

PLAYER 1



HIGHSCORE 9000

PLAYER 2



LEVELING UP GHIDRA: LEARN GHIDRA PLUGINS WITH A GAME BOY GAME

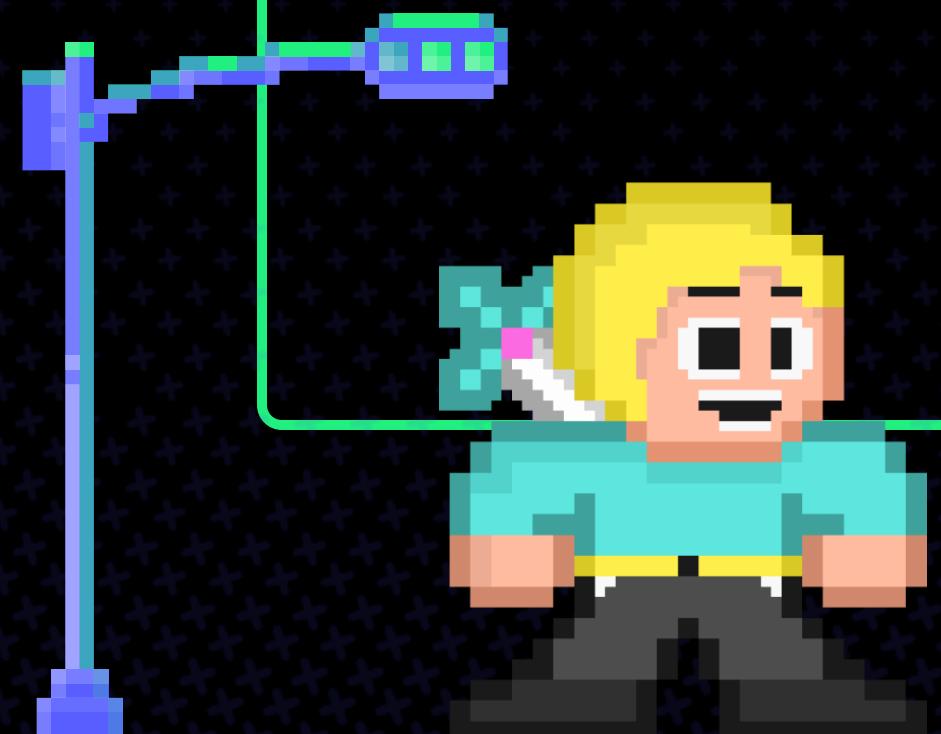
SANS DFIR SUMMIT - Aug 23rd, 2024

JACOB LATONIS

MENU

START

SIGN IN



MENU

01

07

12



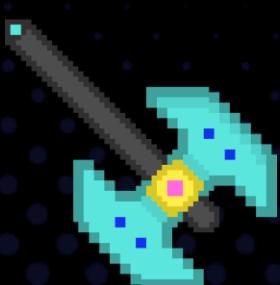
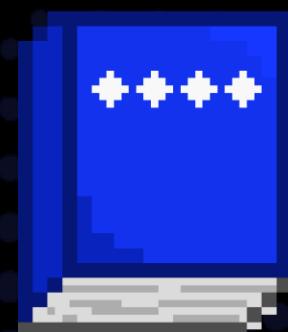
WHOAMI

Name: Jacob Latonis

What I do: Staff Software Engineer, Threat Research @ proofpoint

Residing in: Boulder, CO

What i do for fun: running, reading, hiking, open-source development



AGENDA

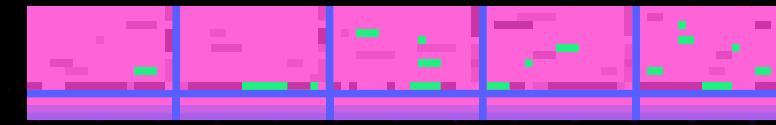
- PLUGINS?
- GHIDRA ECOSYSTEM
- POSSIBILITIES
- WRITING THE PLUGIN
- A LOT MORE!



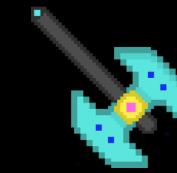
GHIDRA



The possibilities!



Implementation



A lot more!

MENU

01

07

12



BEFORE WE START...

We will discuss a whole lot here today, not every topic will have the time and/or focus to be fully completed and discussed.

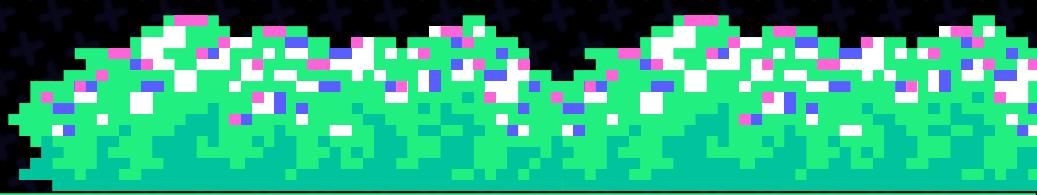
curious on what a full plugin implementation and design looks like? I have published a fully working plugin on [github](#) under the APACHE 2.0 license.



SIGN IN



INTRODUCTION





GHIDRA!

- Open-Source Reverse engineering Toolkit
 - Developed by the NSA because IDA costs a billion dollars
 - Highly extensible and customizable
 - FREE!

The screenshot shows the CodeBrowser interface with several windows open. The main window displays assembly code for the function `_ungetc_nolock` from the file `oo.exe`. The assembly code is as follows:

```
00406118 e8 98 02 CALL    __unlock_file
0040611d 00 00
0040611e 59      POP     ECX
0040611f c3      RET
```

Below the assembly code, the decompiled C code is shown:

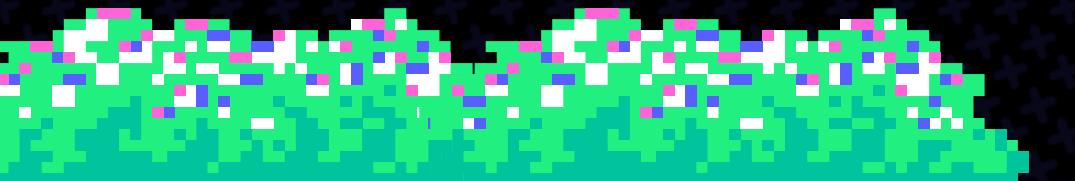
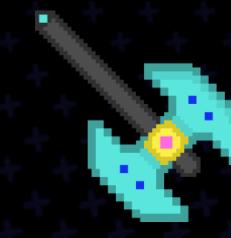
```
1  /* Library Function - Single Match
2   * Name: __ungetc_nolock
3   * Library: Visual Studio 2010 Release */
4
5   int __cdecl __ungetc_nolock(int _Ch
6
7   {
8       char *pCVar1;
9       uint uVar2;
10      undefined *puVar3;
11      int *piVar4;
12
13      if (((byte *)&_File->_flag & 0x40) == 0) {
14          uVar2 = _fileno(_File);
15          if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
16              puVar3 = &DAT_00417750;
17          }
18          else {
19              puVar3 = (undefined *)((uVar2 & 0x1f) * 0x40);
20          }
21          if ((puVar3[0x24] & 0x7f) == 0) {
22              if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
23                  puVar3 = &DAT_00417750;
24      }
25  }
```

The right side of the interface shows the decompiled C code for `_ungetc_nolock`, which is identical to the assembly code above. The bottom right corner of the interface has a status bar with the text "Defined Str".

SIGN IN

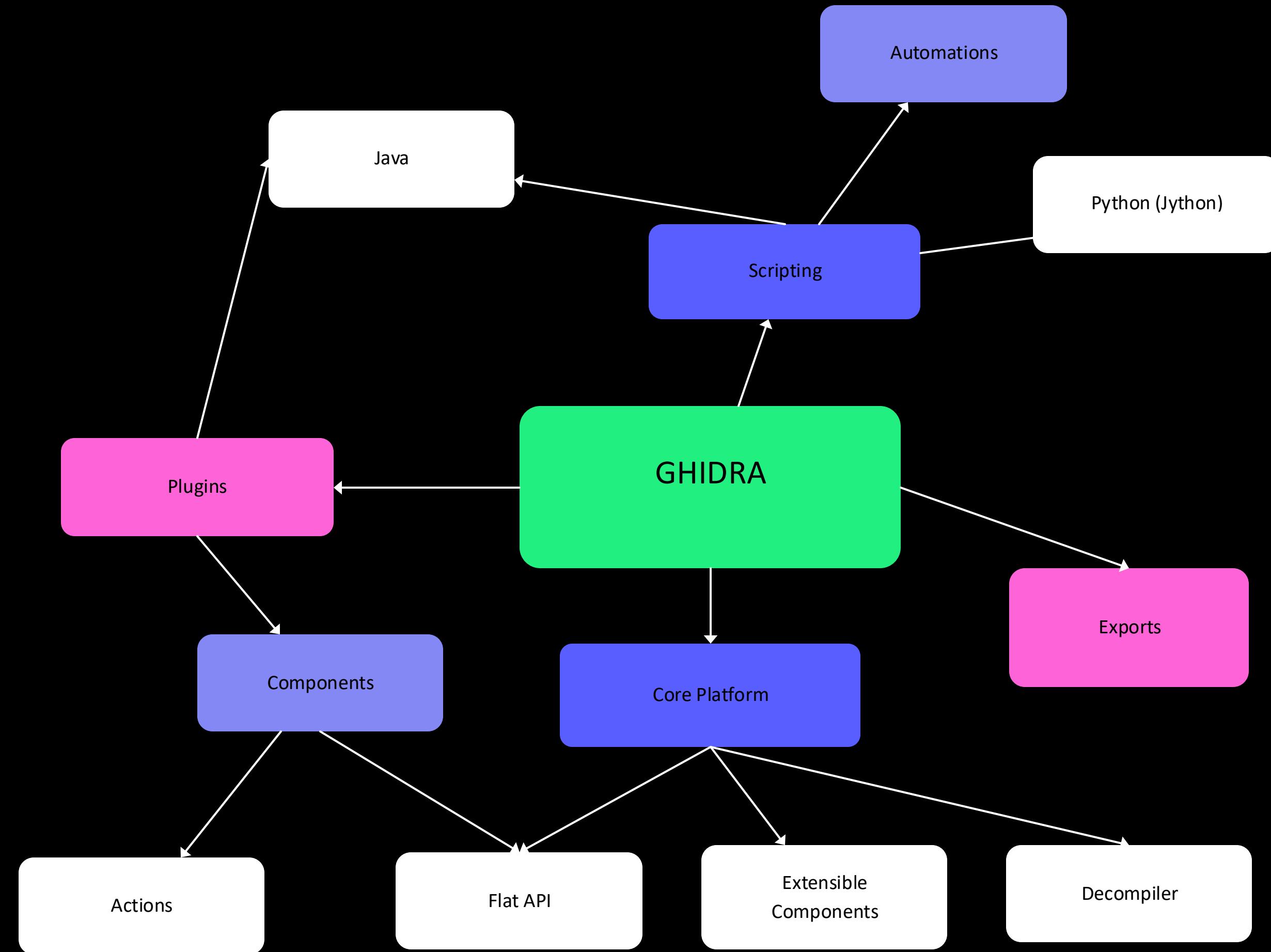
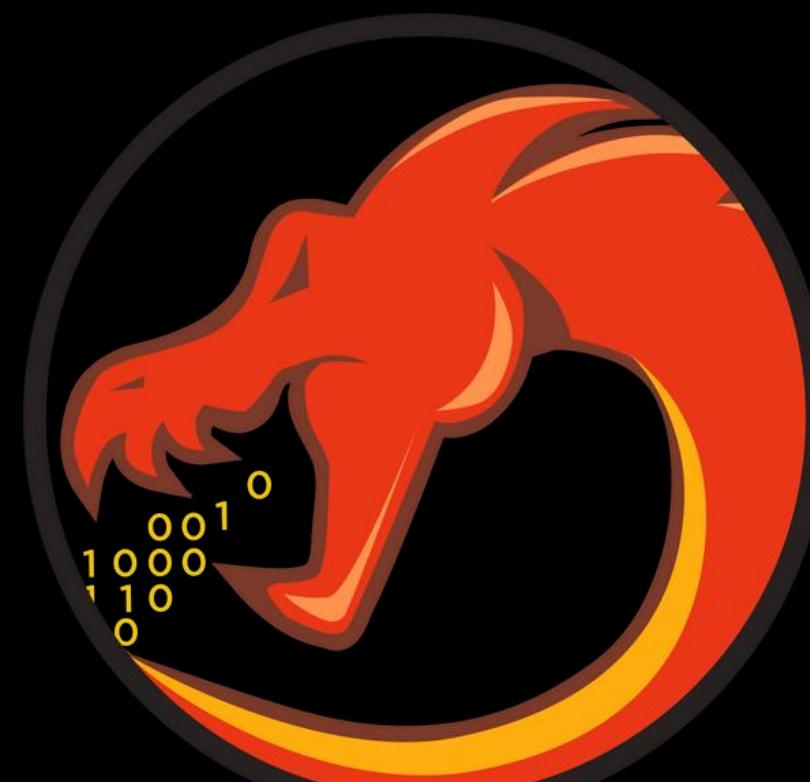


