THOMAS MORE

EMBEDDED DEVICES ADVANCED

Game Boy

2023 - 2024

1. Woord vooraf

Mijn naam is Jesse Gabriëls. Ik ben een student aan Thomas More hogeschool in Geel en ik volg de richting IoT (Internet of Things) in het 3de jaar. Voor het vak *Embedded Deviced Advanced* kregen wij de keuze uit een aantal projecten. Ik heb gekozen voor: *Creatie van een GBA Emulator game in C met een documentatie van de interne werking en het creatieproces.*

Een Game Boy game creëren is een zeer interessant project. Het is en zal voor altijd een enorm bekende gameconsole blijven. Zelf ben ik niet opgegroeid in het tijdperk van de Game Boy, maar net zoals vele anderen ben ik er wel bekend mee. Het maken van dit project was enorm leuk en leerrijk. Het gebruik maken van oude technologie is iets wat we tegenwoordig niet meer zoveel doen. Hierdoor voelde het tof om toch eens te doen. Het in het achterhoofd houden dat je programma dat je hebt gemaakt met hedendaagse technologie kan worden uitgevoerd op technologie van 25 jaar oud is verbazingwekkend.

De Game Boy lijn is al een tijdje niet meer te vinden in de winkels. Toch is er nog een grote community die zich bezighouden met deze retro consoles; Zij maken het gemakkelijker voor anderen die zelf eens willen experimenteren met een Game Boy.

Graag wil ik docent Quinten Desmyter bedanken voor de inspiratie van mogelijke games.

Inhoud

2. Lijst met figuren

3. Inleiding

In dit verslag toon ik aan hoe je zelf een game kan maken voor een Game Boy. Hiervoor zal er eerst worden uitgelegd wat een Game Boy nu precies is en hoe het allemaal in elkaar zit. Ook zal de tijdlijn van alle uitgebrachte *handheld Game Boy consoles* aan bod komen. Verder zullen er ook een aantal tools ter sprake komen dat het maken van een game vergemakkelijkt.

4. Game Boy

4.1. Omschrijving

Een Game Boy is een game-console ontwikkeld door het Japanse bedrijf Nintendo dat voor het eerst werd uitgebracht op 21 april 1989. De Game Boy was een revolutionair apparaat dat een immense invloed had op de game-industrie en een onuitwisbare stempel heeft gedrukt op de gamingwereld. Dit draagbare systeem, hoewel technologisch eenvoudig vergeleken met hedendaagse apparaten, wist een enorm succes te behalen.

De Game Boy gebruikte een 8-bit Sharp-processor, de LR35902. Deze is gebaseerd op de Z80-architectuur. Ondanks zijn bescheiden kloksnelheid van 4,19 MHz, was het krachtig genoeg voor grafische en rekenkundige vereisten van zijn tijd. Het had 8 KB RAM, 8 KB VRAM en kon gegevens opslaan op externe cartridges.

Het monochrome LCD-scherm van de Game Boy was zowel een kracht als een beperking. Met een resolutie van 160x144 pixels, kon het slechts vier grijstinten weergeven. Toch maakte deze eenvoud het apparaat energiezuinig, wat essentieel was voor draagbare gameconsoles.

Het ontwerp van de Game Boy controller was intuïtief met een D-pad en knoppen A, B, Start en Select. Deze eenvoudige besturing maakte de Game Boy toegankelijk voor een breed publiek.

De mogelijkheid om games op cartridges te spelen, gaf de Game Boy een enorm voordeel. Spelers konden hun favoriete games meenemen en eenvoudig verwisselen, wat de draagbaarheid aanzienlijk vergrootte.

4.2. Succes

De Game Boy was de eerste succesvolle poging om gaming mobiel te maken. Het concept was ongekend en opende nieuwe mogelijkheden voor zowel recreatieve games en voor ontwikkelaars om nieuwe soorten games te bedenken.

Een grote reden van het succes van de Game Boy was het spel Tetris. Een simpel maar verslavend puzzelspel. Het droeg aanzienlijk bij aan de acceptatie van de Game Boy en maakte het tot een “must-have”.

De Game Boy had een indrukwekkend bibliotheek met klassieke titels zoals Super Mario Land, The Legend of Zelda, Pokémon,… Deze games droegen bij aan de blijvende aantrekkingskracht van het systeem.

Figuur 1 Tetris cartridge

4.3. Tijdlijn

4.3.1. Game Boy - 1989

De originele Game Boy was de eerste draagbare gameconsole van Nintendo en was een doorbraak in de gaming wereld. Er werd een aangepaste 8-bit processor geïmplementeerd, gebaseerd op de Intel 8080 en Zilog Z80. Het had een monochroom LCD-scherm met een groenachtige tint dat later iconisch werd voor de Game Boy. En alles gevoed door vier AA-batterijen voor een spelduur tot 30 uur. Dit alles bewees dat draagbare gameconsoles een succesvol product konden zijn voor jongeren en volwassenen over heel de wereld.

Figuur 2 Game Boy

4.3.2. Game Boy Pocket - 1996

De Game Boy Pocket kwam uit in het jaar 1996. Deze was een kleinere en lichtere versie van de originele Game Boy met een helderder scherm en een betere beeldkwaliteit. Verder werkte de console nu op twee AAA-batterijen in plaats van vier AA-batterijen. Dit zorgde ervoor dat hij enorm makkelijk was om mee te nemen.

