PLAYER 1

HIGHSCORE 9000

PLAYER 2

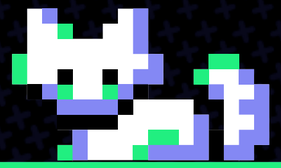
LEVELING UP GHIDRA:

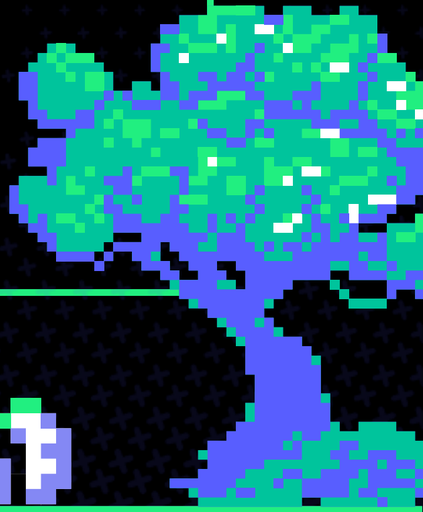
LEARN GHIDRA PLUGINS WITH A GAME BOY GAME

SANS DFIR SUMMIT - Aug 23rd, 2024

JACOB LATONIS

MENU START SIGN IN

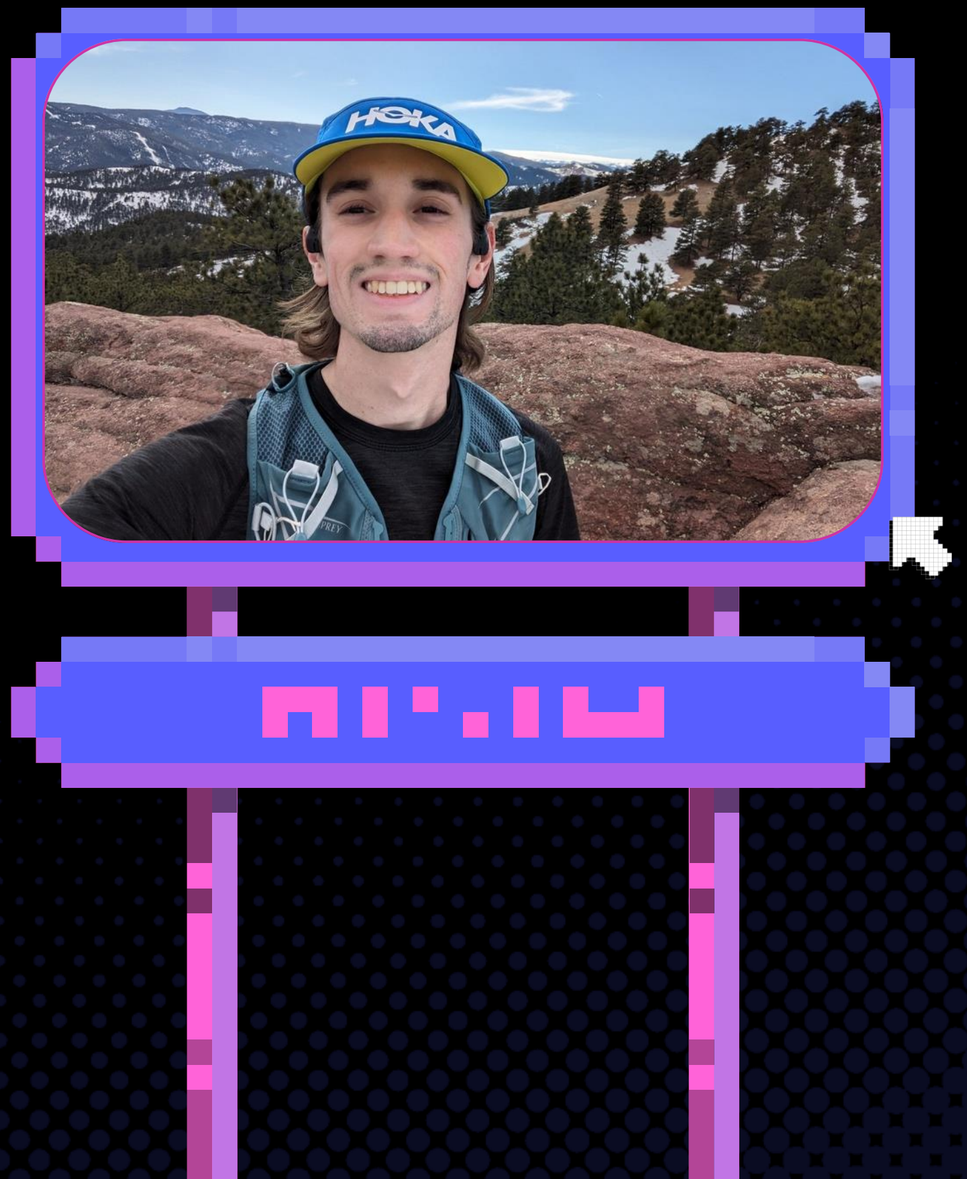




MENU 01 07 12

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



MENU 01 07 12

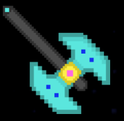
AGENDA

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

GHIDRA

The possibilities!

Implementation



A lot more!

MENU 01 07 12

BEFORE WE START...

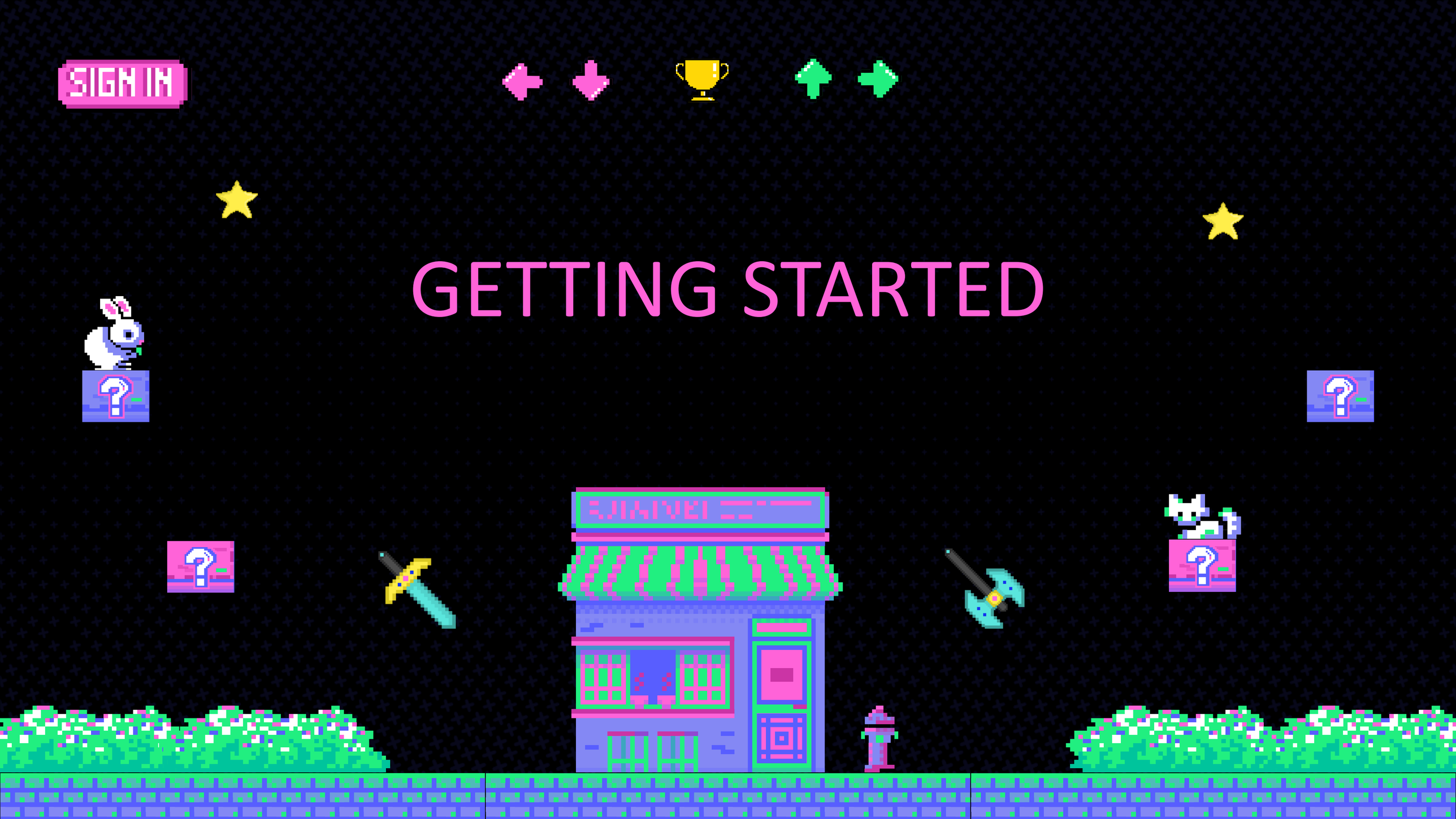
We will discuss a whole lot here today, not every topic will have the time and/or focus to be fully completed and discussed.