GETTING STARTED



Ghidra Architecture

There's quite a few moving parts in Ghidra itself, even before we start adding stuff.



SIGN IN



SCRIPTING AND PLUGINS





01



07



12



SCRIPTING IN GHIDRA



Written in Python (Jython) or Java

Access to the FLATAPI and more

Can be ran from CLI or Action



01



07



12



PLUGINS IN GHIDRA



Written in JAVA



Allows for more in-Depth Use than Scripting (keep state, GUI, etc)

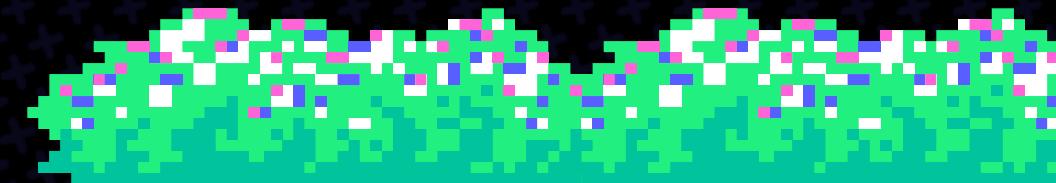
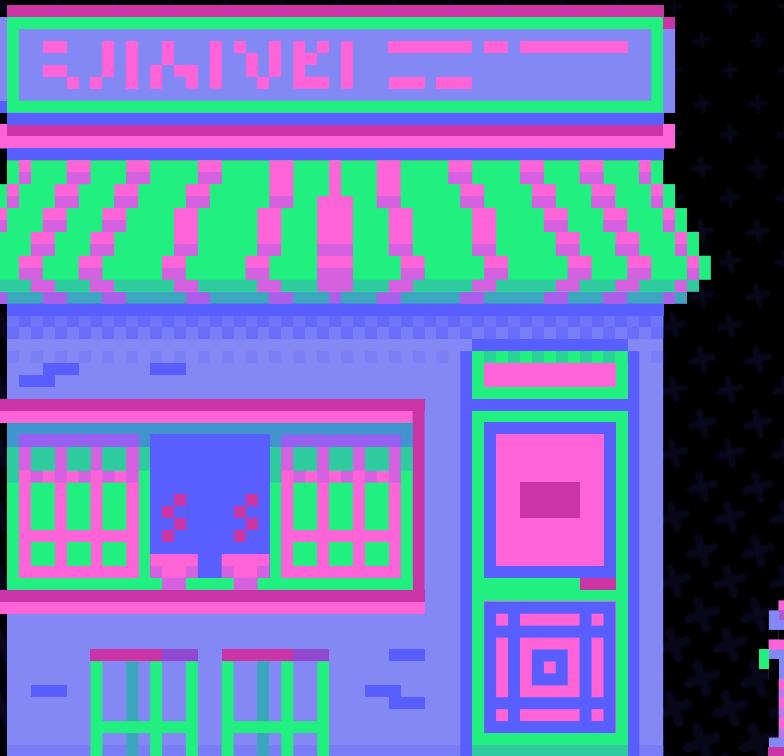


Can be built and distributed, instead of a script

SIGN IN



GAME BOY?



GAME BOY!

+ Developed by Nintendo

+ Originally released in 1989

+ lots, and lots, of games to play, games still being developed in indie scene

+ still used for leisure and competitive play world wide



01

07

12



+ small, handheld video games (lots)

+ 8 bit CPU similar to 8080 SHARP CPU

+ 1.94304 MHz CPU | 160 x 144 Display

GAME BOY RESOURCES



[GEKKIO/GB-RESEARCH](#)

[GAME BOY PANDOCS](#)

[GEKKIO/GHIDRABOY](#)

[GEKKIO/GB-CTR](#)
(COMPLETE TECHNICAL REFERENCE)



01



07



12



GAME BOY CARTRIDGE

Each cartridge has a header

Header contains information about the game

We want that information; how do we get it?

- 0100-0103 — ENTRY POINT
- 0104-0133 — NINTENDO LOGO
- 0134-0143 — TITLE
- 013F-0142 — MANUFACTURER CODE
- 0143 — CGB FLAG
- 0144-0145 — NEW LICENSEE CODE
- 0146 — SGB FLAG
- 0147 — CARTRIDGE TYPE
- 0148 — ROM SIZE
- 0149 — RAM SIZE
- 014A — DESTINATION CODE
- 014B — OLD LICENSEE CODE
- 014C — MASK ROM VERSION NUMBER
- 014D — HEADER CHECKSUM
- 014E-014F — GLOBAL CHECKSUM



01



07



12



HEADER BROKEN DOWN!

0100-0103 — ENTRY POINT

0104-0133 — NINTENDO LOGO

0134-0143 — TITLE

013F-0142 — MANUFACTURER CODE



01



07



12



HEADER BROKEN DOWN!



0143 — CGB FLAG

0144–0145 — NEW LICENSEE CODE

0146 — SGB FLAG

0147 — CARTRIDGE TYPE



01



07



12



HEADER BROKEN DOWN!

0148 — ROM SIZE

0149 — RAM SIZE

014A — DESTINATION CODE

014B — OLD LICENSEE CODE



01



07



12



HEADER BROKEN DOWN!



014C – MASK ROM VERSION NUMBER

014D – HEADER CHECKSUM

014E-014F – GLOBAL CHECKSUM

MENU

01

07

12



WHO'S THAT POKEMON?

```
00000100: 00c3 ab01 ceed 6666 cc0d 000b 0373 0083
00000110: 000c 000d 0008 111f 8889 000e dccc 6ee6
00000120: dddd d999 bbbb 6763 6e0e eccc dddc 999f
00000130: bbb9 333e 504f 4b45 4d4f 4e20 5945 4c4c
00000140: 4f57 0080 3031 031b 0503 0133 0097 047c
00000150: f0b8 f578 cd7e 3e2a 4f2a 472a 573e 033d
```

?

Pokémon

MENU

01

07

12



WRITE A PARSER?

 Cartridge header has the info we want; time to parse byte by byte.

EACH CARTRIDGE CONTAINS A HEADER,
LOCATED AT THE ADDRESS RANGE
\$0100—\$014F.

THE CARTRIDGE HEADER PROVIDES
INFORMATION ABOUT THE GAME ITSELF
AND THE HARDWARE IT EXPECTS TO RUN
ON.



» I/O Ranges

The Game Boy uses the following I/O ranges:

Start	End	First appeared	Purpose
\$FF00		DMG	Joypad input
\$FF01	\$FF02	DMG	Serial transfer
\$FF04	\$FF07	DMG	Timer and divider
\$FF10	\$FF26	DMG	Audio
\$FF30	\$FF3F	DMG	Wave pattern
\$FF40	\$FF4B	DMG	LCD Control, Status, Position, Scrolling, and Palettes
\$FF4F		CGB	VRAM Bank Select
\$FF50		DMG	Set to non-zero to disable boot ROM
\$FF51	\$FF55	CGB	VRAM DMA
\$FF68	\$FF6B	CGB	BG / OBJ Palettes
\$FF70		CGB	WRAM Bank Select

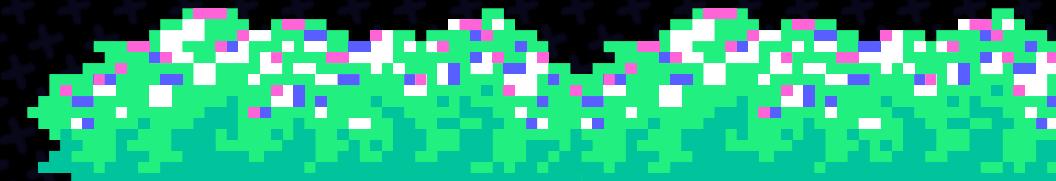
Custom CPU/Instruction Set

Lots of I/O and Addressing to Get FAMILIAR WITH

SIGN IN



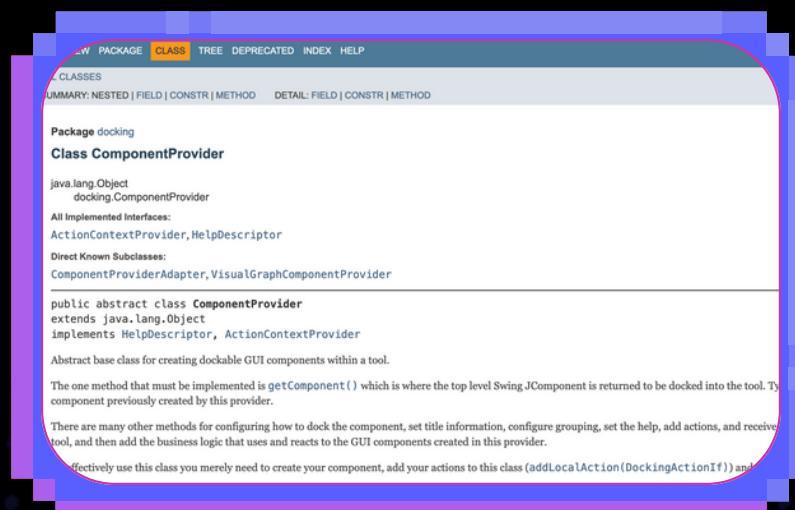
PLUGIN DEEP DIVE



MENU



WRITING A PLUGIN: WHAT DO WE NEED?



