DraconicGB: Creating a Game Boy Emulator

David Soler Bartomeu
Grau d'Enginyeria Informàtica
Àrea de Videojocs

Gisela Vaquero Juanola
Joan Arnedo Moreno

06/06/2020

FITXA DEL TREBALL FINAL

Títol del treball:	*DraconicGB: Creating a Game Boy emulator*
Nom de l'autor:	*David Soler Bartomeu*
Nom del consultor/a:	*Gisela Vaquero Juanola*
Nom del PRA:	*Joan Arnedo Moreno*
Data de lliurament (mm/aaaa):	*06/2020*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Àrea de Videojocs*
Idioma del treball:	*Anglès*
Paraules clau	*Videojocs, Emulador, C++*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

En aquest treball s'explica el desenvolupament d'un emulador de la consola Game Boy de Nintendo. Primer de tot es repassa la història de la consola i dels emuladors existents i després s'explica la metodologia emprada, que consisteix en la realització d'una àmplia recerca per a informar-se de l'estat de l'art actual així com de l'arquitectura de la consola i emuladors existents tot seguit de la definició de l'abast del projecte: La creació d'un emulador funcional de Gameboy sense so ni guardat d'estats.

El resultat final ha estat satisfactori i s'ha pogut crear un emulador que pot carregar ROMs, permet veure per pantalla el resultat a l'usuari i aquest pot jugar als jocs. De la mateixa manera, l'emulador ofereix capacitats de debugging bàsiques i una característica addicional de selecció de paletes de color.

Com a conclusió s'ha vist que la realització d'un projecte d'aquest tipus es tracta d'una tasca difícil i que requereix molta recerca, però donat un abast realista es tracta d'un gran projecte d'aprenentatge en el funcionament de hardware més antic, i enriquidor a l'hora d'entendre una mica més com funcionen els emuladors. Es plantegen millores de futur com afegir la possibilitat de guardar l'estat de l'emulador, portar l'emulador a altres plataformes o afegir funcionalitat d'àudio.

Abstract (in English, 250 words or less):

This work explains the development of an emulator of Nintendo's Game Boy console. First of all, the history of the console and the existing emulators is reviewed and then the methodology used is explained, which consists of carrying out extensive research to find out the current state of the art as well as the architecture of the console and existing emulators followed by the definition of the scope of the project: The creation of a functional Gameboy emulator without sound or state save.

The end result has been satisfactory and it has been possible to create an emulator that can load ROMs, allows the user to see the result on the screen and the user can play the games. Similarly, the emulator offers basic debugging capabilities and an additional color palette selection feature.

In conclusion, carrying out such a project is a difficult task and requires a lot of research, but given a realistic scope it is a great learning project and it allows us to understand a little bit more how emulators work. Future improvements are proposed such as adding the ability to save the status of the emulator, bringing the emulator to other platforms, or adding audio functionality.

Contents

1. Introduction **10**
1.1. Context and justification of the Work 10
1.2. Goals of the Work 10
1.3. Approach and method followed 11
1.4. Work Plan 11
1.4.1. PAC1 - Disseny del videojoc 11
1.4.2. PAC2 - Versió Parcial 12
1.4.3. PAC3 - Versió Jugable 12
1.4.4. PAC4 - Versió Final 13
1.4.5. PAC5 - Defensa 13
1.5. Brief summary of products obtained 13
1.6. Brief description of the other chapters of the report 14
2. History of Emulation **15**
2.1. History of the gameboy console 15
2.1.1. The First Gameboy 15
2.1.2. The Link Cable 16
2.1.3. Kirby 16
2.1.4. The Super Gameboy 16
2.1.5. The Virtual Boy 16
2.1.6. Pokémon 17
2.1.7. The Gameboy Pocket 17
2.1.8. The Gameboy Light 17
2.1.9. The Gameboy Color 18
2.2. History of Gameboy Emulators 19
2.2.1. Introduction 19
2.2.2. GB SIM 19
2.2.3. Virtual Gameboy 19
2.2.4. GB 97 20
2.2.5. Wzonkalad 20
2.2.6. AmiGameboy 20
2.2.7. GodBoy 20
2.2.8. No$GMB 21

2.2.9. Current History 21

3. Emulator project 22

3.1. Emulator project description 22

3.2. State of the art 23

3.2.1. Visual Boy Advance 23

3.2.2. BizHawk 23

3.2.3. RetroArch 23

3.2.4. gbemu 23

3.2.5. Gearboy 23

3.2.6. Gameboy Emulator 24

3.2.7. Coffee GB 24

3.2.8. PyBoy 24

3.2.9. Cinoop 24

3.2.10. BGB 24

4. Technical Design 25

4.1. Game Boy Architecture 25

4.1.1. Processor 25

4.1.2. PPU 26

4.1.3. Background Layer 26

4.1.4. Window Layer 26

4.1.5. Sprite Layer 27

4.1.6. Interrupts 27

4.1.7. Audio 27

4.1.8. Cartridge 28

4.1.9. Memory 28

4.2. Programming Language 29

4.3. Tools 29

4.3.1. Microsoft Visual Studio 29

4.3.2. Sublime Text 3 29

4.4. Libraries 30

4.4.1. OpenGL 30

4.4.2. SDL 30

4.4.3. Dear ImGui 30

4.5. Emulator Architecture 31

4.5.1. Draconic Emulator 32
4.5.2. Base Hardware 32
4.5.3. Draconic CPU 32
4.5.4. Registers 32
4.5.5. Draconic Memory 32
4.5.6. Draconic Cartridge 32
4.5.7. Draconic State 33
4.5.8. Draconic GPU 33
4.5.9. Input Manager 33
4.5.10. Timer Manager 33
4.5.11. Interrupt Manager 33

5. User Manual 34

5.1. Launching the emulator 34
5.2. User interface 34
5.2.1. Global Layout 34
5.2.2. Main Screen 35
5.2.3. Registers Panel 35
5.2.4. GPU Debugger 36
5.2.5. Memory Viewer 36
5.3. Controls 37
5.4. ROM loading 37
5.5. Additional Features 37
5.6. Requirements 38

6. Conclusions 39

6.1. Development Process 39
6.2. Missing Features 39
6.3. Goals 40
6.4. Planning and Methodology 40
6.5. Future Work 40
6.6. Final Thoughts 41

7. Glossary 42

8. Bibliography 43

1. Introduction

1.1. Context and justification of the Work

Emulators are an interesting topic, they allow users to simulate a piece of hardware without the need to have it themselves even if it is no longer available. At the same time emulators allow developers to better understand software created using said hardware in order to see how such software works or even to modify it and upgrade it.

The Gameboy is one of the most well known consoles and at the same time it is simple enough for a work like this to be able to happen.

There is not a lot of information regarding the topic of emulator development readability available to the public and creating a console emulator from scratch is a really good learning experience for a game developer in order to understand how consoles work.

In this case, choosing the Gameboy has been a combination of it being one of the most influential consoles of all time and at the same time its hardware being simple enough for a single person to be able to develop an emulator for it. is a really good learning experience for a software developer.

1.2. Goals of the Work

This work has two main goals. First, the creation of an emulator of the Nintendo Gameboy which can load game ROMs and play them. Second, to be a learning experience and allow the author as well as other people who read this work to learn more about emulator development.

1.3. Approach and method followed

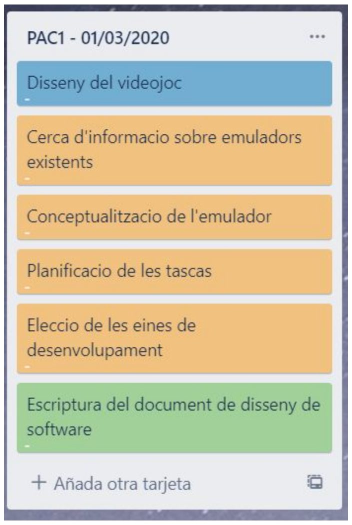
The main method to develop this work has been that of first gathering the necessary information doing research, then planning how the emulator will be developed and finally acting upon this plan. During the course of the development the whole plan has not been able to be followed exactly but more research has been conducted as the development has progressed and problems have arisen.

Developing a game product can have a more concise plan such as first creating a game design document, then implementing game mechanics, level design and finally the game systems such as AI, pathfinding, 3D rendering etc. However, emulator development is still a very unknown field, so fixing a robust plan has not been possible and instead a more flexible approach has been deemed better for this task.

1.4. Work Plan

The plan designed to accomplish this task is the following (titles have been maintained in original language as this was the name of those deliverables in the subject).

1.4.1. PAC1 - Disseny del videojoc


PAC1 - 01/03/2020

Disseny del videojoc

Cerca d'informacio sobre emuladors existents

Conceptualitzacio de l'emulador

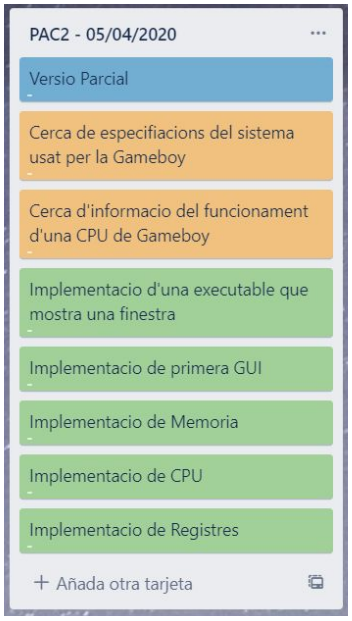
Planificacio de les tascas

Eleccio de les eines de desenvolupament

Escriptura del document de disseny de software

The first PAC has been centered around finding information about existing emulators and planning the following tasks. The development tools have also been chosen and more research has been conducted. This fist stage is a planning stage.

