CÉGEP DE MATANE

FORUM INNOVATION
INGÉNIERIE | INFORMATIQUE |
ENTREPRENEURIAT | UQAR

PixelBoy un émulateur de GameBoy en C

par Kylian Eury

Présentés au FI3E par le Cégep de Matane

FORUM INNOVATION
INGÉNIERIE | INFORMATIQUE |
ENTREPRENEURIAT | UQAR

PixelBoy un émulateur de GameBoy en C

par Kylian Eury

Un émulateur de GameBoy écrit de toutes pièces en C.

Il n'est pas encore terminé, mais la majorité des jeux sont entièrement jouables et de nombreuses fonctionnalités ont été émulées.

Pourquoi

Dans un marché en plein essor pour les jeux rétro, marqué par un retour nostalgique aux expériences classiques de gaming, il existe un besoin distinct de solutions capables de recréer fidèlement ces expériences sur des appareils modernes.

Le défi réside non seulement dans l'émulation technique de l'architecture matérielle d'origine, comme le CPU, le GPU, la gestion de la mémoire de la GameBoy,

Mais aussi dans la capture de l'essence esthétique et ludique qui a fait le charme de ces systèmes.

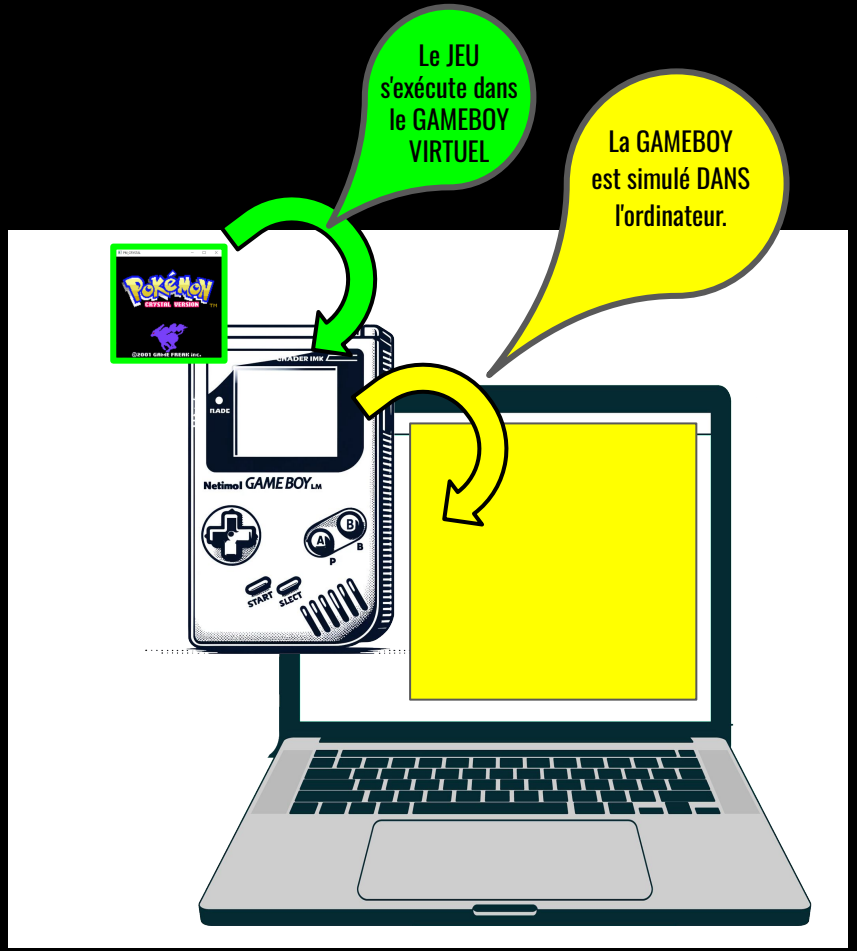
Illustration par l'intelligence artificielle

Objectifs

Développer un émulateur de GameBoy écrit entièrement en C,

- visant à offrir une expérience de jeu authentique
- tout en étant léger et performant,
- pour préserver l'expérience originale de GameBoy.

