”
- These tiles, in combination with a map, allow for large graphics made of recycled sections.
- Used extensively in video and computer games to draw area maps.

How maps are assembled



```
01 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 01 01 00 05 00 05 00 01 00 01 00 00
05 00 00 00 00 00 00 00 00 00 00 00 00 00
04 00 00 00 00 00 00 00 00 00 00 00 00 00
05 00 00 01 00 00 00 00 01 01 00 01 01 00
00 00 01 05 05 04 01 00 05 05 00 05 05 00
00 00 05 04 04 04 05 00 05 04 04 04 05 00
01 00 05 04 04 04 05 00 05 04 04 04 05 00
00 00 05 05 04 05 04 00 05 04 04 04 05 00
00 00 05 00 05 00 05 00 05 04 04 04 05 00
00 00 00 00 00 00 00 00 04 04 04 04 05 00
```



00 01 02 03 04 05 06 07

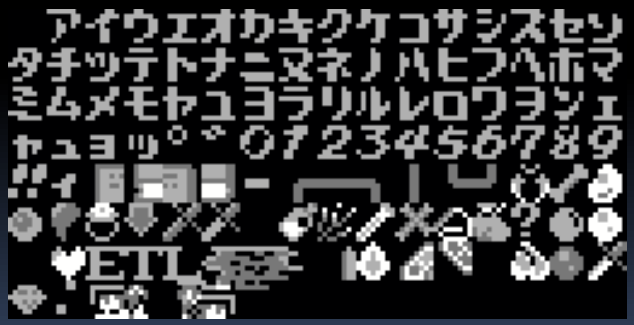
08 09

- Don’t forget! **Programmers are lazy people just like you!**
- If you find relevant data, you can almost always assume the programmers used it in the order they stored it.

Run-length encoding

```
0000h:  1800 87FC 0626 3430 2040 0900 840C 1E38
0010h:  D003 1809 0087 30FC C6C6 060C 300A 0081
0020h:  7E03 1882 30FE 0900 870C FE1C 1C3C 6CCC
0030h:  0900 8260 FC03 6682 CC18 0900 8730 30FE
0040h:  18FE 0C0C 0900 873C 2666 C40C 1860 0900
0050h:  8760 7E6C CC0C 0C18 0A00 817C 0406 817E
0060h:  0900 876C FE6C 6C0C 1830 0900 8770 0272
0070h:  040C 7830 0900 87FC 0E0C 183C 6EC6 0900
```

```
0000h:  0000 0000 0000 0000 0000 0000 0000 0000
0010h:  0000 0000 0000 0000 FC06 2634 3020 4000
0020h:  0000 0000 0000 0000 0C1E 38D0 1818 1800
0030h:  0000 0000 0000 0000 30FC C6C6 060C 3000
0040h:  0000 0000 0000 0000 007E 1818 1830 FE00
0050h:  0000 0000 0000 0000 0CFE 1C1C 3C6C CC00
0060h:  0000 0000 0000 0000 60FC 6666 66CC 1800
0070h:  0000 0000 0000 0000 3030 FE18 FE0C 0C00
```



Network packets

- Think about what data is being transferred and what port is being used.
  - Software using a Port 80 connection for its connection may be using a standard HTTP interface.
- The more data you can log, the better. Unroll packets to single lines and compare which bytes change to isolate events.

Case Study: Shiren

- Chunsoft decided to stop operating service for *Shiren the Wanderer*.
  - This made it impossible to play because the game constantly validated your account.
- Packets to the server collected using Wireshark (then Ethereal).
- Protocol used HTTP connection. 4 hours to reverse and re-engineer in Apache+PHP.
  - 2 days to reverse engineer game data payload.