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

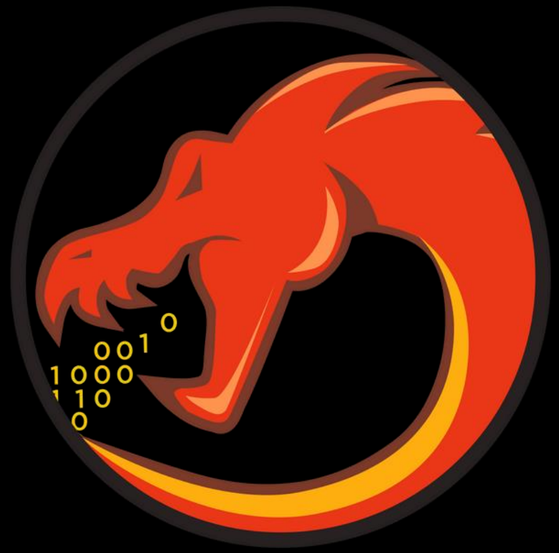
GHIDRA!

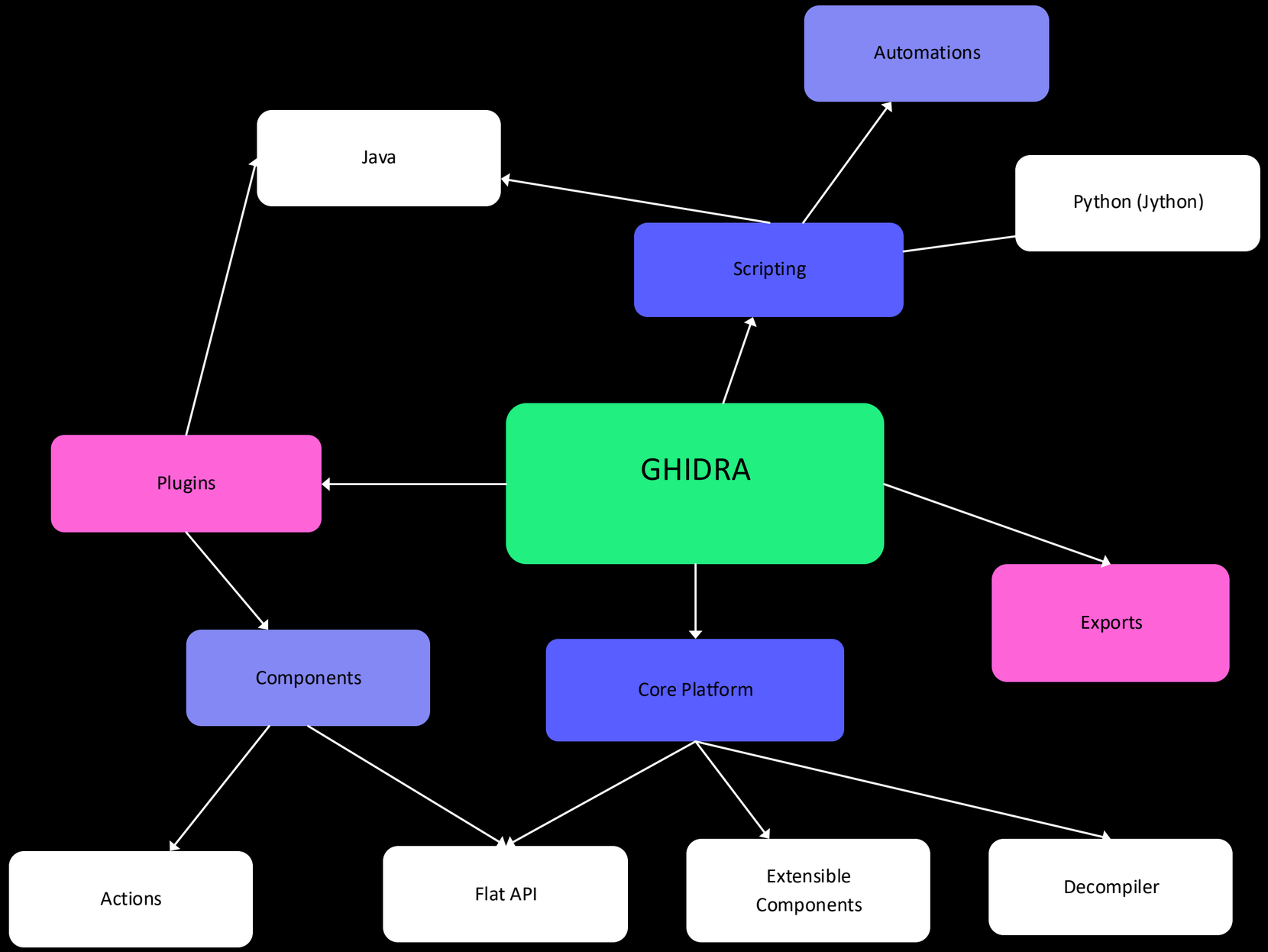
- Open-Source Reverse engineering Toolkit
- Developed by the NSA because IDA costs a billion dollars
- Highly extensible and customizable
- FREE!


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

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
- 4.194304 MHz CPU | 160 x 144 Display

GAME BOY RESOURCES

GEKKIO/GB-RESEARCH

GAME BOY PANDOCS

GEKKIO/GHIDRABOY

GEKKIO/GB-CTR
(COMPLETE TECHNICAL REFERENCE)

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


HEADER BROKEN DOWN!

0100-0103 — ENTRY POINT

0104-0133 — NINTENDO LOGO

0134-0143 — TITLE

013F-0142 — MANUFACTURER CODE

HEADER BROKEN DOWN!

0143 — CGB FLAG

0144–0145 — NEW LICENSEE CODE

0146 — SGB FLAG

0147 — CARTRIDGE TYPE

HEADER BROKEN DOWN!

0148 — ROM SIZE

0149 — RAM SIZE

014A — DESTINATION CODE

014B — OLD LICENSEE CODE

HEADER BROKEN DOWN!

