
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# NÁVRH A IMPLEMENTACE HRY PRO KONZOLI NINTENDO GAME BOY

**DESIGN AND IMPLEMENTATION OF A GAME FOR NINTENDO GAME BOY**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **ŠTĚPÁN KREJČÍŘ**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Doc. Ing. MARTIN ČADÍK, Ph.D.**
**SUPERVISOR**



**BRNO 2022**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

# Zadání bakalářské práce


Student: **Krejčíř Štěpán**
Program: Informační technologie
Název: **Návrh a implementace hry pro konzoli Nintendo Game Boy**
**Design and Implementation of a Game for Nintendo Game Boy**
Kategorie: Počítačová grafika

Zadání:

1. Prozkoumejte historii a aktuální stav návrhu her pro herní konzole.
2. Seznamte se s konzolí Nintendo Game Boy s ohledem na tvorbu her.
3. Navrhněte novou hru na základě získaných poznatků.
4. Navrženou hru implementujte a v postupných iteracích experimentuje s různými návrhy a herními mechanikami.
5. Prezentujte výslednou hru formou plakátu a krátkého videa.

Literatura:

- Koster, Raph. Theory of fun for game design. O'Reilly Media, Inc., 2013.
- Schell, Jesse. The Art of Game Design: A book of lenses. CRC press, 2008.
- Unity Learn. Unity, https://learn.unity.com/.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Čadík Martin, doc. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

# Abstrakt

Tato bakalářská práce čtenáře stručně seznamuje s kontextem, v jakém byla na trh uvedena přenosná herní konzole Nintendo Game Boy. Zaměřuje se na způsob vývoje pro tuto konzoli, uvádí vlastnosti konzole, které je důležité znát pro psaní efektivního kódu a navrhuje řešení k některým častým problémů. V praktické části je popsán návrh a implementace rytmické hry pro Game Boy s využitím jazyka C a toolkitu GBDK-2020.



# Abstract

This bachelor's thesis briefly introduces the reader to the context surrounding the launch of the handheld console Nintendo Game Boy. It focuses on the development process for this console, lists the console's quirks that are essential to know in order to write effective code, and suggests solutions to some common problems. The practical part describes the design and implementation of a rhythm game for Game Boy using C language and GBDK-2020 toolkit.



# Klíčová slova

hra, Nintendo, Game Boy, herní konzole, handheld, 8-bit, návrh her, vývoj her, GBDK

# Keywords

game, Nintendo, Game Boy, game console, handheld, 8-bit, game design, game development, GBDK



# Citace

KREJČÍŘ, Štěpán. *Návrh a implementace hry pro konzoli Nintendo Game Boy.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Martin Čadík, Ph.D.

# Návrh a implementace hry pro konzoli Nintendo Game Boy

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana docenta Martina Čadíka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Štěpán Krejčíř
7. května 2022


## Poděkování

Rád bych poděkoval panu doc. Ing. Martinu Čadíkovi, Ph.D. za vedení práce a ochotu online i při osobních konzultacích. Dále komunitě GBdev za rady při řešení některých problémů.

# Obsah

# Kapitola 1

# Úvod

Retro hry jsou stále populární. Můžeme spekulovat o tom, zda za to může nostalgie, nové nezávislé hry, které v mnoha případech volí pixelovanou stylizaci z estetických či budgetových důvodů, nebo tehdy hry „byly prostě lepší“. Faktem ale zůstává, že v posledních letech je trendem posílat na trh starší herní konzole v nových, zmenšený podobách – namátkou například *NES Classic Mini* nebo *Sega Genesis Mini* – a ceny za původní hardware a hry neklesají, v nemalém množství případů mnohonásobně převyšují původní cenovku [5].

Hry pro starší systémy pro mnoho lidí zkrátka mají nezaměnitelné kouzlo i napříč – nebo snad díky – svým nedokonalostem a mnohým omezením a okolo retro konzolí se vytvořila relativně početná, aktivní vývojářská komunita, jež konzole nadále zásobuje svými neoficiálními (tzv. homebrew) hrami dlouho po tom, co byla ukončena oficiální podpora původních výrobců.

Tato práce si klade za cíl představit herní konzole z dodnes oblíbené čtvrté generace a prozkoumat způsob a praktiky vývoje softwaru pro handheld Game Boy od firmy Nintendo (dále také zkracován jako GB) z roku 1989 – jednoho ze zástupců této generace. Výstupem práce je hra napsaná v jazyce C, hratelná jak v emulátoru, tak především na původním handheldu a všech jeho zpětně kompatibilních následnících.

Kapitola 2 shrnuje a srovnává konzole čtvrté generace, které vyšly v podobném časovém rozmezí jako Game Boy. Krátce se také věnuje historii Game Boye a jeho hlavnímu návrháři.

V kapitole 3 je čtenář seznámen s bližšími hardwarovými specifikacemi handheldu a s tím spojenými omezeními. Dále je kapitola zaměřena na nejdůležitější principy vývoje softwaru pro handheld Game Boy a na praktiky, které je dobré při psaní kódu dodržovat. Kapitola také představuje některé z vývojářských nástrojů, jež se dnes k vývoji používají.

Kapitola 4 hovoří o návrhu konkrétní hry, popisuje hlavní inspirace a objasňuje některá designová rozhodnutí. Následující kapitola 5 popisuje nejzajímavější detaily z implementace. Kapitola 6 potom krátce informuje o zdařilosti použitých optimalizací a shrnuje závěry z uživatelského testování výsledné hry.

# Kapitola 2

# Herní konzole čtvrté generace

V sekci budou představeny nejvýznamější videoherní konzole tzv. čtvrté generace. Generace převážně 16bitových konzolí, která trvala přibližně od roku 1987 do roku 1994 [3], byla dominována firmami Nintendo a Sega, jež zvládly navázat na úspěch svých herních zařízení z předchozí generace.

## 2.1 Domácí konzole čtvrté generace

Domácí konzole jsou herní zařízení s výstupem do zobrazovacího zařízení, nejčastěji televize, a dedikovaným ovladačem určeným k interakci s hrami. Ty byly pro tyto konzole distribuovány zejména na vyměnitelných kazetách (anglicky cartridge), které se skládaly z integrovaných obvodů typu ROM. V dnešní době byly herní cartridge nahrazeny převážně optickými disky.

### NEC PC Engine

V USA známý pod názvem TurboGrafx-16, PC Engine je s datem vydání v roce 1987 považován za první herní konzoli čtvrté generace. Vyvinut společnostmi Hudson Soft a NEC Home Electronics nabízel 8bitovou CPU jednotku, dva 16bitové grafické procesory schopné zobrazit až 482 barev najednou (z 512 dostupných) a 8 KB paměti RAM [3].

PC Engine se těšil úspěchu v domácím Japonsku, ale nepodařilo se mu proniknout na americký trh, kde byl zastíněn firmami Nintendo a Sega zejména kvůli nižším nákladům na marketing a neochotě vývojářů systém podporovat svými hrami [12].

### Sega Mega Drive

Mega Drive poprvé vyšel jako nástupce konzole Master System v Japonsku v roce 1989. Byl první komerčně úspěšnou herní konzolí s 16bitovým procesorem [8]. Dále měl k dispozici 64 KB paměti RAM, ale narozdíl od PC Engine zvládal najednou zobrazit pouze 61 barev z dostupných 512 [3].

Konzoli se v domácím Japonsku nedařilo tolik jako konkurenci. Začala se ale hojně prodávat po uvedení na americký trh, kde byla pod názvem Genesis inzerovaná jako konzole pro náctileté a mladé dospělé, zatímco konkurence v čele s Nintendem se snažila cílit převážně na mladší publikum [23]. Úspěchu také dopomohla postava Sonica. Pro Mega Drive vyšla historicky první hra s tímto ikonickým ježkem, jenž se stal oficiálním maskotem firmy Sega.

### Super Nintendo Entertainment System

Konzole firmy Nintendo, jejíž název je často zkracován na SNES, vyšla nejprve v Japonsku v roce 1990. Navazovala na úspěšnou domácí konzoli třetí generace – Nintendo Entertainment System. SNES byl osazen 16bitovou CPU jednotkou a měl přístup k celým 128 KB RAM paměti. Byl schopen najednou zobrazit 256 barev z celkové palety 32768 barev [3].

Konzole se stala hitem na japonském trhu, kde rychle porazila veškerou konkurenci. Složitou pozici ale měla v USA, kde ji především z počátku škodila agresivní reklamní kampaň firmy Sega. Prvenství nakonec dosáhla i v Americe díky silnému hernímu katalogu [12].

## 2.2 Přenosné konzole čtvrté generace

Přenosné konzole, také zvané handheldy, jsou malá herní zařízení vhodná k hraní na cestách. Zpravidla obsahují vestavěný displej, reproduktory a ovládací prvky. Stejně jako u domácích konzolí, i software pro handheldy je dodáván primárně na vyměnitelných cartridgích.

### Nintendo Game Boy

Předchozí série handheldů od Nintenda s názvem Game & Watch nabízela v každém zařízení pouze jedinou hru, která nešla vyjmout a vyměnit za jinou. Game Boy byl s vydáním v roce 1989 první masově úspěšnou přenosnou herní konzolí s vyměnitelnými cartridgemi [21].

Game Boy měl v porovnání s přenosnými tehdejší doby slabý hardware [22]. Uměl nabídnout pouze 8bitový procesor a 8 KB paměti RAM (později se začala používat i přídavná RAM zabudovaná přímo v herních cartridgích, zvyšujíc celkovou RAM až na 32 KB). Jeho největši slabinou byl nepodsvícený maticový displej schopný zobrazit jen 4 barvy – navíc pouze v odstínech šedi.

Co ztrácel po technické stránce, vynahrazoval svou dostupnou cenovkou, odolnou konstrukcí, v porovnání s konkurencí obrovskou výdrží (výrobce udával až 30 hodin na čtyři AA baterie), ale hlavně bohatým katalogem kvalitních her [13] [22].

Protože podstatná část této bakalářské práce se věnuje právě Game Boyi, jeho historie je detailněji rozebrána v samostatné podkapitole 2.3.

### Atari Lynx

První handheldový počin firmy Atari – konzole Lynx – na trh nastoupila v roce 1989 dva měsíce po Game Boyi, vůči kterému byl nezanedbatelně napřed v rámci hardwaru. 8bitová CPU jednotka, 64 KB RAM, ale především na tehdejší dobu velký, podsvícený displej se schopností zobrazovat barvy – mezi handheldy první na světě [19].

Neúspěch handheldu zapříčinila nepříliš výrazná marketingová kampaň, na handheld velké rozměry, dvojnásobná cena oproti Game Boyi a malá výdrž baterií (4-5 hodin na 6 AA článků). Na Lynx komerčně vyšlo pouhých 76 her a prodeje konzole zůstaly pod očekáváními [19].

### Sega Game Gear

Konzole vydaná v roce 1990 s hardwarem téměř totožným s domácí konzolí Master System ze třetí generace – 8bitovým CPU a 8 KB RAM. Displej nabízel podsvícení a z celkové palety 4096 barev mohl zároveň zobrazit až 32.

Game Gear tedy obsahoval silnější komponenty než Game Boy, ale zaostával za Lynx, což pro Segu nepředstavovalo tak velký problém vzhledem k slabé marketingové kampani firmy Atari.

Díky architektuře podobné Master Systemu pro Game Gear vyšlo velké množství portů her z této konzole. Nových a originálních titulů ale bylo poskrovnu, což, spolu s nízkou výdrží (2-4 hodin na 6 AA baterií), zapříčinilo, že ač se nejednalo o úplný tržní neúspěch, handheld zůstal prodeji daleko za Game Boyem [7].

### NEC PC Engine GT

PC Engine GT vydaný v roce 1990 v Japonsku a USA (kde je známý též jako TurboExpres), se evropskému kontinentu úplně vyhnul. Jednalo se o zmenšenou verzi domácí konzole PC Engine (2.1), obsahující velmi podobné komponenty – modifikovanou verzi 8bitové CPU jednotky z původní konzole a 8 KB RAM. Podsvícený displej měl dvakrát větší rozlišení než konkurence a uměl, stejně jako jeho větší předchůdce, zobrazit až 482 barev z palety 512.

Přes to, že se jednalo o technologicky nejvyspělejší handheld své doby, který navíc hned při vydání podporoval přehrávání kompletní knihovny her z PC Engine, kvůli špatnému marketingu, nejvyšší ceně ze všech kapesních konzolí své doby a pouze 3hodinové výdrži na šest AA baterií PC Engine GT nepřesvědčil dostatečný počet lidí [14].

## 2.3 Z historie Game Boye

Sekce se věnuje okolnostem vzniku GB, jeho životnímu cyklu, odkazu a významu pro budoucnost kapesního hraní.

### Gunpei Yokoi – otec Game Boye

Gunpei Yokoi byl dlouhodobý zaměstnanec firmy Nintendo, který výrazně ovlivnil budoucnost firmy a velkou měrou přispěl k jejímu vzestupu během sedmdesátých a osmdesátých let minulého století.

Razil filozofii používání starších a dobře známých technologií, které mohou být vyráběny masově za nízkou cenu, ale používat tyto technologie novými, inovátorskými způsoby. Zastával názor, že hry a hračky nepotřebují být postaveny na nejmodernějších a drahých komponentech, důležitá je zábava, kterou umí nabídnout [20]. Nintendo se tímto přístupem řídí do jisté míry dodnes.

Yokoi se této filozofie držel i při vývoji jednoduchých handheldů ze série *Game & Watch*, jeho významného milníku, který Nintendo posunul dále na dráhu videoherní společnosti. Tato zařízení ještě neobsahovala vyměnitelné cartridge, jednotlivé verze byly dedikovány vždy pro jednu konkrétní hru. Některé z nich se ovládaly pomocí směrového kříže (tzv. D-padu), ovládacího prvku z dílny Gunpeie, jehož design se na herních ovladačích stále používá.

Od *Game & Watch* už mnoho nechybělo ke Game Boyi. Levné, přenosné herní konzoli s vyměnitelnými hrami, která zpopularizovala kapesní hraní po celém světě [10].