Figuur 4 Game Boy Pocket

4.3.3. Game Boy Light - 1998

Twee jaar nadat de Game Boy Pocket op de markt was, kwam de Game Boy Light uit. Dit was de eerste versie met een verlicht scherm, wat het spelen in het donker mogelijk maakte. De Game Boy Light werd nooit officieel buiten Japan gedistribueerd.

Figuur 3 Game Boy Light

4.3.4. Game Boy Color - 1998

In hetzelfde jaar kwam ook de Game Boy Color op de markt. Hierbij werd een kleurenscherm geïntroduceerd en daarbovenop was het compatibel met oudere Game Boy games. De console kon automatisch kleurenpaletten toewijzen aan monochrome games. Dit zorgde ervoor dat de oudere games weer nieuw leven kregen. Verder had de Game Boy Color een krachtigere processor dan zijn voorgangers; maar liefst 2 keer zo snel en 3 keer zoveel geheugen.

Figuur 5 Game Boy Color

4.3.5. Game Boy Advance - 2001

In het jaar 2001 werd de Game Boy Advance (GBA) uitgebracht. Dit was een aanzienlijk krachtigere draagbare gameconsole met geavanceerdere graphics en de mogelijkheid om SNES-achtige games te spelen. Het was de eerste 32-bits gameconsole van Nintendo. Het nieuwe design moest er moderner en zachter uitzien. Ook had de console nu 2 schouderknoppen en een TFT-kleurenscherm.

Figuur 6 Game Boy Advance

4.3.6. Game Boy Advance SP - 2003

Figuur 7 Game Boy Advance SP

Twee jaar later kwam de opvouwbare versie van de GBA op de markt. Het had dezelfde spelprestaties maar alles in een geheel nieuwe vorm. Nintendo wilde gamers een formaat aanbieden dat makkelijk in een broekzak paste. Dankzij het nieuwe ontwerp kon de console worden gesloten en worden weggestopt zonder dat het scherm of de knoppen beschadigd raken.

De Game Boy Advance SP had ingebouwde verlichting en een oplaadbare batterij. Hierdoor moest je geen nieuwe batterijen meer gaan kopen maar kon je de console een paar uur in het stopcontact steken en weer verder spelen.

Figuur 8 Game Boy Micro

4.3.7. Game Boy Micro - 2005

In het jaar 2005 kwam de Game Boy Micro uit. Dit was de kleinste en laatste versie van de Game Boy lijn en ondersteunde alleen GBA-games. De nadruk werd voornamelijk gelegd op de draagbaarheid. Verder konden gebruikers de look van hun console aanpassen met verwisselbare faceplates.

Na de Game Boy Micro stopte Nintendo met de productie ten gunste van de Nintendo DS. De DS en 3DS volgden en bouwden voort op het erfgoed van draagbare gameconsoles dat door de Game Boy was gevestigd.

Verder doorheen dit document leggen we de focus bij de originele Game Boy.

4.4. Graphics

De Game Boy was niet zo technologisch geavanceerd op het gebied van graphics. De grafische beperkingen hadden een directe invloed op het ontwerp van games. Games waren vaak eenvoudiger in termen van graphics. Dit dwong ontwikkelaars om op inventieve wijze gebruik te maken van de beschikbare middelen, wat resulteerde in een aantal van de meest gedenkwaardige en tijdloze spellen in de geschiedenis van mobile gaming.

4.4.1. Tiles

Net zoals andere retro gameconsoles, worden pixels niet individueel gemanipuleerd. Dit zou namelijk te veel energie vragen van de CPU. Hierdoor worden pixels gegroepeerd in een 8x8 patroon dat ook wel eens een “tile” wordt genoemd.

Een tile codeert geen kleurinformatie maar wijst een tile een colorID toe aan elke pixel. Dit kan variëren van 0 tot 3. Om deze reden worden Game Boy graphics ook wel 2bpp (2 bits per pixel) genoemd.

4.4.2. Palettes

Omdat de originele Game Boy een monochroom scherm heeft, wil dat zeggen dat er slechts één kleur tegelijk kan worden weergegeven. Omdat het scherm groenachtig was, leidde dit tot een beperkt kleurenpalet dat bestond uit verschillende tinten grijs en groen. Het kleurenpalet van de Game Boy bestaat uit een reeks van 4 kleuren; Namelijk zwart, donkergrijs, lichtgrijs en wit (doorschijnend).

Elke kleur in het palet werd toegewezen aan een numerieke index die correspondeerde met specifieke bits in het geheugen. Door deze indexering konden developers aangeven welke kleur op welke locatie op het scherm moest worden gebruikt.

4.4.3. Layers

Een Game Boy heeft 3 layers. Background, Window en Objects/Sprite. De verschillende lagen en technieken op de Game Boy maakte het mogelijk om gevarieerde en boeiende visuele ervaringen te creëren binnen de beperkingen van de hardware.

4.4.3.1. Background

De Background Layer is verantwoordelijk voor het weergeven van de statische elementen van het spel, zoals de spelomgeving en de achtergrondafbeeldingen. Dit kunnen landschappen, muren of andere niet-bewegende onderdelen zijn.