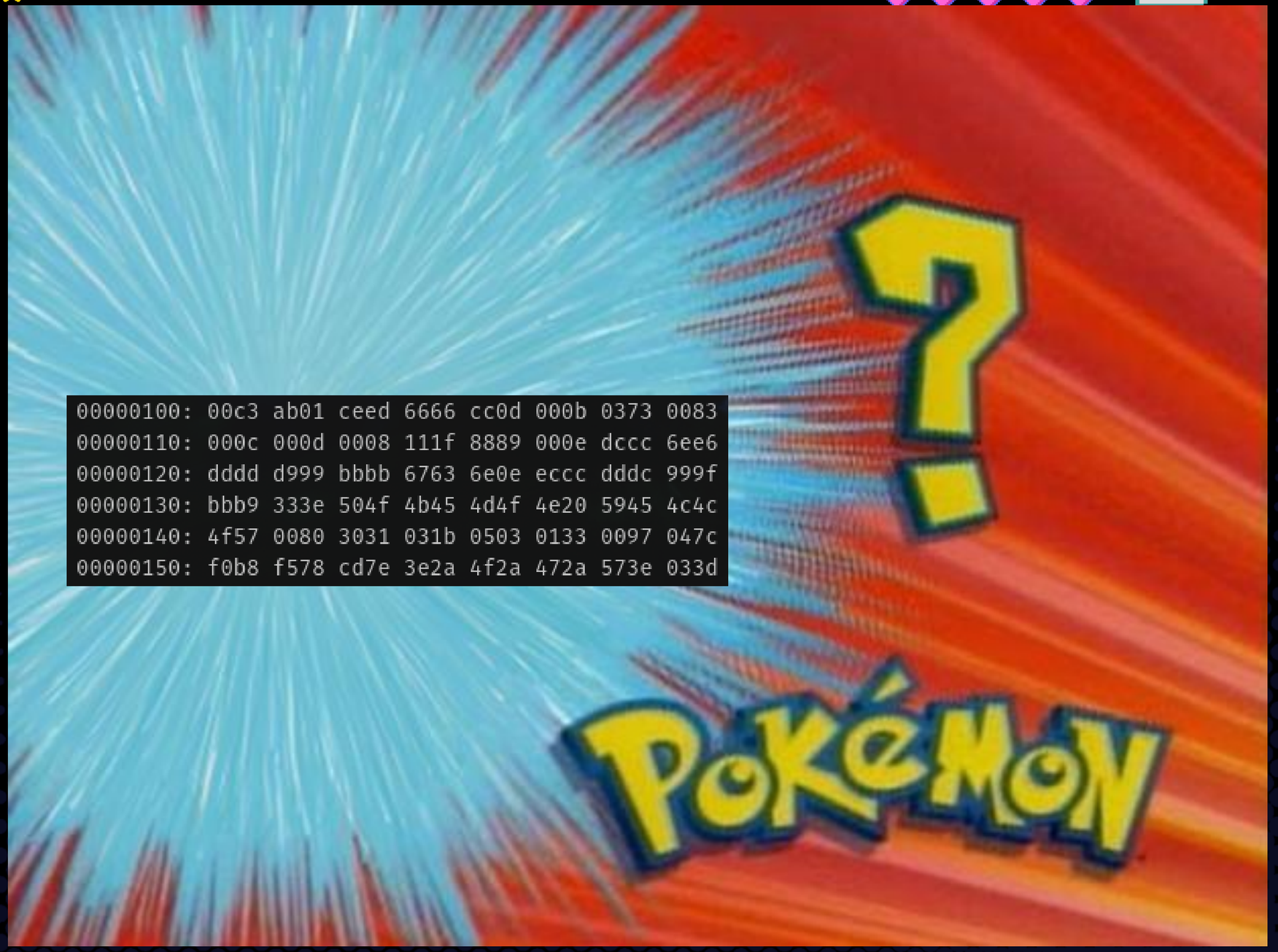
014C — MASK ROM VERSION NUMBER

014D — HEADER CHECKSUM

014E-014F — GLOBAL CHECKSUM

MENU 01 07 12

WHO’S THAT POKEMON?



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

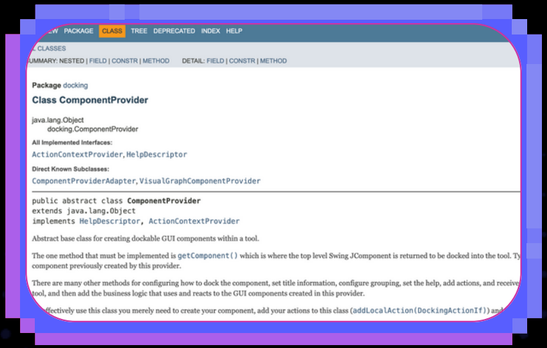
WRITING A PLUGIN: WHAT DO WE NEED?

- 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

MENU 01 07 12

EXAMPLES!

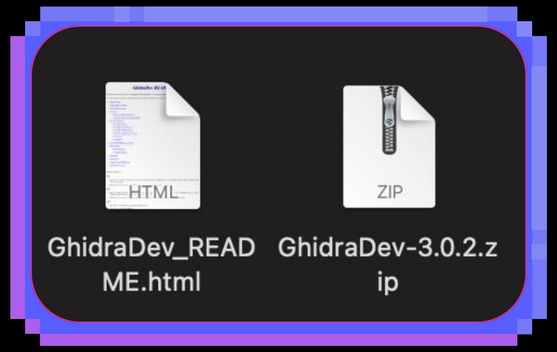
- Ghidra provides quite a few examples for us
- Plugin examples
- Scripting examples
- located in <ghidra_dir>/Extensions/Ghidra/

MENU

WRITING THE PLUGIN

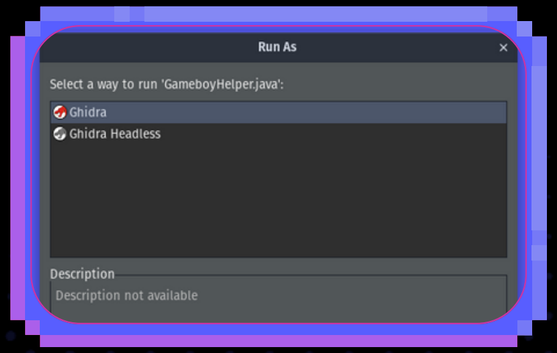
TEAM ECLIPSE

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!