### Životní cyklus Game Boye

Game Boy byl na domácí japonský trh vypuštěn v dubnu roku 1989, do Severní Ameriky v červenci téhož roku a do Evropy se dostal o rok později v září 1990. I přes to, že pro GB

byly na začátku dostupné pouze čtyři hry, z nichž pouze *Super Mario Land* je významná i z dnešního pohledu, se handheldu nedařilo špatně. Z části se na tom mohla podílet dobrá reputace Nintenda po vydání jejich domácí konzole *Nintendo Entertainment System* [12].

Největší zlom však přišel o pár měsíců později, kdy se Nintendu podařilo zajistit exkluzivní práva na kapesní vydání puzzle hry *Tetris*. Tento titul firma začala v Americe a Evropě přikládat ke každému novému handheldu, což se ukázalo jako dobrý tah. Oproti původnímu plánu přikládat *Super Mario Land* se logická hra lépe cílila na širší publikum. *Tetris* oslovil kromě dětí i dospělé a proklestil Game Boyi cestu k úspěchu [10] [12].

Nintendo neváhalo a systém následujících let po vydání dál podporovalo uznávanými tituly ze svých sérií jako *Mario*, *The Legend of Zelda*, *Metroid* a *Kirby*. Úspěch GB nalákal i další vývojařské firmy, které také přispěly značkami ze svých katalogů. Relativně nízké náklady na vývoj ale zapříčinily i lavinu hůře hodnocených a dnes zapomenutých her a softwaru.

Stárnoucí konzole chytila v roce 1996 nový dech díky sérii *Pokémon*. Tato slavná značka odstartovala svůj život právě na GB a prodloužila jeho život nad rámec očekávání Nintenda, které tak mělo dostatek času k soustředění na vývoj neméně úspěšného nástupce *Game Boy Color*.

### Odkaz pro budoucnost

Nintendo svým Game Boyem nevsadilo na nejnovější hardware, ale na zábavnost svých her a dostupnou cenu. Touto filozofií se řídili další desítky let při konstrukci nových produktů z řad *Game Boy* nebo následující *Nintendo DS* a *3DS*, které dále zpopularizovaly kapesní hraní a připravily cestu mobilnímu hraní, jež nakonec nad handheldy převládlo.

V dnešní době má málokdo odvahu vytvořit handheldové zařízení na hraní dedikovaných her (výjimku představuje například *Panic Playdate*). Stoupá ale popularita hybridních zařízení, které dokážou přehrávat plnohodnotné hry z domácích konzolí a počítačů na cestách v čele s *Nintendo Switch* a *Valve Steam Deck*.

# Kapitola 3

# Vývoj softwaru pro Game Boy

Kolem Game Boye se v dnešní době stále pohybuje poměrně aktivní scéna vývojářů [15] a sběratelů poháněná především nostalgií, případně nutností při programování přijmout nové návyky a obcházet limitace hardwaru, což může pro programátora představovat zajímavou výzvu.

Společnost Nintendo od roku 2001 pro handheld nelicencuje a nevydává nové hry. Většina dnešních projektů je proto vydávána digitálně a zdarma ve formě spustitelných ROM souborů pro přehrátí v emulátorech nebo pomocí flash cartridgí[1] na konzoli samotné. Menší část projektů je vydávána také komerčně na samostatných fyzických cartridgích, typicky pomocí crodwfundingových platforem jako Kickstarter nebo specializovaných vydavatelů jako například Incube8 Games. Takto vydaný software nemá oficiální podporu Nintenda. Společnost ale do vývojářské scény nijak nezasahuje, pokud nedojde k porušení jejích vlastnických práv [4].

Tato kapitola představí hardware, z něhož se konzole skládá spolu s jeho specifiky a nedostatky, na které je potřeba při programování myslet, dobré praktiky při psaní softwaru pro GB (v mnohém se liší od obecných dobrých praktik při programování) a některé další obecné vlastnosti handheldu. Zmíněny budou i v současnosti používané vývojové nástroje.

## 3.1 Hardwarové specifikace Game Boye

Game Boy běží na hardwaru, jenž byl i ve své době poněkud zastaralý a přinášel s sebou četná omezení jako například hůře čitelný displej [18]. Tato podkapitola blíže popisuje technické detaily součástek, z nichž se handheld skládá a jaké jsou jejich slabiny.

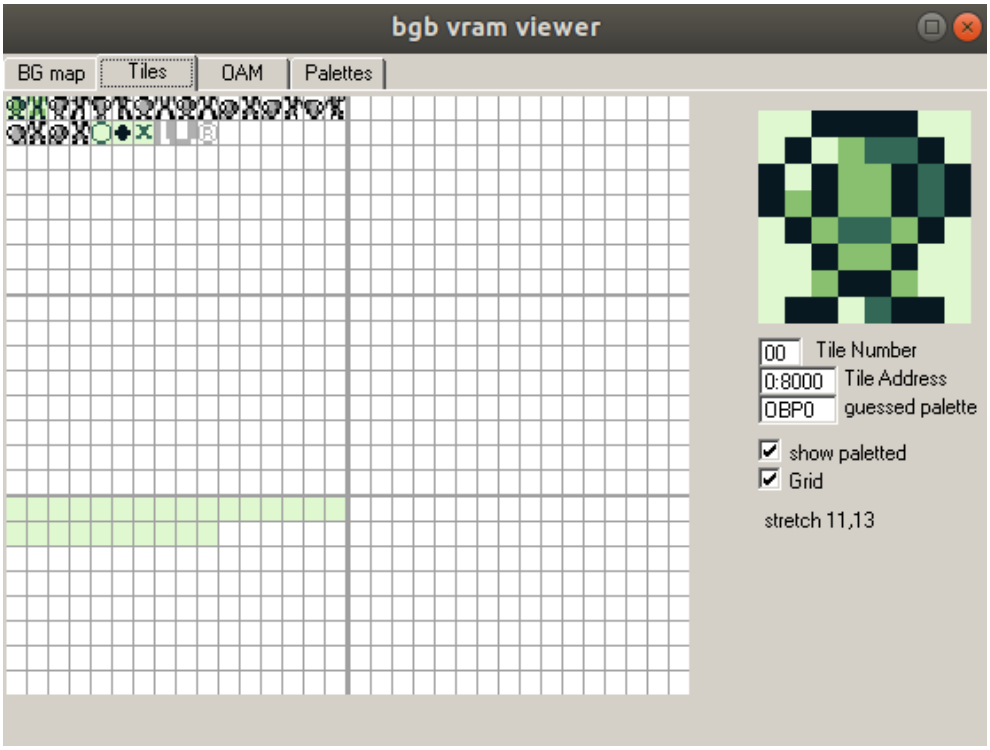
### Centrální procesorová jednotka (CPU)

Centrální procesorovou jednotkou v Game Boyi je *Sharp SM83* založený na procesoru *Zilog Z80*, jímž byla osazena například základní deska počítače *Sinclair ZX Spectrum*.

SM83 operuje na frekvenci 4,19 MHz. Každá operace ale trvá minimálně čtyři hodinové tiky, takže bychom mohli uvažovat o reálné frekvenci 1,05 MHz [17].

Procesor má sedm 8bitových obecně účelných (general purpose) registrů, jeden 8bitový registr určený pro přiznaky (flagy) jako přetečení po vykonání instrukce a dva speciální 16bitové registry pro čítač instrukcí (program counter) a ukazatel na zásobník (stack poin-

[1] Cartridge velikostí podobná těm, na kterých v minulosti vycházely oficiálně licencované hry. Je do ní vložena SD karta obsahující spustitelné ROM soubory.




Obrázek 3.1: Prohlížeč dlaždic nahraných do VRAM v emulátoru BGB.

ter). Na určité dvojice z 8bitových general purpose registrů je možné nahlížet jako na jeden 16bitový registr pro snadnější práci s většími hodnotami.

Jedním z velkých, ale vzhledem k jeho stáří pochopitelných, nedostatků procesoru SM83 je absence instrukcí pro násobení a dělení. Tyto operace tedy mohou zabrat velké množství výpočetního času. Procesor také nenabízí podporu plovoucí řádové čárky.

### Paměti RAM, VRAM, ROM a tabulka OAM

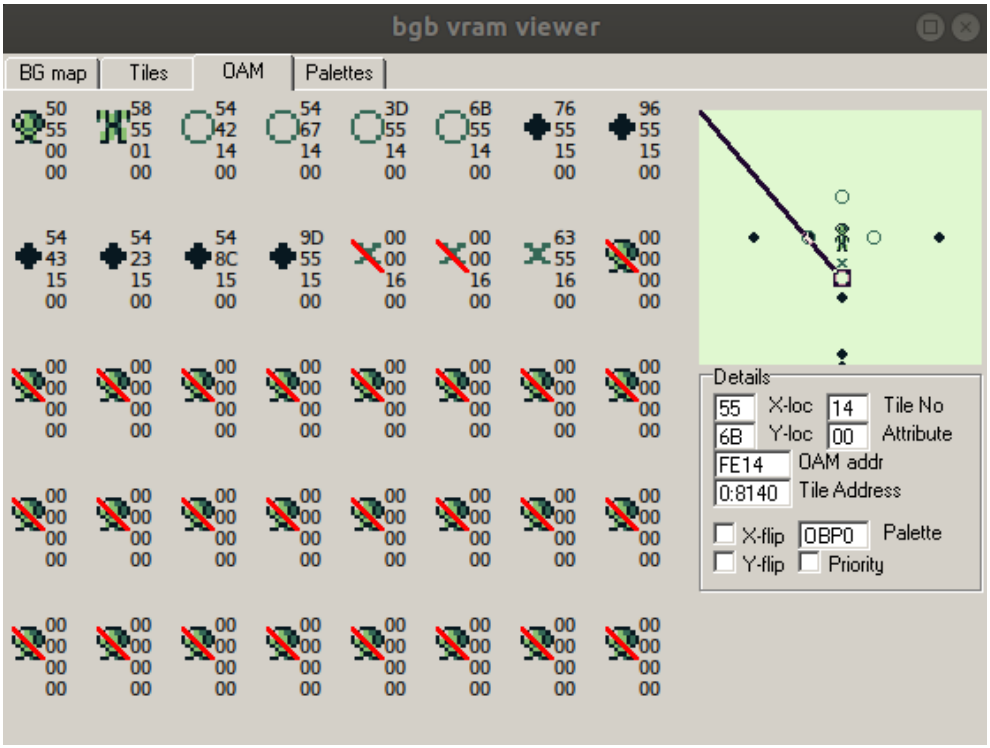
Handheld disponuje 8 KB interní paměti RAM a 8 KB video RAM. Ve VRAM jsou uložena data grafických dlaždic[2], kterých umí pamět najednou pojmout až 256. Na obrázku 3.1 je možné vidět dlaždice nahrané ve VRAM pro sprajt hráče v různých polohách.

OAM (Object Attribute Memory) je tabulka odkazující se na dlaždice. Obsahuje data o herních objektech jako například jejich grafickou reprezentaci (dané offsetem dlaždice ve VRAM) a souřadnice na osách displeje. OAM může nést informace až o 40 grafických objektech současně. Na obrázku 3.2 lze vidět 13 ze zmiňovaných 40 slotů aktuálně zabraných hernímy objekty na obrazovce. Na tomto příkladu je sprajt hráče složen ze 2 dlaždic. Každá z nich v OAM zabírá jeden slot.

Nabízená velikost paměti RAM je dostačující pro velkou část her, ale existují i výjimky, proto některé herní cartridge obsahují svou vlastní externí RAM o velikosti typicky až 128 KB, ke které GB umí přistupovat v úsecích po 8 KB. Tato přidaná RAM zvaná SRAM je často držena pod napětím pomocí baterie a používá se k ukládání postupu ve hře[3].

[2]Dlaždice (angl. tiles) jsou v GB obrázky o velikosti 8x8 pixelů. Ze skládání těchto dlaždic k sobě poté vznikají celé sprajty – grafické reprezentace herních objektů. Detailnější popis se nachází v sekci 3.3.

[3]http://web.archive.org/web/20200109094313/https://eldred.fr/gb-asm-tutorial/memories.html




Obrázek 3.2: Data o herních objektech v OAM zobrazená v emulátoru BGB.

Pamět ROM bývá součástí vyměnitelných cartridgí a je určená pro ukládání grafiky, hudby a kódu. V základu je velká 32 KB. Na rozdíl od RAM je tato kapacita pro mnoho her nedostatečná, tudíž podobně jako v případě externí RAM cartridge mohou disponovat i přidanou ROM. To dává hrám a softwaru přístup typicky až k 8 MB ROM, z nichž v jeden moment může být zpřístupněno maximálně 32 KB.

Proces přístupu k jednotlivým úsekům pamětí ROM a RAM je blíže vysvětlen v podkapitole 3.2.

### Displej

Součástí konzole je malý maticový LCD displej o velikosti 47x43 mm s rozlišením 160x144 pixelů a vykreslovací frekvencí cca. 60 snímků za vteřinu.

Displej GB byl i ve své době považován za jednu z největších slabin handheldu. Nemá žádné podsvícení, tudíž je bez přímého osvětlení viditelný jen obtížně. Nedisponuje ani podporou barev, dostupné jsou pouze černá, bílá (transparentní) a dva odstíny šedi, které vzhledem k zabarvení pozadí displeje působí spíše zeleně.

Obrazovka též trpí na poměrně výrazný ghosting – rychle se pohybující objekty za sebou zanechávají stopu a mohou se rozmazávat vlivem pomalé obnovy pixelů [18].

Další limitací displeje je to, že dokáže spolehlivě zobrazit maximálně 10 sprajtů na jednom řádku. V případě většího počtu sprajty střídavě „problikávají“.

### Audio hardware

Audio čip Game Boye umí přehrávat až 4 zvukové kanály současně. Každý z kanálů je určen pro jiný druh zvuků. Dva kanály s vyššími frekvencemi se nejběžněji používají pro zvukové

efekty hry nebo melodickou hudební linku, třetí s nižšími frekvencemi často zastupuje basově položené tóny a ten poslední umí generovat statický šum o různých frekvencích. Jeho nejčastější využití bývá pro ambientní efekty (např. šum moře) nebo perkuse.