Whole lot of Java

Unlike scripting which can be written in Java or Python (Jython), plugins for Ghidra are written in Java

A bit of Gradle

Gradle is the build system that is used for building Ghidra plugins ready for distribution.

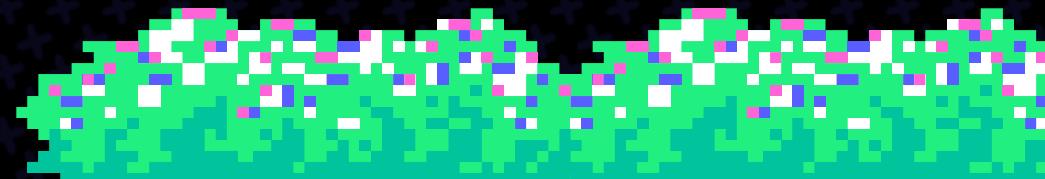
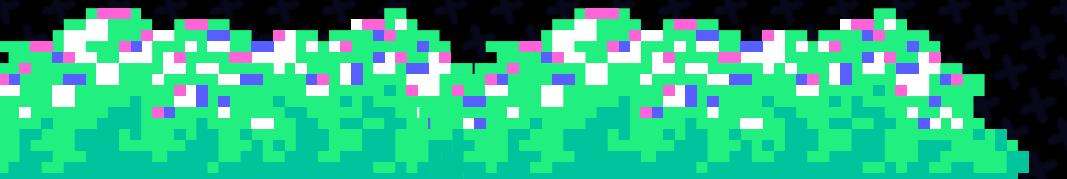
Many Patience to read many docs

The docs for/in Ghidra can get pretty elaborate.

SIGN IN



WRITING THE PLUGIN



EXAMPLES!

- Ghidra provides quite a few examples for us
- Plugin examples
- Scripting examples
- located in <ghidra_dir>/Extensions/Ghidra/

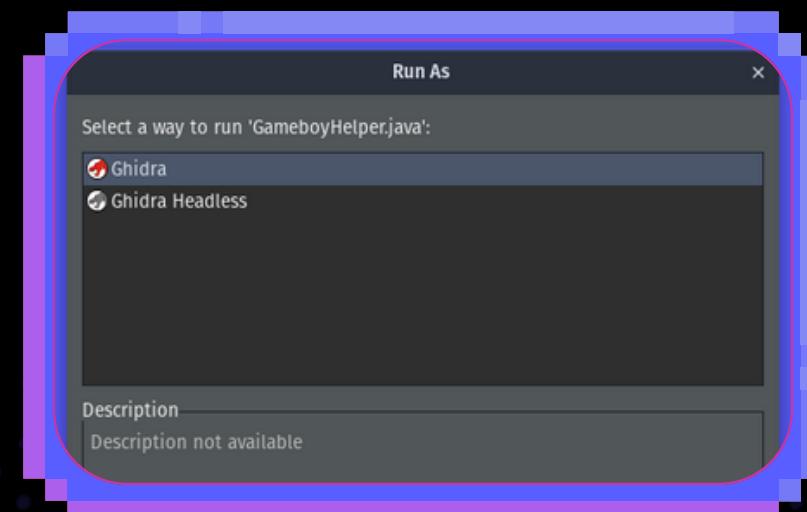
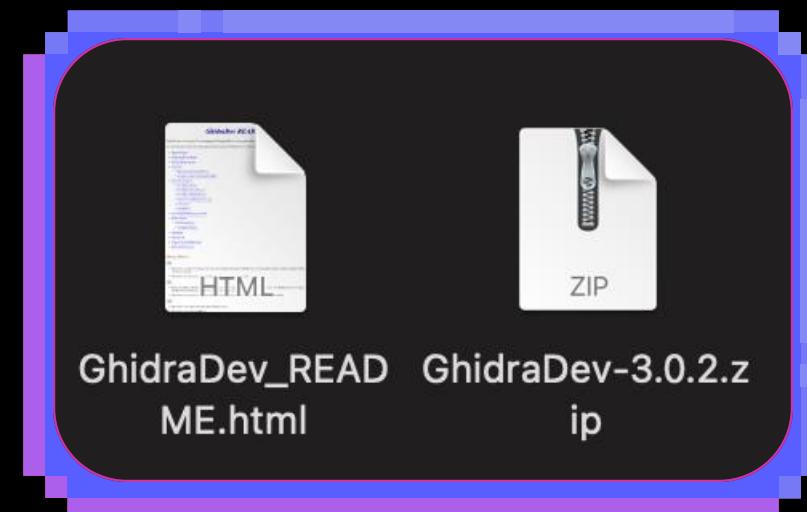
The screenshot shows the Ghidra debugger interface with multiple windows. The main window displays assembly code for the `_ungetc_nolock` function. The assembly code includes instructions like `CALL _unlock_file`, `POP ECX`, and `RET`. Below the assembly window is a decompiled C code window, which shows the following:

```
1 // Library Function - Single Match
2 * Name: __ungetc_nolock
3 * Library: Visual Studio 2010 Release
4
5 int __cdecl __ungetc_nolock(int _Ch,FILE *_File)
6 {
7     char *pcVar1;
8     uint uVar2;
9     undefined *puVar3;
10    int *piVar4;
11
12    if (((byte *)&_File->_flag & 0x40) == 0)
13    {
14        uVar2 = _fileno(_File);
15        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
16            puVar3 = &DAT_00417750;
17        else
18            puVar3 = (undefined *)((uVar2 & 0xf) * 0x40);
19        if ((puVar3[0x24] & 0x7f) == 0)
20            if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
21                puVar3 = &DAT_00417750;
22    }
23    else
24        puVar3 = (undefined *)((uVar2 & 0xf) * 0x40);
25
26    if ((puVar3[0x24] & 0x7f) == 0)
27        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
28            puVar3 = &DAT_00417750;
29
30    if ((puVar3[0x24] & 0x7f) == 0)
31        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
32            puVar3 = &DAT_00417750;
33
34    if ((puVar3[0x24] & 0x7f) == 0)
35        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
36            puVar3 = &DAT_00417750;
37
38    if ((puVar3[0x24] & 0x7f) == 0)
39        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
40            puVar3 = &DAT_00417750;
41
42    if ((puVar3[0x24] & 0x7f) == 0)
43        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
44            puVar3 = &DAT_00417750;
45
46    if ((puVar3[0x24] & 0x7f) == 0)
47        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
48            puVar3 = &DAT_00417750;
49
50    if ((puVar3[0x24] & 0x7f) == 0)
51        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
52            puVar3 = &DAT_00417750;
53
54    if ((puVar3[0x24] & 0x7f) == 0)
55        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
56            puVar3 = &DAT_00417750;
57
58    if ((puVar3[0x24] & 0x7f) == 0)
59        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
60            puVar3 = &DAT_00417750;
61
62    if ((puVar3[0x24] & 0x7f) == 0)
63        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
64            puVar3 = &DAT_00417750;
65
66    if ((puVar3[0x24] & 0x7f) == 0)
67        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
68            puVar3 = &DAT_00417750;
69
70    if ((puVar3[0x24] & 0x7f) == 0)
71        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
72            puVar3 = &DAT_00417750;
73
74    if ((puVar3[0x24] & 0x7f) == 0)
75        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
76            puVar3 = &DAT_00417750;
77
78    if ((puVar3[0x24] & 0x7f) == 0)
79        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
80            puVar3 = &DAT_00417750;
81
82    if ((puVar3[0x24] & 0x7f) == 0)
83        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
84            puVar3 = &DAT_00417750;
85
86    if ((puVar3[0x24] & 0x7f) == 0)
87        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
88            puVar3 = &DAT_00417750;
89
90    if ((puVar3[0x24] & 0x7f) == 0)
91        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
92            puVar3 = &DAT_00417750;
93
94    if ((puVar3[0x24] & 0x7f) == 0)
95        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
96            puVar3 = &DAT_00417750;
97
98    if ((puVar3[0x24] & 0x7f) == 0)
99        if ((uVar2 == 0xffffffff) || (uVar2 == 0xffff))
100       puVar3 = &DAT_00417750;
```

MENU



WRITING THE PLUGIN



◆ TEAM ECLIPSE

If you want the nice helper utilities for Ghidra development, welcome to Eclipse. It's only a lot more painful than IntelliJ IDEA.

◆ USE GHIDRADEV

The GhidraDev extension for eclipse generates skeleton files, allowing us to start quickly!

◆ Debugging is a go

The GhidraDev extension for Eclipse allows us to run and debug the plugin between changes.