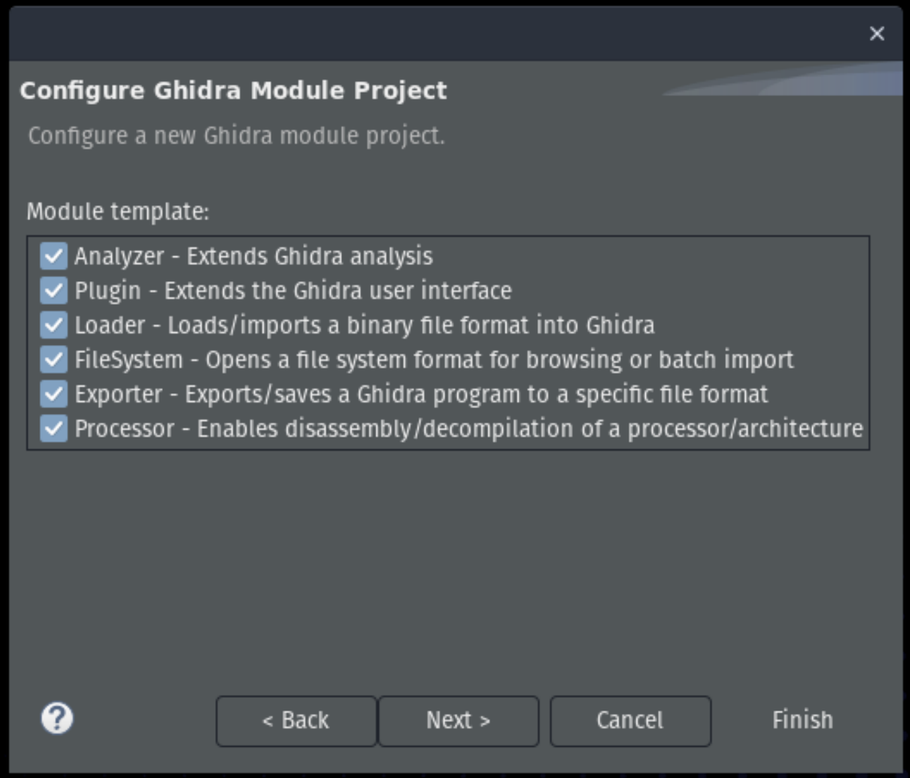


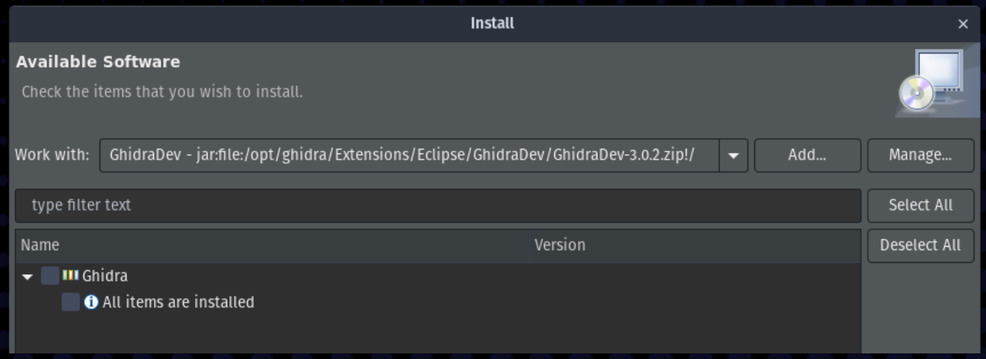
Debugging is a go

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

LEVERAGING GHIDRADEV

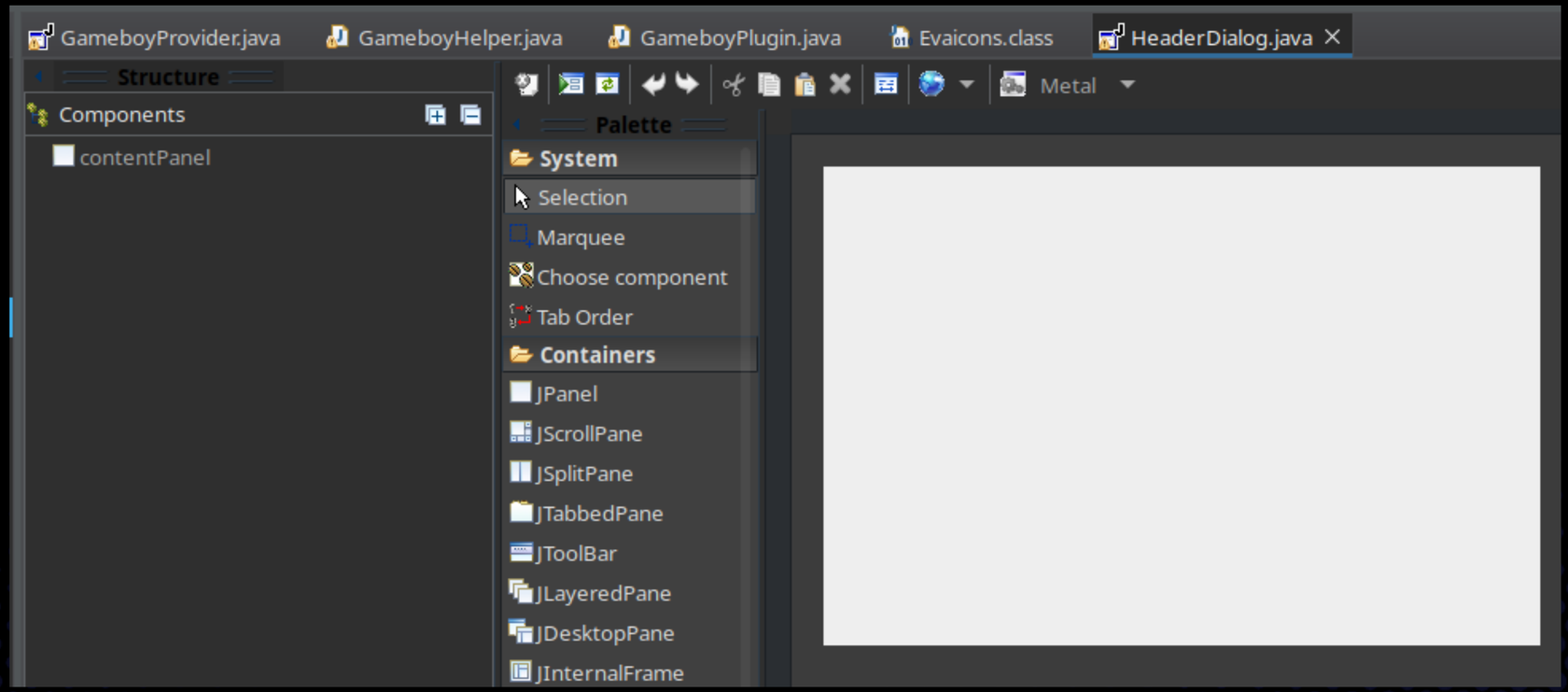
located in <ghidra_dir>/Extensions/Eclipse/GhidraDev/





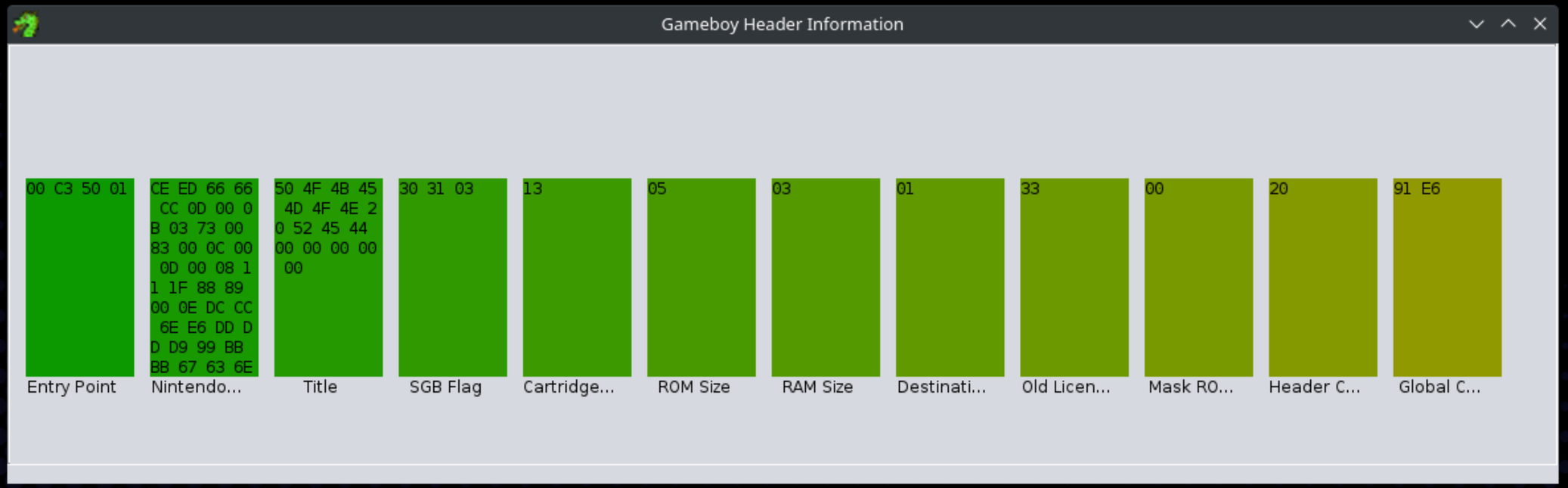
WORKING WITH UI STUFF

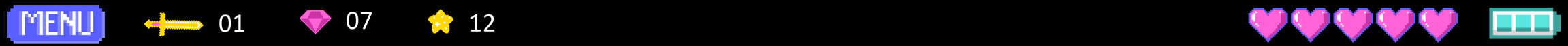
WindowBuilder!



MENU 01 07 12

MY FIRST UI ATTEMPT





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 COM

- This is wher
- Allows you t

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

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

```
to.

is not desired)

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

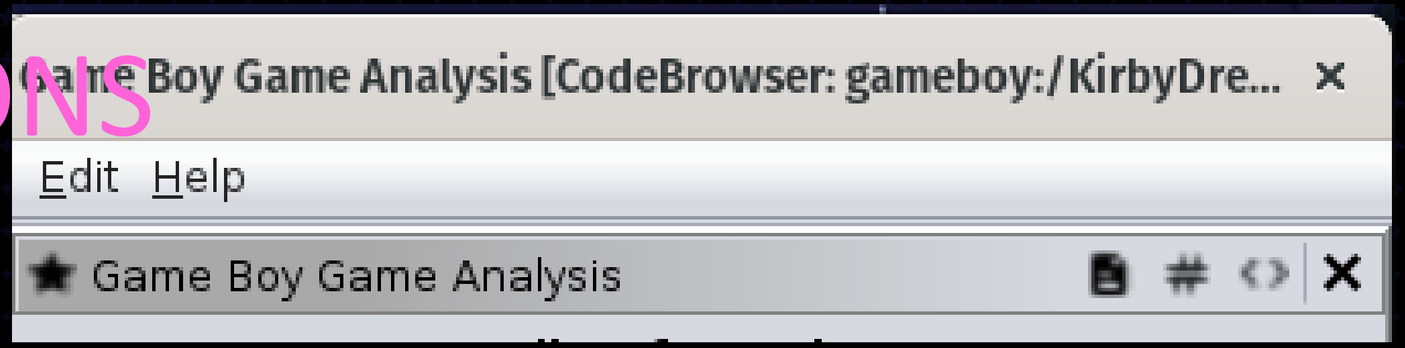
PRO
COM

- This is whe
  sort of stuf
- You initializ

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

```
ion("GameBoyHelp", "HELPME");

ner) {
```

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(ActionContext 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);
```


HELPER COM

- Helper co
- Keep stat
- Leverage

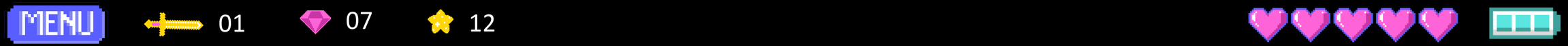
```java
public class GameboyHelper {
    private static PluginTool tool = null;
    private static FlatProgramAPI api = null;

    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;
}
```



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  OW..01.....3...|
00000150: f0b8 f578 cd7e 3e2a 4f2a 472a 573e 033d  ...x.~>*O*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  ................
```


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 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 52
```

```java
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>();

}
```