1.4.2. PAC2 - Versió Parcial


PAC2 - 05/04/2020

Versio Parcial

Cerca de especifiacions del sistema usat per la Gameboy

Cerca d'informacio del funcionament d'una CPU de Gameboy

Implementacio d'una executable que mostra una finestra

Implementacio de primera GUI

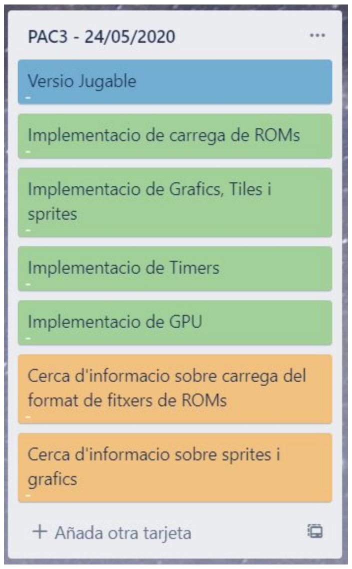
Implementacio de Memoria

Implementacio de CPU

Implementacio de Registres

The work on the second part has been to research how the hardware of the game boy functions as well as its technical specifications. In a practical level in this PAC a program that opens a window with basic GUI should be implemented and in a basic level Memory, CPU and registers should also be implemented.

1.4.3. PAC3 - Versió Jugable


PAC3 - 24/05/2020

Versio Jugable

Implementacio de carrega de ROMs

Implementacio de Grafics, Tiles i sprites

Implementacio de Timers

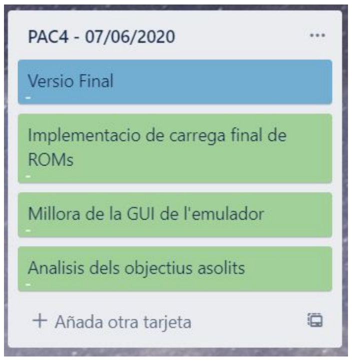
Implementacio de GPU

Cerca d'informacio sobre carrega del format de fitxers de ROMs

Cerca d'informacio sobre sprites i grafics

In this deliverable ROMs should be able to be loaded into the emulator and graphics, such as sprites and tiles also implemented. A timer system should be implemented as well as the GPU Component designed to draw everything on the screen. This deliverable should then also contain research regarding the above mentioned topics.

1.4.4. PAC4 - Versió Final


PAC4 - 07/06/2020

Versio Final

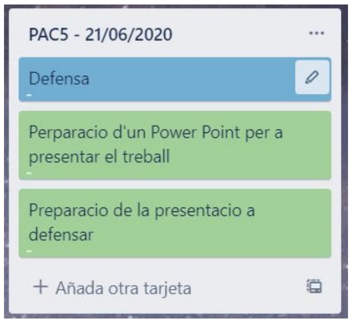
Implementacio de carrega final de ROMs

Millora de la GUI de l'emulador

Analisis dels objectius asolits

Here ROMs should finally be able to be selected by the user and the GUI should be final as well. Finally an analysis of the goals should be performed to see if they have been achieved.

1.4.5. PAC5 - Defensa


PAC5 - 21/06/2020

Defensa

Perparacio d'un Power Point per a presentar el treball

Preparacio de la presentacio a defensar

This deliverable relates to the presentation and defense of the work. As such a powerpoint presentation should be created and a video of presenting the work done should also be created.

1.5. Brief summary of products obtained

The final product has been a Gameboy emulator running on Windows that is able to run ROM files. The emulator supports graphics as well as video output but lacks audio and save/load state functionality.

With the emulator one can play commercial games given that a dump of them is performed as well as homebrew ROMs created by other users.

1.6. Brief description of the other chapters of the report

Chapter 2 will talk about the history of the Nintendo Gameboy console as well as a little bit of the history of Nintendo Gameboy emulators.

Chapter 3 will describe in more detail the scope of the emulator project and show some of the emulators that have been used as reference.
Chapter 4 will go over the technical design of the emulator as well as the tools and libraries used to create it.

Chapter 5 will be the user manual. In this chapter the emulator functionality will be described as well as additional features added to it and how to run it.

Chapter 6 will show the conclusion of this report as well as possible future work.

Chapter 7 will be a glossary of terms used during this work.

Chapter 8 will contain the bibliography of sources consulates during the development of the project as well as during the writing of this report.

2. History of Emulation

2.1. History of the gameboy console

2.1.1. The First Gameboy

The gameboy was the first portable console developed by Nintendo. It was first released in Japan on April 21 of the year 1989 and steadily became one of the most popular games consoles of all time.
We can say that Gunpei Yokoi could be considered the father of the gameboy as he was one of the most innovatives employees at Nintendo. The creation of inventions such as the Ultra Hand or the Game & Watch impressed Nintendo's president at that time so he was allowed to continue inventing new products. One of such inventions is the D-pad on the Gameboy which would later become a trademark on all game controllers.

One of the requirements for the Gameboy was that it needed to be designed to look like a toy but feel like a computer. This was the core premise. In order to fulfil such goal the development team took inspiration on certain aspects of the NES (Nintendo Entertainment System) and the Game & Watch and combined it with a robust build in order for it to be able to sustain a lot of damage. One final aspect to be considered is that the team wanted the battery to last long so that increased the build size.

The final specs of the Gameboy resulted in a device that was a 8-bit portable console that could display 4 shades of green on a 160x144 pixel display. It was powered by 4 AA batteries (less than their competitors) and its retail price was also lower than the competition.

It was released in North America the 14th of June of 1989 with launch titles such as Tetris (which came with the console) and Super Mario Land, resulting in an astounding success.

2.1.2. The Link Cable

In 1991 the Link Cable was introduced to the market which allowed users to connect two Gameboy systems in order to play together, this would allow games like Tetris to be played in multiplayer, but the relevance of the Link Cable would come later with the launch of the Pokémon games.

2.1.3. Kirby

In 1992 Kirby debuted on the Gameboy, this was interesting because other IPs (Intellectual Properties) like Mario and Zelda were established on Nintendo home consoles. But the Kirby series started on the Gameboy and became a long-live IP for nintendo.

2.1.4. The Super Gameboy

Another accessory related to the Gameboy launched in 1994 and it was the Nintendo Super Game Boy, a cartridge adapter which allowed Gameboy games to be played on the SNES (Super Nintendo Entertainment System).

2.1.5. The Virtual Boy

On August 14th of 1995 a revolutionary device that would go down in history as one of Nintendo's greatest failures was released. This was the Virtual Boy, a handheld system that simulated a 3D view of the games such as being in virtual reality. Although today virtual reality has been a massive success those ideas were too much ahead of its time and the technology for a pleasant 3D viewing experience was still not there and thus resulted in the commercial failure of the Virtual Boy.

2.1.6. Pokémon

If one year has to be remembered that is 1996. On the 27th of February of 1996 Pokémon was released for the first time in Japan. It would still take 2 years until 1998 for the game to release in North America. But this game became one of the best-selling games on the Nintendo Game Boy and introduced a lot of people to the world of videogames to the point that today almost everyone has heard the name Pikachu, the mascot of the Pokémon games. With the release of Pokémon the sales of the Link cable also soared, as it allowed players to trade Pokémons with one another and take part in multiplayer battles.

2.1.7. The Gameboy Pocket

Nintendo had a great success with the Gameboy, it had changed the industry forever so they wanted to cash in the opportunity as much as they could so on July 21st of 1996 a new Gameboy version was revealed: the Gameboy Pocket. It was a much smaller and lighter device that could do the same things as the original but better. The display this time was a black and white display which allowed for more contrast and more distinguishable animation and also reduced the blur that happened on the original gameboy on fast paced games. At the same time, instead of 4 AA batteries it was powered by 3 AAA batteries while keeping the 10 hour gameplay battery life. This allowed the gameboy pocket to be much thinner.

2.1.8. The Gameboy Light

In 1998 another revision of the Gameboy Pocket was released this time called Gameboy Light. The main selling point of the device was that the screen was retro illuminated allowing users to play the console on dark spaces. They also changed the batteries to 2 AA batteries which increased the total playtime to 20 hours if the backlight was not being used. Finally they recovered the battery life indicator which was removed on the Gameboy Pocket and many users complained that caused them to lose progress as they did not know when the console was going to turn off due to lack of power.

2.1.9. The Gameboy Color

After 6 month of the release of the Gameboy Light in 1998 Nintendo announced the new Gameboy Color. The main feature of the console is that it was capable of displaying 56 colors simultaneously from a palette of a total of 32768 colors. It was also backwards compatible with all the older Gameboy games so only at launch it had an incredible library of games. With this new console new titles were announced such as Pokemon Gold/Silver, The Legend of Zelda: Link’s Awakening and Super Mario Bros Deluxe.

This new iteration also had better specs with 4 times the amount of RAM and 2 times the amount of video RAM and the CPU could run at double the clock of the original Gameboy.

This new iteration of the Gameboy was also a total success, selling an incredible amount of copies.