MENU

01

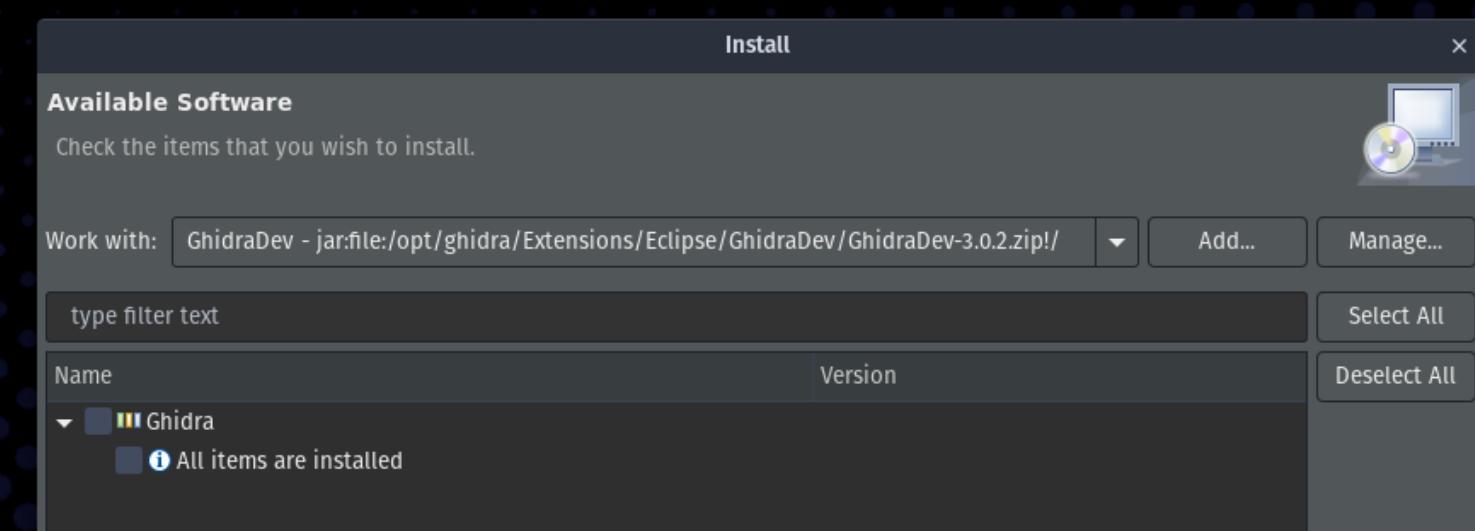
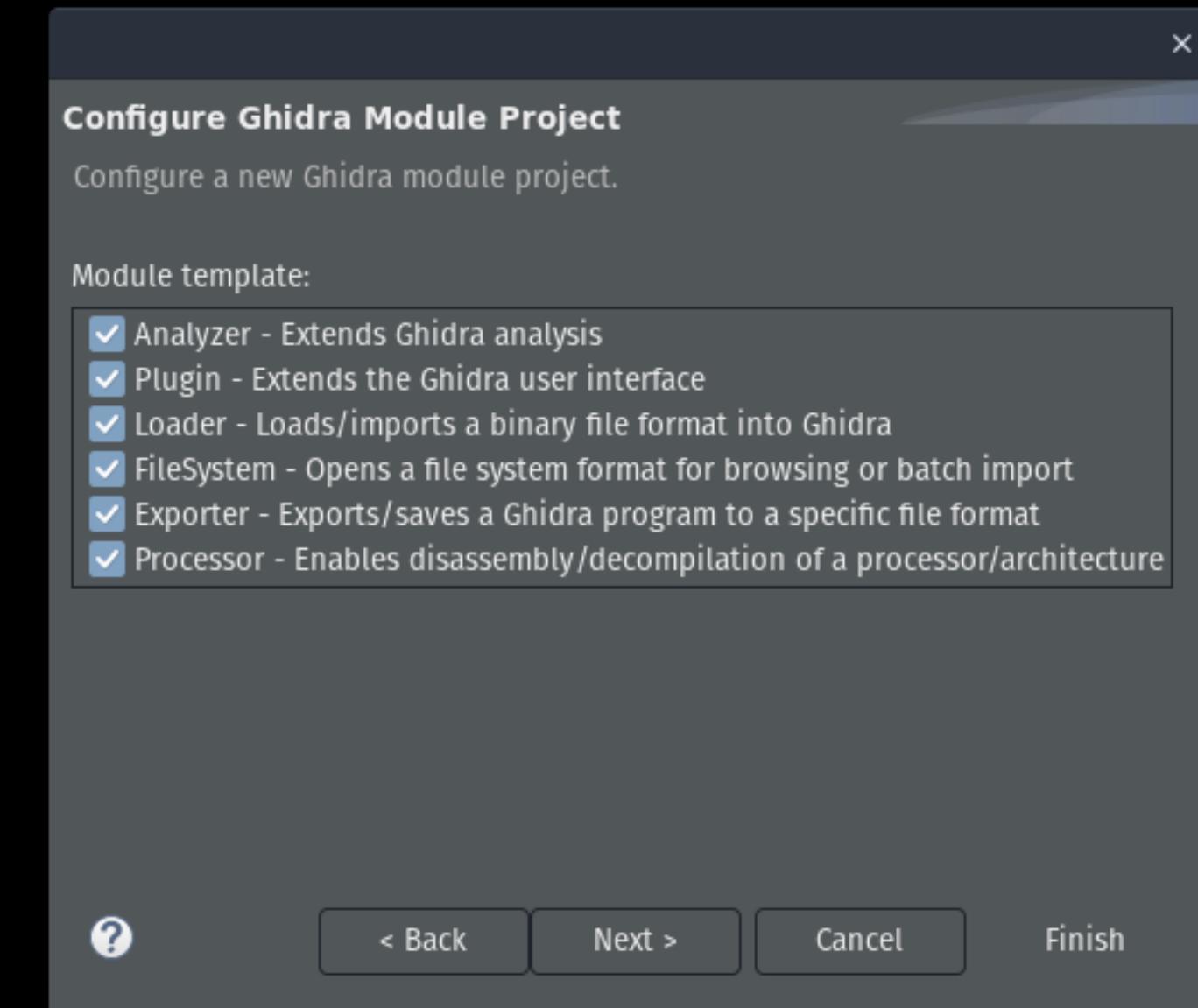
07

12



LEVERAGING GHIDRADEV

located in <ghidra_dir>/Extensions/Eclipse/GhidraDev/



MENU

01

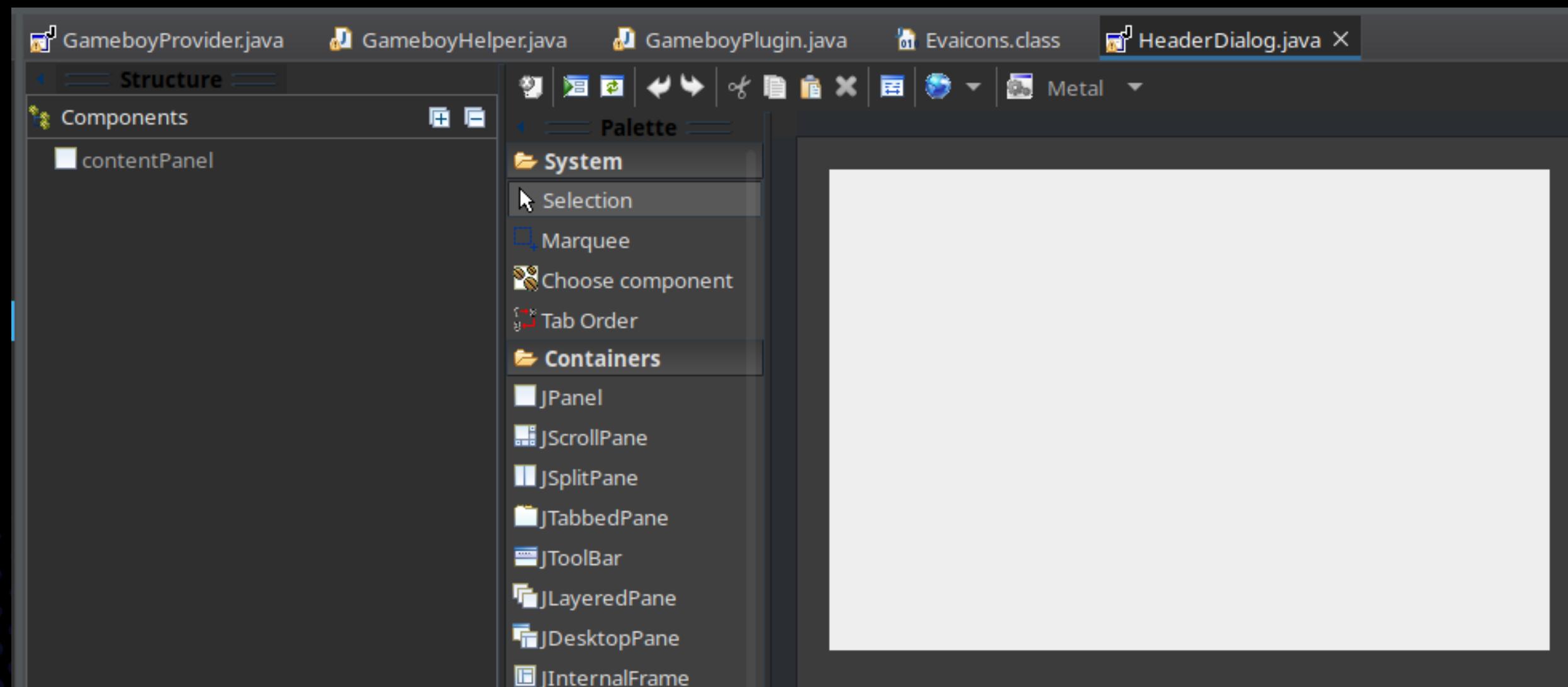
07

12



WORKING WITH UI STUFF

WindowBuilder!



MENU



01



07



12



MY FIRST UI ATTEMPT

Gameboy Header Information											
Entry Point	Nintendo...	Title	SGB Flag	Cartridge...	ROM Size	RAM Size	Destinati...	Old Licens...	Mask RO...	Header C...	Global C...
00 C3 50 01	CE ED 66 66 CC 0D 00 0 B 03 73 00 83 00 0C 00 0D 00 08 1 1 1F 88 89 00 0E DC CC 6E E6 DD D D D9 99 BB BB 67 63 6E	50 4F 4B 45 4D 4F 4E 2 0 52 45 44 00 00 00 00 00	30 31 03	13	05	03	01	33	00	20	91 E6

FRUSTRATION?

Consider migrating away from Java/Swing! #4188

Closed

ghost opened this issue on Apr 24, 2022 · 0 comments



ghost commented on Apr 24, 2022

It is frustrating to work with the current user interface which is written using Java/Swing.

I can understand that at the point when development was initiated, Java/Swing might have been a good choice for cross-platform user interface development, but there are options available these days which are more feature rich while delivering improved responsiveness and performance than Java/Swing.

Would you please "consider" moving the user interface to [Qt](#)?

Qt is written in C++ (mostly) and delivers near native platform widget responsiveness than Java/Swing.



NationalSecurityAgency locked and limited conversation to collaborators
on Apr 24, 2022

Assignees

No one assigned

Labels

[Feature: GUI](#)

Projects

None yet

Milestone

No milestone

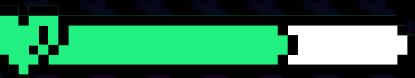
Development

No branches or pull requests



★★★★★

HIGHSCORE 2500



PLAYER 2

PLUGIN COMPONENT

- ◆ This is where you can add logic to your component.
- ◆ Allows you to interact with the game.

```
@Override
public void programActivated(Program program) {
    super.programActivated(program);

    api = new FlatProgramAPI(this.currentProgram);
    GameboyHelper.init(tool, api);
    provider.addDetailsPanel();
}

String topicName = this.getClass().getPackage().getName();
String anchorName = "HelpAnchor";
provider.setHelpLocation(new HelpLocation(topicName, anchorName));
}
```



HIGHSCORE 2500



PLAYER 2

PRO COM

◆ This is where sort of stuff

◆ You initialize

```
// TODO: Customize actions
private void createActions() {
    DockingAction parseHeaderAction = new DockingAction("Parse Header", getName()) {
        @Override
        public void actionPerformed(ActionEvent context) {

            for (Map.Entry<String, byte[]> entry : GameboyHelper.headerBytes.entrySet()) {
                System.out.print(entry.getKey() + ": ");
                for (byte a : entry.getValue()) {
                    System.out.print(String.format("%02X ", a));
                }
                System.out.println();
            }
            tool.showDialog(new HeaderDialog("Gameboy Header Information"));
        }
    };

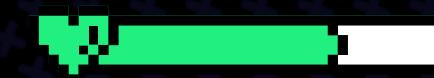
    parseHeaderAction.setToolBarData(new ToolBarData(FontIcon.of(Evaicons.FILE_TEXT), null));
    parseHeaderAction.setEnabled(true);
    parseHeaderAction.markHelpUnnecessary();
    parseHeaderAction.setHelpLocation(HELP);
    dockingTool.addLocalAction(this, parseHeaderAction);

    DockingAction checksumAction = new DockingAction("Calculate and Verify Checksums", getName()) {
        @Override
        public void actionPerformed(ActionEvent context) {
            int checksum = GameboyHelper.calcHeaderChecksum();
            System.out.println("Calculated Checksum: " + checksum);
            System.out.println("Given Checksum: " + Byte.toUnsignedInt(GameboyHelper.getHeaderChecksum()));
            System.out.println(
                "Checksum Valid? - " + (checksum == Byte.toUnsignedInt(GameboyHelper.getHeaderChecksum())));
            tool.showDialog(new ChecksumDialog("Gameboy Checksum Information"));
        }
    };

    checksumAction.setToolBarData(new ToolBarData(FontIcon.of(Evaicons.HASH), null));
    checksumAction.setEnabled(true);
    checksumAction.markHelpUnnecessary();
    checksumAction.setHelpLocation(HELP);
    dockingTool.addLocalAction(this, checksumAction);
}
```