The protocols

- Wireshark reported HTTP GET packets exchanged with shiren.jp server on Port 80.
- Data hit 4 programs: shirenDownload, authori, ranking, rescue.
  - Download hit by installer. Used to assign mirror.
  - authori hit at 5 points. Checked if account valid. Locked all but first mission if failed.
  - ranking hit at mission end or death, sent status
  - rescue hit when requesting password

authori methods

- From the packets and executable, we can see the server has three methods:
  - chg (change password)
    - Serial, oldPass, newPass
  - reg (register account)
    - yourName, yourPass, Serial
  - aut (authentication)
    - yourName, counter, Serial, check1, check2

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT=o>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
    charset="EUC-JP>
<TITLE>authori.php</TITLE>
</HEAD>
<!--
Shiren authentication system 89a0uKT2
-->
<!--
result=success
-->
<BODY>
OK<BR>
</BODY>
</HTML>
```

```
0030h:   4470 9d60 0000 504f 5354 202f 6367 692d   Dp.`..POST /cgi-
0040h:   6269 6e2f 7261 6e6b 696e 672e 6367 6920   bin/ranking.cgi
0050h:   4854 5450 2f31 2e31 0d0a 5265 6665 7265   HTTP/1.1..Refere
0060h:   723a 2077 7777 2e73 6869 7265 6e2e 6a70   r: www.shiren.jp
0070h:   0d0a 5573 6572 2d41 6765 6e74 3a20 4765   ..User-Agent: Ge
0080h:   7448 746d 6c0d 0a48 6f73 743a 2077 7777   tHtml..Host: www
0090h:   2e73 6869 7265 6e2e 6a70 0d0a 436f 6e74   .shiren.jp..Cont
00a0h:   656e 742d 4c65 6e67 7468 3a20 3231 370d   ent-Length: 217.
00b0h:   0a43 6163 6865 2d43 6f6e 7472 6f6c 3a20   .Cache-Control:
00c0h:   6e6f 2d63 6163 6865 0d0a 0d0a 6d65 6d6f   no-cache....memo
00d0h:   3d35 3336 3836 3937 3236 3536 4532 3036   =53686972656E206
00e0h:   4436 3537 3432 3036 3836 3937 3332 3036   D657420686973206
00f0h:   3536 4536 3432 3036 4636 4532 3034 3636   56E64206F6E20466