2.2. History of Gameboy Emulators

2.2.1. Introduction

Not much is known about Gameboy emulation before 1995. One of the first known Gamboy emulators was Toyboy, an emulator developed in-house by Argonaut Software, a developer for Nintendo, that was stolen from their servers. They used this emulator as a learning tool to be able to learn development for the Gameboy before the hardware was even released. The emulator itself does not run any games as it is intended for development purposes. The platform it ran on was the Commodore Amiga.

2.2.2. GB SIM

Developed by Jens Remstemeier was a DOS emulator that originally only played the game Tetris. It was programmed in TurboPascal and Assembly language and later was updated with a debugger. Although it was quite advanced at the time, it was abandoned due to the ease of use of other emulators such as Virtual Gameboy. Although it’s powerful debugger was ported to C code and later integrated into Virtual Gameboy itself. The development of this emulator allowed Jens Remstemeier to later get a job at Rare to work on porting Donkey Kong Country to the Gameboy Color.

2.2.3. Virtual Gameboy

VirtualGameboy is one of the oldest Gameboy emulators and was first released in 1995 for an unknown platform and was ported to PC in 1996. It was developed by Marat Fayzullin and it is still used and is extremely accurate and versatile. It is considered by many to be the first usable emulator that could support comercial games. The emulator was written in the C language. It can be considered the standard at that time of Gameboy emulation and having the debugger from GBSim this emulator was one of the most powerful and flexible emulators around. One of it’s caveats is that it was a paid emulator so later emulators that were free pushed it into obscurity.

2.2.4. GB 97

An emulator developed by Paul Robson in Assembly Language and released in 1997. For a short time this emulator competed with Virtual Gameboy as an alternative as it was extremely fast and very efficient. It was abandoned in 1998 but the source code was released which allowed other people to learn from its code and expanded the emulation scene even further. But after being abandoned and open sourced no one continued the work on the emulator.

2.2.5. Wzonkalad

Wzonkalad was an emulator developed by Ville Helin in Assembly Language for Amiga systems and released in 1997. It did not have sound emulation for any games. Wzankalad was slightly slower than other emulators on the Amiga but had better game compatibility and allowed the user to change the game speed.

2.2.6. AmiGameboy

Was another gameboy emulator for the Amiga released in 1997. It was written by Juan Anotnio Gomez and had similar compatibility support as Wzonkalad but with sound support. It was originally a paid emulator but was switched to freeware and later abandoned in 1999.

2.2.7. GodBoy

The GodBoy was an emulator for the Atari Falcon created by a group called the Reservoir Gods. It was released in 1997 and had cheat trainers, colored sprites and redone music. These emulators were packed directly with the games which allowed for these recolors and music tweaks. It was coded in Assembly and a total of 7 games were redone using this emulator.

2.2.8. No$GMB

Released in 1997 and developed by Martin Korth No$GMB was a MS-DOS and Windows emulator was really fast due to being written in Assembly language and was really accurate.
It contained a really advanced debugger and it could emulate up to 4 Gameboys at the same time allowing you to hook up these virtual Gameboys together like using a Link Cable in real life.
The emulator had 3 versions: A free version that limited the play time to 5 minus, a 20$ private version and a 750$ comercial version.

2.2.9. Current History

As of today many of the modern Gameboy Advance emulators can also run Gameboy games. And these emulators are available on a lot of devices including PCs, smartphones and even as homebrew for other consoles. There are even Gameboy Emulators that run on browsers using Javascript, showing how technology has advanced these days.

3. Emulator project

3.1. Emulator project description

An emulator is a software application that allows the user to replicate the behavior of a piece of hardware in another one. Due to this specification the creation of such a piece of software is a complex task that requires a great deal of time and effort.

As stated in the introduction this academic work has been focused on the creation of a function Gameboy Emulator. Due to the difficulty of the task the goal of the project is not the creation of the emulator itself, as being a difficult task the project may not be able to be finished or functional but as a learning exercise in the necessary knowledge for the creation of such an emulator.

That being said the main goal of the work has been the creation of an emulator with the following features:

- Memory emulation
- Registers emulation
- Flag emulation
- CPU emulation
- Timer emulation
- Tilesets emulation
- Sprites emulation
- ROM loading capabilities

As an example, the following features are out of the scope of the work:

- Saving and loading states
- Audio emulation
- Precise emulation
- Integrated debugging
- Gamepad support

With this definition it is assumed that the project will be able to be completed in due time with the working emulator being finished at the end of the project.

3.2. State of the art

The following emulators have been taken into account in order to develop the emulator. Some pieces of code from the emulators that are open source have been studied in order to know how to implement some difficult aspects of the emulator.

3.2.1. Visual Boy Advance

http://www.emulator-zone.com/doc.php/gba/vboyadvance.html

Open source Gameboy Advance emulator that also supports Gameboy and Gameboycolor games. It is regarded as one of the best Gameboy Emulators

3.2.2. BizHawk

https://www.emulator-zone.com/misc/bizhawk

Emulator created in order to give support to TAS (Tool-Assisted Speedruns). These types of speedruns are created using the help of an emulator or other tools which the runner uses to optimize the route or perform near impossible frame-perfect tricks.
https://en.wikipedia.org/wiki/Speedrun

3.2.3. RetroArch

https://www.emulator-zone.com/misc/retroarch

Multi system emulator that allows the emulation of a lot of different platforms. It is really complex and highly customizable.

3.2.4. gbemu

https://github.com/jgilchrist/gbemu

gbemu is a Nintendo Gameboy emulator written in C++. It was written as an exercise (and for fun!) so its goals are exploration of modern C++ and clean code rather than total accuracy.

3.2.5. Gearboy

https://github.com/drhelius/Gearboy
Gearboy is a cross-platform Game Boy / GameBoy Color emulator written in C++ that runs on Windows, macOS, Linux, iOS, Raspberry Pi and RetroArch.

3.2.6. Gameboy Emulator

https://github.com/mattbruv/Gameboy-Emulator

An emulator for the first version of the Nintendo Gameboy. Made for higher learning purposes, written from scratch in C++ using the SFML library for windowing & IO.

3.2.7. Coffee GB

https://github.com/trekawek/coffee-gb

Coffee GB is a Gameboy Color emulator written in Java 8. It's meant to be a development exercise.

3.2.8. PyBoy

https://github.com/Baekalfen/PyBoy

Game Boy emulator written in Python

3.2.9. Cinoop

https://github.com/CTurt/Cinoop

A multiplatform Game Boy emulator written in C; currently available for: Windows, OS X, Linux based OSes, Nintendo DS, Nintendo 3DS, Nintendo GameCube, Sony PSP, and Sony PS4.

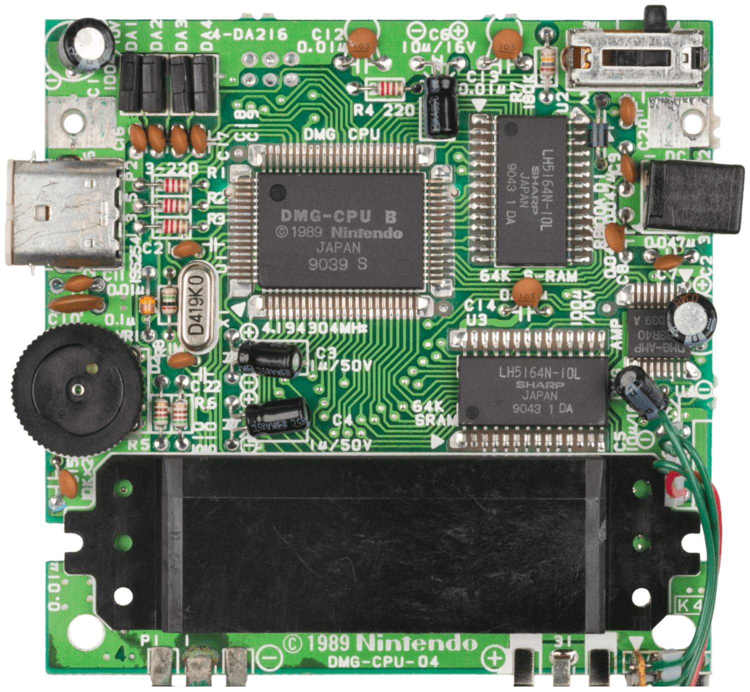
3.2.10. BGB

https://bgb.bircd.org/

BGB is a Gameboy emulator, a program that lets you play Gameboy and Gameboy Color games on a pc. It does this with many features that give a good gameplay experience, such as gamepad support, high quality sound and graphics, smooth Vsync animation, and low latency. In addition, it contains a debugger that lets you analyze/look into the emulation, create cheat codes, and assist in creating and modifying roms.

4. Technical Design

4.1. Game Boy Architecture



https://www.copetti.org/images/consoles/gameboy/motherboard.1f907a9e44df7b7a800894eb3d137c719156bd264419e3767f040487f897664f.png

4.1.1. Processor

The processor of the Game Boy is a Sharp LR35902 which is a mix of the z80 and the Intel 8080 processors and runs at 4.19 MHz.

The processor has eight registers of 8 bit each called A, B, C, D, E, F, H and L and two 16 bit registers called SP and PC. The 8 bit registers are all-purpose registers and the SP is the Stack Pointer registers and the PC is the Program Counter register.

4.1.2. PPU

The graphic calculations are all done by the CPU and then the PPU (Picture Processing Unit) renders them on the screen. The integrated LCD has a resolution of 160x144 pixels and can display 4 shades of gray. The PPU uses the information stored in the VRAM which the CPU has previously setup and using that information renders the game in 3 layers and finally onto the screen.