HIGHSCORE 2500



PLAYER 2

DOCKING ACTIONS

- ➔ Actions that fire when state change, etc)
- ➔ small functions that ca

```
DockingAction checksumAction = new DockingAction("Calculate and Verify Checksums", getName()) {  
    @Override  
    public void actionPerformed(ActionEvent context) {  
        int checksum = GameboyHelper.calcHeaderChecksum();  
        System.out.println("Calculated Checksum: " + checksum);  
        System.out.println("Given Checksum: " + Byte.toUnsignedInt(GameboyHelper.getHeaderChecksum()));  
        System.out.println(  
            "Checksum Valid? - " + (checksum == Byte.toUnsignedInt(GameboyHelper.getHeaderChecksum())));  
        tool.showDialog(new ChecksumDialog("Gameboy Checksum Information"));  
    }  
};  
  
checksumAction.setToolBarData(new ToolBarData(FontIcon.of(Evaicons.HASH), null));  
checksumAction.setEnabled(true);  
checksumAction.markHelpUnnecessary();  
checksumAction.setHelpLocation(HELP);  
dockingTool.addLocalAction(this, checksumAction);
```



HIGHSCORE 2500



PLAYER 2

HELPFR COM

➔ Helper co

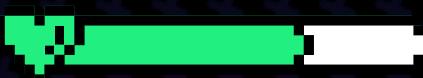
➔ Keep stat

➔ Leverage

```
public class GameboyHelper {  
    private static PluginTool tool = null;  
    private static FlatProgramAPI api = null;  
  
    public static LinkedHashMap<String, Integer> headerEntries = new LinkedHashMap<String, Integer>();  
    public static LinkedHashMap<Integer, String> ioEntries = new LinkedHashMap<Integer, String>();  
    private static LinkedHashMap<Integer, String> oldLicensees = new LinkedHashMap<Integer, String>();  
    private static LinkedHashMap<String, String> newLicensees = new LinkedHashMap<String, String>();  
    public static LinkedHashMap<String, byte[]> headerBytes = new LinkedHashMap<>();  
  
    public static int headerChecksum = 0;  
  
    public static void init(PluginTool tool_, FlatProgramAPI api_) {  
        GameboyHelper.tool = tool_;  
        GameboyHelper.api = api_;  
        buildMap();  
        buildHeader();  
        buildLicensees();  
    }  
}
```



HIGHSCORE 2500



PLAYER 2

MAKES THINGS SIMPLER

```
}

public static String getProgName() {
    return api.getCurrentProgram().getDomainFile().getName();
}

public static String getPath() {
    return api.getCurrentProgram().getExecutablePath();
}

public static String getSHA256() {
    return api.getCurrentProgram().getExecutableSHA256();
}

public static String getMD5() {
    return api.getCurrentProgram().getExecutableMD5();
}

public static void setCodeComments() {
    setComments();
}

public static String getLicensee() {
    try {
        int addr_val = api.getByte(api.toAddr(headerEntries.get("Old Licensee Code")));
        if (addr_val == 0x33) {
            String licenseeCode = String.format("%c%c", (char) api.getByte(api.toAddr(0x144)),
                (char) api.getByte(api.toAddr(0x145)));
            System.out.println(licenseeCode);
            return newLicensees.get(licenseeCode);
        }
        return oldLicensees.get(addr_val);
    } catch (MemoryAccessException e) {
        e.printStackTrace();
    }
    return null;
}
```

MENU

01

07

12



REMEMBER THAT PARSER WE NEEDED?

Earlier we discovered we needed to parse the cartridge header. Let's Parse it.

```
00000100: 00c3 ab01 ceed 6666 cc0d 000b 0373 0083 .....ff.....s...
00000110: 000c 000d 0008 111f 8889 000e dccc 6ee6 .....n.....
00000120: dddd d999 bbbb 6763 6e0e eccc dddc 999f .....gcn.....
00000130: bbb9 333e 504f 4b45 4d4f 4e20 5945 4c4c ..3>POKEMON YELL
00000140: 4f57 0080 3031 031b 0503 0133 0097 047c 0W..01....3...|
00000150: f0b8 f578 cd7e 3e2a 4f2a 472a 573e 033d ...x..~>*0*G*W>.=
00000160: 20fd cd99 01cd a501 cd99 01cd a501 cd99 ..... .
00000170: 01cd a501 cd99 01cd a501 cd99 01cd a501 ..... .
00000180: cd99 01cd a501 cd99 01cd a501 cd99 010b ..... .
```

MENU

01

07

12



PARSE IT

```
FLATapi.getByte(FLATapi.toAddr(ADDR_  
    POINTER))
```

Entry Point: 00 C3 50 01

Nintendo Logo: CE ED 66 66 CC 0D 00 0B 03 73 00 83 00 0C 00 0D 00 08 11 1F 88 89 00 0E DC CC 6E E6 DD D9 99 BB BB 67 63 6E 0E EC CC DD DC 99 9F BB B9 33 3E

Title: 53 50 49 44 45 52 2D 4D 41 4E 00 00 00 00 00 00

SGB Flag: 00 00 00

Cartridge Type: 01

ROM Size: 01

RAM Size: 00

Destination Code: 01

Old Licensee Code: 56

Mask ROM Version: 00

Header Checksum: BE

Global Checksum: 9B 5

```
public static LinkedHashMap<String, byte[]> getHeader() {
    Integer pointer = 0x100;
    LinkedHashMap<String, byte[]> headerBytes = new LinkedHashMap<>();
    try {
        for (Map.Entry<String, Integer> entry : headerEntries.entrySet()) {
            ArrayList<Byte> curBytes = new ArrayList<Byte>();
```

1

MENU

01

CONVERT

```
private static void buildLicensees()
{
    oldLicensees.put(0x00, "None");
    oldLicensees.put(0x01, "Nintendo");
    oldLicensees.put(0x08, "Capcom");
    oldLicensees.put(0x09, "HOT-B");
    oldLicensees.put(0x0A, "Jaleco");
    oldLicensees.put(0x0B, "Coconuts");
    oldLicensees.put(0x0C, "Elite Sy");
    oldLicensees.put(0x13, "EA (Elec");
    oldLicensees.put(0x18, "Hudson S");
    oldLicensees.put(0x19, "ITC Ente");
    oldLicensees.put(0x1A, "Yanoman");
    oldLicensees.put(0x1D, "Japan Cl");
    oldLicensees.put(0x1F, "Virgin G");
    oldLicensees.put(0x24, "PCM Comp");
    oldLicensees.put(0x25, "San-X");
    oldLicensees.put(0x28, "Kemco");
    oldLicensees.put(0x29, "SETA Cor");
    oldLicensees.put(0x30, "Infogram");
    oldLicensees.put(0x31, "Nintendo");
    oldLicensees.put(0x32, "Bandai");
    oldLicensees.put(0x33, "Indicate");
    oldLicensees.put(0x34, "Konami");
    oldLicensees.put(0x35, "HectorSo");
    oldLicensees.put(0x38, "Capcom");
    oldLicensees.put(0x39, "Banprest");
    oldLicensees.put(0x3C, ".Enterta");
    oldLicensees.put(0x3E, "Gremlin");
    oldLicensees.put(0x41, "Ubi Soft");
    oldLicensees.put(0x42, "Atlus");
    oldLicensees.put(0x44, "Malibu I");
}
```

```

newLicensees.put("00", "None");
newLicensees.put("01", "Nintendo Research & Development 1");
newLicensees.put("08", "Capcom");
newLicensees.put("13", "EA (Electronic Arts)");
newLicensees.put("18", "Hudson Soft");
newLicensees.put("19", "B-AI");
newLicensees.put("20", "KSS");
newLicensees.put("22", "Planning Office WADA");
newLicensees.put("24", "PCM Complete");
newLicensees.put("25", "San-X");
newLicensees.put("28", "Kemco");
newLicensees.put("29", "SETA Corporation");
newLicensees.put("30", "Viacom");
newLicensees.put("31", "Nintendo");
newLicensees.put("32", "Bandai");
newLicensees.put("33", "Ocean Software/Acclaim Entertainment");
newLicensees.put("34", "Konami");
newLicensees.put("35", "HectorSoft");
newLicensees.put("37", "Taito");
newLicensees.put("38", "Hudson Soft");
newLicensees.put("39", "Banpresto");
newLicensees.put("41", "Ubi Soft");
newLicensees.put("42", "Atlus");
newLicensees.put("44", "Malibu Interactive");
newLicensees.put("46", "Angel");
newLicensees.put("47", "Bullet-Proof Software");
newLicensees.put("49", "Irem");
newLicensees.put("50", "Absolute");
newLicensees.put("51", "Acclaim Entertainment");
newLicensees.put("52", "Activision");
newLicensees.put("53", "Sammy USA Corporation");
newLicensees.put("54", "Konami");
}
```



ATION

```

edgeTypes() {
    ILY";
    ;
    IAM";
    IAM+BATTERY";
    ;
    BATTERY");
    M";
    M+BATTERY");
    );
    RAM");
    RAM+BATTERY");
    IMER+BATTERY");
    IMER+RAM+BATTERY");
    ;
    IAM");
    IAM+BATTERY");
    ;
    RUMBLE");
    RUMBLE+RAM");
    RUMBLE+RAM+BATTERY");
    ;
    SENSOR+RUMBLE+RAM+BATTERY");
    CAMERA");
    TAMA5");
    ;
    IAM+BATTERY");
}
```