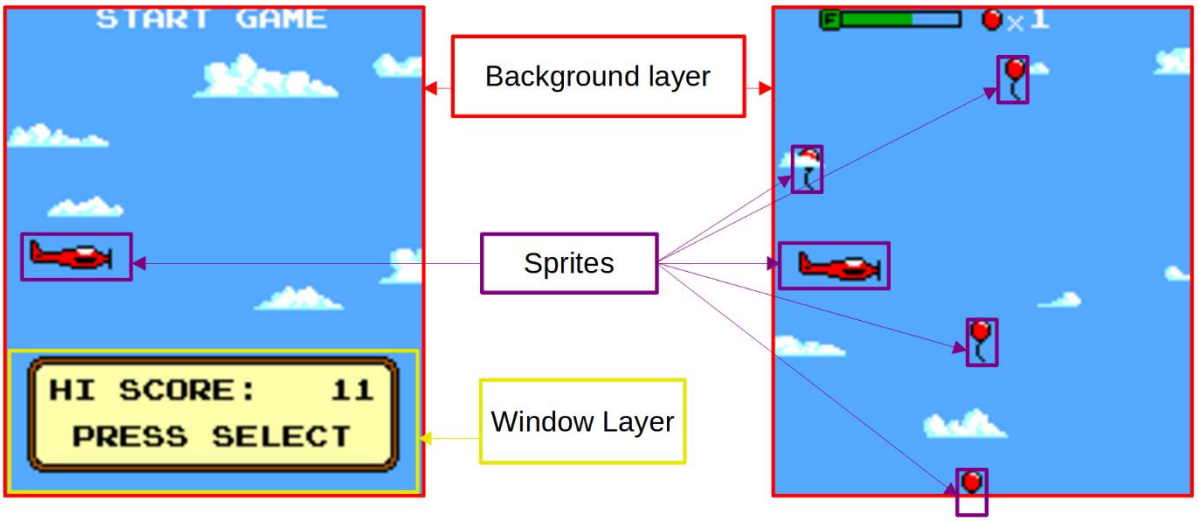
De Background Layer bestaat uit een tilemap; Een groot raster van tiles. Tiles worden niet direct naar tilemaps geschreven, ze bevatten slechts verwijzingen naar de tiles. Dit maakt het hergebruik zeer goedkoop in CPU-tijd en geheugenruimte.

4.4.3.2. Window

De Window Layer maakte het mogelijk om specifieke delen van het scherm te definiëren waarin sprites konden worden weergegeven. Dit stelde developers in staat om bepaalde delen van het scherm te reserveren voor specifieke game-elementen. Het wordt vaak gebruikt om een statusbalk weer te geven met levels, scores of tekst.

4.4.3.3. Objects/Sprites

De Sprite Layer is verantwoordelijk voor het weergeven van bewegende objecten, personages of andere dynamische elementen in het spel. Sprites zijn onafhankelijke grafische objecten die kunnen bewegen en overlappen met de Background Layer.



Figuur 9 Game Boy scherm lagen

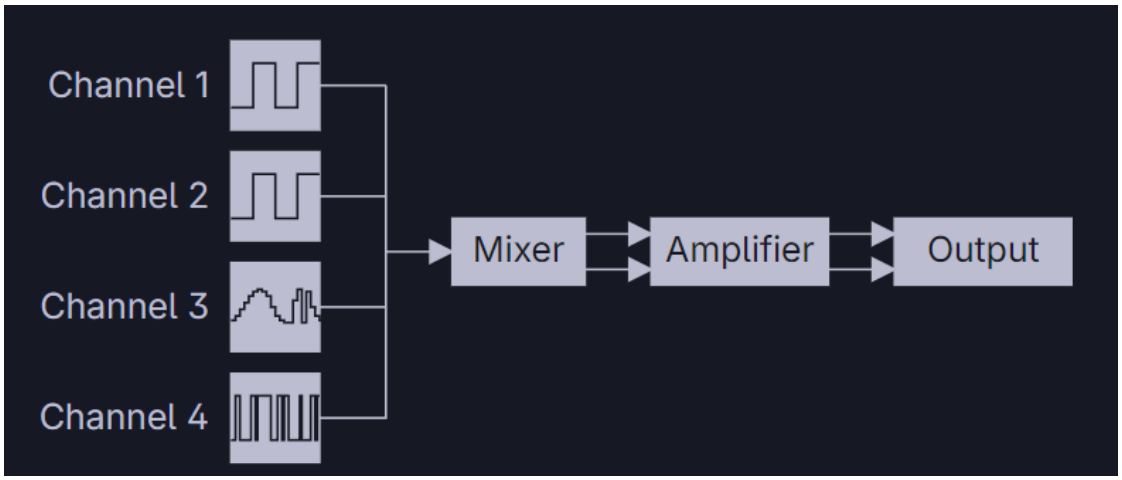
4.5. Audio

De audio is een integraal onderdeel van de ervaring en heeft kenmerkend geluid dat nostalgisch is voor veel gamers. De originele Game Boy heeft een aangepast geluidschip die verantwoordelijk is voor het genereren van alle geluiden. Deze chip (Game Boy Sound System) speelde een cruciale rol bij het creëren van muziek en geluidseffecten.

De Game Boy heeft 4 geluidsgeneratoren: kanaal 1 tot en met 4 en genoteerd als “CH1”, “CH2”, “CH3” en “CH4”. Elk geluidskanaal is gespecialiseerd op een manier die grotendeels verschilt van de andere kanalen.

- CH1 en CH2 (Pulse Wave Channels): Deze kanalen genereren puls-golfvormen en konden worden aangepast om verschillende tonen te produceren.
- CH3 (Wave Channel): Dit kanaal maakt gebruik van een golfvormtabel om complexere geluiden te genereren, vergelijkbaar met een synthesizer.
- CH4 (Noise Channel): Dit kanaal genereerd willekeurige geluiden en wordt vaak gebruikt voor percussiegeluiden en speciale effecten.