The PPU uses tiles as the basic building block to display sprites and backgrounds. A gameboy tile is a 8x8 bitmap that is stored on the ROM of the game but they have to be copied previously to the VRAM. In the VRAM the tiles are organized in what's called a Pattern Table.

4.1.3. Background Layer

The background is a 256x256 map containing static tiles however as the screen has a resolution of 160x144 pixels only a portion of the background is displayed at a time. The viewable portion of the background can be moved during gameplay in order to achieve camera panning or a scrolling effect.

4.1.4. Window Layer

The window layer is a 160x144 layer that contains tiles that will be displayed on top of the background and sprites. This layer does not scroll and is usually used to display HUD (Heads Up Display) such as lives or score.

4.1.5. Sprite Layer

The sprites are tiles that can move around the screen and overlap each other and even appear behind background. This is decided using the priority value of the sprite. Sprites have an extra color available which is the Transparent color but due to this they can only display in 3 shades of gray instead of 4. The OAM part of the memory allows us to specify which tiles will be used as sprites as well as containing an index, a position and multiple attributes of the sprite.

The PPU can only display up to 40 sprites per frame and overflowing this will result in the sprites not being drawn.

4.1.6. Interrupts

In order to sync with the PPU the CPU uses a set of interrupts that let it know that the PPU is idle so the corresponding RAM tables can be modified. This is done using the Horizontal Blank interrupt and the Vertical Blank interrupt.

4.1.7. Audio

The audio of the system is processed by the APU (Audio Processing Unit), which is a PSG chip with four channels, each of one reserved to one type of waveform.

Pulse waves are used for melody and sound effects and the APU reserves two channels for two kinds of pulse waves. These channels use one of four different tones. Due to the limited number of channels melody will often be interrupted when playing sound effects during gameplay.

Another channel is allocated to a noise band, which is usually used for percussion or ambient effects.

Finally, the APU allows the usage of 32 custom wave forms to be played in the fourth channel. Each wave is composed of a 4-bit sample that will be played repeatedly. This channel also allows to control its frequency, which can be used to change the pitch of the notes, as well as the volume.

4.1.8. Cartridge

The Game Boy cartridges have a maximum size of 32 KB so a lot has to be done to make the games fit this size. However, some cartridges can use a special Memory Bank Controller to reach bigger sizes such as 1MB. Such cartridges can also include real time clocks and external batteries along with an SRAM, which allows to keep save data.

4.1.9. Memory

The Game Boy RAM is divided into various regions and adds to a total of 32KB of RAM and 16KB of VRAM.

The RAM itself is then divided into the following memory mapping:

Cartridge ROM: This region is divided into various subregions and stores the program of the cartridge made available to the CPU. A special chip on some cartridges switches the banks and allows for different memory regions on the cartridge to be accessible.

Cartridge Header: This section contains data about the game such as the name, creator and other useful info.

Graphics RAM: Also called VRAM, this stores data required to render the different layers and show them on screen using the PPU.

ERAM: This is a small amount of writable memory that is stored on some cartridges and is made available using the cartridge memory bank controllers.

Memory-mapped I/O: Control values to allow the programs to interface with the hardware such as graphics, sound and input.

Working RAM: This is the internal 8KB of RAM which the CPU can use.

Zero-page RAM: A high speed 128 bytes area of the memory to communicate with the Gameboy hardware.

4.2. Programming Language

The language chosen to develop this emulator is C++. This language has been chosen for 2 main reasons. First, it is a high performance language, the performance that one can get programming using C++ is really high and that is a requirement for an emulator. The second reason is the familiarity of the language, being familiar with the language used to develop software is also an important part in order to be able to program fast.

At first the Rust language was also considered in order to be able to learn this modern powerful language but it was dropped in favor of C++ because the task of creating an emulator was already a difficult one and there was no time to have the problems of not only implementing the emulator but also learning a new language.

4.3. Tools

4.3.1. Microsoft Visual Studio

Visual Studio has been chosen as the IDE (Integrated Development Environment) as it offers a really good support for C++ programming as well as an incredible in-depth debugger.

4.3.2. Sublime Text 3

Sublime Text 3 is a text editing program and has been used to take notes and make high level plans regarding the development.

4.4. Libraries

4.4.1. OpenGL

OpenGL is the main rendering library that has been chosen to display the graphics on screen. It has been chosen as it is available on all systems and even low end hardware can run OpenGL applications. DirectX was also an option but then the project was limited to only run on Windows.

4.4.2. SDL

Simple DirectMedia Layer is a cross-platform development library designed to provide low level access to audio, keyboard, mouse, joystick, and graphics hardware. This library has been used to create the window that displays the emulator as well as to receive the input from the user.

4.4.3. Dear ImGui

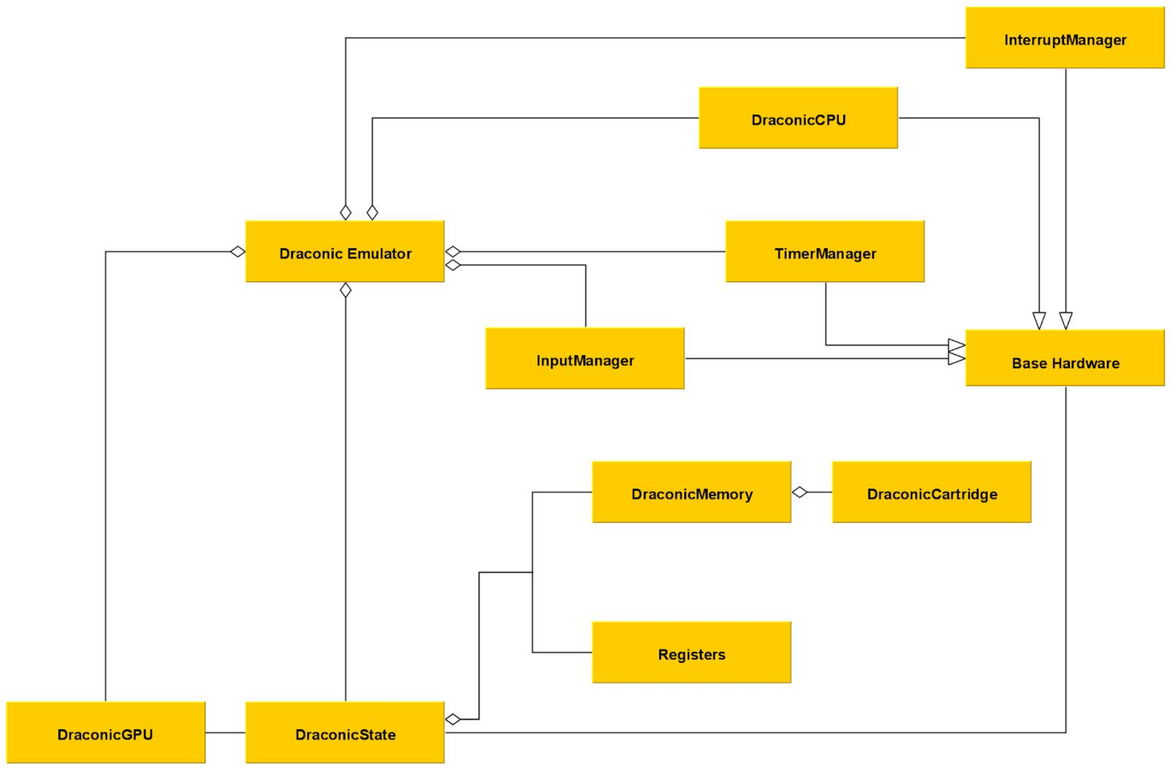
Dear ImGui is designed to enable fast iterations and to empower programmers to create content creation tools and visualization / debug tools (as opposed to UI for the average end-user). It favors simplicity and productivity toward this goal, and lacks certain features normally found in more high-level libraries.

This library has been used to create the Graphical User Interface displayed on the emulator. An immediate GUI has been selected instead of something like Qt as prototyping with an immediate GUI is much faster than using other methods.

4.5. Emulator Architecture

The emulator architecture has changed many times during the development as it was not decided since the start as not enough info was present to make such a decision. This architecture does not pretend to be perfect but is a result of the ongoing development and learning procedure about emulator development. If the emulator had to be redone from scratch surely a different architecture would be used.

This is the final emulator architecture:



4.5.1. Draconic Emulator

This class is responsible for holding the whole application together. It is the starting point of the emulator and handles window creation using SDL, OpenGL context initialization, drawing the GUI using imgui as well as receiving the user input, loading ROMs and running the emulator main loop.

4.5.2. Base Hardware

This is a base class for those classes that need the functionality to init themselves with a pointer to the emulator state.

4.5.3. Draconic CPU

This class is the main core of the emulator, it is responsible for implementing all the different opcodes as well as handling which opcode must be run. It also handles state transformation due to opcode operations.

4.5.4. Registers

This structure holds information relevant to the state of the different emulator base registers.

4.5.5. Draconic Memory

This class represents the RAM of the emulator. It splits the RAM in its different sections VRAM, OAM, WRAM and ZRAM and handles the task of reading and writing to corresponding memory. It also contains the object of the cartridge.

4.5.6. Draconic Cartridge

This class handles the ROM data as well as the ERAM.

4.5.7. Draconic State

This represents the whole state of the emulator. It contains the objects of the registers as well as the memory and other information such as if the CPU is halted, the number of cycles or the CPU clock speed.