MENU 01

CONVERT ATION

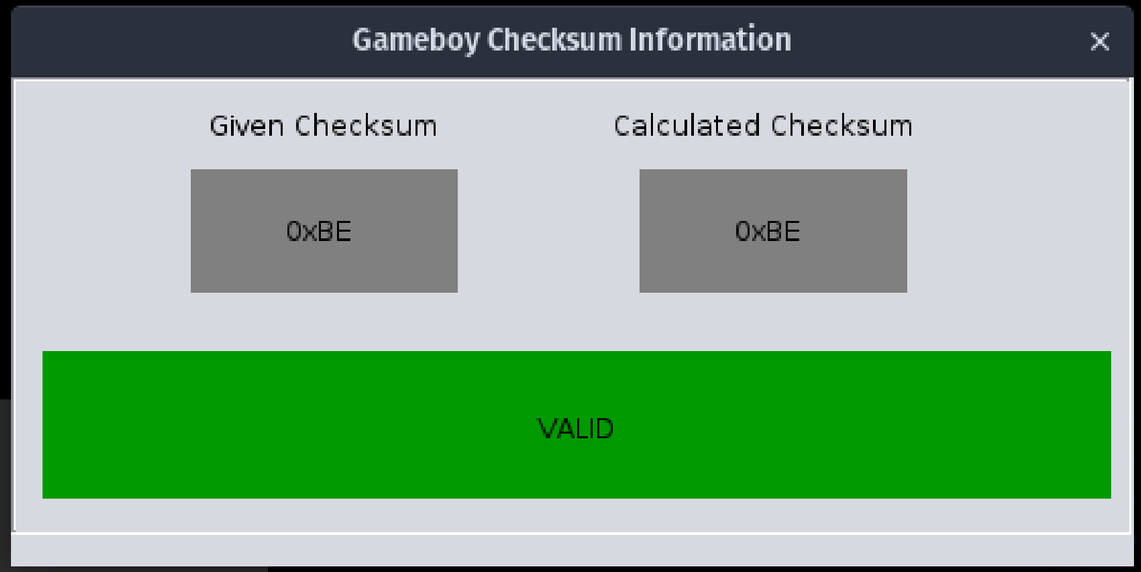
```
newLicensees.put("00", "None");
newLicensees.put("01", "Nintendo Research & Development 1");
newLicensees.put("08", "Capcom");
newLicensees.put("13", "EA (Electronic Arts)");
newLicensees.put("18", "Hudson Soft");
newLicensees.put("19", "B-AI");
newLicensees.put("20", "KSS");
newLicensees.put("22", "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");
```

```
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
```

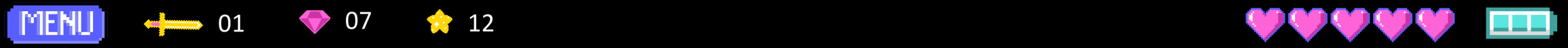
```
.dgeTypes() {
ILY");
;
AM");
AM+BATTERY");
;
ATTERY");
M");
M+BATTERY");
);
RAM");
RAM+BATTERY");
IMER+BATTERY");
IMER+RAM+BATTERY");
;
AM");
AM+BATTERY");
;
AM");
AM+BATTERY");
UMBLE");
UMBLE+RAM");
UMBLE+RAM+BATTERY");
;
ENSOR+RUMBLE+RAM+BATTERY");
 CAMERA");
 TAMA5");
;
AM+BATTERY");
```

VERIFYING HEADER INFO

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

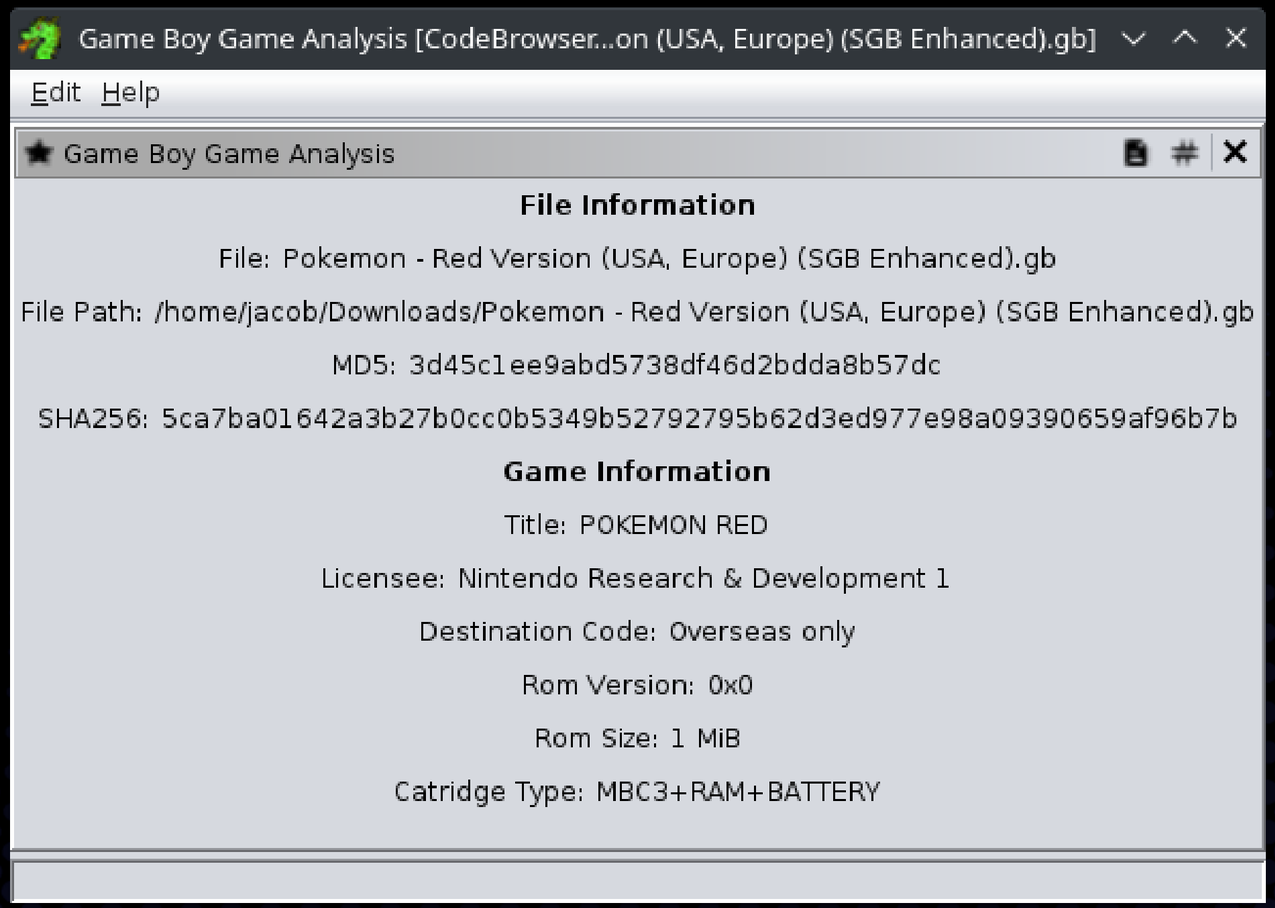



```java
public static int calcHeaderChecksum() {
    int checksum = 0;
    try {
        for (int addr = 0x134; addr <= 0x14C; addr++) {
            checksum = checksum - api.getByte(api.toAddr(addr)) - 1;
        }
    } catch (MemoryAccessException e) {
        e.printStackTrace();
    }

    headerChecksum = checksum & 0xFF;
    return headerChecksum;
}
```



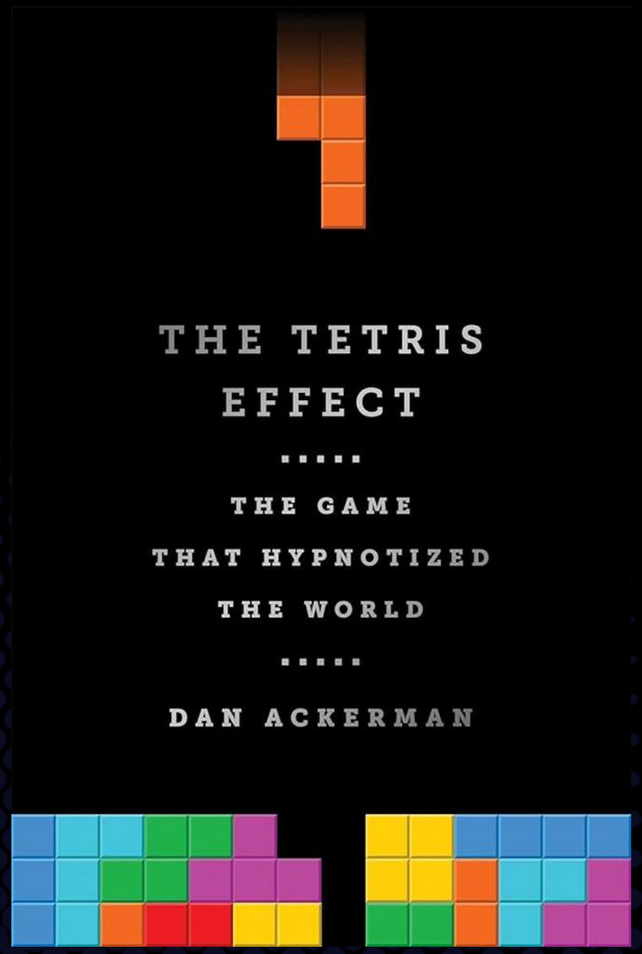
GOOD INFO? GOOD!