Deze signalen worden vervolgens gemengd tot 2 nieuwe kanalen die voor stereo dienen. Eén voor het linkeroor en één voor het rechteroor. Deze kunnen afzonderlijk worden versterkt en vervolgens worden uitgevoerd naar de hoofdtelefoonaansluiting of de luidspreker.



Figuur 10 Game Boy Audio

4.6. Gamepad

De Game Boy gebruikt een gamepad/joypad als invoerapparaat. De Game Boy-joypad had een compact ontwerp. Dit was essentieel voor een draagbare gameconsole. Het ergonomische ontwerp zorgde ervoor dat spelers comfortabel konden gamen. Het ontwerp van de joypad is zeer herkenbaar en symboliseert een tijdperk van handheld gaming. Het eenvoudige, maar effectieve ontwerp heeft een blijvende impact gehad.

Het ontwerp van de joypad heeft bijgedragen aan de evolutie van gamecontrollers. De D-pad, A- en B-knoppen zijn invloedrijk geweest bij de ontwikkeling van toekomstige gamecontrollers.

4.6.1. D-pad

Het meest opvallende kenmerk van de Game Boy joypad is het D-pad (ook wel bekend als de Directional Pad.) Het is een kruisvormige schakelaar die wordt gebruikt voor het navigeren door menu's en het besturen van personages in games.

De D-pad bestaat uit vier richtingen:

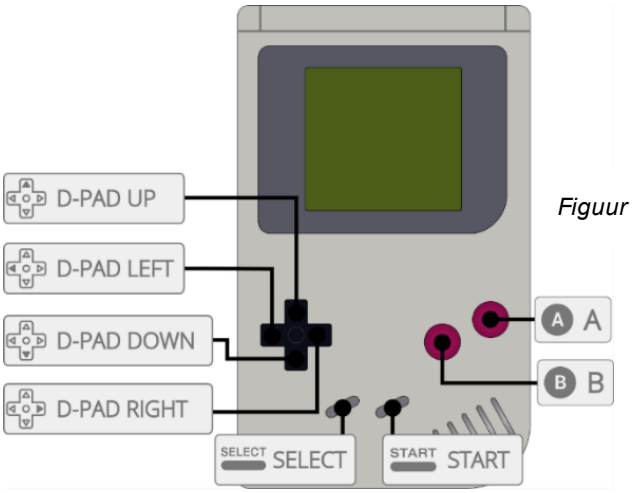
- Omhoog: Druk omhoog op de D-pad om personages naar boven te bewegen of omhoog door menu-opties te navigeren
- Omlaag: Druk omlaag om personages naar beneden te bewegen of omlaag door menu-opties te navigeren
- Links: Druk naar links om personages naar links te bewegen of links door menu-opties te navigeren
- Rechts: druk naar recht om personages naar rechts te bewegen of rechts door menu-opties te navigeren

4.6.2. A en B

Naast het D-pad heeft de Game Boy joypad twee actieknoppen. A en B. Deze knoppen worden gebruikt voor verschillende in-game. De A knop wordt vaak gebruikt voor bevestigingsacties, zoals het selecteren van een menu-item of het bevestigen van keuzes in een game. De B knop wordt gebruikt als een secundaire actieknop voor interactie in games.

4.6.3. Start en select

De joypad bevat ook een Start- en Select knop. Deze Start knop wordt gebruikt om een game te starten, om door menu's te navigeren of om het spel te pauzeren. De Select knop wordt gebruikt voor verschillende doeleinden. Dit kan verschillen van game tot game.



Figuur 11 Game Boy Gamepad

5. Onderzoeksvragen

5.1. Hoe maakt men een GBA-game in C, ondersteund door een proof-of-concept?

Het maken van een Game Boy game in C kan door het gebruik van GBDK (Game Boy Developers Kit). Dit is een pakket dat gebruik maakt van de programmeertaal C. Verder kan je ook gebruik maken van Graphics Tools; Namelijk Game Boy Tile Designer (GBTD) en Game Boy Map Builder (GBMB). Om uiteindelijk je spel te laten runnen, kan je BGB gebruiken. Dit is een emulator die is ontworpen voor het runnen van Game Boy games.

5.1.1. GBDK

De Game Boy Developer Kit is een ontwikkelingsomgeving die speciaal is ontworpen voor het schrijven van games en programma's voor de Nintendo Game Boy. GBDK biedt een set tools en bibliotheken aan om het ontwikkelingsproces te vergemakkelijken en de complexiteit van het programmeren op het Game Boy-platform te verminderen. GBDK is voornamelijk gericht op C-programmering.

Deze kit kan ook voor andere consoles gebruikt worden:

- Analogue Pocket (AP)
- Sega Master System (SMS)
- Game Gear (GG)
- Mega Duck (DUCK)
- …

5.1.1.1. Compiler

GBDK maakt gebruik van een Small Device C Compiler (SDCC) als de kerncompiler. SDCC is een open-source C-compiler die geschikt is voor het ontwikkelen van software voor beperkte resources. Deze compiler zet C-code om in machinetaal die later kan worden uitgevoerd op een Game Boy. Verder is GBDK geconfigureerd om optimale output te genereren voor de Game Boy-architectuur. Het ondersteunt de Z80-achtige chipset die de Game Boy CPU gebruikt.

5.1.1.2. Library

GRAPHICS

De graphics library bevat functies voor het werken met grafische elementen op de Game Boy, waaronder sprites, tiles en achtergronden.

INPUT

De input library biedt functies voor het verwerken van knopinvoer van de speler. Hiermee kun je reageren op de toetsaanslagen van de joypad.