Každý z kanálů má vlastní nastavení hlasitosti, které je možné v průběhu měnit (či kanál úplně ztlumit). Podporován je i stereo zvuk. GB ale má pouze jeden reproduktor, stereo zvuk je proto možné slyšet pouze po připojení sluchátek.

Vydané hry v mnoha případech pro hudbu využívají všechny čtyři kanály, pro přehrávání herních efektů je tedy nutné v jednom kanálu hudbu přerušit, přehrát požadovaný efekt a skladbu obnovit.

### Ovládací prvky

K interakci se softwarem uživateli slouží osmisměrný křížový D-pad, nejběžněji používán k ovládání směru hráčské postavy, a akční tlačítka A, B, Start a Select, z nichž poslední dvě se kvůli jejich umístění hodí spíše na interakci s herním kontextovým menu.

Rozložení ovládacích prvků je tedy velmi podobné tomu z ovladače k domácí konzoli *Nintendo Entertainment System*.

## 3.2 Správa paměti

Vědět jak spravovat pamět je jedním z důležitých aspektů vývoje softwaru pro GB, bez kterého se obejde jen málokterý projekt [9]. Tato sekce představí hlavní způsob správy paměti na GB – tzv. ROM/RAM banking – aby nedocházelo k přetečení do neočekávaných adresových prostor.

### ROM banking

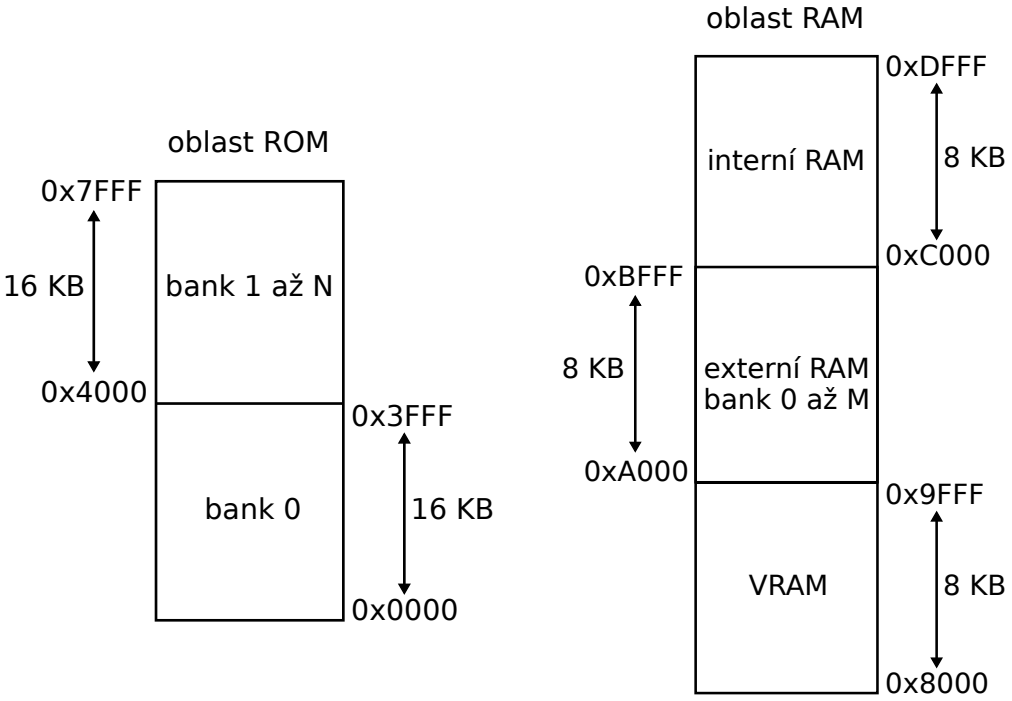
Read-only sekce paměti se nachází na adresách `0x0000` až `0x7FFF` a používá se pro grafické a zvukové assety a pro uložení samotného spustitelného kódu softwaru. Její základní rozsah je 32 KB, což často může být nedostačující. Jak je znázorněno na obrázku 3.3, tato pamět je rozdělená v polovině. Spodní polovina paměti v rozsahu `0x0000` až `0x3FFF` se nazývá *bank 0* a je handheldu přístupná vždy. Měla by tedy obsahovat především nejčastěji používané proměnné a funkce.

Rozsah od `0x4000` do `0x7FFF` je určen pro vyměnitelné banky, a tím pádem nemusí obsahovat ta stejná data po celou dobu běhu programu. Programátor může tyto 16kilobajtové banky střídat dle potřeby. Vhodné použití může být například nahrání mapy konkrétního levelu do jedné z vyměnitelných bank. Tato banka je poté zpřístupněna pouze při načítání daného levelu, kdy je potřeba ke zkopírování mapy do paměti RAM.

Konkrétní počet vyměnitelných bank se nedá blíže specifikovat, protože závisí na paměťových čipech a přepínači bank (Memory Bank Controller – MBC), jímž je každá daná cartridge osazená. Pokročilejší oficiální MBC dovolují spravovat celkově až 8 MB paměti ROM, tedy 512 bank. Při běžném vývoji na GB je obtížné se takové velikosti reálně přiblížit.

### RAM banking

Handheld má v sobě zabudovanou 8kilobajtovou video RAM v adresovém rozsahu od `0x8000` do `0x9FFF` určenou pro grafické dlaždice. Tuto pamět nelze nijak rozšířit.




Obrázek 3.3: Grafické znázornění adresového prostoru pamětí ROM a RAM.

Na adresách `0xC000` až `0xDFFF` se nachází interní 8kilobajtová paměť RAM určená pro výpočty. Tato kapacita většině softwaru stačí. Game Boy ale podporuje taktéž adresování dalších 8 KB v rozsahu `A000` až `0xBFFF`. Na tyto adresy se mapuje případná externí paměť RAM obsažená v cartridgi. Tento rozsah je díky MBC možné přepínat podobně jako horní polovinu paměti ROM s tím rozdílem, že RAM banky mají poloviční velikost (8 KB oproti 16 KB ROM bankám) a pokročilejší oficiální MBC dovolují spravovat nanejvýš 128 KB externí RAM (16 RAM bank oproti 512 ROM bankám).

Externí RAM je mnohdy napájena baterií pro zachování dat i po vypnutí handheldu a využívána pro ukládání hráčova postupu ve hře.

### Přetečení bank

Může se stát například to, že programátor nahraje příliš mnoho read-only dat do banky 0, nadbytečná data přetečou do banky 1, která je poté vyměněna za jinou. GB následně potřebuje přistoupit k datům banky 1, ale v adresovém prostoru `0x4000` až `0x7FFF` se už nachází data jiné banky a dochází k neočekávanému chování.

Kompilátory GB softwaru zpravidla nevarují před přetečením do jiných oblastní paměti, proto vývojářská komunita hojně využívá open-source nástroj *romusage*[4], jenž je schopen odhadnout volné místo v použitých ROM a RAM bankách a varovat před přetečením.

[4] https://github.com/bbbbbr/romusage


Obrázek 3.4: Úvodní obrazovka ze hry *Super Mario Land* rozdělená na jednotlivé dlaždice.

## 3.3 Dlaždicová povaha grafických assetů

Ke správnému pochopení systému je dobré znát také to, jak handheld pracuje s grafikou. Konzole neukládá informace o každém zobrazovaném pixelu. Místo toho je shlukuje do čtverečků velkých 8x8 pixelů – takzvaných dlaždic (anglicky tiles). Tuto techniku ve své době používalo mnoho dalších herních konzolí, neboť umožňovala výrazně šetřit kapacitu VRAM [2].

Pokud bychom ukládali barvu každého zobrazovaného pixelu zvlášť, s rozlišením 160x144 pixelů a dvěma bity pro barvu pixelu bychom se dostali na 46080 bitů, tzn. 5760 bajtů z celkové kapacity VRAM 8192 bajtů, tedy přes dvě třetiny VRAM využité pouze na informace o barvách jednotlivých bodů. Naproti tomu jednu dlaždici lze použít opakovaně a odkazovat se přitom pořád na to stejné místo v paměti, jak lze vidět na příkladu na obrázku 3.4, kde všechny červeně zvýrazněné dlaždice pod cedulí s nápisem odkazují na tu stejnou položku v paměti a zabírají díky tomu mnohonásobně méně dat, než kdyby byl každý pixel uložen individuálně. Ostatně i prázdná plocha bez obarvených pixelů, ze které se skládá většina obrázku, je též nakopírovaná dlaždice, jež má bity nastaveny na nulovou hodnotu. I tady dochází k velkému šetření VRAM.

Jednotlivé dlaždice je možné dále překlopit podél horizontální či vertikální osy (nebo i obou zároveň) v případě, že se nacházejí na sprite layer. Takto překlopené objekty se stále odkazují na originální dlaždici a umožňují další úsporu prostředků. Například k vykreslení bočních pohledů na postavu o rozlišení 8x16 pixelů z levé a pravé strany stačí nakreslit pouze jednu stranu a u druhé využít překlopení, čímž se ušetří 2 dlaždice.

## 3.4 Grafické vrstvy

Dalším z pilířů porozumění vývoje na Game Boy je pochopení, že je jeho displej na logické úrovni rozdělen na vrstvy a jak s nimi pracuje. Každá z jeho tří vrstev se chová jinak a slouží k jinému účelu. Jakýkoliv grafický asset objevující se na displeji musí patřit do jedné z těchto vrstev.

### Sprite layer

Hlavní vrstva, která má prioritu před všemi ostatními, tzn. všechny objekty umístěné v této vrstvě budou překrývat objekty ostatních vrstev.

Všechny objekty na sprite layer mohou být spravovány individuálně. Díky tomu se nejčastěji používá pro sprajty, s nimiž je třeba pohybovat a animovat je nezávisle na sobě. Nejčastěji se jedná o hráčovu postavu, nepřátele, projektily apod.

Na obrázku 3.5 je příklad ze hry *Donkey Kong*, kde jsou objekty umístěné na sprite layer označeny červenými obdélníky.

### Background layer

Název této vrstvy napovídá, že slouží převážně k zobrazování map a pozadí levelů. Pokud to hra vyžaduje, dají se naprogramovat kolize objektů ve sprite layer s objekty v background layer.

Background layer je možné buď celou posouvat s tím, jak úrovní prochází hráč, nebo přepisovat jednotlivá políčka za běhu (není nutné kvůli změně jedné dlaždice kompletně načítat celou mapu znovu) například k rozanimování objektu v pozadí, ale jedná se o méně intuitivní proces než animovat objekt na sprite layer.

Na obrázku 3.5 jsou žebříky, trámy a stojící sudy součástí background layer. V levém dolním rohu je vidět hořící barel, který je taktéž zobrazován v pozadí. Plamen nad barelem je rozpohybován a kvůli jednodušší implementaci je vykreslován na sprite layer.

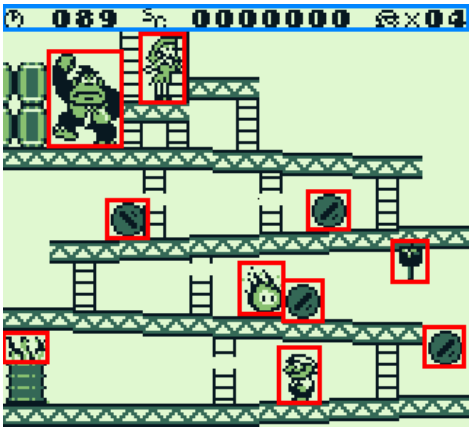
### Window layer

Window layer se nejčastěji používá na zobrazování ukazatelů skóre, životů, kontextových menu a dialogových oken.

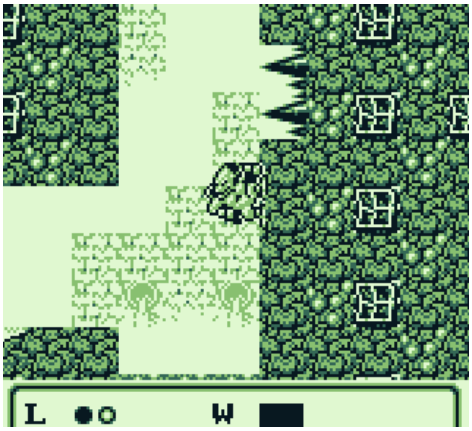
Tato vrstva nemá žádnou možnost transparence a dostává prioritu před pozadím, takže ho může překrýt. Vrstva je ale plovoucí, tudíž je možné ji celou posunout dolů a nechat jen její malý kousek „vykukovat“ ze spodní části obrazovky. To je důvodem, proč skóre a další informace bývají na GB často umisťovány dolů. Jako příklad byla vybrána hra Gargoyle's Quest na obrázku 3.6, kde lze ve spodních rozích vidět i překrytí pozadí bílými pixely vrstvy window layer i v okolí zaobleného obdélníku značícího hranice HUD[5].

Obrázek 3.5 ukazuje window layer vyznačenou modře. Za normálních okolností by tato vrstva při takovémto posunutí nahoru kvůli své neprůhlednosti překryla i celé pozadí. Hra Donkey Kong využívá toho, že konzole displej obnovuje po řádcích shora dolů. Window layer nechává aktivovanou jen pro prvních 8 řádků a pro 9. a další řádky ji hardwarovým přerušením vypne. Stejný způsob je implementován i v projektu vytvořeném pro tuto práci a je blíže popsán v kapitole 5.9.

[5]HUD = head-up display. Ve hrách slouží většinou k zobrazování informací jako například počet životů či nasbírané skóre.




Obrázek 3.5: Screenshot ze hry *Donkey Kong* se zvýrazněnými sprajty.




Obrázek 3.6: Screenshot ze hry *Gargoyle's Quest*.

## 3.5 Barevné palety

Handheld umí pracovat s celkem třemi barevnými paletami. Všechny dovolují zacházet nanejvýš se čtyřmi barvami, a to bílou, světle šedou, tmavě šedou a černou. Dalo by se namítat, že kvůli zbarvení displeje se jedná spíše o odstíny zelené. Přesto na ně v této práci bude odkazováno v odstínech šedi, protože se jednalo o ústupek technologii a následné pokročilejší verze konzole Game Boy zelenou barvu eliminovaly.