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?

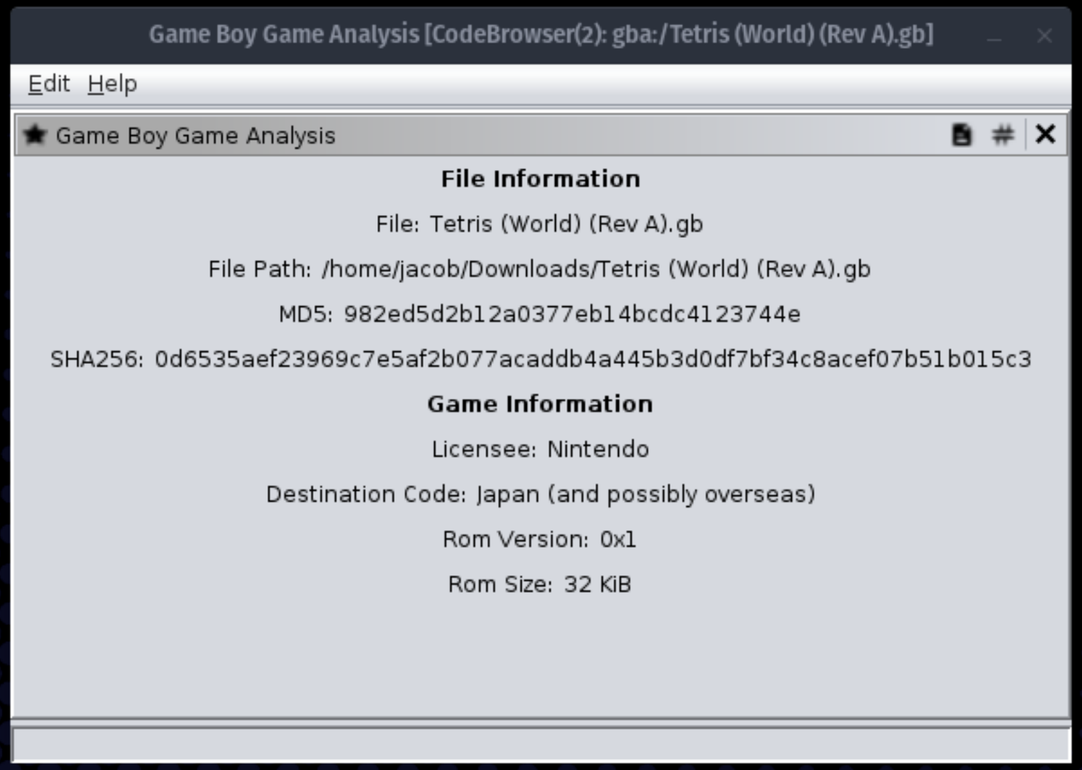


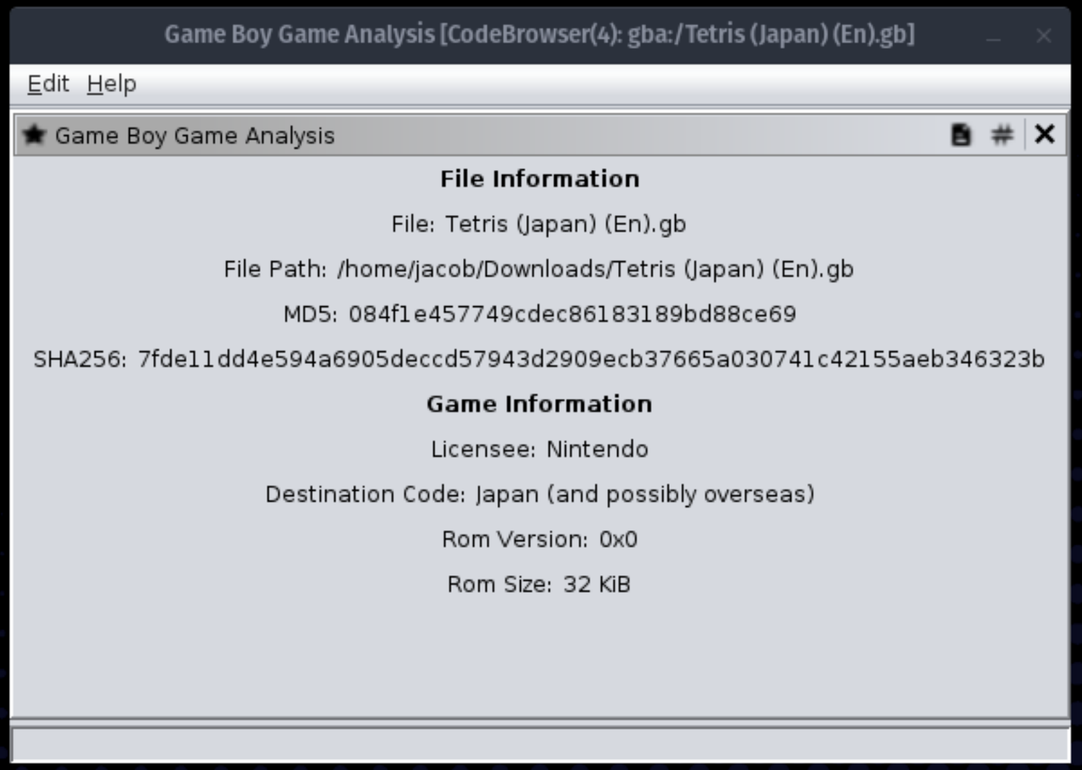
MENU 01 07 12

VARIOUS VERSIONS OF TETRIS

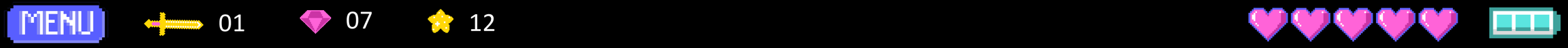
MENU 01 07 12

VARIOUS VERSIONS OF TETRIS





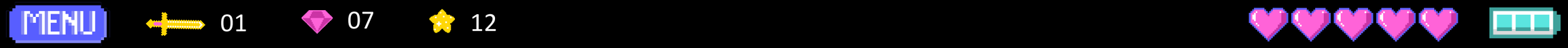

SIGN IN
NEXT USE CASE: I/O



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
```



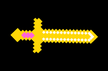
TO THIS ...

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");
```