SOUND

De sound library ondersteunt geluidsweergave met de Game Boy-geluidschips. Hiermee kan je geluidseffecten en muziek in je games implementeren

5.1.1.3. Linker

GBDK bevat een linker die de verschillende onderdelen van een Game Boy-programma samenvoegt in een uitvoerbaar bestand dat op de Game Boy kan worden geladen.

5.1.1.4. Debugger

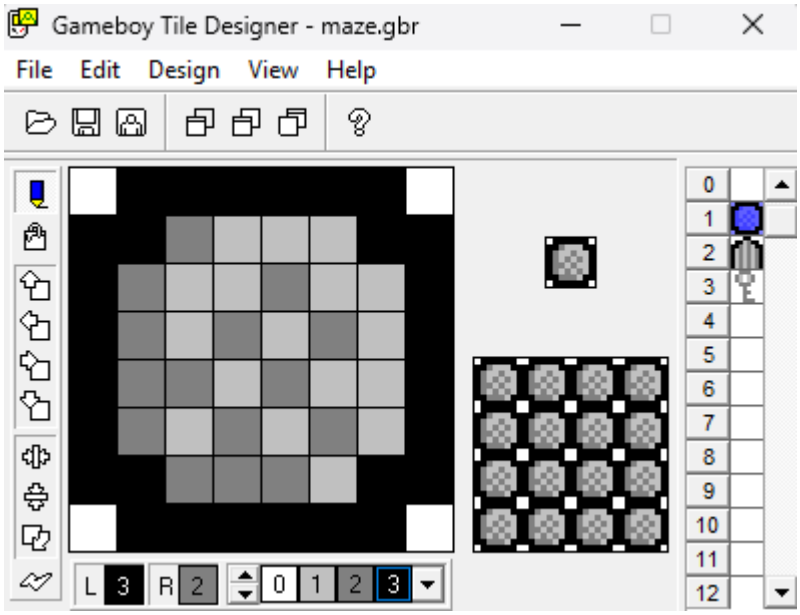
Er zijn verschillende tools en emulators die compatibel zijn met GBDK en kunnen worden gebruikt voor het debuggen van Game Boy-software. Deze tools helpen ontwikkelaars bij het analyseren van code en het opsporen van fouten.

5.1.2. GBTD

GBTD staat voor “Game Boy Tile Designer”. Het is een grafische tool die speciaal is ontwikkeld om het ontwerpen van tiles te vergemakkelijken.

GBTD biedt pixelniveau bewerkingsmogelijkheden waardoor individuele pixels kunnen worden bewerkt om gedetailleerde tiles te maken. Met deze tool kan je zowel voor de originele Game Boy als voor de Game Boy Color tiles designen.

Het implementeren van je eigen tiles is een fluitje van een cent. GBTD kan je design exporteren in .asm of in .c extensie. Het genereert ook automatisch een include- en headerbestand.

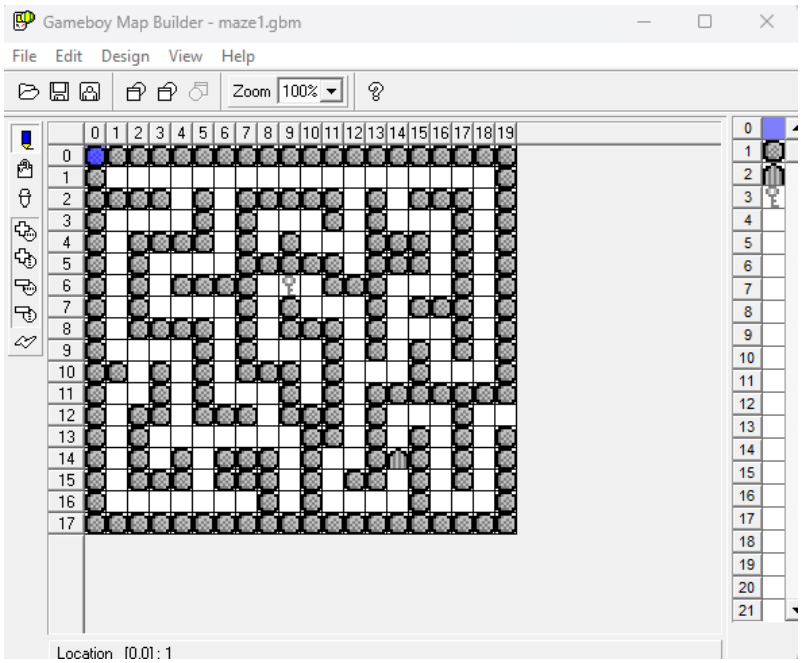


Figuur 12 Game Boy Tile Designer

5.1.3. GBMB

Met Game Boy Map Builder kun je kaarten maken die je in je eigen Game Boy applicatie kunt verwerken. GBMB bied je een groot canvas aan van maximaal 1024x1024 tiles waarop je tiles kunt tekenen die gemaakt zijn met GBTD. Dit wordt meestal in combinatie met GBTD gebruikt. Je kan namelijk makkelijk je zelfontworpen tiles importeren om er zo een kaart van te maken.

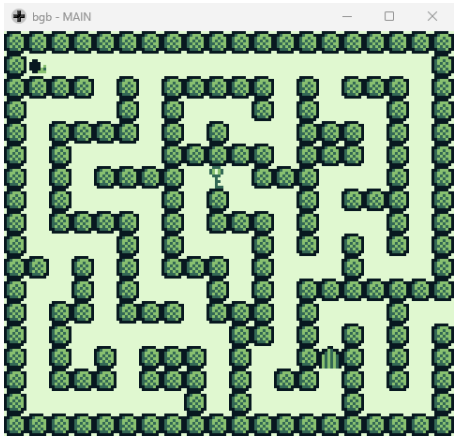
Net zoals GBTD kan je je ontwerpen exporteren in verschillende extensies. Ook het automatisch genereren van een include- en headerbestand is hier aanwezig.