MENU

01

07

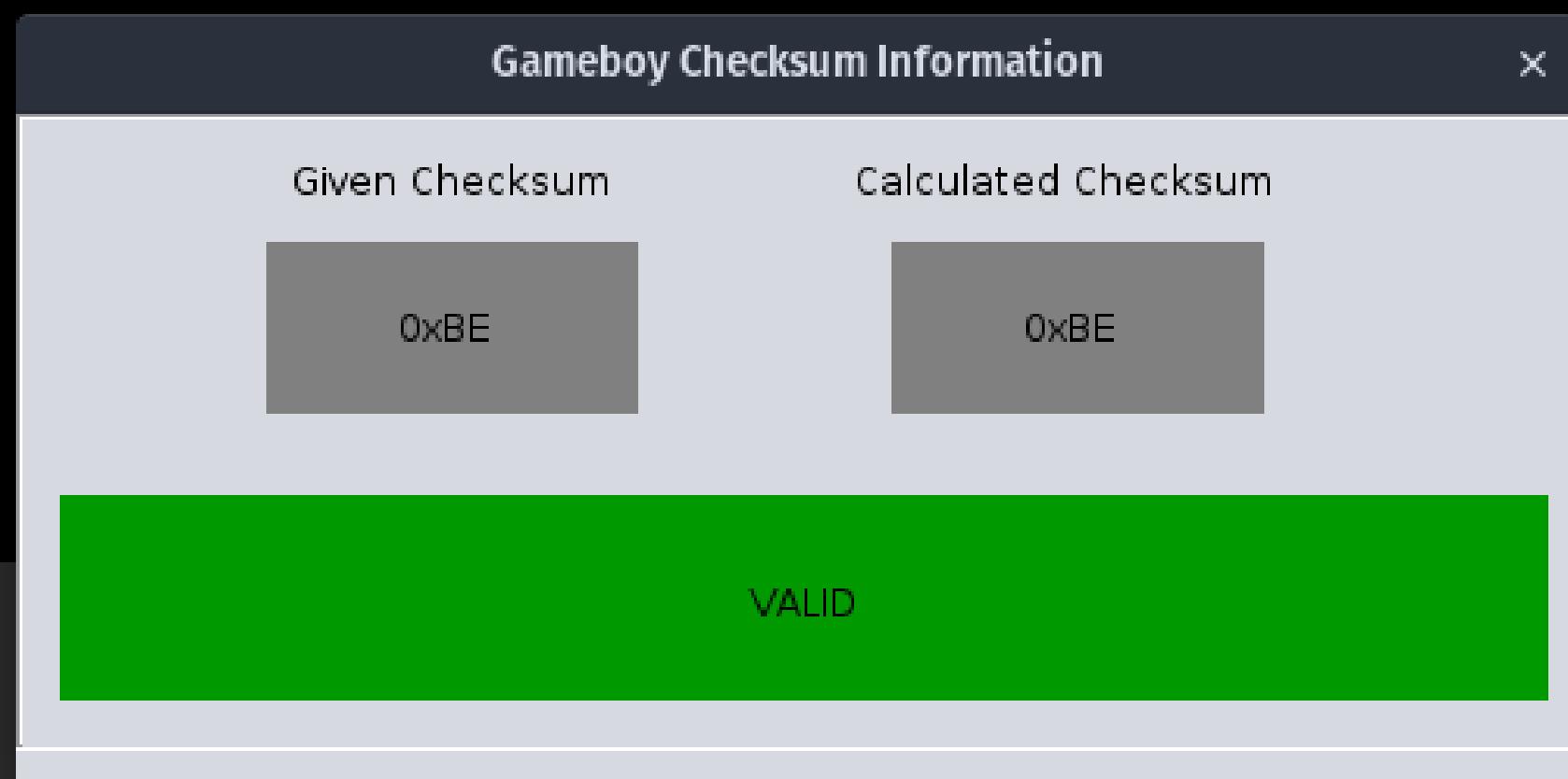
12



VERIFYING HEADER INFO

Luckily, if we parse correctly, we can verify it with a checksum.

```
public static int calcHeaderChecksum() {  
    int checksum = 0;  
    try {  
        for (int addr = 0x134; addr <= 0x14C; addr++) {  
            checksum = checksum - api.getByte(api.toAddr(addr)) - 1;  
        }  
    } catch (MemoryAccessException e) {  
        e.printStackTrace();  
    }  
  
    headerChecksum = checksum & 0xFF;  
    return headerChecksum;  
}
```



MENU

01

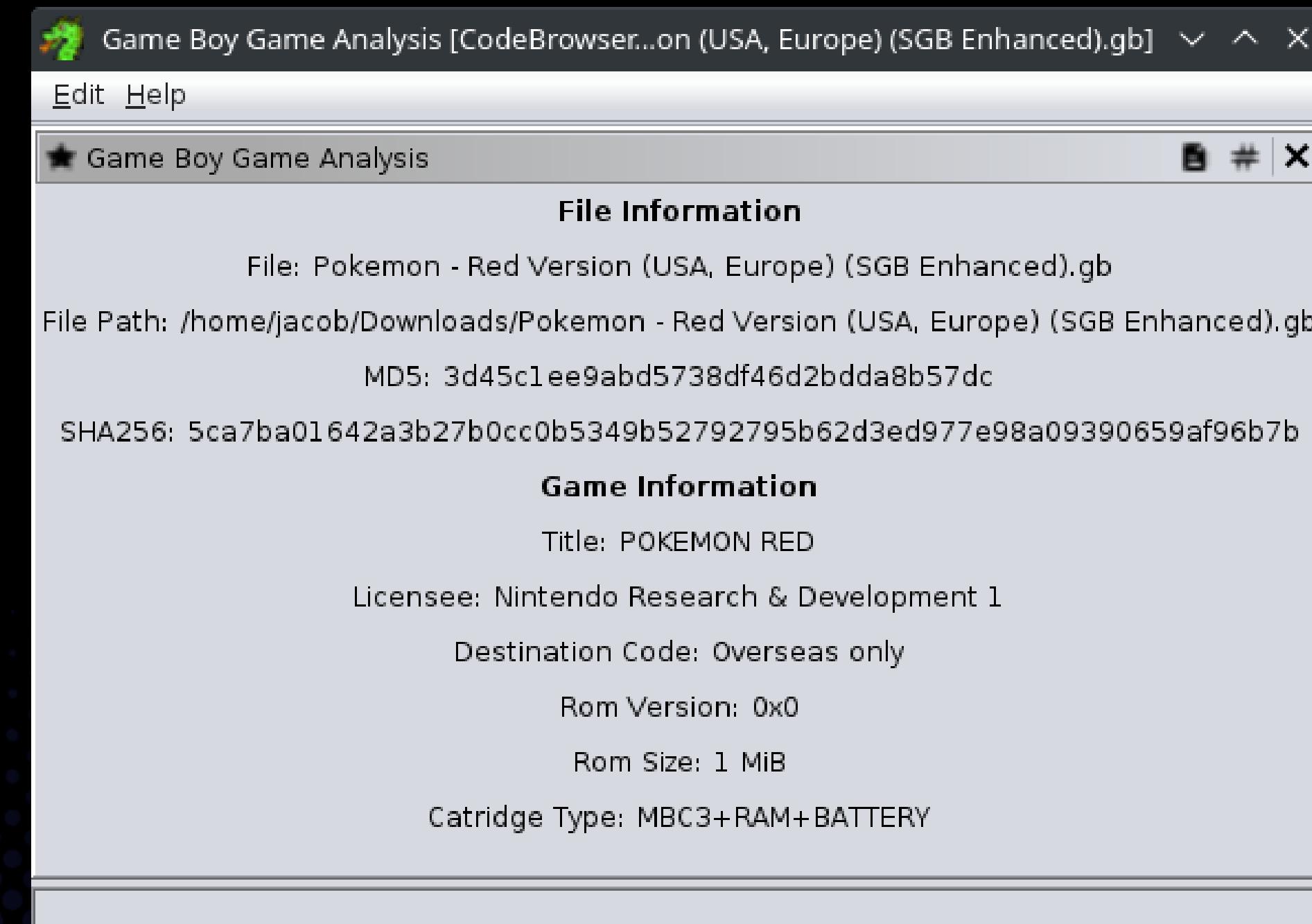
07

12



GOOD INFO? GOOD!

As we know if we parsed things correctly, we can now display it to the user for quick info.



MENU

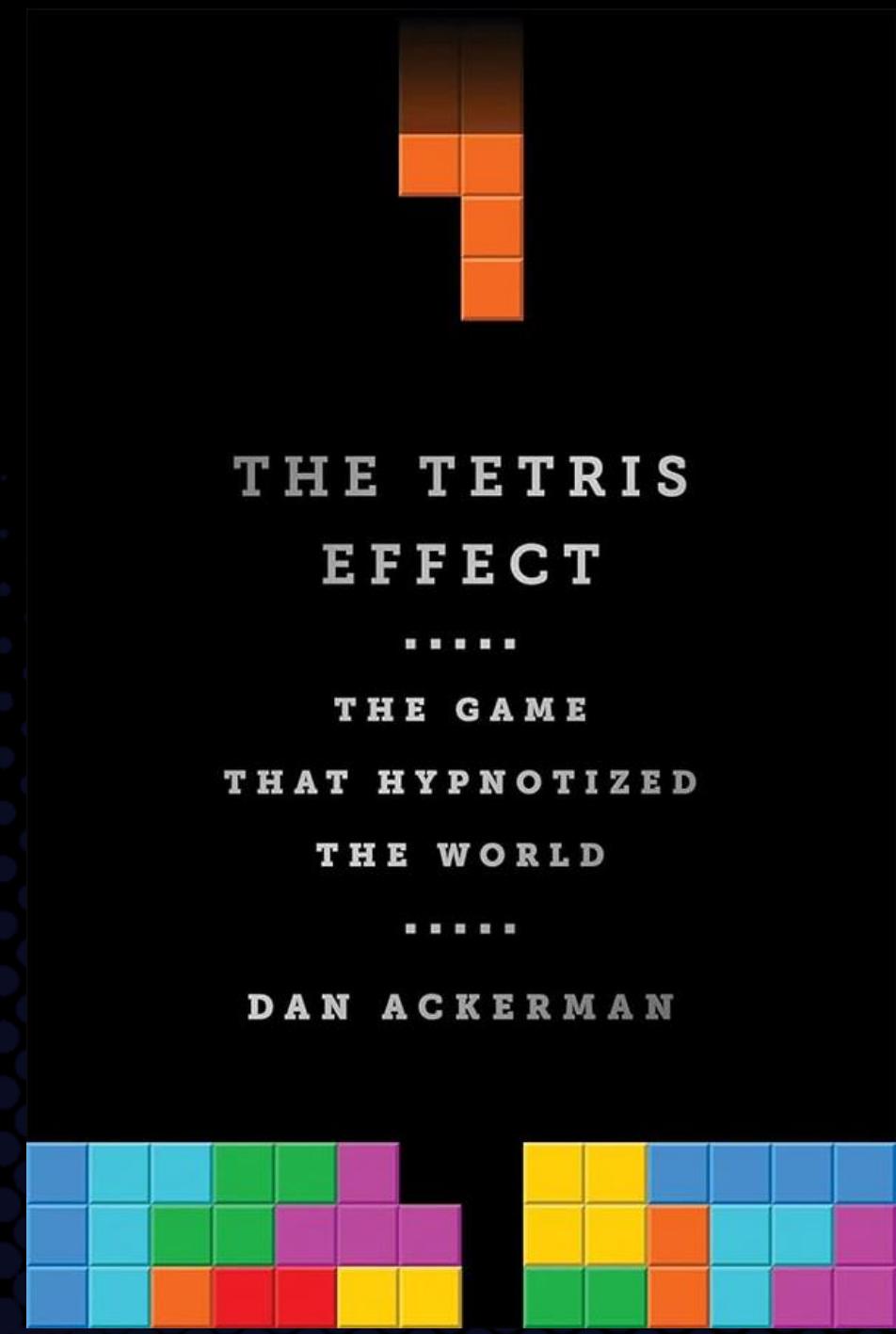
01

07

12



WHO KNOWS TETRIS?