I když všechny palety pracují s identickými barvami, jsou v nich lehké rozdíly ve využití i vlastnostech, které budou v této sekci představeny.

### Paleta pozadí

Pro background a window layer je vyhrazena jedna paleta zvaná zkráceně BGP. Její použití je vcelku přímočaré. Paleta se skládá ze čtyř barev, z nichž všechny čtyři je možné použít pro vykreslování pozadí. Programátor může pořadí barev v paletě za běhu libovolně přehazovat nebo dokonce nastavit paletu tak, aby nějakou z barev obsahovala více než jednou (v tom případě se ale musí jiné barvy vzdát).

Přehazování pořadí barev lze využít například k inverzi barevného schématu pozadí bez toho, aby se musely v paměti nacházet ty stejné dlaždice akorát s invertovanými barvami. Toho se využívá mimo jiné k efektům bouřkového blesku, kdy se pozadí na krátkou chvíli přebarví z tmavého na bílé či naopak.

Opakování barev v paletě se dá použít například k postupnému zatmavení obrazu při změně herních obrazovek, kdy postupně všechny barvy palety tmavnou až do doby, než celá obrazovka zčerná. Detaily implementace tohoto efektu jsou přiblíženy v kapitole 5.8.

### Palety sprajtů

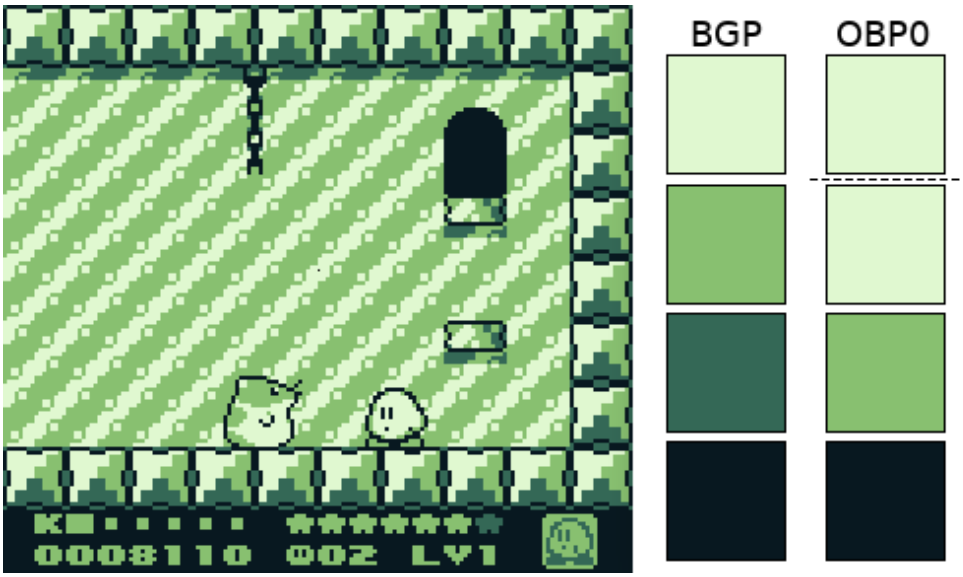
Palety objektů na sprite layer jsou dvě a nazývají se OBP0, respektive OBP1. Jsou na sobě nezávislé, pro každý ze sprajtů může být zvolena kterákoliv z nich a opět je možné jejich barvy za běhu měnit. Na rozdíl od BGP sprajty nemohou využívat všechny čtyři odstíny, protože jeden z nich je vždy vyhrazen pro transparenci a handheld ho nezobrazuje.

Obrázek 3.7 znázorňuje barevné palety použité ve hře *Kirby's Dream Land 2*. OBP1 byla na obrázku zanedbána, protože ji v tento moment žádný sprajt nevyužíval. Paleta

BGP zůstala ve výchozím nastavení a pozadí využívá všechny čtyři barvy. V paletě OBP0 byla odstraněna tmavě šedá barva (dá se povšimnout, že ji postavičky neobsahují) a místo ní se ve schématu dvakrát vyskytuje bílá. Jednou, oddělená čerchovanou čarou, jako barva pro transparenci a jednou jako běžná netransparentní bílá, kterou jsou vybarveny postavičky a díky tomu přes ně neprosvítá pozadí.

Jaká barva je zvolena pro transparenci z vizuálního hlediska nehraje žádnou roli, jelikož tuto barvu konzole nikdy nebude vykreslovat. Je třeba si dávat pozor pouze při kreslení sprajtů. V případech podobných předchozímu příkladu, kde transparentní bílá a neprupustná bílá jsou technicky vzato odlišné barvy, by při záměně došlo k tomu, že tam, kde na obrázku 3.7 mají postavičky bílé pixely, by začala prosvítat grafika pozadí a tam, kde původně byla transparentní (tzn. okolo postaviček, protože sprajty jsou složeny z dlaždic a mají čtvercový tvar), by se objevila neprůhledná bílá.

Hra *Metroid II* na obrázku 3.8 se vyznačuje nejen tím, že používá na GB poměrně nezvyklou barvu pozadí. Pro objekty na sprite layer využívající OBP0 nebyla použita černá (ta oddělená čerchovaně je ve skutečnosti transparentní a nevykresluje se). Navzdory tomu se sprajt parazita na první pohled skládá ze všech čtyř barev. Vývojáři využili jednolitého černého pozadí a sprajt parazita proložili transparentními pixely. Černé části sprajtu jsou tím pádem ve skutečnosti pouze prosvítající background layer.



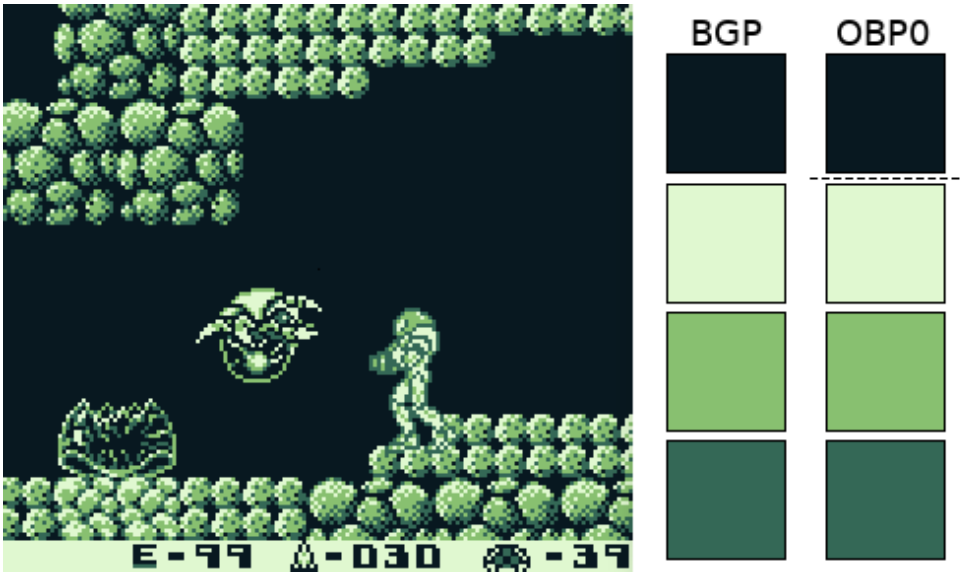

Obrázek 3.7: Ukázka barevných palet ve hře *Kirby's Dream Land 2*.

## 3.6 Programovací jazyky pro vývoj na Game Boy

Pro psaní softwaru pro GB je možné využít assembler, jazyk C, kombinaci obou předchozích možností nebo grafický nástroj. Každá z možností má své výhody i nevýhody, které budou popsány v této sekci.

### Assembler

Assembler je pravděpodobně nejsilnější nástroj k psaní softwaru pro GB. Vzhledem k nízkoúrovňové povaze tohoto jazyka může dobře napsaný assemblerový kód běžet na GB nej-




Obrázek 3.8: Ukázka barevných palet ve hře *Metroid II: Return of Samus*.

rychleji a nejefektivněji. Je proto vhodný pro velké a náročné projekty, kde je třeba šetřit už tak omezenými prostředky, které hardware handheldu nabízí [1].

Psaní dobře optimalizovaného kódu v assembleru je ale náročné a se vzrůstající velikostí projektu a počtem řádku může trpět také přehlednost a udržitelnost kódu. Efektivita je tedy vykoupena zvýšenou uživatelskou náročností.

### Jazyk C

Oproti assembleru je pro člověka čitelnější a udržitelnější. Může ale svádět k používání konstrukcí, které se pro GB nehodí (například rekurzivní funkce). Dostupné kompilátory navíc nemusí kód v jazyce C přeložit do assembleru s takovou efektivitou jako ručně psaný strojový jazyk. Šetření hardwarových prostředků tím pádem může být složitější [1].

Jazyk C však dovoluje vkládání assembleru do C kódu. Toho se dá využít při optimalizacích například tak, že se do assembleru přepíšou pouze nejpoužívanější funkce, čímž se dá dosáhnout kompromisu mezi přehledností kódu a výkonem.

Programovací jazyk C byl s přihlédnutím na všechna pozitiva, negativa a časové možnosti zvolen i pro tuto bakalářskou práci.

### Vizuální skriptování

GB software lze tvořit i bez znalosti předchozích dvou programovacích jazyků pomocí drag-and-drop editoru GB Studio. Jeho detailnější popis se nachází v podkapitole 3.7.1.

## 3.7 Vývojové nástroje

Sekce popisuje některé z nejrozšířenějších nástrojů vhodných pro vývoj GB softwaru v dnešní době.

### 3.7.1 Enginy a toolkity

- **RGBDS** – Aktivně vyvíjený toolkit, které skládají assembler kód do ROM souboru spustitelném na GB. Dle vlastních slov nejpoužívanější sada nástrojů pro vývoj v assembleru, z čehož pramení velká, aktivní komunita a obsáhlá dokumentace.
- **GBDK-2020** – V roce 2020 vzkříšený soubor nástrojů navazující na původní GBDK. Kompiluje kód v jazyce C, ale i v assembleru, do spustitelného ROM souboru pro herní konzole s procesory SM83 či Z80, tedy kromě Game Boye například také pro Sega Master System. Obsahuje API podporující většinu hardwarových funkcí GB. Stejně jako RGBDS, i GBDK-2020 se těší velké komunitě, aktivní podpoře a rozsáhlé dokumentaci. Nástroj byl zvolen i pro tuto bakalářskou práci.
- **ZGB** – Engine postavený na GBDK-2020. Umí za programátora řešit nejběžnější situace jako nahrávání grafických dlaždic do RAM paměti, zobrazování a mizení sprajtů, animace, detekci kolizí apod. Díky tomu dovoluje začít s programováním i bez bližších znalostí fungování hardwaru Game Boye, což ale může vést k nepochopení slabin a některého specifického chování handheldu.
- **GB Studio** – Nástroj umožňující tvorbu GB her bez znalosti programovacích jazyků. Obsahuje grafické rozhraní pro skriptování scén, animací, akcí a zvuků. GB Studio je vysokoúrovňový nástroj a může se stát, že vygenerovaný kód nebude pro danou situaci nejvodnější možné řešení. Také zamýšlené herní mechaniky mohou narazit na limity dostupných skriptovacích možností. GB Studio proto pro zkušenější uživatele nabízí export projektu do kódu v C postaveném na GBDK-2020.

### Emulátory a flash cartridge

- **BGB** – Emulátor vyznačující se vysokou přesností vzhledem k emulovanému hardwaru včetně simulovaného chování původního LCD displeje (maximálně 10 sprajtů na jednom řádku v jeden moment) a co nejpřesnějšího načasování všech operací a hodinového signálu. BGB nabízí mimo jiné také výkonný disassembler (spustitelný binární soubor rozloží na assemblerový kód) a real-time debugger s možností kód přepsat a upravenou hru uložit (tzv. ROM hacking). Umožňuje též nahlížet do paměti VRAM, což výrazně usnadňuje paměťové optimalizace a odstraňování grafických bugů.
- **Emulicious** – Podobné možnosti jako BGB, navíc ale navíc umí také debugging zdrojového kódu v jazyce C pomocí pluginu pro vývojové prostředí Microsoft Visual Studio nebo emulaci dalších konzolí s procesorem Z80.
- **EverDrive-GB** – Pro největší přesnost při testování her je nejlepší je spouštět přímo na originálním hardwaru. K tomu lze využít například EverDrive-GB, mezi jejíž výhody patří nízká energetická spotřeba [11] a podpora přídavných ROM a RAM.

### Grafika

- **RBGFX** – Program je už v základu součástí nástroje RGBDS a slouží ke konvertování obrázků ve formátu PNG do formátu pro Game Boy. RBGFX sám převádí barvy původního obrázku do 4barevné palety handheldu. Výstupní soubor ale není podporován toolkity založenými na jazyce C, které používají vlastní formáty, proto se hodí pouze při vývoji v čistém assembleru.
- **SuperFamiconv** – Má podobné vlastnosti jako RBGFX, ale navíc umí konvertovat i do formátů pro některé další retro konzole.
- **GBTD** – Nástroj umožňující kreslení grafických dlaždic přímo v něm. Vedle toho ale také podporuje import obrázků ve formátech jako PNG a JPG. Dlaždice umí exportovat do formátů jak pro GBDK-2020, tak pro toolkity založené na assembleru jako je RGBDS.
- **GBMB** – Program od stejného autora a s podobným rozhraním, funkcemi a možnostmi exportu jako GBTD. Narozdíl od GBTD, jenž je přizpůsoben tvorbě menších sprajtů (např. herní postavičky) je GBMB zaměřen na tvorbu velkých map, levelů a obrázků sloužící jako pozadí pro hru.

### Hudba

- **GBT Player** – Program pro převod hudby z formátu MOD[6] do vlastního formátu a ovladač, který po integraci do hry konvertuje převedenou hudbu na zvukové instrukce pro GB. GBT Player je primárně vyvíjen pro použití s RGBDS. Existuje zastaralá a nadále neudržovaná verze pro GBDK.
- **hUGEDriver** – Ovladač principem podobný GBT Playeru, ale s aktuální podporou GBDK i RGBDS. Od stejného autora je dostupný program hUGETracker určený ke skládání hudby přímo pro GB, jenž bere v potaz limitace audio hardwaru handheldu (popsané v podkapitole 3.1) a nedovolí nic, co by konzole neuměla replikovat svým audio čipem.