Figuur 13 Game Boy Map Builder

5.1.4. BGB

BGB is een Game Boy emulator. Dit is een programma waarmee je Game Boy games op een pc kunt spelen. Het ondersteunt alle functies die op een fysieke Game Boy zitten. Zoals de gamepad, het geluid en de graphics. Daarnaast bevat het een debugger waarmee je de emulatie kunt analyseren en cheatcodes kunt aanmaken. Omdat BGB een hoge nauwkeurigheid heeft, zal een applicatie dat werkt in BGB waarschijnlijk ook werken op een fysieke Game Boy.



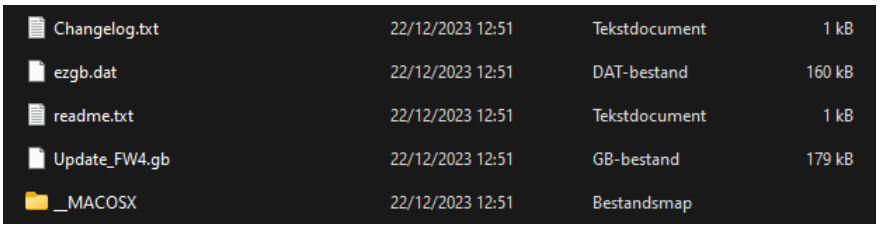
Figuur 14 BGB Emulator

5.1.5. Cartridge

Uiteraard kan je ook je spel op een fysieke Game Boy spelen. Hiervoor zal je wel een speciale cartridge moeten aanschaffen. Voor een originele Game Boy is een EZ-Flash Junior een goede keuze. Deze cartridge stelt je in staat om je eigen game te flashen zodat je je game op een echte Game Boy kan spelen.

Figuur 15 EZ-Flash Junior

Het gebruik is enorm simpel; Eerst en vooral moet je op de officiële webpagina van EZ-Flash de juiste firmware downloaden. (https://www.ezflash.cn/download/) In deze folder vind je aan aantal bestanden.



Figuur 16 EZ-Flash Junior firmware

Het bestand dat nodig is om je game te spelen is “ezgb.dat”. Dit bestand kan je simpelweg kopiëren naar je micro SD kaart. Hierna kan je ook je game(s) op je micro SD kaart zetten. Als alles goed is verlopen kan je de SD kaart in de Game Boy cartridge stoppen. Vervolgens plaats je de cartridge in je Game Boy en kan je je zelfgemaakte game spelen op een fysieke Game Boy.

5.2. Wat is de werking van de code en de interactie met de (emulated) hardware?

Voor dit project heb ik besloten om een “maze” game te maken. De reden hierachter is omdat er een aantal basiselementen aan bod komen.

- Sprites
- Joypad
- Background
- Background Collision

Het is een simpel voorbeeld maar het geeft wel de basis weer. Als je deze onderdelen onder de knie hebt, kom je al een heel eind.

5.2.1. Werking

De werking van het spel is relatief simpel. Als de game opstart, zal eerst het doolhof en de speler ingeladen worden. Het doel is om de uitgang te bereiken. Om het wat moeilijker te maken zal je eerst door de deur moeten gaan. Om deze deur te openen heb je uiteraard een sleutel nodig. Eens je de deur hebt geopend, kan je naar de uitgang gaan.

Onderstaande link zal een video openen waarmee je de code in werking ziet.