0100h:   4336 4636 4637 3232 3033 3232 3033 4132   C6F6F722032203A2
0110h:   3826 5365 7269 616c 3d30 3030 3030 3030   8&Serial=0000000
0120h:   3030 3030 3030 3030 3030 3030 3030 3030   0000000000000000
0130h:   3026 7370 6177 3d42 4242 4349 4849 4247   0&spaw=BBBCIHIBG
0140h:   4842 5842 4242 4442 4b42 4442 4243 4642   HBXBBBDBKBDBBCFB
0150h:   4243 4445 4242 4258 4258 4244 4344 4642   BCDEBBBXBXBDCDFB
0160h:   4846 424a 424a 4242 4243 5442 4242 4242   HFBJBJBBBCTBBBBB
0170h:   4242 4242 4242 4242 424a 4547 484a 454a   BBBBBBBBBJEGHJEJ
0180h:   554a 454b 4526 746f 6b65 3d41 4141 4141   UJEKE&toke=AAAAA
0190h:   4341 4141 4141 4844 5226 6163 7469 6f6e   CAAAAAHDR&action
```

Looking at the spaw var

- Letters present:
  - BCDEFGHIJKSTUVWX
  - 0123456789ABCDEF
- If we think in hex, 0=0x30, 1=0x31, B=0x42, C=0x43 …
  - All values line up with hexadecimal if you subtract 0x12

28 bits: Score

8 bits: Year

4 bits: Month

8 bits: Day

16 bits: Floor number

8 bits: Area name

8 bits: Level

8 bits: Current HP

8 bits: Max HP

20 bits: Gold

16 bits: Attempt #

24 bits: Mission number

8 bits: Current food

8 bits: Max food

8 bits: Current STR

8 bits: Max STR

20 bits: EXP

8 bits: Weapon name

8 bits (s): Weapon adjustment

8 bits: Weapon state

8 bits: Shield name

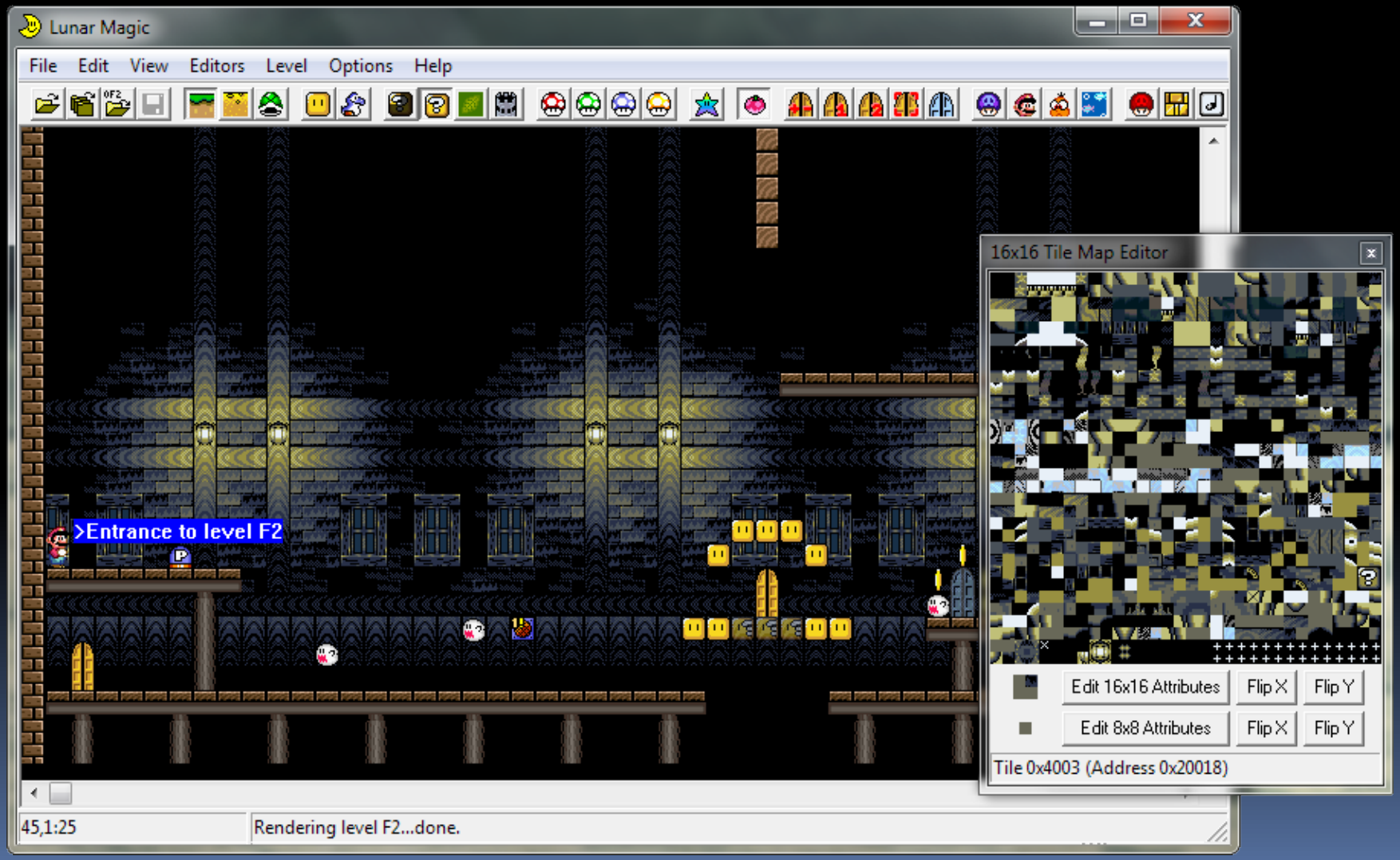
8 bits (s): Shield adjustment

8 bits: Shield state

4 bits: Unknown

8 char: Player name in hex

A fantastic example of RE



Useful resources

- MadEdit – the only Linux hex editor that can display Chinese
  - http://madedit.sourceforge.net/
- NANA – closed-source raw graphic viewer that works well in DOSBox
  - http://neillcorlett.com/nana/
- Lunar Magic – an incredible RE project that edits everything in Super Mario World
  - http://fusoya.eludevisibility.org/lm/