## 3.8 Dobré praktiky při psaní kódu pro Game Boy

Pro psaní efektivního kódu pro Game Boy je dobré dodržovat níže uvedené zásady. Některé z nich jsou v rozporu s dnešními obecnými programovacími návyky, ale mohou být důležité pro šetření omezených zdrojů, kterými Game Boy disponuje.

Vzhledem k tomu, že je tento projekt postaven na jazyce C s toolkitem GBDK-2020, budou některé uvedené praktiky zaměřeny na něj, ale část z nich je uplatitelná i pro psaní assemblerového kódu.

Velká část těchto praktik vychází z toho, že zásobníkové operace na Game Boyi vždy pracují s 16bitovými hodnotami [17], ale procesor konzole je pouze 8bitový. Proto tyto operace, pokud jsou používány příliš často, prodlužují výpočetní čas. Z toho důvodu většina následujících bodů slouží k minimalizaci potřeby zásobníkových operací.

- **Globální proměnné** – Aby k manipulaci se zásobníkem docházelo co nejméně, vyplatí se používat globální proměnné a minimalizovat počet lokálních. Především v často volaných funkcích. S tím souvisí také následující bod.

[6] Rozšířený hudební formát často používaný pro chiptunovou muziku.

- **Krátké seznamy parametrů funkcí** – Při volání funkcí je dobré předávat jim co nejméně argumentů, což je opět možné řešit tím, že nejčastěji používané funkce budou brát vstupní hodnoty z globálních proměnných.
- **Klíčové slovo `inline`** – Krátké a jednorázově používané funkce je výhodné co nejvíce „inlajnovat“. Opět se tím předchází potřebě použití zásobníku.
- **Klíčové slovo `const`** – Je velmi důležité označovat proměnné s neměnnými hodnotami klíčovým slovem `const`. Proměnné bez tohoto slova jsou při práci kopírovány z ROM paměti do RAM, čímž je zbytečně zabírán procesorový čas a místo v operační paměti, pokud nemáme v plánu hodnotu proměnné měnit. Na omezeném systému jako GB tohle může hrát velkou roli.
- **8bitové hodnoty** – Operace s hodnotami většími než 8 bitů vyžadují více instrukcí a prodlužují výpočetní čas.
- **Procházení polí** – Pole o mnoha položkách je typicky rychlejší procházet vytvořením pomocného pointeru a jeho zvyšováním než indexací pomocí hranatých závorek.
- **Násobení, dělení a modulo** – Compiler SDCC, jehož GBDK-2020 využívá, umí optimalizace pro násobení a dělení hodnotami dělitelnými dvěma[7]. S jinými operandy jsou tyto operace drahé a není dobré je používat příliš často, protože pro procesor mohou představovat zbytečnou zátěž.
- **Statická alokace** – GBDK-2020 ve své standardní knihovně obsahuje vlastní implementaci funkce `malloc()` pro dynamickou alokaci paměti. Od jejího použití ale členové vývojářské komunity často odrazují a doporučují spíše statickou alokaci[8]. Například pro herní nepřátele, kteří se při hraní dynamicky objevují a po zneškodnění zase mizí, se často používá staticky alokované pole struktur s vhodně zvolenou velikostí. Všichni nepřátelé tedy zůstávají po celou dobu alokováni v paměti a po jejich zneškodnění je jejich sprajt pouze „schován“ mimo obraz.
- **Rekurzivní funkce** – Rekurzivním funkcím je ve většině případů lepší se zcela vyhnout.

[7] http://fivedots.coe.psu.ac.th/~cj/masd/resources/sdcc-doc/SDCCUdoc-6.html
[8] https://gbdev.gg8.se/forums/viewtopic.php?id=504

# Kapitola 4

# Návrh hry pro Game Boy

Kapitola popisuje základní koncept hry, představuje hlavní inspirace a další podobné hry a vysvětluje některá designová rozhodnutí, k nimž došlo během vývoje.

## 4.1 Inspirace a podobné hry

Hlavní inspirací k tomuto projektu byly různé hudebně a rytmicky zaměřené hry napříč různými platformami. Tyto hry jsou často spjaty se svým soundtrackem, jemuž je přizpůsobeno dění na obrazovce.

### Guitar Hero

Série *Guitar Hero* (na obrázku 4.1), která se stala největší inspirací pro tento projekt, se snaží o velmi zjednodušenou a arkádovou simulaci hry na elektrickou kytaru. Na obrazovce se pohybují barevné body (dále bude používáno označení *nota*) ve směru shora dolů. Hráčův úkol je v moment, kdy nota dorazí do spodní části obrazovky, stisknout specifické tlačítko podle toho, ve kterém z pěti sloupců se daná nota nachází. Rychlost not je uzpůsobena tak, aby správný čas ke stisku tlačítka vždy vycházel do rytmu hudby a za špatné načasování je hráč penalizován. Na tomhle nebo obdobném stylu hraní je postavena velká část rytmických videoher.

### osu!

V současnosti jedna z nejpopulárnějších rytmických her, v níž se noty mohou zobrazit na kterémkoliv místě na obrazovce a jsou nepohyblivé. Hráč musí na notu ve správnou dobu kliknout myší. Nejsilnější stránkou *osu!* je zapojení komunity, jednoduchost vytváření vlastních beatmap[1] a jejich sdílení s ostatními [6].

### Beat Saber

Česká hra *Beat Saber* bere koncept podobný *Guitar Hero* a zasazuje ho do 3D prostoru díky zaměření na VR platformy. Vývojáři prodávají oficiálně licencované skladby s jejich vlastními beatmapami, ale podobně jako *osu!* umožňují i stahování neoficiálních uživatelských map.

[1] V kontextu rytmických videoher se jedná o skladbu s naskriptovaným načasováním jednotlivých interaktivních herních prvků, například časy, ve kterých se na obrazovce má objevit nota, případně na kterém místě herní plochy se má objevit.

Zdroj: https://www.polygon.com/2015/2/24/8099797/guitar-hero-ps4-xbox-one-e3-2015

Obrázek 4.1: Screenshot ze hry *Guitar Hero III: Legends of Rock*.

## 4.2 Návrh nové hry

Sekce 4.1 mohla napovědět, že pro tento projekt byl zvolen žánr rytmických her. Důvodů k tomu bylo hned několik, tím hlavním byla moje osobní záliba v rytmicky a hudebně zaměřených hrách a vášeň k tvorbě hudby obecně. Po průzkumu trhu se mi nepodařilo najít žádnou pro Game Boy komerčně ani nekomerčně vydanou hru podobného stylu. To přisuzuji tomu, že popularita těchto her přišla až později, kdy už byly k dispozici novější platformy. Audio čip GB je nicméně dostatečně výkonný pro přehrávání plnohodnotné chiptunové muziky (čip je více popsaný v kapitole 3.1) a pro hudebně laděné hry není překážkou.

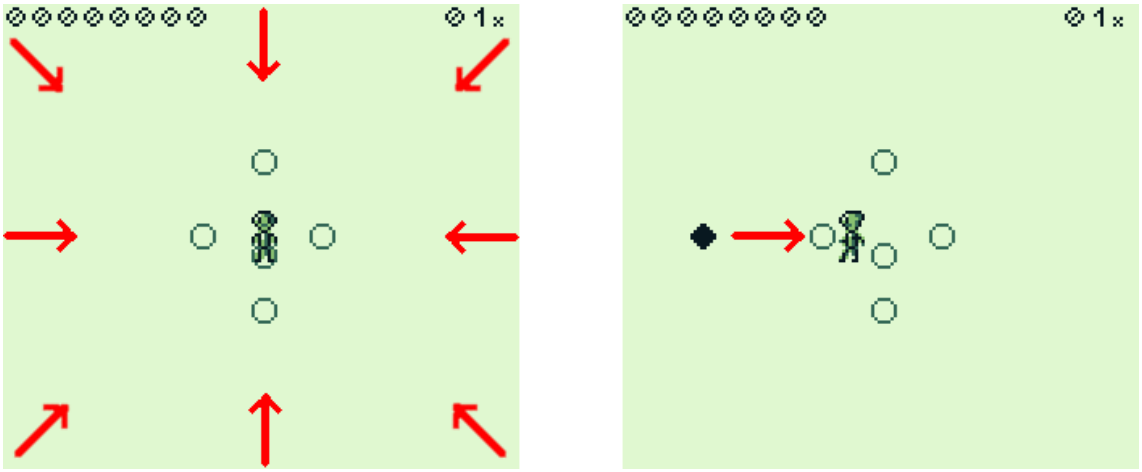
Tato kapitola popisuje postup návrhu této hry a jejích pravidel. Při tom vysvětluje některá rozhodnutí učiněná při vývoji vlivem ústupků cílové platformě, nevhodností původního řešení nebo z jiných důvodů.

### Koncept a souhrn pravidel

Hráčův avatar je zasazen doprostřed obrazovky. Ze severního, jižního, západního a východního okraje směrem k němu přichází noty. Hráč musí přidržením směrového tlačítka správně navolit směr, ze kterého nota přichází a stisknout akční tlačítko v moment, kdy se nota nachází v kolečku daného směru (vizuální znázornění na obrázku 4.2). Zvolený směr je indikován přesunutím avatara do této pozice. Přesun je instantní, avatar se „teleportuje“ kvůli co nejsvižnější odezvě. Dále přichází noty z rohů obrazovky šikmým úhlem směrem k hráči. K získání těchto not hráč nevolí směr, ale stiskne akční tlačítko, když se jeho avatar nachází ve výchozí pozici uprostřed.

Rychlost not je nastavená tak, aby správný okamžik ke stisknutí tlačítka vyšel vždy na rytmickou dobu skladby, jež hraje v pozadí, což hráči usnadňuje odhad, pokud při hře vnímá i muziku. Implementační detaily se nachází v kapitole 5.4.

Po zdařeném pokusu jsou hráči přičteny body a o jedničku zvýšen násobič skóre, tudíž po dalším úspěchu je bodový nárůst větší. Násobič se zvedá až do hodnoty 50, což je tím pádem největší možný počet bodů, jež je možné získat za jednu notu. Pokud hráč nestihne notu „chytit“ a ta přejede kolečko, nota zmizí, body nejsou přičteny a násobič se vrací na hodnotu 1.



Obrázek 4.2: Screenshoty znázorňující směry not a jejich pohybu.

Hra obsahuje i další typ not, u nichž hráčovým úkolem není je získávat, ale naopak se jim vyhýbat. Tyto noty se pohybují stejně jako základní a jsou odlišeny graficky. Body jsou za ně přičteny jakmile dorazí do středu obrazovky bez toho, aby se dotkly hráčova avatara. Pokud se střetnou s avatarem, zmizí, násobič skóre je vrácen na hodnotu 1 a hráči je po krátkou dobu znemožněno ovládání avatara.

V dřívější verzi hra obsahovala i pokročilejší možnost vyhnutí se omračovací notě. Například omračovací notě jdoucí zleva se dá uhnout do všech ostatních směrů, tedy nahoru, dolů a doprava. Kromě toho bylo (díky instantnímu přesunu hráčova avatara po stisku směrového tlačítka) možné notu „přeskočit“ také přesunem do jejího směru tak, že hráč stojící uprostřed počkal, až se nota přiblíží těsně k němu a poté stisknul směrové tlačítko doleva, čímž se jeho postava objevila nalevo s notou za zády.

Ačkoliv tato mechanika byla zamýšlená a počítalo se s ní pro pokročilejší úrovně, podle prvních ohlasů působila spíše jako chyba a ze hry byla odstraněna.

Pro tento projekt bylo plánováno také dát uživatelům možnost jednoduše do hry importovat vlastní beatmapy podobně jako ve hře *osu!*, ale výzkum ukázal, že by se v případě Game Boye jednalo o náročné řešení, které by svým rozsahem mohlo vydat na samostatnou bakalářskou práci, protože by s velkou pravděpodobností bylo nutné přepisovat binární soubor zkompilované hry.

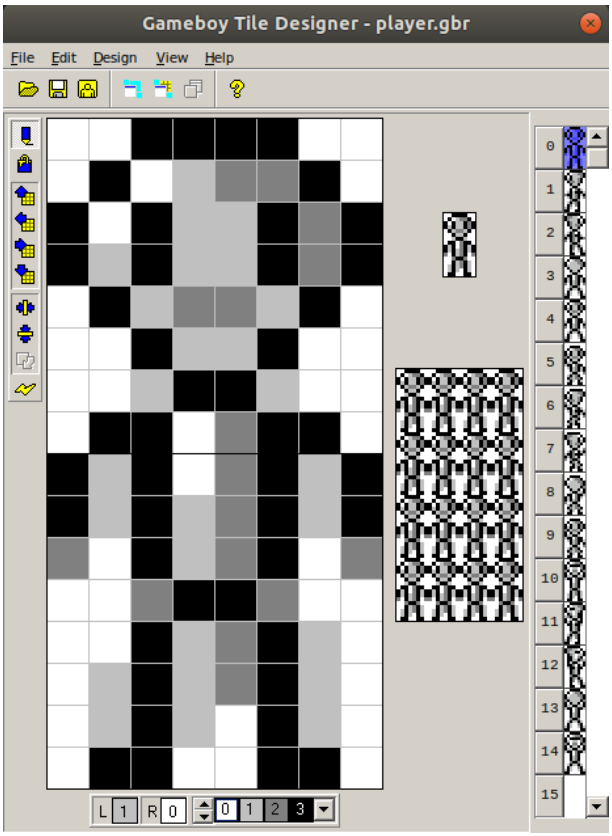
### Grafika

Grafická prezentace hry je velice jednoduchá vzhledem k malým zkušenostem v oblasti grafického návrhu. U rytmické hry pro Game Boy může simplistická grafika bez rušivých elementů být žádoucí. Navzdory tomu by projektu výraznější grafické zpracování bez pochyb prospělo.