4.5.8. Draconic GPU

This class is responsible for drawing the final output on an array buffer and presenting the buffer into the screen. It does so by using an OpenGL texture.

4.5.9. Input Manager

The Input Manager handles the SDL key events, processes them and writes the corresponding values on the memory positions associated with inputs.

4.5.10. Timer Manager

The Timer Manager handles the Game Boy timer. It can be used to update the timer as well as to request a new timer and write it to the corresponding position in memory.

4.5.11. Interrupt Manager

The Interrupt Manager has the job to handle interrupts such as gameboy input button events as well as timer interrupts or LCD interrupts.

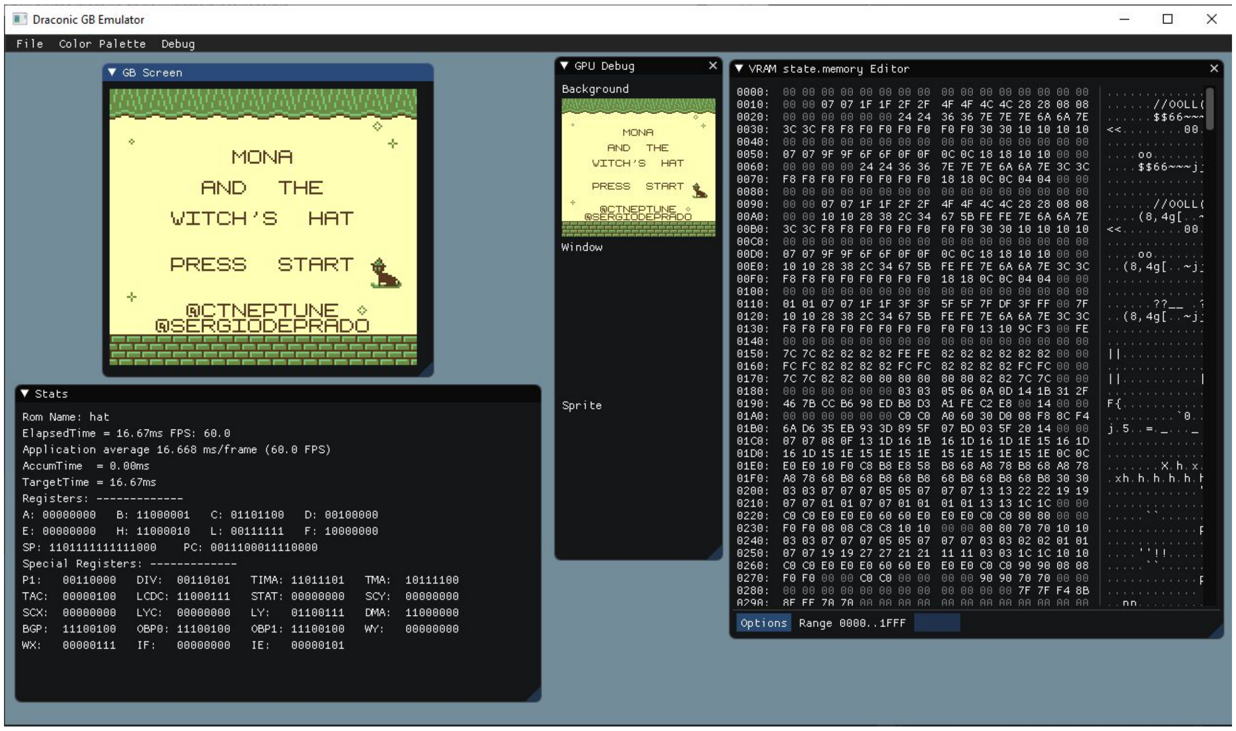
5. User Manual

5.1. Launching the emulator

The emulator is designed to run in x64 Windows machines. To launch the emulator the executable DragonicGB.exe must be run from the Windows explorer or from the command line.

5.2. User interface

5.2.1. Global Layout



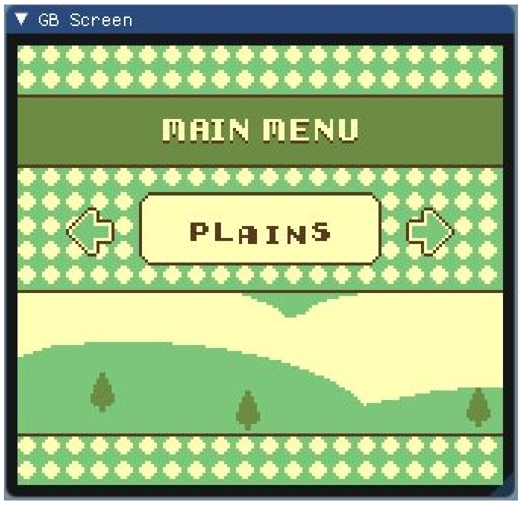
```
Stats
Rom Name: hat
ElapsedTime = 16.67ms FPS: 60.0
Application average 16.668 ms/frame (60.0 FPS)
AccumTime  = 0.00ms
TargetTime = 16.67ms
Registers: -------------
A: 00000000   B: 11000001   C: 01101100   D: 00100000
E: 00000000   H: 11000010   L: 00111111   F: 10000000
SP: 1101111111111000    PC: 0011100011110000
Special Registers: -------------
P1:   00110000   DIV:  00110101   TIMA: 11011101   TMA:  10111100
TAC:  00000100   LCDC: 11000111   STAT: 00000000   SCY:  00000000
SCX:  00000000   LYC:  00000000   LY:   01100111   DMA:  11000000
BGP:  11100100   OBP0: 11100100   OBP1: 11100100   WY:   00000000
WX:   00000111   IF:   00000000   IE:   00000101
```

```
VRAM state.memory Editor
0000:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0010:  00 00 07 07 1F 1F 2F 2F  4F 4F 4C 4C 28 28 08 08
0020:  00 00 00 00 00 00 24 24  36 36 7E 7E 7E 6A 6A 7E
0030:  3C 3C F8 F8 F0 F0 F0 F0  F0 F0 30 30 10 10 10 10
0040:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0050:  07 07 9F 9F 6F 6F 0F 0F  0C 0C 18 18 10 10 00 00
0060:  00 00 00 00 24 24 36 36  7E 7E 7E 6A 6A 7E 3C 3C
0070:  F8 F8 F0 F0 F0 F0 F0 F0  18 18 0C 0C 04 04 00 00
0080:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0090:  00 00 07 07 1F 1F 2F 2F  4F 4F 4C 4C 28 28 08 08
00A0:  00 00 10 10 28 38 2C 34  67 5B FE FE 7E 6A 6A 7E
00B0:  3C 3C F8 F8 F0 F0 F0 F0  F0 F0 30 30 10 10 10 10
00C0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00D0:  07 07 9F 9F 6F 6F 0F 0F  0C 0C 18 18 10 10 00 00
00E0:  10 10 28 38 2C 34 67 5B  FE FE 7E 6A 6A 7E 3C 3C
00F0:  F8 F8 F0 F0 F0 F0 F0 F0  18 18 0C 0C 04 04 00 00
0100:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0110:  01 01 07 07 1F 1F 3F 3F  5F 5F 7F DF 3F FF 00 7F
0120:  10 10 28 38 2C 34 67 5B  FE FE 7E 6A 6A 7E 3C 3C
0130:  F8 F8 F0 F0 F0 F0 F0 F0  F0 F0 13 10 9C F3 00 FE
0140:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0150:  7C 7C 82 82 82 82 FE FE  82 82 82 82 82 82 00 00
0160:  FC FC 82 82 82 82 FC FC  82 82 82 82 FC FC 00 00
0170:  7C 7C 82 82 80 80 80 80  80 80 82 82 7C 7C 00 00
0180:  00 00 00 00 00 00 03 03  05 06 0A 0D 14 1B 31 2F
0190:  46 7B CC B6 98 ED B8 D3  A1 FE C2 E8 00 14 00 00
01A0:  00 00 00 00 00 00 C0 C0  A0 60 30 D0 08 F8 8C F4
01B0:  6A D6 35 EB 93 3D 89 5F  07 BD 03 5F 20 14 00 00
01C0:  07 07 08 0F 13 1D 16 1B  16 1D 16 1D 1E 15 16 1D
01D0:  16 1D 15 1E 15 1E 15 1E  15 1E 15 1E 15 1E 0C 0C
01E0:  E0 E0 10 F0 C8 B8 E8 58  B8 68 A8 78 B8 68 A8 78
01F0:  A8 78 68 B8 68 B8 68 B8  68 B8 68 B8 68 B8 30 30
0200:  03 03 07 07 07 05 05 07  07 07 13 13 22 22 19 19
0210:  07 07 01 01 07 07 01 01  01 01 13 13 1C 1C 00 00
0220:  C0 C0 E0 E0 E0 60 60 E0  E0 E0 C0 C0 80 80 00 00
0230:  F0 F0 08 08 C8 C8 10 10  00 00 80 80 70 70 10 10
0240:  03 03 07 07 07 05 05 07  07 07 03 03 02 02 01 01
0250:  07 07 19 19 27 27 21 21  11 11 03 03 1C 1C 10 10
0260:  C0 C0 E0 E0 E0 60 60 E0  E0 E0 C0 C0 90 90 08 08
0270:  F0 F0 00 00 C0 C0 00 00  00 00 90 90 70 70 00 00
0280:  00 00 00 00 00 00 00 00  00 00 00 00 7F 7F F4 8B
0290:  8F FF 70 70 00 00 00 00  00 00 00 00 00 00 00 00
```

The emulator has a top menu bar with its options. This is the main point of interaction with the emulator user interface.