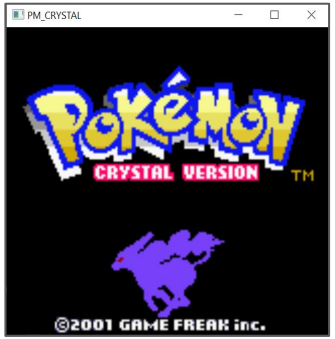
Dans ce système : Le JEU s'exécute donc dans le GAMEBOY qui s'exécute dans l'ordinateur.
On a un programme dans un programme.

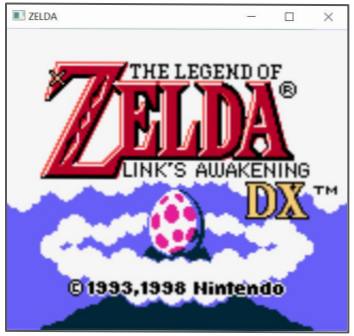


Objectifs

L'émulateur vise à supporter le plus de jeux,

- avec une implémentation complète du CPU
- du PPU (puce graphique)
- de l'APU (puce sonore)
- des dizaines de registres matériels
- et un contrôle mémoire,
- ainsi qu'une interface utilisateur classique basée sur OpenGL.





Implémentations

Testées dans les jeux

🗄️ **Emulation du CPU :**

Implémentation de tous les opcodes et réussite des ROM de test de blargg (11/11).

💾 **Contrôleur de mémoire :**

Gestion complète des opérations de lecture/écriture sur les composants et registres matériels

🎨 **Unité de traitement graphique (PPU) :**

Dessin ligne par ligne, bien que non cycle-précis, entraînant quelques bugs graphiques.

📣 **Puce de traitement audio (APU) :**

Émulation en grande partie complète. Génération des sons à la volée pour le driver audio

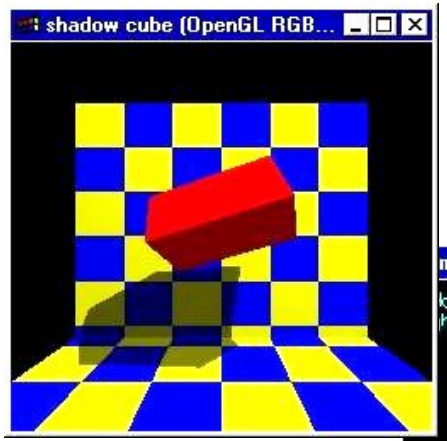
Support ROM MBC1/2/3/5 :

Avec plans pour ajouter plus de mappers. permet déjà de lancer la majorité des jeux

Choix technologique

J'ai choisi OpenGL pour :

- Sa compatibilité multiplateforme, OpenGL est un standard reconnu depuis plus de 30 ans
- Son contrôle détaillé des graphiques pour recréer les graphiques vintage
- La richesse de ses fonctionnalités tels que les shaders pour adapter les jeux originaux aux écrans modernes
- Sa communauté éducative



Choix technologique

J'ai choisi CMake pour :

- Création de builds multi-plateformes (génération de makefile, solutions Visual Studio, projets XCode etc.)
- Automatisation du processus de compilation
- Simplification du développement sur d'autres machines


```
cmake_minimum_required(VERSION 3.28)
project(PixelBoy LANGUAGES C)

find_package(OpenGL REQUIRED)

# Add library subfolders
add_subdirectory(libs/glfw EXCLUDE_FROM_ALL)
add_subdirectory(libs/portaudio EXCLUDE_FROM_ALL)
```

Problèmes et Solutions

🔎 REVERSE ENGINEERING

La documentation de ce matériel ancien et propriétaire était incomplète et pour déduire certaines spécifications, il a fallu passer le code dans un débugger avec des points d'arrêts en assembleur et comparer avec une référence (console ou émulateur fiable).