Při návrhu grafiky bylo nutné přistupovat ke kompromisům ve velikosti sprajtů. Hráč potřebuje dostatek času zareagovat na přicházející notu, proto by sprajty neměly být příliš velké, aby nezabíraly nepřiměřeně mnoho prostoru. Zároveň by ale mělo být dostatečně výrazné na to, aby byly bez potíží rozeznatelné také na originálním, nepodsvíceném displeji.

Velká část komerčních her používá sprajt hráčova avatara o velikost 16x16, nebo i 16x32 pixelů. Pro tento projekt byla z výše popsaných důvodů zvolena velikost 8x16, která nenechává mnoho místa pro umělecké vyjádření. Proto byl jako hlavní postava použit humano-

idní mimozemšťan, jenž má možnost lépe vyniknout v daném rozlišení. Na obrázku 4.3 je vidět, že postava zabírá celou šířku i výšku vyhrazeného prostoru a na větší detaily nezbývá místo.




Obrázek 4.3: Screenshot z programu GBTD ukazující hráčova avatara.

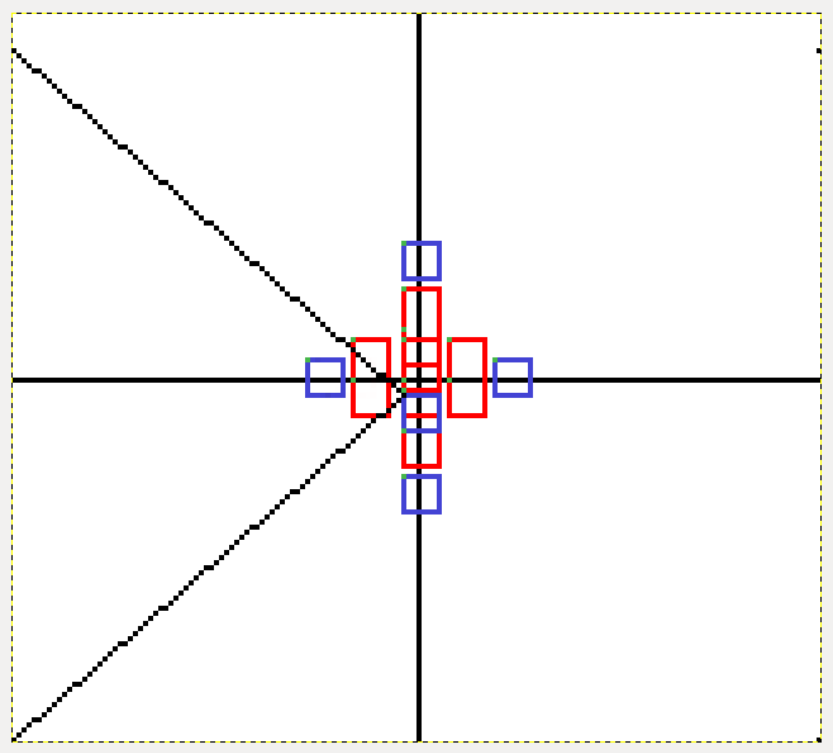
### Rozložení sprajtů na obrazovce

Mnoho sprajtů v této práci je statických a při hře nemění svoje souřadnice nebo se pohybují pouze v rámci předem známých mezí. K lepší představě kam sprajty umístit jsem v programu GIMP vytvořil mapu sprajtů ve stejném měřítku jako displej GB, která ulehčila plánování rozložení jednotlivých prvků. Mapa sprajtů je znázorněna na obrázku 4.4, kde si lze povšimnout také toho, že bylo nemožné umístit hráčova avatara přesně doprostřed obrazovky, protože displej má na šířku sudý počet displejů. Toto jemné vychýlení jde ale ve hře poznat jen stěží.

## 4.3 Cílová skupina

Cílovou skupinou nejsou běžní hráči videoher, které pravděpodobně hra na Game Boy nezaujme. Práce si nedává nutně za cíl ani nabídnout komplexnější zážitek než jiné rytmické hry, jež mají díky zaměření na pokročilejší platformy větší možnosti, ale obtížností by měla uspokojit i zkušenější hráče tohoto typu her.

Především tato práce cílí na příznivce starých konzolí a retro her se zálibou tyto hry hrát na originálním hardwaru. Na hráče, kteří mají kladný vztah k aktuální homebrew



Obrázek 4.4: Mapa sprajtů vytvořená v programu GIMP.

scéně a baví je sledovat, jaké nápady přinesou noví vývojáři na staré stroje dnešní optikou. Rytmická hra na systému, kde žádné takové nebyly (nebo jich bylo jen velmi málo), by je mohla zaujmout.

# Kapitola 5

# Implementace hry na Game Boy

V této kapitole je čtenáři popsán postup implementace hry pro konzoli Game Boy v jazyce C za pomoci toolkitu GBDK-2020 a hudebního ovladače hUGEDriver (popis těchto a dalších nástrojů se nachází v kapitole 3.7.1). Dále se kapitola zaměřuje na některé konkrétní detaily implementace a také optimalizace, které jsou být pro GB specifické.

## 5.1 Makra a globální proměnné

Jak bylo zmíněno dříve v sekci 3.8, na Game Boyi je výhodné co nejvíce omezovat zásobníkové operace. Proto soubor `src/main.c` obashuje poměrně hodně globálních proměnných, což by za běžných okolností při programování pro moderní procesory nebylo úplně žádoucí. Globální proměnné minimalizují nutnost jejich předávání funkcím, které k nim místo toho přistupují do globálního prostoru. Ze stejného důvodu je ve funkcích používáno pokud možno co nejmenší množství pomocných lokálních proměnných, jež by se opět tvořily na zásobníku.

Také nemalé množství maker slouží především k tomu, aby nebylo nutné tak často přistupovat k proměnným ve strukturách a místo toho se používaly konstanty, kde to jen jde a dává smysl.

## 5.2 Struktury

V kódu je používán malý počet velmi jednoduchých struktur jazyka C implementovaných ve složce `src` v souboru `sprite_structs.h`.

Struktura `player_t` obsahuje pouze pole o dvou prvcích typu `uint8_t`, které slouží jako indexy do tabulky OAM (blíže popsaná v kapitole 3.1). Objektům v OAM na těchto indexech jsou později přiřazeny patřičné grafické dlaždice a souřadnice na obrazovce, čímž vznik ne sprajt hráčova avatara. GDBK nemá funkci, jež by vracela souřadnice objektů na specifikovaném indexu v OAM, proto je na programátorovi, aby si informace o pozici sprajtů hlídal sám. Původní návrh tím pádem ukládal ve struktuře `player_t` i souřadnice na osách X a Y kvůli kontrole kolizí s ostatními sprajty. Vzhledem k tomu, že hráč se nemůže volně pohybovat po obrazovce a sprajt se pohybuje pouze na 5 předem daných pozic, byl tento údaj vypuštěn. Údaj o pozici hráče je místo toho uchováván enumem `pl_state`, který zároveň drží informaci o tom, zda je hráč omráčen a tím pádem je mu znemožněno ovládání. Tento způsob je rychlejší než by bylo přistupování do struktury pro kontrolu souřadnic.

Analogicky je implementována struktura `pad_t` pro sprajty vyznačující místa pro včasné stisknutí akčního tlačítka k „chycení“ not, která mají také předem danou, neměnnou pozici.

Naproti tomu noty se po obrazovce plynule pohybují a je potřeba si ve struktuře `note_t` ukládat také souřadnice. Mimo to je zde navíc ukládána i informace o typu noty a ukazatel na další notu. Jak jsou noty vázány k sobě do seznamů a výpočet souřadnic pro správnou rychlost pohybu not je přiblíženo v následujících podkapitolách.

## 5.3 Pole not

Původní návrh počítal s tím, že pro každou notu bude dynamicky alokována pamět, která se uvolní po zániku noty. GBDK-2020 má ve své knihovně vlastní implementaci funkce `malloc()`, ke které se mi ale nepodařilo dohledat dostatek informací z hlediska stability a efektivity, pouze jedno téma na vývojářském fóru, které od použití odrazuje[1].

V konečné implementaci je použito statické pole `notes`, pro nějž byla zvolena pevná velikost 20 prvků. V jeden moment tak může být na obrazovce nanejvýš 20 not a pokud se jich skladba pokusí vytvořit více, jsou noty navíc ignorovány. Tento počet se nicméně osvědčil jako více než dostačující, při běžné hře pouze zřídka kdy počet aktivních not přesáhne číslo 10.

Kromě toho jsou vytvořeny dva ukazatele na strukturu `note_t` – `free_notes` a `used_notes` – jež slouží jako počátky jednosměrných seznamů. Při prvotní inicializaci not se použije nota na nultém indexu pole `notes` jako kořen pro seznam `free_notes` a všechny následující noty jsou s ní provázány.

V průběhu hry jsou aktivní noty přesouvány do seznamu `used_notes`. Výsledné řešení dovoluje funkcím, jež manipulují pouze aktivními notami, projíždět tento seznam bez nutnosti zabývat se také těmi, co zrovna nejsou využívány, jako kdyby se iterovalo přes původní pole `notes`.

## 5.4 Aproximace rychlosti pohybu not

Ve hře bylo třeba zařídit, aby se noty pohybovaly takovou rychlostí, která bude vycházet do tempa hudby, což hráče nutí je chytat v rytmu. To je hlavní podstata rytmických her a výsledkem je intuitivnější zážitek, než kdyby noty přicházely náhodně bez synchronizace se skladbou na pozadí.

V původním návrhu se počítalo s využitím časových registrů handheldu a spuštěním přerušení každých několik jednotek času, které by zavolalo funkci, jež by pohnula s notami na obrazovce o jeden pixel. Četnost spouštění přerušení měla záviset na tempu aktuální skladby, které je v tomto kontextu předem daná konstantní hodnota definovaná v softwaru hUGETracker při komponování skladby a představující délku trvání jedné doby v počtech průchodů herní smyčkou[2]. Přehrávání hudby je v tomto projektu účelně navázáno na rychlost běhu hry. To proto, aby při případném zpomalení zapříčiněném například přetížením procesoru došlo i ke zpomalení skladby. Od řešení s pohybem not pomocí přerušení bylo upuštěno po uvědomění, že by mohlo dojít k desynchronizaci zapříčiněné tímto zpomalením procesoru (a tím pádem i zpomalením přehrávání hudby), ale hodinový čip by pracoval a spouštěl přerušení ve své obvyklé rychlosti.

[1] https://gbdev.gg8.se/forums/viewtopic.php?id=504

[2] Nekonečný cyklus kontrolující hráčovy vstupy a volající další herní logiku.

Dalším řešením bylo vzít vzdálenost v pixelech, kterou musí nota urazit od okraje obrazovky ke kolečku značícímu místo, v němž má být chycena pro získání bodů (k vidění na obrázku 4.2), a tuto hodnotu vydělit tempem skladby vynásobeném číslem N. Podíl byl uložen do proměnné `ppf` a představoval počet pixelů, o nějž se nota musí v každém proběhnutí herní smyčky posunout, aby v požadovaném čase dorazila do kolečka. Hodnota čísla N určuje, kolik dob bude trvat cesta noty od kraje ke kolečku. Ve finální implementaci bylo použito konstantní číslo 8, z čehož vyplývá, že všechny typy not do kolečka dorazí za 8 hudebních dob od jejich objevení na okraji obrazovky. Dostatek pro to, aby hráč stíhal reagovat.

Řešení nastíněné v předchozím odstavci se potýká s problémem, že po dělení může vyjít i desetinné číslo a Game Boy nepodporuje datové typy s plovoucí desetinnou čárkou. Kvůli tomu bylo nutné nutné proměnnou `ppf` zvětšit z 8 bitů na 16. Vzdálenost od okraje ke kolečku je poté bitovým posunutem přesunuta do horních 8 bitů a dělitelem je opět 8bitový výsledek násobení tempa skladby s číslem N. To z proměnné `ppf` dělá 16bitové číslo s pevnou řádovou čárkou, kde vrchních 8 bitů reprezentuje hodnotu před čárkou a spodních 8 bitů hodnotu za čárkou.

Proměnná `ppf` je přičítána (případně odečítána) k souřadnicím not, jež jsou také uloženy v 16bitových proměnných. K vykreslení not na obrazovce je používána pouze část souřadnice před desetinnou čárkou, tedy vrchních 8 bitů (protože sprajt se samozřejmě nemůže pohnout o desetinný počet pixelů).

Například při tempu 7 – základním přednastaveném tempu v programu hUGETracker – vychází posun zhruba na 1,1 pixelů za jednu iteraci herní smyčkou. Díky popsané aproximaci pevnou řádovou čárkou se noty obvykle posunují rychlostí 1 pixel za iteraci a v průměru každou 10. iteraci se posunou o 2. Tento rozdíl je pro lidské oko nezaznamenatelný. Aproximace není stoprocentně přesná a má za následek také to, že nota za daných 8 dob nemusí dorazit do přesného středu kolečka. Opět se ale jedná o zanedbatelnou nepřesnost.

## 5.5 Rozhraní pro komunikaci hudby se hrou

V běžné hře často stačí vložit do úrovně skladbu a tu přehrávat v nekonečné smyčce. V rytmické hře je potřeba, aby hra věděla, ve které fázi se zrovna skladba nachází a podle toho odpovídajícím způsobem reagovat.

K vytvoření takového rozhraní jsem využil funkci, již nabízí hUGEDriver. Při skládání hudby v hUGETrackeru se ke každé hudební době dá přiřadit efekt. Ve většině případů se jedná o hudební efekty jako vibráto nebo slide. Jeden z nehudebních efektů je zavolání jedné z až 16 rutin – programátorem definované bloky kódu v assembleru nebo jazyce C.

Implementované rozhraní lze najít v souboru `src/routines.c` a příslušném hlavičkovém souboru ve stejné složce. Soubor obsahuje vynulovanou 16bitovou proměnnou `new_notes`. Každý bit reprezentuje jeden typ noty, jež se má objevit na obrazovce. Zavolaná rutina logickým operátorem OR změní příslušný bit z 0 na 1.