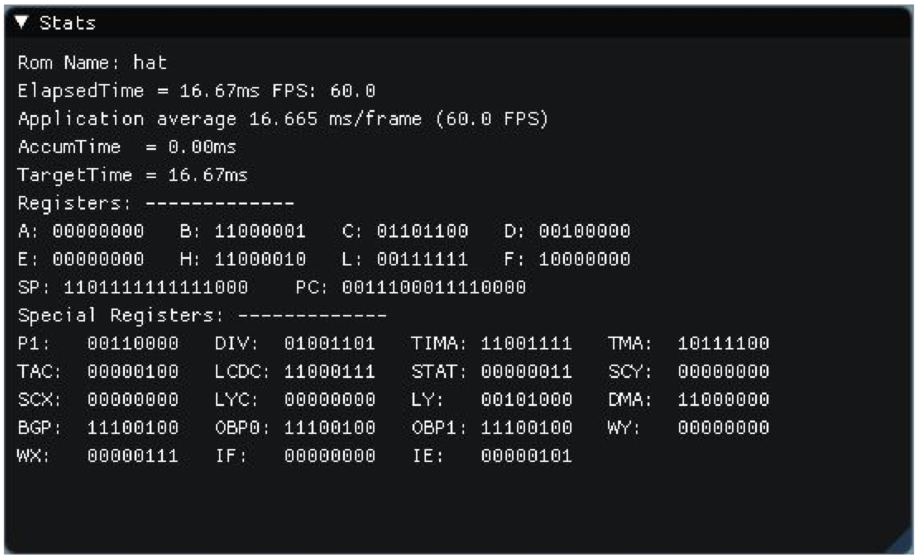
5.2.2. Main Screen

In the top left the emulator main screen is displayed, here the final image presented by the Gameboy emulator will be displayed.



5.2.3. Registers Panel

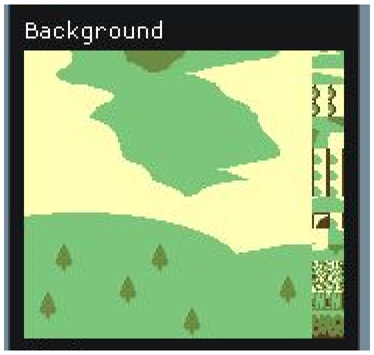
On the bottom left the registers debug panel can be seen. This panel contains stats about the current performance of the emulator as well the current value of the most important registers. The elapsed time displays the time it has taken for the emulator to render 1 frame. The emulator tries to cap itself to the vsync of the screen, which is usually 60 FPS o 16.6ms.

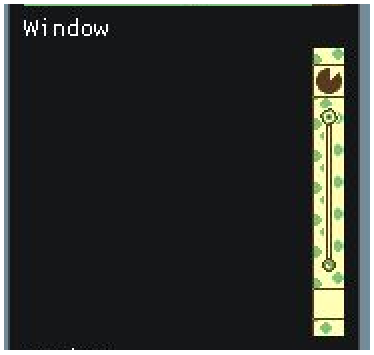


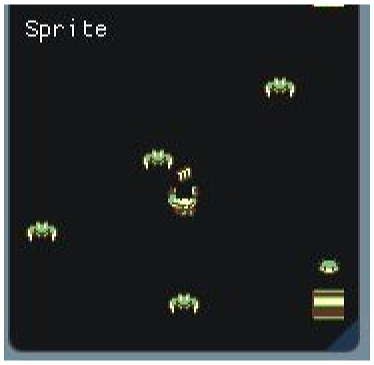
```
▼ Stats
Rom Name: hat
ElapsedTime = 16.67ms FPS: 60.0
Application average 16.665 ms/frame (60.0 FPS)
AccumTime  = 0.00ms
TargetTime = 16.67ms
Registers: -------------
A: 00000000   B: 11000001   C: 01101100   D: 00100000
E: 00000000   H: 11000010   L: 00111111   F: 10000000
SP: 1101111111111000    PC: 0011100011110000
Special Registers: -------------
P1:   00110000   DIV:  01001101   TIMA: 11001111   TMA:  10111100
TAC:  00000100   LCDC: 11000111   STAT: 00000011   SCY:  00000000
SCX:  00000000   LYC:  00000000   LY:   00101000   DMA:  11000000
BGP:  11100100   OBP0: 11100100   OBP1: 11100100   WY:   00000000
WX:   00000111   IF:   00000000   IE:   00000101
```

5.2.4. GPU Debugger

In the center the GPU debug is displayed. These screens are useful to debug what is going on with the Background, the Overlay Window or the rendering of the sprites when a problem occurs.

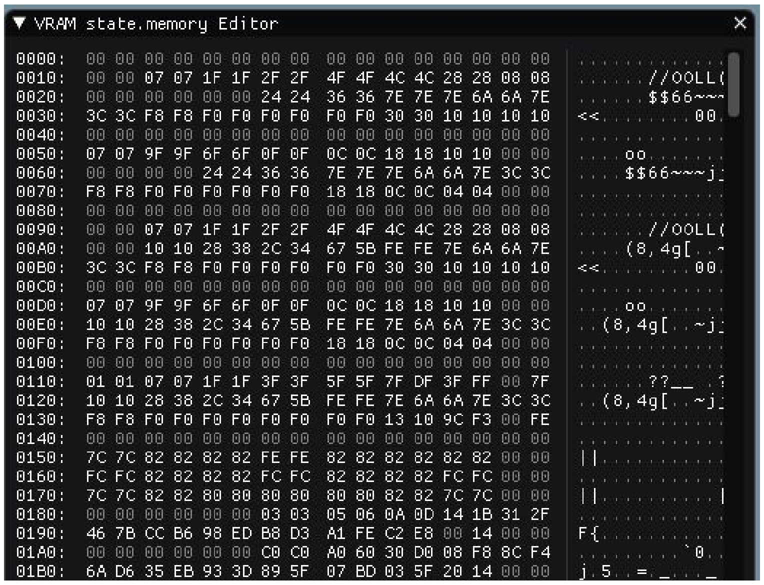






5.2.5. Memory Viewer

Finally on the right the memory debugger is rendered. By default the memory debugger points at the VRAM memory but this can be changed by selecting the Debug menu on the main menu and choosing what debug visualization the user wants to see.



```
▼ VRAM state.memory Editor                                                    ×
0000:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ..............
0010:  00 00 07 07 1F 1F 2F 2F  4F 4F 4C 4C 28 28 08 08   ......//OOLL(
0020:  00 00 00 00 00 00 24 24  36 36 7E 7E 7E 6A 6A 7E   ......$$66~~~
0030:  3C 3C F8 F8 F0 F0 F0 F0  F0 F0 30 30 10 10 10 10   <<........00.
0040:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ..............
0050:  07 07 9F 9F 6F 6F 0F 0F  0C 0C 18 18 10 10 00 00   ....oo........
0060:  00 00 00 00 24 24 36 36  7E 7E 7E 6A 6A 7E 3C 3C   ....$$66~~~j
0070:  F8 F8 F0 F0 F0 F0 F0 F0  18 18 0C 0C 04 04 00 00   ..............
0080:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ..............
0090:  00 00 07 07 1F 1F 2F 2F  4F 4F 4C 4C 28 28 08 08   ......//OOLL(
00A0:  00 00 10 10 28 38 2C 34  67 5B FE FE 7E 6A 6A 7E   ....(8,4g[..~
00B0:  3C 3C F8 F8 F0 F0 F0 F0  F0 F0 30 30 10 10 10 10   <<........00.
00C0:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ..............
00D0:  07 07 9F 9F 6F 6F 0F 0F  0C 0C 18 18 10 10 00 00   ....oo........
00E0:  10 10 28 38 2C 34 67 5B  FE FE 7E 6A 6A 7E 3C 3C   ..(8,4g[..~j
00F0:  F8 F8 F0 F0 F0 F0 F0 F0  18 18 0C 0C 04 04 00 00   ..............
0100:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ..............
0110:  01 01 07 07 1F 1F 3F 3F  5F 5F 7F DF 3F FF 00 7F   ......??__..?
0120:  10 10 28 38 2C 34 67 5B  FE FE 7E 6A 6A 7E 3C 3C   ..(8,4g[..~j
0130:  F8 F8 F0 F0 F0 F0 F0 F0  F0 F0 13 10 9C F3 00 FE   ..............
0140:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ..............
0150:  7C 7C 82 82 82 82 FE FE  82 82 82 82 82 82 00 00   ||............
0160:  FC FC 82 82 82 82 FC FC  82 82 82 82 FC FC 00 00   ..............
0170:  7C 7C 82 82 80 80 80 80  80 80 82 82 7C 7C 00 00   ||...........|
0180:  00 00 00 00 00 00 03 03  05 06 0A 0D 14 1B 31 2F   ..............
0190:  46 7B CC B6 98 ED B8 D3  A1 FE C2 E8 00 14 00 00   F{............
01A0:  00 00 00 00 00 00 C0 C0  A0 60 30 D0 08 F8 8C F4   ..........`0..
01B0:  6A D6 35 EB 93 3D 89 5F  07 BD 03 5F 20 14 00 00   j.5..=._...._
```

5.3. Controls

Once a ROM is loaded the emulator controls for the game are the following:

- **A:** Z key
- **B:** X key
- **Start:** A Key
- **Select:** S Key
- **D-Pad:** Arrow Keys

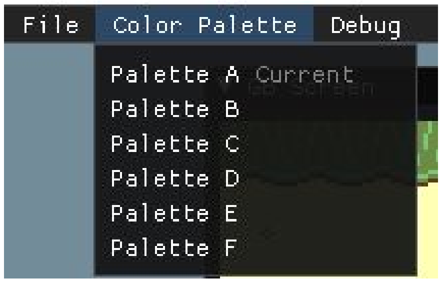
5.4. ROM loading

In order to run the emulator a ROM is needed, otherwise the emulator can not be run. This can be done by selecting FIle->LoadROM and selecting an appropriate ROM file.