MENU

01

07

12



VARIOUS VERSIONS OF TETRIS



MENU

01

07

12



VARIOUS VERSIONS OF TETRIS

Game Boy Game Analysis [CodeBrowser(2): gba:/Tetris (World) (Rev A).gb]

Edit Help

★ Game Boy Game Analysis

File Information

File: Tetris (World) (Rev A).gb
File Path: /home/jacob/Downloads/Tetris (World) (Rev A).gb
MD5: 982ed5d2b12a0377eb14bcd4123744e
SHA256: 0d6535aef23969c7e5af2b077acaddb4a445b3d0df7bf34c8acef07b51b015c3

Game Information

Licensee: Nintendo
Destination Code: Japan (and possibly overseas)
Rom Version: 0x1
Rom Size: 32 KiB

Game Boy Game Analysis [CodeBrowser(4): gba:/Tetris (Japan) (En).gb]

Edit Help

★ Game Boy Game Analysis

File Information

File: Tetris (Japan) (En).gb
File Path: /home/jacob/Downloads/Tetris (Japan) (En).gb
MD5: 084f1e457749cdec86183189bd88ce69
SHA256: 7fdel1dd4e594a6905decccd57943d2909ecb37665a030741c42155aeb346323b

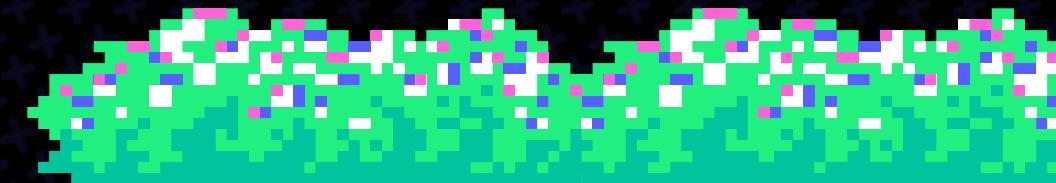
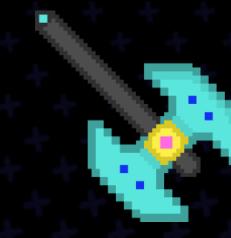
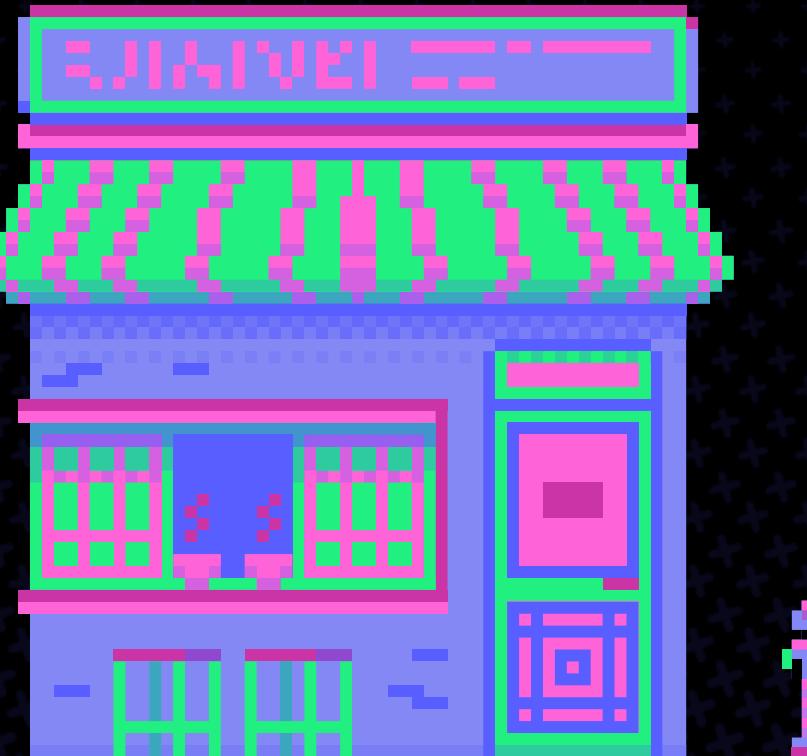
Game Information

Licensee: Nintendo
Destination Code: Japan (and possibly overseas)
Rom Version: 0x0
Rom Size: 32 KiB

SIGN IN



NEXT USE CASE: I/O



GOING FROM THIS ...

A bit ago, I showed the hardware address table, let's use that context!

		LAB_2125	XREF[1]:	0058(j)
2125	f5	PUSH AF		
2126	c5	PUSH BC		
2127	d5	PUSH DE		
2128	e5	PUSH HL		
2129	f0 aa	LDH A, (offset DAT_ffaa)	= ??	
212b	3c	INC A		
212c	28 14	JR Z, LAB_2142		
212e	f0 01	LDH A, (offset SB)	= ??	
2130	e0 ad	LDH (offset DAT_ffad), A	= ??	
2132	f0 ac	LDH A, (offset DAT_ffac)	= ??	
2134	e0 01	LDH (offset SB), A	= ??	
2136	f0 aa	LDH A, (offset DAT_ffaa)	= ??	
2138	fe 02	CP 0x2		
213a	28 26	JR Z, LAB_2162		
213c	3e 80	LD A, 0x80		
213e	e0 02	LDH (offset SC), A	= ??	
2140	18 20	JR LAB_2162		

MENU

01

07

12



TO THIS ...

Showing
Hardware (I/O)
Addresses!

	LAB_2125		XREF[1]: 0058(j)
2125	f5	PUSH AF	
2126	c5	PUSH BC	
2127	d5	PUSH DE	
2128	e5	PUSH HL	
2129	f0 aa	LDH A, (offset DAT_ffaa)	= ??
212b	3c	INC A	
212c	28 14	JR Z, LAB_2142	
212e	f0 01	LDH A, (offset SB)	SB: Serial transfer data
2130	e0 ad	LDH (offset DAT_ffad), A	= ??
2132	f0 ac	LDH A, (offset DAT_ffac)	= ??
2134	e0 01	LDH (offset SB), A	SB: Serial transfer data
2136	f0 aa	LDH A, (offset DAT_ffaa)	= ??
2138	fe 02	CP 0x2	
213a	28 26	JR Z, LAB_2162	
213c	3e 80	LD A, 0x80	
213e	e0 02	LDH (offset SC), A	SC: Serial transfer control
2140	18 20	JR LAB_2162	

MAP INSTRUCTIONS TO HARDWARE

```
private static void setComments() {
    ioEntries.put(0xFF00, "P1/JOYP: Joypad");
    ioEntries.put(0xFF01, "SB: Serial transfer data");
    ioEntries.put(0xFF02, "SC: Serial transfer control");
    ioEntries.put(0xFF04, "DIV: Divider register");
    ioEntries.put(0xFF05, "TIMA: Timer counter");
    ioEntries.put(0xFF06, "TMA: Timer modulo");
    ioEntries.put(0xFF07, "TAC: Timer control");
    ioEntries.put(0xFF0F, "IF: Interrupt flag");
    ioEntries.put(0xFF10, "NR10: Sound channel 1 sweep");
    ioEntries.put(0xFF11, "NR11: Sound channel 1 length timer & duty cycle");
    ioEntries.put(0xFF12, "NR12: Sound channel 1 volume & envelope");
    ioEntries.put(0xFF13, "NR13: Sound channel 1 period low");
    ioEntries.put(0xFF14, "NR14: Sound channel 1 period high & control");
    ioEntries.put(0xFF16, "NR21: Sound channel 2 length timer & duty cycle");
    ioEntries.put(0xFF17, "NR22: Sound channel 2 volume & envelope");
    ioEntries.put(0xFF18, "NR23: Sound channel 2 period low");
    ioEntries.put(0xFF19, "NR24: Sound channel 2 period high & control");
    ioEntries.put(0xFF1A, "NR30: Sound channel 3 DAC enable");
    ioEntries.put(0xFF1B, "NR31: Sound channel 3 length timer");
    ioEntries.put(0xFF1C, "NR32: Sound channel 3 output level");
    ioEntries.put(0xFF1D, "NR33: Sound channel 3 period low");
    ioEntries.put(0xFF1E, "NR34: Sound channel 3 period high & control");
    ioEntries.put(0xFF20, "NR41: Sound channel 4 length timer");
    ioEntries.put(0xFF21, "NR42: Sound channel 4 volume & envelope");
    ioEntries.put(0xFF22, "NR43: Sound channel 4 frequency & randomness");
    ioEntries.put(0xFF23, "NR44: Sound channel 4 control");
    ioEntries.put(0xFF24, "NR50: Master volume & VIN panning");
    ioEntries.put(0xFF25, "NR51: Sound panning");
    ioEntries.put(0xFF26, "NR52: Sound on/off");
    ioEntries.put(0xFF30, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF31, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF32, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF33, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF34, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF35, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF36, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF37, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF38, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF39, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF3A, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF3B, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF3C, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF3D, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF3E, "Wave RAM: Storage for one of the sound channels' waveform");
    ioEntries.put(0xFF3F, "Wave RAM: Storage for one of the sound channels' waveform");
}
```

MAP INSTRUCTIONS TO HARDWARE

```
var id = api.getCurrentProgram().startTransaction("Set line comment from Kernel");

InstructionIterator insIter = api.getCurrentProgram().getListing().getInstructions(true);
while (insIter.hasNext()) {
    Instruction ins = insIter.next();
    for (Reference ref: ins.getReferencesFrom()) {
        int dest = (int) ref.getToAddress().getPhysicalAddress().getOffset();
        if (ioEntries.containsKey(dest)) {
            CodeUnit cu = api.getCurrentProgram().getListing().getCodeUnitContaining(ins.getAddress());
            cu.setComment(CodeUnit.EOL_COMMENT, ioEntries.get(dest));
        }
    }
}

for (Map.Entry<Integer, String> entry : ioEntries.entrySet()) {
    CodeUnit cu = api.getCurrentProgram().getListing().getCodeUnitContaining(api.toAddr(entry.getKey()));
    cu.setComment(CodeUnit.EOL_COMMENT, entry.getValue());
}
api.getCurrentProgram().endTransaction(id, true);
```