Pro příklad, když v hUGETrackeru nastavím, aby skladba, respektive ovladač, který skladbu přehrává, na každou první dobu v taktu zavolal rutinu č. 1 a tuto skladbu naimportuji do hry, rutina v nastavených intervalech bude nastavovat druhý bit proměnné `new_notes` na 1. Ve funkci `main()` je každý bit této proměnné při hře kontrolován v nekonečném cyklu. Kdykoliv narazí na jedničku, zavolá funkci `spawn_note()` ze souboru `src/notes.c` a argumentem předá makro korespondující s kontrolovaným bitem (například druhý bit reprezentuje klasickou notu přicházející z pravé strany obrazovky). Po zkontrolování všech bitů je proměnná opět vynulována a připravena pro další průchod cyklem.

Původní implementace byla taková, že se proměnná přestala kontrolovat po nalezení prvního jedničkového bitu. Nešlo tedy na hráče poslat v jednom okamžiku více not. V případě klasických not to není žádoucí, hráč by měl šanci „chytit“ jen jednu z nich. Výsledná implementace kontroly celé proměnné se ale může hodit při posílání omračovacích not, kdy skripter skladby může nastavit příchod omračovacích not ze 3 směrů současně a hráč je tím pádem přinucen se jim vyhnout do posledního volného směru.

## 5.6 Omračovací noty

Omračovací noty po kontaktu s hráčovým avatarem hráči znemožní ovládání po dobu dvou hudebních taktů. Jejich kolize s hráčem je kontrolována ve funkci `move_notes()` v souboru `src/notes.c`. Funkce v případě omračovacích not sleduje pozici hráče uloženou v `player_pos` a pokud se hráč nachází v cestě noty, dívá se i na aktuální pozici noty. Když dojde ke kolizi, zavolá funkci `stun_player()`, která nastaví nejvyšší bit 8bitové proměnné `stun_counter` na 1.

Tento bit je kontrolován ve funkci `main()` v hlavní herní smyčce před čtením hráčova inputu a pokud je nastaven na 1, čtení inputu je přeskočeno (hráči je tedy znemožněno ovládání) a zbylých 7 bitů se používá jako časovač omráčení, který je opět nastaven na 8 dob. Číslo 8 je násobeno tempem skladby. Násobení je na GB typicky drahá operace, vyjma násobení násobky dvou, kdy je proces zoptimalizován v obyčejný bitový posun [16]. Ani takto opakovaně volané násobení osmi tím pádem nijak výrazně nezatíží procesor.

## 5.7 Probliknutí displeje při omráčení

Pokud hráč přijde do kontaktu s omračovací notou, obrazovka na malý okamžik problikne černě díky tomu, že všechny barvy barevné palety pozadí jsou funkcí `screen_flash_on()` nastaveny na černou, což efektivně zatmaví celý displej.

Délka tohoto zatmavení je počítána interním hardwarovým časovačem konzole. Registr TAC pro kontrolu časovače je po omráčení hráče nastaven na spuštění časovače rychlostí 4096 Hz a začne inkrementovat 8bitový registr TIMA, jenž po přetečení spustí přerušení časovače, které zavolá funkci `screen_flash_off()`, jež nastaví barevnou paletu pozadí na původní barvy a vypne časovač. Aby zatmavení netrvalo příliš dlouho, počáteční hodnota registru TIMA bývá nastavována na 128, takže k přetečení registru a tím pádem také ke spuštění přerušení dojde v polovičním čase.

## 5.8 Přechody mezi obrazovkami

Hra pro plynulejší změny obrazovek obsahuje fade out a fade in efekty, při kterých displej postupně celý přejde do bílé barvy a z bílé se poté opět plynule vykreslí nová obrazovka. Přechodů je opět dosáhnuto přepisy barevných palet a jejich implementaci lze nalézt ve funkcích `fade_out_white()` a `fade_in_white()` implementovaných v `src/main.c`.

Základní čtyřbarevná paleta od bílé po černou je binárně reprezentována jako `11100100`, kde každé dva bity zastupují jednu barvu (`00` je bílá, `11` černá, `01` a `10` potom světle, respektive tmavě šedá). Zesvětlování je tím pádem implementováno v několika krocích, v nichž jsou přepisovány hodnoty palet BGP a OBP0 tím způsobem, že v jednotlivých krocích je od každé bitové dvojice odečtena jednička až do chvíle, kdy se hodnoty obou

palet rovnají nule, což znamená čtyři dvojice nul – Game Boy všechny barvy nahradí pouze bílou.

Mezi jednotlivými kroky zesvětlení je několikrát volána věstavěná funkce toolkitu GBDK `wait_vbl_done()`, jež slouží k tomu, že na daný takt uvede procesor do klidového režimu až do překreslení celého displeje. Tato funkce má v tomto případě za účel pouze to, aby vložila pauzu mezi jednotlivé kroky zesvětlování. Jejím vynecháním by k „animaci“ postupného zbělení došlo příliš rychle a lidskému oku by se přechod jevil jako okamžitý.

Funkce `delay()` známá z jazyka C by docílila stejného efektu jako `wait_vbl_done()`, pouze bez uspání procesoru, což by znamenalo zbytečné plýtvání bateriemi.

## 5.9 Vrchní umístění ukazatelů skóre a násobiče

Jak bylo řečeno dříve v kapitole 3.4, window layer je netransparentní vrstva. Pokud tedy chce programátor umístit HUD do vrchní části obrazovky, musí počátek vrstvy umístit do levého horního rohu, což má za normálních okolností za následek překrytí celého pozadí neprůhlednou window layer.

V tomto projektu jsou skóre a aktuální násobič vypisovány nahoře. Kvůli tomu byl v tomto projektu implementován postup, díky němuž se window layer vykresluje pouze po dobu prvních osmi řádků displeje a po zbytek doby vykreslování je vrstva deaktivována.

Je toho docíleno nastavením dvou registrů a voláním přerušení. Game Boy si v registru LY uchovává číslo řádku, který je na displeji momentálně vykreslován. Další registr – LYC – je zapisovatelný a číslo v něm uložené je neustále porovnáváno s číslem aktuálně vykreslovaného řádku v registru LY. Jakmile se LY rovná LYC, je nastaven patřičný bit registru STAT, jenž udržuje informace o tom, v jakém stavu se zrovna nachází LCD displej. V tomto registru lze taktéž handheldu říci, zda má volat přerušení související s displejem a pokud ano, tak kdy. Přepsáním šestého bitu na 1 konzole volá přerušení právě když se LY rovná LYC.

Do registru LYC byla tedy uložena hodnota 8. Ve funkci `main()` je na začátku hlavní herní smyčky registr LCDC pro ovládání displeje nastaven tak, aby byla vykreslována window layer. Konzole tím pádem vykresluje tuto vrstvu od prvního do osmého řádku a jakmile příjde k devátému, spouští přerušení, které volá funkci `disable_win()`, jež má za úkol pouze přepsání LCDC za účelem vypnutí vykreslování window layer. Pro zbývající řádky je dále vykreslována pouze vrstva pozadí a sprajtů.

## 5.10 Optimalizace pozadí

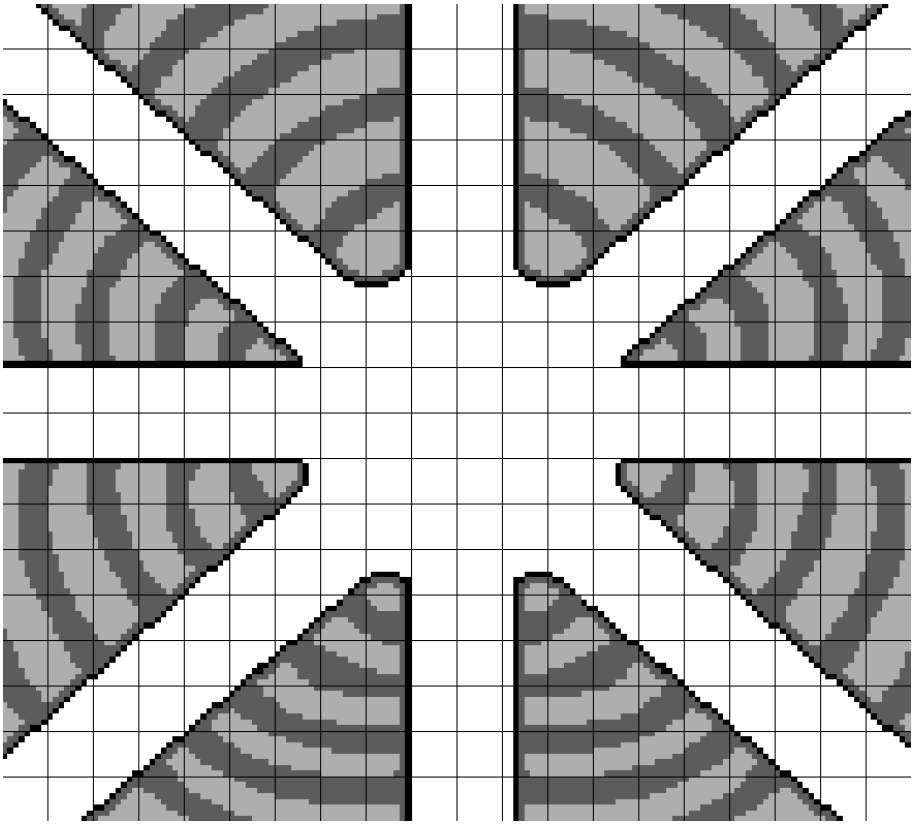
Pozadí bylo nakresleno v programu GIMP a překonvertováno na grafické dlaždice kompatibilní s GBDK-2020 pomocí nástroje *png2asset*[3]. Game Boy má ve VRAM místo pro 256 dlaždic pro sprajty a dalších 256 pro pozadí.

Původně zamýšlené pozadí bylo příliš komplikované na to, aby se dalo rozdělit do maximálně 256 dlaždic velkých 8x8 pixelů. Bylo tím pádem nutné využít vlastnosti popsané v sekci 3.3, tedy optimalizovat obrázek tak, aby se dal poskládat z co nejvíce totožných, znovupoužitelných dlaždic.

Optimalizace pozadí probíhala taktéž v GIMPu zapnutím mřížky, nastavením rozestupů mřížky na 8x8 pixelů a následným překreslováním tak, aby výsledek využíval co nejmenší

[3] https://github.com/gbdk-2020/gbdk-2020/tree/develop/gbdk-support/png2asset

počet unikátních dlaždic. Zvětšena byla zejména oblast s plně bílými dlaždicemi. Finální pozadí se skládá z 235 unikátních dlaždic a i s mřížkou lze vidět na obrázku 5.1.



Obrázek 5.1: Pozadí nakreslené v programu GIMP se zvýrazněnými dlaždicemi o rozměru 8x8 pixelů.

# Kapitola 6

# Testování

Testování se skládalo ze dvou částí, z nichž jedna se zaměřovala na optimalizaci kódu k dosažení co nejmenší zátěže na procesor a s tím spojenou nižší spotřebou baterií. Druhá se soustředila na sběr zpětné vazby na samotnou hru od testerů a jejím zapracováváním do herního designu.

## 6.1 Testování výkonu

Už od začátku byl kód psán se snahou o co nejlepší možnou optimalizaci pomocí technik nastíněných v kapitole 3.8, zejména omezením operací se zásobníkem.

Na Game Boyi dle mého vědomí bohužel neexistuje žádná cesta ke snadnému benchmarkování optimalizace softwaru. Bylo proto nutné se spolehnout na emulátor BGB, jenž nabízí možnost zobrazení snímků za sekundu a vytížení CPU, které by při výchozím nastavení emulace měly být věrné omezeným možnostem reálného handheldu.

Monitorováním těchto hodnot po dobu asi 10 minut bylo dosaženo závěru, že hra si na emulátoru drží stabilních 60 snímků za vteřinu a simulovaný procesor je málokdy zatížen na více jako 50 procent. Při testování na pravém hardwaru nebyla možnost tyto hodnoty ověřit, ale při hraní lidské oko žádné propady snímků za vteřinu nepostřehne. Z toho důvodu považuji optimalizaci za zdařilou.

## 6.2 Uživatelské testování

K závěrečnému testování docházelo na skutečném hardwaru jednak na původní, neupravené konzoli Game Boy, ale také na novějším modelu Game Boy Advance s neoficiální (ale v dnešní době rozšířenou) modifikací displeje tak, aby nabízel podsvícení. Testování v žádném z případů neprobíhalo na emulátoru, protože dle mého názoru nereprezentuje hru tak, jak se hraje na opravdovém handheldu a při hraní je o něco náročnější chytat noty v požadovaných intervalech. Jedním z možných důvodů by mohly být mírné rozdíly v časování mezi reálným a emulovaným hardwarem.

Testovaná skupina se skládala z pěti lidí, z nichž všichni měli předchozí zkušenosti s hraním na handheldech. Dva se dříve setkali přímo s originálním Game Boyem nebo jeho pozdějšími modely. Tři z této skupiny v minulosti hráli rytmické hry jako *Beat Saber* nebo *Guitar Hero*. Všichni členové testovací skupiny byli vybráni tak, aby měli kladný vztah k retro hrám. Měli tedy potenciál patřit do cílové skupiny.

Uživatelé byli testováni nezávisle na sobě a žádný z nich se se hrou před samotným testem nesetkal. Před hraním byl testerům vysvětlen základní koncept hry a ovládání. Následně byl každý z nich požádán o odehrání jedné 2minutové skladby prvně na podsvíceném a poté na původním displeji. Po dvou „povinných“ kolech bylo testerům nabídnuto pokračovat v hraní na libovolném modelu konzole a k podpoře kompetitivnosti bylo zapisováno a porovnáváno nejvyšší skóre jednotlivých testerů. Této možnosti využili všichni testeři. Stejně tak se každý z nich vrátil k modifikovanému modelu s podsvíceným displejem. V průměru si každý tester skladbu po úvodních dvou kolech zopakoval ještě třikrát. Po dohrání byli testeři požádáni o co nejobjektivnější ohodnocení hry v několika kategoriích na stupnici od jedné do pěti, kde číslo pět reprezentovalo nejlepší hodnocení.