https://youtu.be/AkkkIASvocs

5.2.2. Code

```
//include necessary libs
#include <gb/gb.h>
#include <stdio.h>
#include "MazeSprites.c"
#include "MazeMap.c"
#include "Snail.c"

//variable declaration
const char blankmap[1] = {0x00};    //array with an empty tile, used for
grabbing the key and opening the door
UINT8 playerlocation[2];            //array with the positions for the sprite
(player)
UBYTE debug, haskey, gamerunning;
//wait function
void performantdelay(UINT8 numloops){
    UINT8 i;
    //loop
    for(i = 0; i < numloops; i++){
        //vertical blanking period --> https://gbdk-2020.github.io/gbdk-
2020/docs/gbdk_manual.pdf
        wait_vbl_done();
    }
}
```

```
//check if player can move to a new position based on the map. return value
indicates whether the movement is possible
UBYTE canplayermove(UINT8 newplayerx, UINT8 newplayery){
    //local variables
    UINT16 indexTLx, indexTLy, tileindexTL;
    UBYTE result;

    //calculate the x-coordinate index with an offset of 8 pixels
    indexTLx = (newplayerx - 8) / 8;
    //calculate the y-coordinate index with an offset of 16 pixels
    indexTLy = (newplayery - 16) / 8;
    //calculate the overall tile index with a 20-tile wide map
    tileindexTL = 20 * indexTLy + indexTLx;

    //get the location of a specific place on the map, used for debugging
    /*
    if(debug){
        printf("%u %u\n",(UINT16)(newplayerx),(UINT16)(newplayery));
        printf("%u %u
%u\n",(UINT16)indexTLx,(UINT16)indexTLy,(UINT16)tileindexTL);
    }
    */
    //check if the tile on the map is black
    result = MazeMap[tileindexTL] == blankmap[0];
    // collect key
    if(tileindexTL==321){
        //change the key tile to a black tile
        set_bkg_tiles(1,16,1,1,blankmap);
        haskey = 1;
        result = 1;
    }
    // open door
    else if(tileindexTL==263 && haskey){
        //change the door tile to a black tile
        set_bkg_tiles(3,13,1,1,blankmap);
        result = 1;
    }
    // finish game
    else if(tileindexTL==340){
        //end of the game
        gamerunning = 0;
        HIDE_SPRITES;
        printf("\n \n \n \n \n \n \n \n      YOU WIN!");
        result = 1;
    }
    return result;
}
```

```
//animates the sprite on the screen for a smooth experience
void animatesprite(UINT8 spriteindex, INT8 movex, INT8 movey){
    while(movex!=0){
        //scroll sprite horizontally
        /*
        movex < 0 ? -1 : 1 --> check if movex is less than zero. If true (?),
scroll to the left (-1). otherwise (:), scroll to the right (1)
        */
        scroll_sprite(spriteindex, movex < 0 ? -1 : 1, 0);
        movex += (movex < 0 ? 1 : -1);
        wait_vbl_done();
    }
    while(movey!=0){
        //scroll sprite vertically
        scroll_sprite(spriteindex, 0, movey < 0 ? -1 : 1);
        movey += movey < 0 ? 1 : -1;
        wait_vbl_done();
    }
}
void main(){
    //set the background data
    set_bkg_data(0, 4, MazeSprites);
    set_bkg_tiles(0, 0, 20, 18, MazeMap);
    //set the sprite data
    set_sprite_data(0, 1, Snail);
    set_sprite_tile(0,0);
    //starting position of the player
    playerlocation[0] = 16;
    playerlocation[1] = 24;
    //place the sprite to the starting position
    move_sprite(0,playerlocation[0],playerlocation[1]);
    //initializes and displays the sprite and background on the screen
    gamerunning = 1;
    SHOW_SPRITES;
    SHOW_BKG;
    DISPLAY_ON;
    while(gamerunning){
        /*
        //this is only used for debugging - check if the A button is pressed
        if(joypad() & J_A){
            debug = 1;
        }
        */
        //check if D-pad left is pressed
        if(joypad() & J_LEFT){
            //check if sprite can move to the left by 8 pixels based on the
current position
```

```
            if(canplayermove(playerlocation[0]-8,playerlocation[1])){
                //subtract 8 pixels of the current position on x-axis
                playerlocation[0] -= 8;
                animatesprite(0,-8,0);
            }
        }
        //check if D-pad right is pressed
        else if(joypad() & J_RIGHT){
            //check if sprite can move to the right by 8 pixels based on the
current position
            if(canplayermove(playerlocation[0]+8,playerlocation[1])){
                //add 8 pixels to the current position on x-axis
                playerlocation[0] += 8;
                animatesprite(0,8,0);
            }
        }
        //check if D-pad up is pressed
        else if(joypad() & J_UP){
            //check if sprite can move up by 8 pixels based on the current
position
            if(canplayermove(playerlocation[0],playerlocation[1]-8)){
                //subtract 8 pixels of the current position on y-axis
                playerlocation[1] -= 8;
                animatesprite(0,0,-8);
            }
        }
        //check if D-pad down is pressed
        else if(joypad() & J_DOWN){
            //check if sprite can move down by 8 pixels based on the current
position
            if(canplayermove(playerlocation[0],playerlocation[1]+8)){
                //add 8 pixels to the current position on y-axis
                playerlocation[1] += 8;
                animatesprite(0,0,8);
            }
        }
        //wait between each button press
        performantdelay(6);
    }
}
```

5.2.3. Uitbreidingen

Uiteraard zijn er enorm veel mogelijke uitbreidingen op deze game. Er kunnen bijvoorbeeld meerdere levels worden toegevoegd. Er kunnen vijandige karakters rondlopen in het doolhof. Je zou kunnen spelen tegen de tijd. Een mist die boven het doolhof hangt waardoor je zicht beperkt is enzovoort.

5.3. Op welke manier kunnen extra features toegevoegd worden aan een GB(A)-game en wat effect heeft dit dan op de complexiteit van de codebase?

Het effect van extra features op de complexiteit van de codebase hangt af van de specifieke implementatie en het niveau van diepgang. Over het algemeen zal het toevoegen van meer geavanceerde features de codebase complexer maken. Er zijn talloze mogelijkheden voor extra features. Zoals eerder besproken zijn er zeer veel mogelijke uitbreidingen die je kan implementeren in je games. Dit hangt natuurlijk ook af van welke game je gemaakt hebt. Verder zijn er ook nog uitbreidingen die geen betrekking hebben met de code.

5.3.1. Codebased features

5.3.1.1. Multiplayer

Je kan met behulp van een *link-cable* twee Game Boy-systemen rechtstreeks met elkaar verbinden. Zo is het mogelijk om je favoriete game samen te spelen met een vriend(in). Later is ook een linkkabel uitgekomen die je in staat stelt om met vier personen samen te spelen. Alle spelers hebben wel een exemplaar van het spel nodig om samen te kunnen spelen. Als een game multiplayer ondersteunt, zal dit te zien zijn aan het pictogram of label. Er zal een symbool te zien zijn met een *link-cable* of een afbeelding van meerdere Game Boys.

Figuur 17 Multiplayer game

Het maken van je eigen multiplayer-game kan een uitdaging zijn aangezien het meer kennis vraagt op gebied van programmeren en de hardware van de Game Boy.