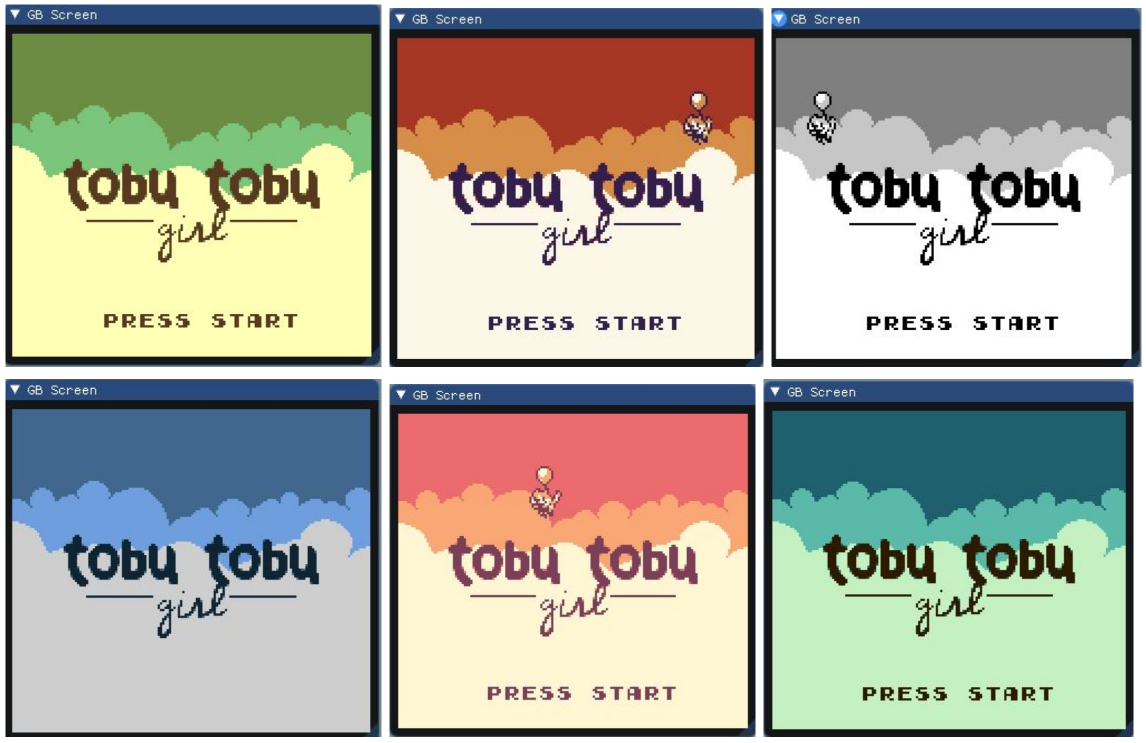
Some freeware test ROMs have been provided as well and can be found under File->Open Preset Files. Some of these ROMs are only for testing purposes and other ones are little games.

5.5. Additional Features

A palette switch menu has been implemented and can be accessed from the main menu. This allows the user to change the color palette used to display the image.



The different palettes can be seen in the images below:



5.6. Requirements

A x64 Windows PC with a graphics card supporting Opengl 2.0 is required in order to be able to run the emulator.

The user may also need to install the provided VC_redist.x64.exe file located on the Redist folder in order to have the needed .dll files.

6. Conclusions

6.1. Development Process

The development process has been harder than initially expected. One of the major problems has been understanding the Gameboy hardware, such as the memory, cpu or gpu and how they communicate to one another using RAM and registers.

In order to understand it a lot of resources have been consulted as well as other already existing open source emulators in order to understand how emulators work. The current emulator architecture is based on a combination of some of the existing emulators as well as personal preference.

Implementing all the Opcodes has also been a very difficult task as well as testing the emulator as homebrew ROMs have had to be used. Initially the plan was to dump an existing Gameboy game and use that as reference, however, due to the problems with the pandemic it was impossible to access to the hardware needed to dump those games and, as such, alternative ROMs have been used as well as generic testing ROMs created by the emulator development community.

6.2. Missing Features

The emulator lacks sound implementation, it is correctly being emulated on the RAM, however, the implementation of sound output has not been created.
The emulator also lacks a feature to save and load states. Some future work could be done in order to implement such a feature, as it is common for an emulator to have it.

6.3. Goals

The initially planned goals have been met. The emulator can load all kinds of ROMs and works correctly. It displays the screen output to the user and the user can interact with it using a keyboard in order to play the games as well as using a mouse to use the menus in the emulator window.

The missing features were features that initially were planned not to be part of the emulator as they were quite time consuming to implement.

An additional feature of Color Palette switching that was not present on the plan has been added which allows the user to change how the emulator displays the final output on the screen.

6.4. Planning and Methodology

The plan has not been following very closely as a lot of problems have been encountered during the development process. Such changes in planning are to be expected as emulator development was an uncharted territory before beginning work on this project. In order to assure the success of the project more sources have had to be consulted as well as taking inspiration on existing emulators regarding how they handle memory management and opcode implementation. That being said, the initial plan has been useful and a good guideline to start working.

6.5. Future Work

As explained on the missing features section audio would be one of the main lines of work that could be developed. Right now the emulator has no audio so a DSP(Digital Signal Processing) unit could be implemented to emulate the Gameboy audio.

A feature to save the current emulation state as well as another feature to load it could be implemented as well. Some work had been started on it but quickly dismissed in order to complete the emulator. In theory it could be as simple as loading the State struct which contains the current state of both CPU and Memory and then make a function that loads that file and restores that state. Going even further, a rewind functionality could be added, saving the state every few frames and allowing the user to rewind the state of the emulator.

The color palette feature could also be expanded. Instead of predefined color palettes, the emulator could let the user choose what colors they want and save those colors as a new color palette, that way each user could have a personalized theme to their emulator.

The emulator could also be ported to other platforms. Right now it only works on Windows, but it could also be ported over to Linux systems or Mac OS systems. Going even further as this would require a complete rewriting of the emulator it could also be ported to smart devices such as iPhone or Android or to console devices as a part of the Homebrew scene.

6.6. Final Thoughts

The creation of this emulator has been a very difficult task that took more work than anticipated but has also been a really good learning experience.
Knowing first hand how a console like the Gameboy works is a really good resource of knowledge that can later be used in creating other pieces of software or coming up with clever solutions to difficult problems.

The creation of this emulator has also been a fun task as the topic of emulation is an interesting one and knowing now how it works participating in emulator development communities sounds like an interesting thing.

I hope that this work also serves others to learn a little bit more about the emulator development scene and to make people gain interest in emulator development.

7. Glossary

PAC: Prova d'Avaluacio Continuada.
Speedrun: An instance of completing a video game, or level of a game, as fast as possible.
TAS: Tool Assisted Speedrun.
GB: Game Boy.
NES: Nintendo Entertainment System.
SNES: Super Nintendo Entertainment System.
RAM: Read Only Memory.
ROM: Read Only Memory (Also used to refer to game data stored on disk).
CPU: Central Processing Unit.
AI: Artificial Intelligence.
Pathfinding: The plotting, by a computer application, of the shortest route between two points.
D-pad: Directional pad, a four-way directional control with one button on each point
DOS: Disk Operating System (Usually refers to Microsoft DOS or MS-DOS)
Flag: A value that can be true or false and changes the behavior of the program.
Register: A small place on the CPU designated to store data.
Timer: A clock that controls a sequence of events.
Tileset: A collection of rectangular images.
Sprite: A bitmap graphic designated to be part of a larger screen.
Debugging: The process of identifying and removing errors from computer hardware or software.
Opcode: Short for Operation Code, the part of a machine code instruction that defines the operation to be performed.

8. Bibliography

1. *Gameboy Emulators*, www.emulator-zone.com/doc.php/gameboy/.
2. *-= The GameBoy Project Homepage =-*, marc.rawer.de/Gameboy/.
3. *Binjgb Rewind*, 31 Dec. 2017, binji.github.io/posts/binjgb-rewind/.
4. *Imran Nazar: GameBoy Emulation in JavaScript: The CPU*, imrannazar.com/gameboy-Emulation-in-JavaScript.
5. *Binjgb Rewind*, 31 Dec. 2017, binji.github.io/posts/binjgb-rewind/.
6. *Debugging Hangs*, 3 May 2017, binji.github.io/posts/debugging-hangs/.
7. *Binjgb on the Web, Part 1*, 26 Feb. 2017, binji.github.io/posts/binjgb-on-the-web-part-1/.
8. *Previously Hosted at Gameboydev.org*, gbdev.gg8.se/files/roms/blargg-gb-tests/.
9. *Index of /Files/Mooneye-Gb/Latest/*, gekkio.fi/files/mooneye-gb/latest/.
10. "# About the Pan Docs." *Pan Docs | Pan Docs*, gbdev.io/pandocs/.
11. /, CodeSlinger. *Codeslinger.co.uk*, www.codeslinger.co.uk/pages/projects/gameboy.html.
12. 262588213843476. "GameBoy Color Boot ROM Disassembly." *Gist*, gist.github.com/drhelius/6063265.
13. Aappleby. "Aappleby/MetroBoy." *GitHub*, 6 June 2020,

github.com/aappleby/MetroBoy.