Ce manque de documentation a été révélé par des fonctionnalités qui étaient imparfaites (émulations non-reproductibles). Du travail d'inspection bas niveau avec l'exécution pas à pas et l'inspection des valeurs des registres matériels ont permis de résoudre les problèmes.

🏛️ ARCHITECTURE LOGICIELLE

Des abstractions de programmation impérative (en C) ont été nécessaires pour gérer une logique de code très complexe.

Un exemple d'application de ces architectures a été la gestion des mappers des cartouches. Des pseudo "interfaces" ont été simulées en utilisant des structures avec des pointeurs de fonctions.

🔄 DÉBORDEMENTS LOGIQUES

N'importe quelle erreur d'implémentation d'une instruction pouvait aussi causer des effets de bords sur les instructions suivantes étant donné les nombreuses interactions entre tous les composants et registres matériels.

Ce problème a été révélé par des jeux qui ne fonctionnaient plus lorsque l'émulation des interruptions matérielles était activée. Cela se déclenchait sur les copies de routines qui servaient à transférer des données à la puce graphique. Une mauvaise implémentation d'une instruction CPU (longueur de paramètre d'un octet spécifiée sur une instruction qui n'en prenait pas) faisait que n'importe quelle instruction suivante n'était pas exécutée.

Résultats

- Émulation complète de la majorité des jeux pour GameBoy classique, et travail presque terminé pour les jeux de GameBoy Color
- Émulation en grande partie complète du son (reproduit par peu d'émulateurs car complexe)
- Système de sauvegarde pour reprendre les jeux plus tard
- Rétrocompatibilité GameBoy / GameBoy Color
- Le langage C permet d'avoir un émulateur très compact et optimisé (600ko de stockage, < de 2 Mo de RAM, < 1% de temps CPU utilisés sur un laptop moderne)


On peut jouer à tous ces jeux !
ZELDA
THE LE
ZEL
LIN
©19
TETRIS
1 PLAY
©1989
PM_CRYSTAL
Pokémon
CRYSTAL VERSION
TM
©2001 GAME FREAK inc.

Futur

Les améliorations futures du projet sont de deux ordres :

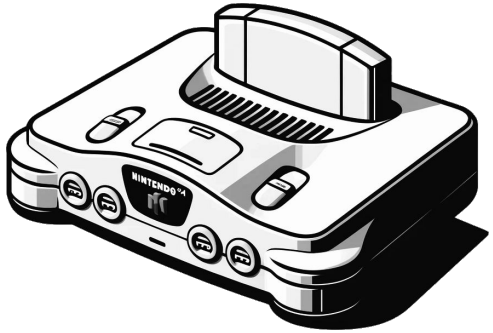
=> Maintenances sur cet émulateur

- Terminer l’émulation de la GameBoy color, finir l’émulation sonore, corriger des bugs

=> Exploration d'autres émulateurs

Cette expérience ayant été des plus enrichissante, j'envisage expérimenter l'émulation de consoles vidéos plus avancées telles que :

- Développer l’émulateur pour émuler la **GameBoy Advance**
- Produire un émulateur Nintendo **N64**
- Produire un émulateur PlayStation **PSX**



Téléchargez l'émulateur ici

https://github.com/razor7877/PixelBoy

Références - Doc techniques

Gameboy CPU (LR35902) instruction set (Tableau opcodes)
https://www.pastraiser.com/cpu/gameboy/gameboy_opcodes.html

GAME BOY Programming Manual Version 1.1 (Détail opcodes)
https://ia803208.us.archive.org/9/items/GameBoyProgManVer1.1/GameBoyProgManVer1.1.pdf

Pan Docs - Documentation du matériel
https://gbdev.io/pandocs/

Références - Inspirations

codeslinger.co.uk - Gameboy - Graphics Emulation.
http://www.codeslinger.co.uk/pages/projects/gameboy/graphics.html

BGB GameBoy Emulator (current version: BGB 1.6.2)
https://bgb.bircd.org/

GameBoy Sound Emulation
https://nightshade256.github.io/2021/03/27/gb-sound-emulation.html