MENU

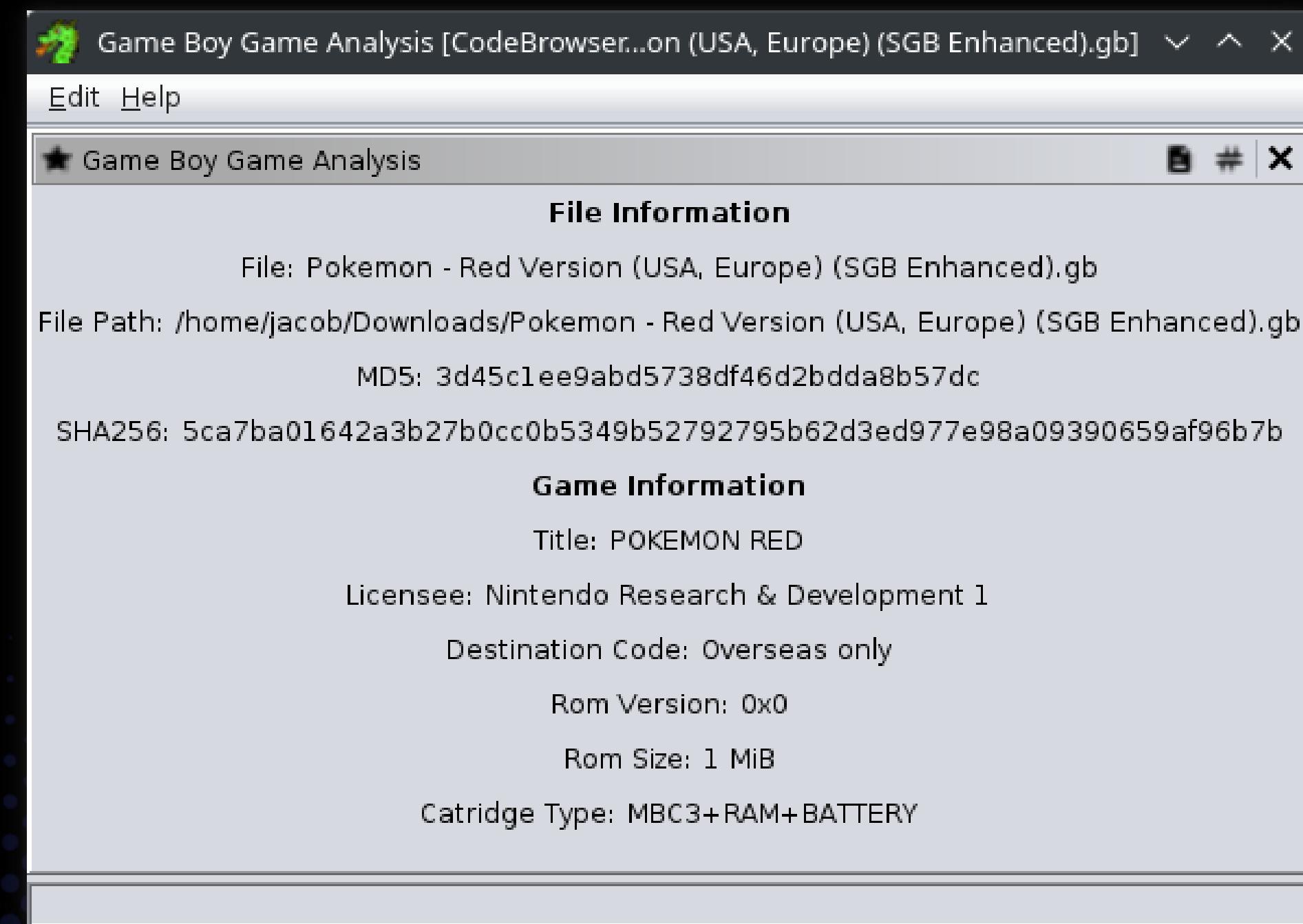
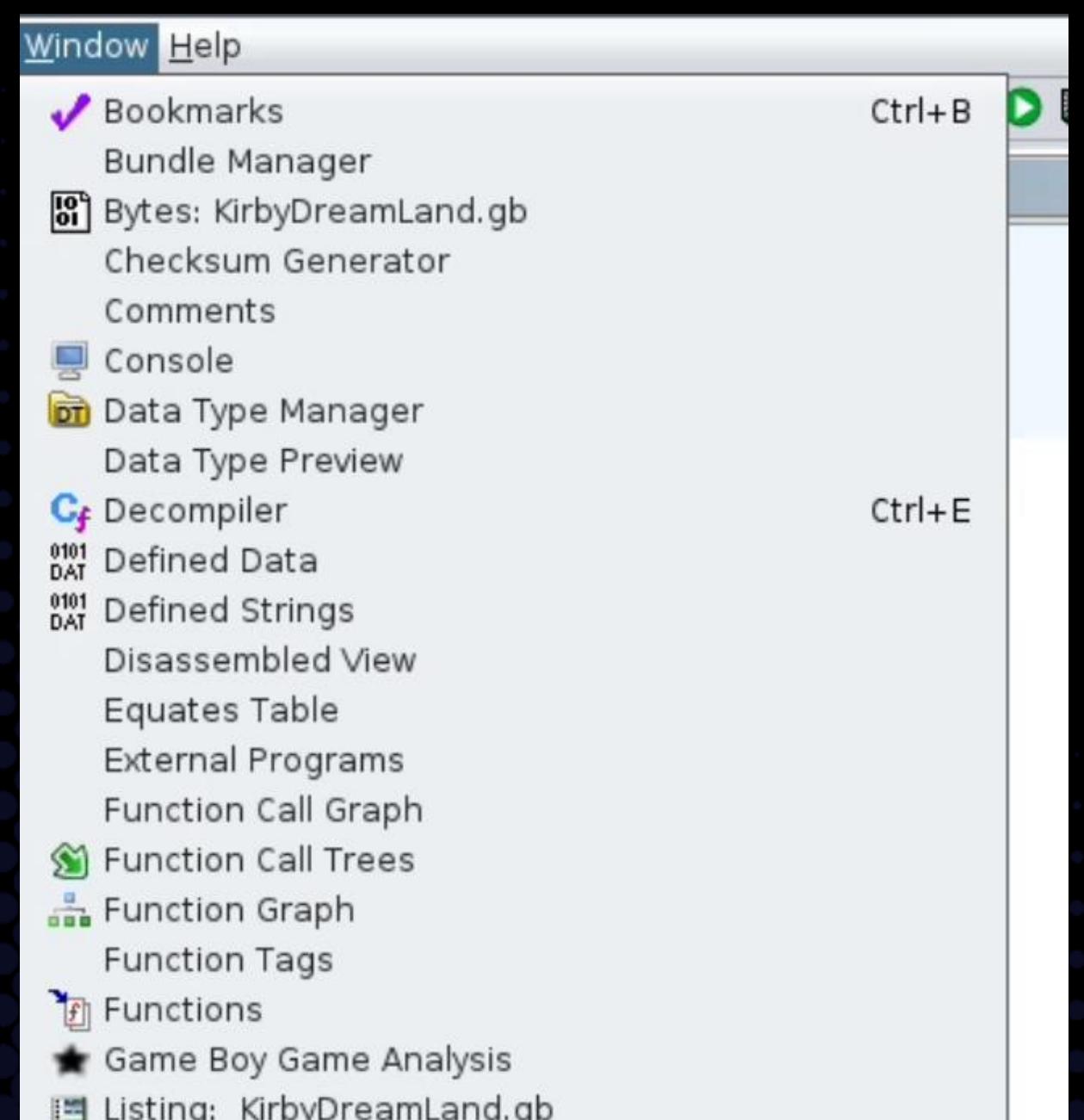
01

07

12



FINISHED (?) PRODUCT!



MENU

01

07

12



POKEMON!

Game Boy Game Analysis [CodeBrowser...on (USA, Europe) (SGB Enhanced).gb] ▾ ⌂ ×

Edit Help

★ Game Boy Game Analysis

File Information

File: Pokemon - Red Version (USA, Europe) (SGB Enhanced).gb

File Path: /home/jacob/Downloads/Pokemon - Red Version (USA, Europe) (SGB Enhanced).gb

MD5: 3d45c1ee9abd5738df46d2bdda8b57dc

SHA256: 5ca7ba01642a3b27b0cc0b5349b52792795b62d3ed977e98a09390659af96b7b

Game Information

Title: POKEMON RED

Licensee: Nintendo Research & Development 1

Destination Code: Overseas only

Rom Version: 0x0

Rom Size: 1 MiB

Cartridge Type: MBC3+RAM+BATTERY

MENU

01

07

12



KIRBY'S DREAM LAND

Game Boy Game Analysis [CodeBrowser: gameboy:/KirbyDreamLand.gb] ▾ ⌂

Edit Help

★ Game Boy Game Analysis

File Information

File: KirbyDreamLand.gb

File Path: /home/jacob/Downloads/Kirby's Dream Land .gb

MD5: a66e4918edcd042ec171a57fe3ce36c3

SHA256: 0f6dba94fae248d419083001c42c02a78be6bd3dff679c895517559e72c98d58

Game Information

Title: KIRBY DREAM LAND

Licensee: Nintendo

Destination Code: Overseas only

Rom Version: 0x0

Rom Size: 256 KiB

Cartridge Type: MBC1

MENU

01

07

12



BATMAN: TAS

Game Boy Game Analysis [CodeBrowser: gameboy:/Batman - The Animated Series... ▾ ▲ ⚙]

Edit Help

★ Game Boy Game Analysis ✖

File Information

File: Batman - The Animated Series.gb
File Path: /home/jacob/Downloads/Batman - The Animated Series.gb
MD5: ae073c63ff7d151dc2dd406830fbbdc2
SHA256: 9ac1f4a299d32ba21cf65f67ab210afeb4c629adbd8e5779f76b6667ca3a0a4a

Game Information

Title: BATMAN ANIMATED
Licensee:
Destination Code: Overseas only
Rom Version: 0x0
Rom Size: 128 KiB
Cartridge Type: MBC1

MENU

01

07

12



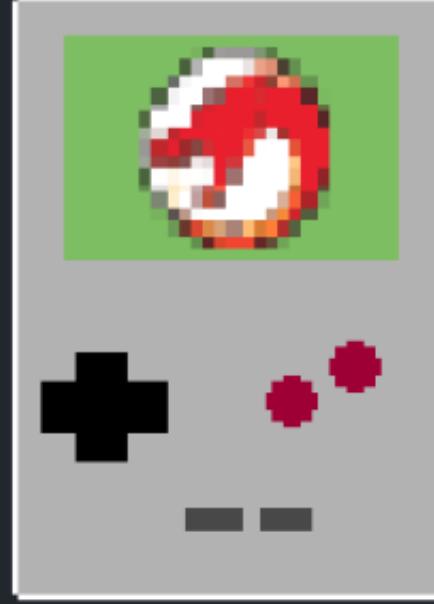
FULL SOURCE CODE?

Available on GitHub!

latonis/GBanalyzer

README

GBAnalyzer



GBANALYZER

GBAnalyzer is a Ghidra plugin that aids in reverse-engineering Game Boy games.

- This is not a decompiler or disassembler. If you're interested in that, check out [GhidraBoy](#).

MENU

01

07

12



./UTILS

.github > workflows > ! build.yml

```
1  name: Build GBAnalyzer
2
3  on: [ push ]
4
5  env:
6    GHIDRA_VERSION: "11.0"
7    GHIDRA_BUILD_DATE: 20231222
8    GHIDRA_SHA256: f1f240f91cf6b1dffc9a4148384ee3c6b269a8ae27c6f981577973e00043ad94
9    GHIDRA_INSTALL_DIR: /home/runner/ghidra
10
11 jobs:
12   build:
13     runs-on: ubuntu-latest
14     steps:
15       - name: Checkout sources
16         uses: actions/checkout@v4
17       - name: Setup Java JDK
18         uses: actions/setup-java@v4
19         with:
20           distribution: 'microsoft'
21           java-version: '17'
22       - name: Install Ghidra
23         run: .github/utils/install_ghidra.sh
24       - name: Setup Gradle**Full Changelog**: https://github.com/latonis/GBAnalyzer/commits/20240612
25         uses: gradle/actions/setup-gradle@v3
26       - name: Build with Gradle
27         run: gradle -P${{env.GHIDRA_INSTALL_DIR}}=/home/runner/ghidra
28       - uses: actions/upload-artifact@v4
29         with:
30           name: "GBAnalyzer_Plugin_${{env.GHIDRA_VERSION}}"
31           path: dist/*GBAnalyzer.zip
32           if-no-files-found: 'error'
```

latonis/GBAnalyzer

MENU

01

07

12



LINKS!

Available on GitHub!



latonis/GBanalyzer



0x00