14. AntonioND. “AntonioND/Giibiiadvance.” *GitHub*, 8 May 2020, github.com/AntonioND/giibiiadvance/blob/master/docs/.
15. AntonioND. “AntonioND/Giibiiadvance.” *GitHub*, 8 May 2020, github.com/AntonioND/giibiiadvance/tree/master/docs.
16. Applefreak. “Applefreak/esp8266-Gameboy-Dev-Board.” *GitHub*, 29 Mar. 2017, github.com/applefreak/esp8266-gameboy-dev-board.
17. Applefreak. “Applefreak/esp8266-Gameboy-Printer.” *GitHub*, github.com/applefreak/esp8266-gameboy-printer.
18. “BGB Homepage.” *BGB GameBoy Emulator (Current Version: BGB 1.5.8)*, bgb.bircd.org/.
19. Baekalfen. “Baekalfen/PyBoy.” *GitHub*, 17 May 2020, github.com/Baekalfen/PyBoy.
20. “Board Index.” *GBC Colorization Palettes in the BIOS*, forums.nesdev.com/viewtopic.php?p=114388&sid=c3d4ce08cfd9d9c834958d4f148750c3#p114388.
21. “Build Software Better, Together.” *GitHub*, github.com/topics/gameboy-emulator.
22. CTurt. “CTurt/Cinoop.” *GitHub*, 15 Nov. 2015, github.com/CTurt/Cinoop.
23. “The Centre for Computing History.” *Centre For Computing History*, www.computinghistory.org.uk/det/4033/Nintendo-Game-Boy/.

24. "DMG Schematics." *DMG Schematics - GbdevWiki*, gbdev.gg8.se/wiki/articles/DMG_Schematics.

25. Dinu, Cristian. "DECODING Gameboy Z80 OPCODES." *Decoding Gamboy Z80 Opcodes*, gb-archive.github.io/salvage/decoding_gbz80_opcodes/Decoding Gamboy Z80 Opcodes.html.

26. Drhelius. "Drhelius/Gearboy." *GitHub*, 23 May 2020, github.com/drhelius/Gearboy.

27. "Emulating a GameBoy Cartridge with an STM32F4. Part 1." *Emulating a GameBoy Cartridge with an STM32F4. Part 1 · Dhole's Blog*, 24 Dec. 2014, dhole.github.io/post/gameboy_cartridge_emu_1/.

28. Furrtek. "Furrtek/DMG-CPU-Inside." *GitHub*, 7 Nov. 2019, github.com/furrtek/DMG-CPU-Inside.

29. "Game Boy." *National Museum of American History*, americanhistory.si.edu/collections/search/object/nmah_1253117.

30. "Game Boy." *Wikipedia*, Wikimedia Foundation, 6 June 2020, en.wikipedia.org/wiki/Game_Boy.

31. "Game Boy Advance." *Wikipedia*, Wikimedia Foundation, 1 June 2020, en.wikipedia.org/wiki/Game_Boy_Advance.

32. "Game Boy Color Bootstrap ROM." *Game Boy Color Bootstrap ROM -*

The Cutting Room Floor, tcrf.net/Game_Boy_Color_Bootstrap_ROM.

33. "Game Boy Family." *Wikipedia*, Wikimedia Foundation, 4 June 2020, en.wikipedia.org/wiki/Game_Boy_family.
34. "Game Boy Hardware Databaseby Gekkio and Contributors." *Home - Game Boy Hardware Database*, gbhwdb.gekkio.fi/.
35. "GameBoy Cartridges." *GB.html*, fms.komkon.org/GameBoy/Tech/Carts.html.
36. "Gameboy Cartridges." *GB.html*, www.devrs.com/gb/files/gb.html.
37. Gbdev. "Gbdev/Awesome-Gbdev." *GitHub*, 24 May 2020, github.com/gbdev/awesome-gbdev#emulators.
38. Gbdev. "Gbdev/Awesome-Gbdev." *GitHub*, github.com/gbdev/awesome-gbdev/blob/master/EMULATORS.md.
39. Gekkio. "Gekkio/Gb-Hardware." *GitHub*, 17 May 2020, github.com/Gekkio/gb-hardware.
40. Gekkio. "Gekkio/Mooneye-Gb." *GitHub*, 18 Apr. 2020, github.com/Gekkio/mooneye-gb.
41. "History of Emulation." *Video Game Emulation Wiki*, emulation.fandom.com/wiki/History_of_emulation#:~:text=Game Boy Advance&text=GBAEmu, released in September 2000,according to its official site.
42. "History of Emulation." *History of Emulation - Emulation General Wiki*,

emulation.gametechwiki.com/index.php/History_of_emulation.

43.“Home.” *Home*, gekkio.fi/.

44.Imgur. “Gameboy Cartridges.” *Imgur*, 21 June 2016, imgur.com/a/D5bpC.

45.LIJI32. “LIJI32/SameBoy.” *GitHub*, 3 June 2020, github.com/LIJI32/SameBoy.

46.LIJI32. “LIJI32/SameSuite.” *GitHub*, 29 Sept. 2019, github.com/LIJI32/SameSuite.

47.“Lazy Foo' Productions.” *Lazy Foo' Productions - Beginning Game Programming v2.0*, lazyfoo.net/tutorials/SDL/.

48.Levick, Ryan. “Oh Boy! Creating a Game Boy Emulator in Rust.” *Home*, Media.ccc.de, media.ccc.de/v/rustfest-rome-3-gameboy-emulator.

49.Mattbruv. “Mattbruv/Gameboy-Emulator.” *GitHub*, github.com/mattbruv/Gameboy-Emulator.

50.Mattcurrie. “Mattcurrie/Mealybug-Tearoom-Tests.” *GitHub*, 24 Oct. 2019, github.com/mattcurrie/mealybug-tearoom-tests.

51. Minotti, Mike. “25 Years of the Game Boy: A Timeline of the Systems, Accessories, and Games.” *VentureBeat*, VentureBeat, 12 Dec. 2018, venturebeat.com/2014/04/21/25-years-of-the-game-boy-a-timeline-of -the-systems-accessories-and-games/.

52.“NAME.” *GBZ80(7)*, rednex.github.io/rgbds/gbz80.7.html.

53. Ocornut. “Ocornut/Imgui.” *GitHub*, 5 June 2020, github.com/ocornut/imgui.
54. Plant, Mike. “A Timeline of Game Boy's Record-Breaking History as Iconic Console Celebrates 30 Years.” *Guinness World Records*, Guinness World Records, 17 Apr. 2019, www.guinnessworldrecords.com/news/2019/4/a-timeline-of-game-boys-record-breaking-history-as-iconic-console-celebrates-30-565921.
55. Saltalamacchia, Brandon, et al. “The History Of The Gameboy.” *Retro Dodo*, 24 Feb. 2020, retrododo.com/gameboy/.
56. Stephen, Bijan. “How the Game Boy Found a New Life through Emulation.” *The Verge*, The Verge, 18 Apr. 2019, www.theverge.com/2019/4/18/18311740/game-boy-emulation-new-life-old-technology.
57. “TASVideos.” *TASVideos RSS News*, tasvideos.org/EmulatorResources/GBAccuracyTests.html.
58. “Thirty Years Ago, Game Boy Changed the Way America Played Video Games.” *Smithsonian.com*, Smithsonian Institution, 29 July 2019, www.smithsonianmag.com/innovation/thirty-years-ago-game-boy-changed-way-america-played-video-games-180972743/.
59. Trekawek. “Trekawek/Coffee-Gb.” *GitHub*, 21 Dec. 2018, github.com/trekawek/coffee-gb.

60. Webb, Kevin. “How Nintendo's Handheld Video Game Consoles Have Evolved over the Past 30 Years, from the Original Game Boy to the Switch.” *Business Insider*, Business Insider, 23 Apr. 2019, www.businessinsider.com/nintendo-game-boy-history-evolution-ds-3ds-switch-2019-4?IR=T.

61. “Why Did I Spend 1.5 Months Creating a Gameboy Emulator?” *Why Did I Spend 1.5 Months Creating a Gameboy Emulator? · Tomek's Blog*, blog.rekawek.eu/2017/02/09/coffee-gb/.

62. “Writing a Game Boy Emulator, Cinoop.” *Cinoop*, cturt.github.io/cinoop.html.

63. “Writing a Game Boy Emulator, Cinoop.” *Cinoop*, cturt.github.io/cinoop.html.

64. “r/Emulation - 🎮 Early History of Gameboy Emulation 🎮.” *Reddit*, www.reddit.com/r/emulation/comments/c9tb97/early_history_of_gameboy_emulation/.

65. serginho89, and serginho89. *RealBoy*, 1 July 2013, realboyemulator.wordpress.com/posts/.

66. “🔗 Game Boy CPU (SM83) Instruction Set (JSON).” *Game Boy CPU (SM83) Instruction Set*, gbdev.io/gb-opcodes/optables/.