01

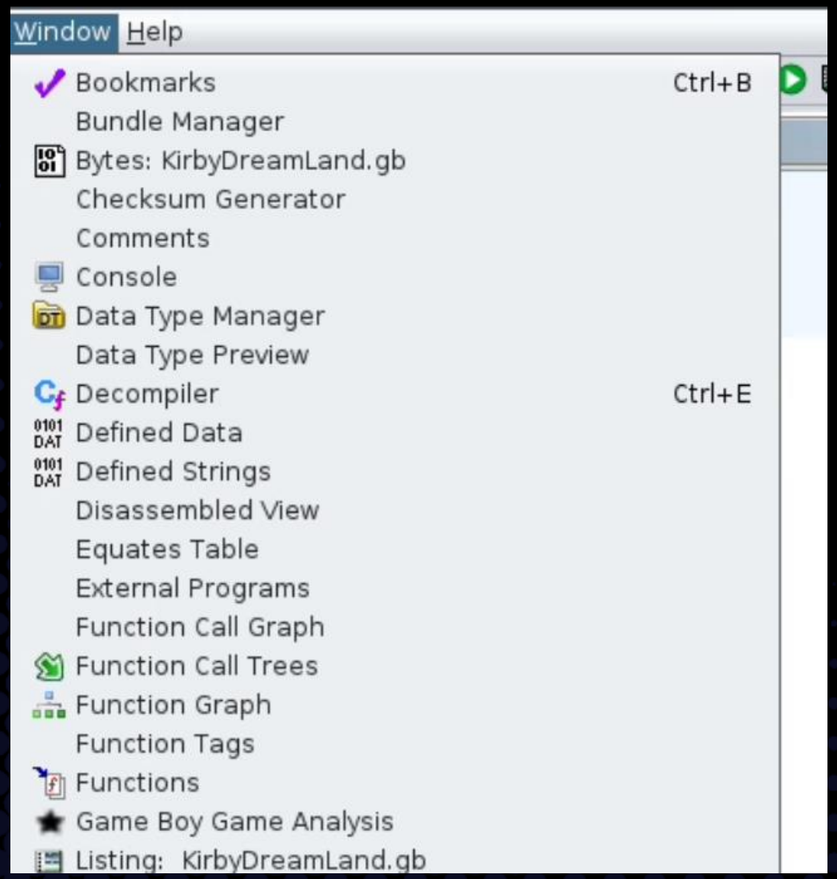
07

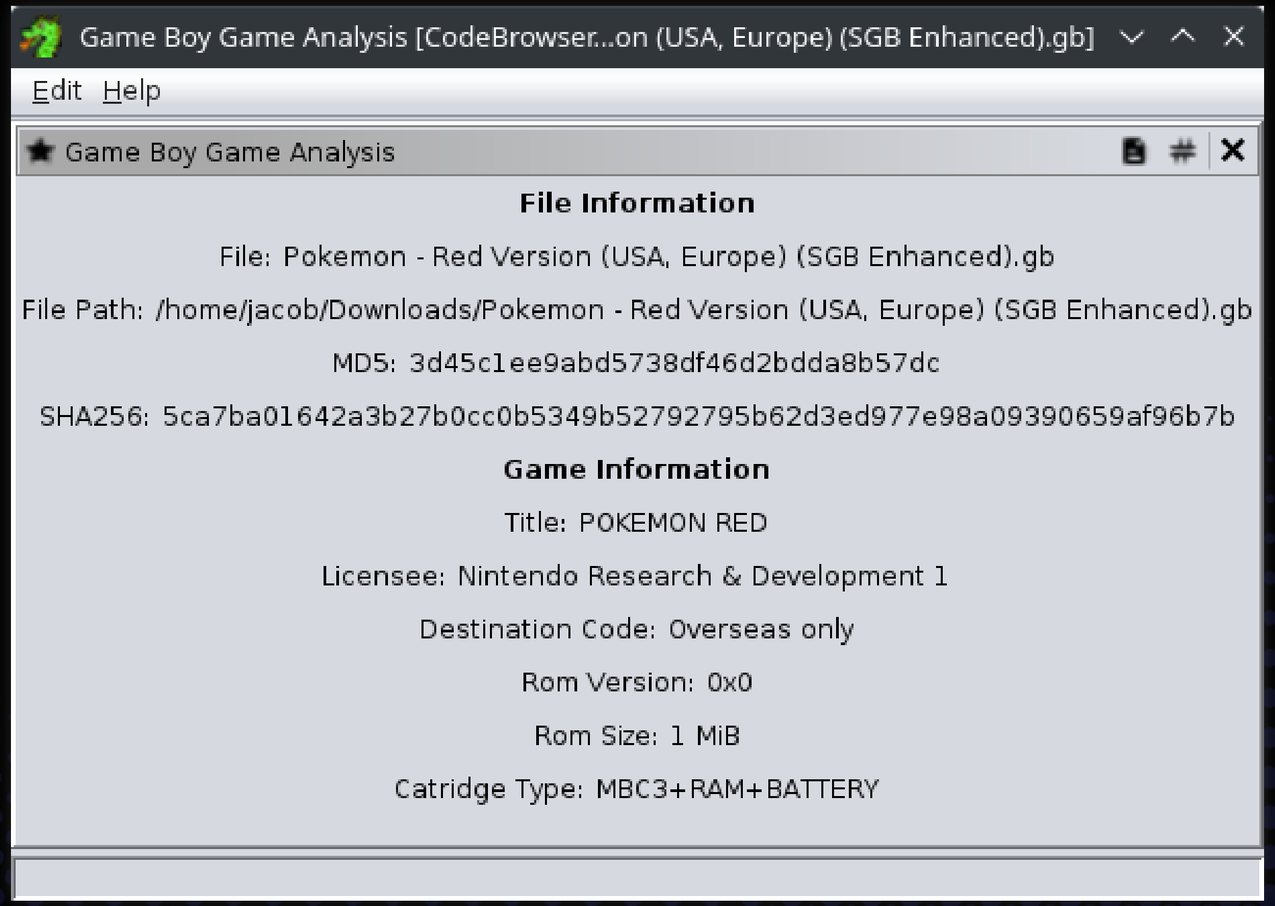
12


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);
```

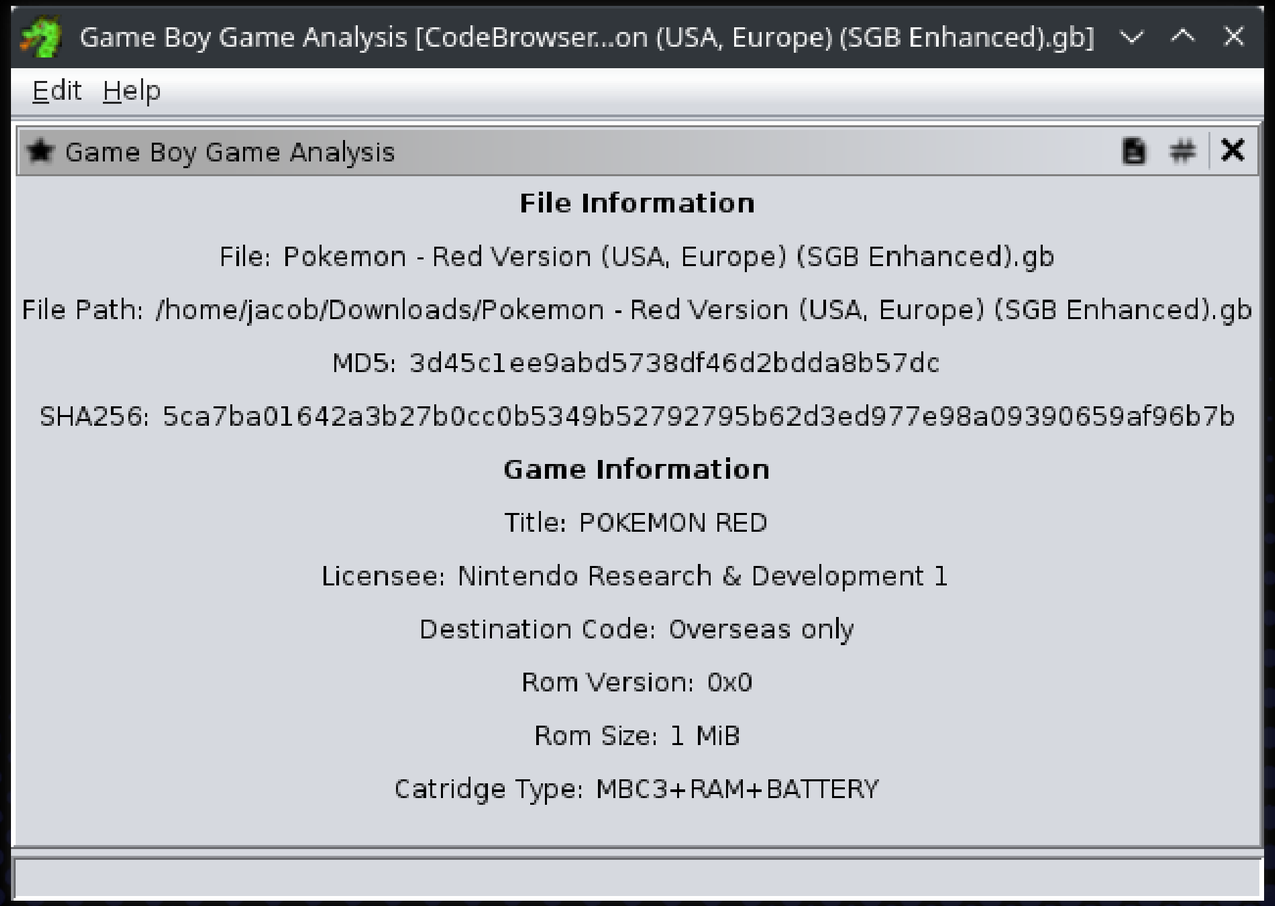
FINISHED (?) PRODUCT!





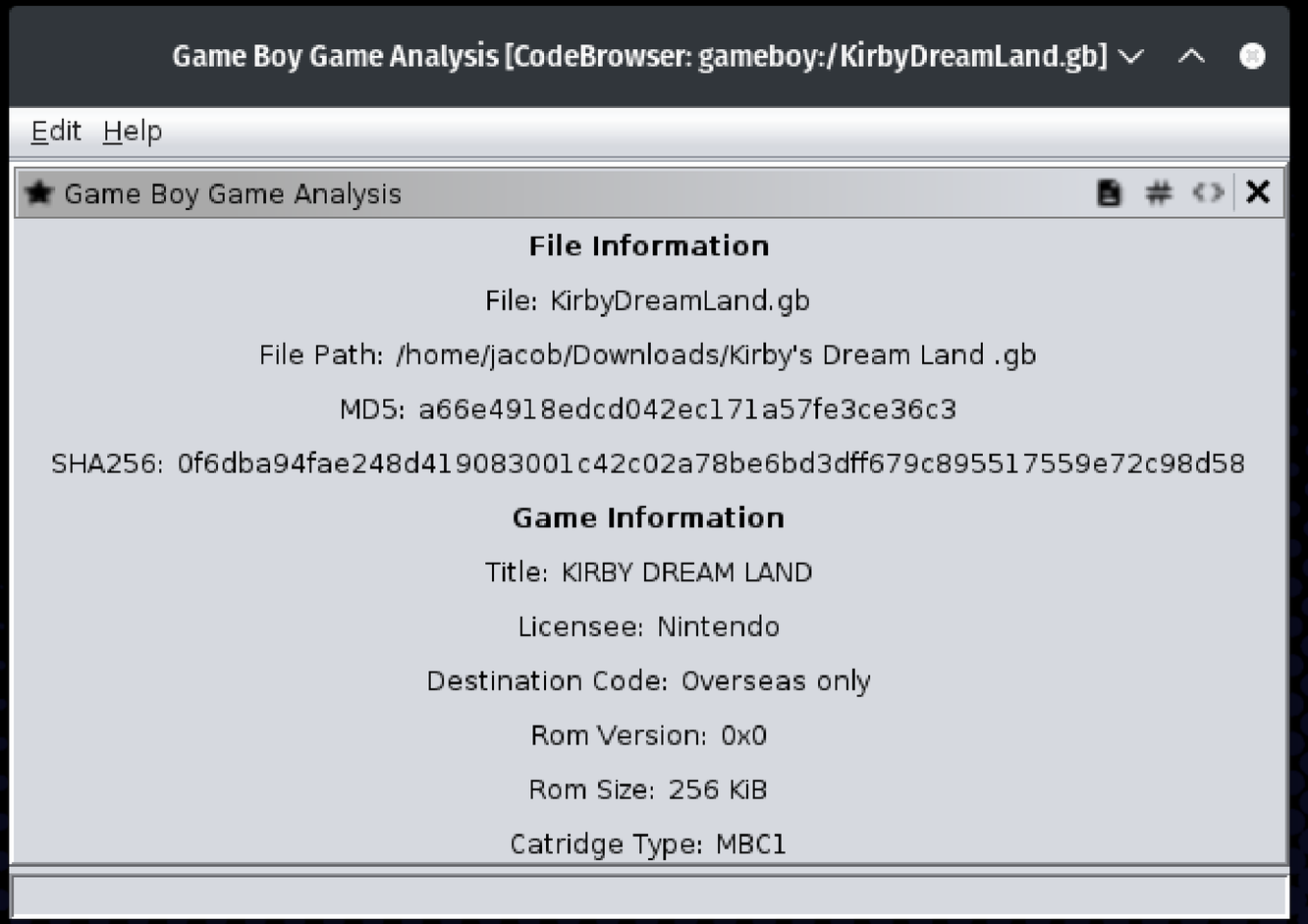
MENU 01 07 12

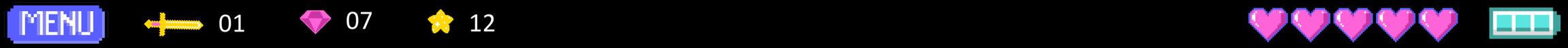
POKEMON!



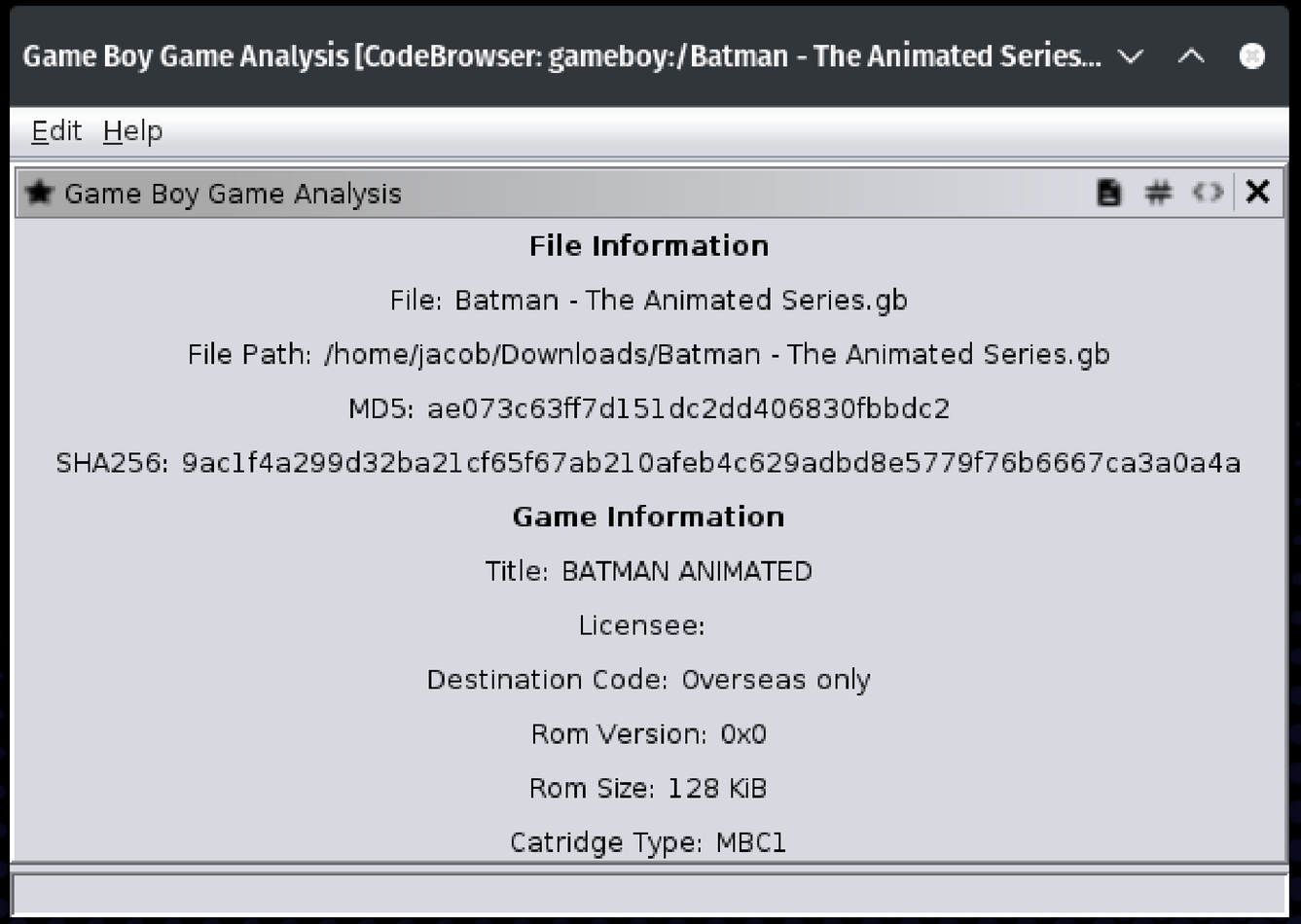
MENU 01 07 12

KIRBY’S DREAM LAND




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
Catridge Type: MBC1


01

07

12

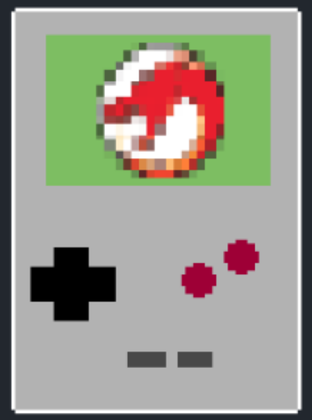

FULL SOURCE CODE?

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.


01

07

12


./UTILS

.github > workflows > ! build.yml

```yaml
name: Build GBAnalyzer

on: [ push ]

env:
  GHIDRA_VERSION: "11.0"
  GHIDRA_BUILD_DATE: 20231222
  GHIDRA_SHA256: f1f240f91cf6b1dffc9a4148384ee3c6b269a8ae27c6f981577973e00043ad94
  GHIDRA_INSTALL_DIR: /home/runner/ghidra

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout sources
      uses: actions/checkout@v4
    - name: Setup Java JDK
      uses: actions/setup-java@v4
      with:
        distribution: 'microsoft'
        java-version: '17'
    - name: Install Ghidra
      run: .github/utils/install_ghidra.sh
    - name: Setup Gradle**Full Changelog**: https://github.com/latonis/GBAnalyzer/commits/20240612
      uses: gradle/actions/setup-gradle@v3
    - name: Build with Gradle
      run:  gradle -PGHIDRA_INSTALL_DIR=/home/runner/ghidra
    - uses: actions/upload-artifact@v4
      with:
        name: "GBAnalyzer_Plugin_${{env.GHIDRA_VERSION}}"
        path: dist/*GBAnalyzer.zip
        if-no-files-found: 'error'

```

latonis/GBanalyzer


01

07

12


LINKS!

Available on GitHub!

latonis/GBanalyzer

0x00