### Pochopitelnost konceptu

Průměrné hodnocení: 4,8
Dle očekávání byla hra snadná na pochopení zejména pro ty, kteří se s rytmickými hrami už setkali. Od všech členů této skupiny byla pochopitelnost hodnocena nejvyšší známkou. Ani další dva testeři ale neměli výraznější problém konceptu porozumět a to, co nepochopili při úvodním slovním vysvětlení, rychle pochytili při samotném hraní. Takto vysoké hodnocení není překvapivé, protože základní koncept rytmických her často bývá poměrně primitivní.

### Ovládání

Průměrné hodnocení: 4,4
Také ovládání bylo všem v testovací skupině vcelku jasné. Hlavní připomínky a nižší známky 4 a 3 si odneslo od dvou testerů s předchozí zkušeností s rytmickými hrami, kterým se nejprve zdálo stisknutí akčního tlačítka *A* k chycení noty po navolení směru zbytečné. Podle jejich názoru by stačilo jednoduše ve správném momentě stisknout pouze směrové tlačítko. Tato poznámka by mohla vyplývat ze zkušeností s hrami jako *Guitar Hero*, v nichž opravdu k získání noty stačí stisknout pouze jedno tlačítko.

Zmiňovaná zpětná vazba nebyla do hry zapracována z toho důvodu, že jako budoucí rozšíření je v plánu do hry přidat nový typ not, který bude vizuálně odlišen a k jeho chycení bude třeba místo akčního tlačítka *A* stisknout akční tlačítko *B*. To dá skripterovi beatmap možnost tvořit komplexnější úrovně, kde hráč nebude muset volit pouze správný směr, ale také akci.

### Grafická prezentace

Průměrné hodnocení: 2,8
Kategorie, která dopadla nejhůře ze všech. Hodnocení je pochopitelné vzhledem k tomu, že grafika je jednou z nejslabších stránek projektu a v budoucnu bude nutné ji od začátku předělat, pokud má hra být na první pohled atraktivní pro běžného uživatele.

Mezi připomínkami se objevily také stížnosti na špatnou čitelnost sprajtů not a nevýrazné značení míst, v nichž se mají noty chytat zejména při hraní na originálním, nepodsvíceném displeji. Sprajty proto byly mírně zvětšeny a místo šedé je nyní používána převážně černá barva.

### Zvuková prezentace

Průměrné hodnocení: 4,8
Prezentovanou skladbu téměř všichni zúčastnění hodnotili nejvyšší známkou, skladba se jim líbila a neměli větších výhrad. Tento výsledek považuji za úspěch, protože u podobného typu her je soundtrack důležitou součástí zážitku.

### Obtížnost

Průměrné hodnocení: 3,8
Hodnocení obtížnosti se zaměřovalo na to, jak danému testerovi sedla. Známka 5 tedy znamená optimální obtížnost a nízká známka že je hra moc lehká nebo naopak těžká. Obtížnost byla jednou z nejrozporuplněji hodnocených kategorií. Od zkušenějších hráčů rytmických her dostala dvakrát hodnocení 5 a jednou 4 a relativně vysoká náročnost byla vyzdvihována jako pozitivum podporující znovuhratelnost. Dvojice testerů méně zběhlých v podobných hrách obtížnost hodnotila známkami 3 a 2 s připomínkou, že je hra příliš náročná a rychlá. Testeři ale měli k dispozici pouze jednu skladbu a při dalším vývoji bude možné vytvořit jednodušší beatmapy zaměřené i na méně zkušené hráče.

Od obou táborů pak zazněly připomínky na to, že časové okno pro chycení noty je možná příliš malé. Proto byla mírně rozšířena rozmezí, v nichž je možné noty získat. Tím byla lehce zvýšena tolerance a hra nyní častěji odpouští, když hráč nechytne notu úplně přesně do rytmu.

### Zábavnost

Průměrné hodnocení: 4,2
Celkově byla zábavnost projektu hodnocena převážně kladně. Nejnižší hodnocení v této kategorii bylo 3, nicméně i tento tester o hře mluvil poměrně pozitivně a vyzdvihl její potenciál v případě pokračujícího vývoje.

# Kapitola 7

# Závěr

Práce nejprve stručně shrnula videoherní konzole z období čtvrté konzolové generace a speciálně se zaměřila na handheld Nintendo Game Boy, jeho hardware, vlastnosti a specifika. Představila nejpoužívanější nástroje dnešní doby pro vývoj softwaru na Game Boy a praktiky, které je dobré dodržovat k udržení efektivního kódu.

Programovým výstupem práce je rytmická hra kompatibilní s původním Game Boyem a jeho nástupci. Hra je napsaná v jazyce C s využitím toolkitu GBDK-2020 a je poměrně zdařile zoptimalizována pro co nejnižší zátěž procesoru. Obsahuje jednu krátkou, jednoduchou skladbu k předvedení základního principu (Track 1) a jednu plnohodnotnou skladbu, jež slouží jako ukázka toho, jak mohou vypadat náročnější úrovně (Track 2). S pěti směry odkud na hráče mohou přijít noty, citlivým použitím omračovacích not a do budoucna plánovaným rozšířením v podobě dalšího typu akčních not je možné tvořit relativně komplexní beatmapy, které by podle uživatelského testování mohli uspokojit i zkušenější hráče rytmických her.

Kostra je implementována tak, že hra umí pracovat s jakoukoliv skladbou vytvořenou v softwaru *hUGETracker*. Rychlost pohybu not a délka omráčení se samy přizpůsobí tempu skladby zvolenému v trackeru.

Výraznou slabou stránkou výsledné hry je její grafická prezentace, u níž zpětně lituji, že jsem na pomoc nepřizval grafika. Zejména absence pozadí hrací plochy hře velkou měrou ubírá na atraktivitě a nenavozuje dobrý první dojem.

Kromě vylepšení grafické stránky by na práci šlo navázat například napsáním nástavby k open-source softwaru *hUGETracker* přizpůsobené na míru přímo této hře. Pohodlné grafické rozhraní pro skriptování volání rutin ke zrození not by mohlo výrazně urychlit práci při vytváření beatmap.

Taktéž by se dalo navázat vymyšlením a implementací způsobu, jak uživatelům hry dovolit naimportovat vlastní skladby bez toho, aby se celá hra musela znovu kompilovat. Teoreticky by mělo být možné například napsat nástroj, který by uměl číst a upravovat binární soubor zkompilované hry.

Před tímto projektem jsem nikdy nevytvářel žádný software pro Game Boy nebo podobný handheld. Díky této práci jsem se naučil, jak celá konzole pracuje a to mi dovoluje více docenit tehdejší i současné hry na Game Boy či další konzole téže generace. Nasbírané zkušenosti zužitkuji při pokračujícím vývoji rytmické hry a jiných projektů pro handheldy Nintendo Game Boy a Panic Playdate, které bych chtěl v budoucnu zrealizovat.

# Literatura


[1] Ahrnbom, M. *Game Boy Assembly Programming for the Modern Game Developer* [online]. 2022 [cit. 2022-03-17]. Dostupné z: https://github.com/ahrnbom/gbapfomgd.

[2] Altice, N. *I am Error: The Nintendo Family Computer / Entertainment System Platform (Platform Studies)*. Reprint. MIT Press, 2017. ISBN 978-0262534543.

[3] Amos, E. *The Game Console 2.0: A Photographic History from Atari to Xbox*. 2. vyd. No Starch Press, 2021. ISBN 978-1718500600.

[4] Camper, B. B. *Homebrew and the Social Construction of Gaming: Community, Creativity, and Legal Context of Amateur Game Boy Advance Development*. 2005. Diplomová práce. University of Washington. Dostupné z: https://dspace.mit.edu/handle/1721.1/42227.

[5] Carlton, S. *Why High Retro-Game Prices Are Here To Stay* [online]. 2021 [cit. 2022-04-12]. Dostupné z: https://medium.datadriveninvestor.com/why-high-retro-game-prices-are-here-to-stay-ac4401628259.

[6] Carpenter, N. *Gamers with godlike reflexes are racing to break world records in this rhythm game* [online]. 2019 [cit. 2022-05-01]. Dostupné z: https://www.pcgamer.com/two-teens-are-on-a-crazy-world-record-race-in-extremely-challenging-rhythm-game-osu/.

[7] Dale, A. *I'll Never Love a Console Like I Loved the SEGA Game Gear* [online]. 2015 [cit. 2021-12-01]. Dostupné z: https://www.vice.com/en/article/gqmwwx/ill-never-love-a-console-like-i-loved-the-sega-game-gear-820.

[8] Edwards, B. *Genesis of success: 20 years of Sega's dark horse console* [online]. 2008 [cit. 2021-11-22]. Dostupné z: https://arstechnica.com/gaming/2008/11/sega-genesis-turns-20.

[9] Great Hierophant [pseud.]. *Cartridge Bankswitching Outside the NES* [online]. 2017 [cit. 2022-04-10]. Dostupné z: http://nerdlypleasures.blogspot.com/2017/03/cartridge-bankswitching-outside-nes.html.

[10] Hansen, D. *Game On!: Video Game History from Pong and Pac-Man to Mario, Minecraft, and More*. 2. vyd. New York: Square Fish, 2021. ISBN 978-1718500600.

[11] Javanainen, J. *Power consumption of Game Boy flash cartridges* [online]. 2021 [cit. 2022-04-10]. Dostupné z: https://gekkio.fi/blog/2021/power-consumption-of-game-boy-flash-cartridges/.

[12] Kent, S. L. *The Ultimate History of Video Games: The Story Behind the Craze that Touched our Lives and Changed the World*. 1. vyd. New York City: Three Rivers Press, 2001. ISBN 0-7615-3643-4.

[13] Kindy, D. *Thirty Years Ago, Game Boy Changed the Way America Played Video Games* [online]. 2019 [cit. 2021-11-29]. Dostupné z: https://www.smithsonianmag.com/innovation/thirty-years-ago-game-boy-changed-way-america-played-video-games-180972743/.

[14] Marriott, S. A. *TurboGrafx-16 TurboExpress* [online]. 2009 [cit. 2021-12-05]. Dostupné z: https://web.archive.org/web/20090406021702/http://www.allgame.com/platform.php?id=17673.

[15] McGlynn, A. *Why are game-makers creating new Game Boy games in 2021?* [online]. 2021 [cit. 2022-04-12]. Dostupné z: https://arstechnica.com/gaming/2021/05/meet-the-developers-making-og-game-boy-games-in-2021/.

[16] Nintendo of America. *Game Boy Programming Manual* [online]. 1999 [cit. 2022-02-30]. Dostupné z: https://archive.org/details/GameBoyProgManVer1.1/mode/2up.

[17] Pan of Anthrox [pseud.] et al. *Game Boy CPU Manual* [online]. 1999 [cit. 2021-12-25]. Dostupné z: https://realboyemulator.files.wordpress.com/2013/01/gbcpuman.pdf.

[18] Parish, J. *Game Boy Works: Nintendo Game Boy (DMG-001)* [online]. 2014 [cit. 2021-12-12]. Dostupné z: https://www.gameboyworks.com/1989/04/21/nintendo-game-boy-dmg-001/.

[19] Parish, J. *Too Powerful for Its Own Good, Atari's Lynx Remains a Favorite 25 Years Later* [online]. 2014 [cit. 2021-12-01]. Dostupné z: https://www.usgamer.net/articles/too-good-for-its-day-ataris-lynx-remains-a-fan-favorite-25-years-later/.

[20] Sheff, D. *Game Over: How Nintendo Zapped an American Industry, Captured Your Dollars, and Enslaved Your Children*. 1. vyd. New York: Random House, 1993. ISBN 978-0679404699.

[21] Steinbock, D. *The Mobile Revolution: The Making of Worldwide Mobile Markets*. 1. vyd. Kogan Page Business Books, 2005. ISBN 978-0749442965.

[22] Stuart, K. *Nintendo Game Boy: 25 facts for its 25th anniversary* [online]. 2014 [cit. 2021-11-29]. Dostupné z: https://www.theguardian.com/technology/2014/apr/21/nintendo-game-boy-25-facts-for-its-25th-anniversary/.

[23] Stuart, K. *Sega Genesis at 30: the console that made the modern games industry* [online]. 2019 [cit. 2021-11-22]. Dostupné z: https://www.theguardian.com/games/2019/aug/16/sega-genesis-at-30-mega-drive-console-modern-games-industry.

# Příloha A

# Obsah přiloženého paměťového média

```
/
├── app/                          - zdrojové soubory hry
├── doc/
│   ├── latex.zip                 - zdrojové soubory textu
│   └── xkrejc71-Hra-pro-GB.pdf   - text bakalářské práce
├── chiptune_rockin.gb            - spustitelný soubor hry
├── poster.png                    - plakát v rozlišení A4
├── README.txt                    - informace ke spuštění,
│                                   kompilaci a licencím
└── video_demo.mp4                - ukázka z hraní
```

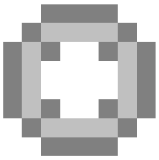
# Příloha B

# Pravidla hry

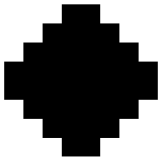
Na hráče přichází noty z několika směrů. U běžných not znázorněných na obrázku B.2 hráč podržením směrového kříže zvolí směr, odkud nota přichází a stiskne tlačítko *A* v moment, kdy dorazí na podložku vyobrazenou na obrázku B.1. Pro chycení not jdoucích šikmo z rohů hráč nevolí směr a zůstává uprostřed. Cítění rytmu skladby může hráči pomoci s odhadem správného načasování chytání not.

Notám na obrázku B.3 je potřeba se vyhýbat. Tyto noty samy zmizí po tom, co dorazí doprostřed obrazovky.

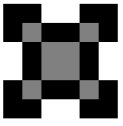
Za správně chycenou notu nebo za vyhnutí se omračovací notě hráč dostane body a je zvýšen jeho násobič skóre. Pokud nestihne chytnout běžnou notu nebo je omráčen, násobič padá zpět na počáteční hodnotu.



Obrázek B.1: Sprajt podložky.



Obrázek B.2: Sprajt běžné noty.



Obrázek B.3: Sprajt omračovací noty.

# Příloha C

# Plakát


Obrázek C.1: Plakát k výsledné hře inspirovaný starými videoherními reklamami.