5.3.1.2. Audio

Officiële games hebben vrijwel altijd muziek en/of geluidseffecten. Dit geeft meer plezier aan de game-ervaring.

Op het internet zijn een aantal tools te vinden waarmee je je eigen muziek kan creëren. Een voorbeeld hiervan is GBT-player. Dit programma maakt ook gebruik van GBDK wat het implementeren van je muziek eenvoudig maakt. Verder is het ook mogelijk om geluidseffecten toe te voegen bij bepaalde acties of prestaties.

5.3.1.3. Color

Als je een game hebt gemaakt voor een Game Boy met monochroom scherm heb je maar 4 grijstinten waarmee je kan werken. Het is mogelijk om je game een upgrade te geven door je applicatie in kleur te maken voor een Game Boy met een kleurenscherm. Dit biedt enorm veel mogelijkheden aangezien je veel meer detail kan verwerken in je game.

5.3.2. Non-codebased features

5.3.2.1. Battery Pack

Het Battery Pack was een pakket dat beschikbaar was voor de originele Game Boy om de levensduur van de batterij te verlengen.

Zonder het batterijpakket kan je ongeveer 10 tot 15 uur spelen. Als je oplaadbare batterijen gebruikt kan de speeltijd verkorten naar 6 tot 10 uur. Het batterijpakket gebruikt 6 AA-batterijen en kan ongeveer 20 uur toevoegen aan je speeltijd.

Omdat veel van deze externe batterijaccessoires werden geproduceerd door derden, waren er vele varianten. Zo werden er ook varianten op de markt gebracht met verschillende batterij-capaciteiten en andere Game Boy versies.

5.3.2.2. Light Boy

Light Boy is een accessoire dat achtergrondverlichting toevoegt aan het originele Game Boy-scherm. De Light Boy kan je boven op je Game Boy klikken. Dit stelde gebruikers met een originele Game Boy in staat om ook in donkere omgevingen te gamen. Latere versies van de Game Boy (zoals de Game Boy Advance SP) hadden ingebouwde achtergrondverlichting waardoor de Light Boy overbodig werd.

Figuur 18 Light Boy

5.3.3. Game Boy Accessoires

Nintendo heeft ook nog enkele andere officiële Game Boy accessoires. Zo kan er een printer of camera worden geïmplementeerd. Met de camera-module kunnen gebruikers zwart-witfoto's maken. Deze camera wordt vaak gebruikt in combinatie met de Game Boy Printer. De Game Boy Printer wordt gebruikt om zwart-witafbeeldingen van Game Boy Camera-foto's af te drukken.

Figuur 19 Game Boy Camera

Figuur 20 Game Boy Printer

6. Besluit

Het ontwikkelen van een Game Boy game was een leerzame ervaring. Gedurende dit project heb ik niet alleen technische vaardigheden verworven, maar ook een dieper begrip gekregen van de complexiteit en creativiteit die komt kijken bij het maken van een interactieve en boeiende spelervaring.

Het begrijpen van de Game Boy-architectuur en het gebruik van de ontwikkeltools was een essentieel onderdeel van dit project. Door de technische aspecten te omarmen en te experimenteren met verschillende elementen, heb ik niet alleen mijn vaardigheden verbeterd, maar ook een dieper begrip gekregen van de complexiteit van het creëren van een functionerende game op dit platform.

Dit project heeft ook het creativiteit en probleemoplossend vermogen uitgedaagd. Het proces van het bedenken van een boeiend verhaal, het ontwerpen van een uitdagend level en het creëren van aantrekkelijke grafische elementen heeft me bewust gemaakt van de vele aspecten die bijdragen aan een succesvolle game-ervaring.

Daarnaast is het buitengewoon indrukwekkend dat de game-ontwikkeling ruim 25 jaar geleden al in staat was om zo’n technologische hoogstandjes te bereiken. In een tijd waarin de technologische mogelijkheden nog in de kinderschoenen stonden, hebben zij de fundamenten gelegd voor wat nu een bloeiende en evoluerende industrie is.

7. Bronnen

Games Database. (sd). Opgehaald van Games Database: https://www.gamesdatabase.org/

gbdev.io. (sd). Opgehaald van gbdev.io: https://gbdev.io/resources.html

gbdk. (2023, Augustus 9). Opgehaald van https://gbdk-2020.github.io/gbdk-2020/docs/gbdk_manual.pdf

gbdk-2020. (2023, Augustus 9). Opgehaald van https://gbdk-2020.github.io/gbdk-2020/docs/api/index.html

HarrisonAllen. (2023, Juni). Opgehaald van GitHub: https://github.com/HarrisonAllen/pebble-gbc-graphics?tab=readme-ov-file

Libretro Docs. (2023, December 8). Opgehaald van Libretro: https://docs.libretro.com/image/controller/gb.png

Lyth. (sd). *BGB*. Opgehaald van BGB: https://bgb.bircd.org/

OpenAI. (sd). Opgehaald van ChatGPT: https://chat.openai.com/chat

Pan Docs. (sd). Opgehaald van gbdev: https://gbdev.io/pandocs/

Wallace, A. (2022, Juli 8). *all game boy models*. Opgehaald van retrododo: https://retrododo.com/all-game-boy-models/

Wikimedia. (2023, December 15). Opgehaald van Wikimedia: https://commons.wikimedia.org/

CONTACT

Gabriëls Jesse | Student

R0882112@student.thomasmore.be

Tel. + 32 496 10 09 59

VOLG ONS

www.thomasmore.be

fb.com/ThomasMoreBE

#WeAreMore

THOMAS MORE