STŘEDNÍ PRŮMYSLOVÁ ŠKOLA

MLADÁ BOLESLAV

**DLOUHODOBÁ PRAKTICKÁ**

**MATURITNÍ PRÁCE**



Matouš Rejzl

Mladá Boleslav 2018

STŘEDNÍ PRŮMYSLOVÁ ŠKOLA

MLADÁ BOLESLAV

**DLOUHODOBÁ PRAKTICKÁ**

**MATURITNÍ PRÁCE**

**Autor: Matouš Rejzl**

**Studijní obor: 18-20-M/01 Informační technologie**

**Vedoucí práce: Ondřej Kolín**

Mladá Boleslav 2018

Obsah

[1. Úvod 5](#_Toc509903804)

[2. Návrh hry, výběr vhodných technologií 6](#_Toc509903805)

[3. Co je to Game Maker Studio 2 8](#_Toc509903806)

[3.1 Sprites 9](#_Toc509903807)

[3.2 Tilesets 10](#_Toc509903808)

[3.3 Sounds 10](#_Toc509903809)

[3.4 Paths 10](#_Toc509903810)

[3.5 Scripts 10](#_Toc509903811)

[3.6 Shaders 11](#_Toc509903812)

[3.7 Fonts 11](#_Toc509903813)

[3.8 Timelines 11](#_Toc509903814)

[3.9 Objects 11](#_Toc509903815)

[3.10 Rooms 12](#_Toc509903816)

[3.11 Notes 12](#_Toc509903817)

[3.12 Included files 12](#_Toc509903818)

[3.13 Extensions 12](#_Toc509903819)

[3.14 Options 12](#_Toc509903820)

[3.15 Configurations 12](#_Toc509903821)

[4. Základní implementace hráče 14](#_Toc509903822)

[5. Implementace herního prostředí, vytvoření úrovní 17](#_Toc509903823)

[6. Vytvoření systému pro náhodné generování 19](#_Toc509903824)

[7. Závěr 21](#_Toc509903825)

[8. Přílohy 22](#_Toc509903826)

[8.1 Zdroje 22](#_Toc509903827)

**Prohlášení**

Prohlašuji, že jsem svou dlouhodobou praktickou maturitní práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti zpřístupňování této závěrečné práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Mladé Boleslavi dne podpis:

# Úvod

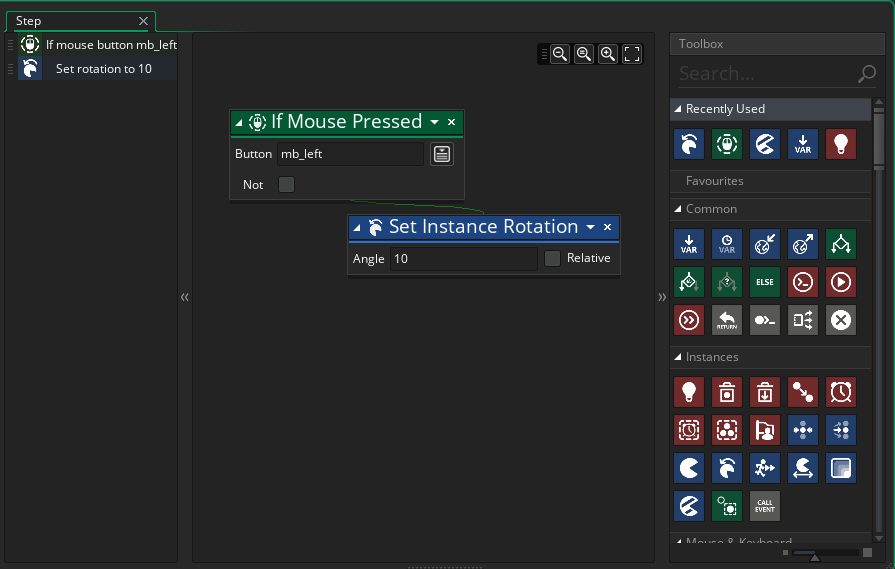
Téma této maturitní práce jsem si zvolil záměrně z jednoho velmi prostého důvodu. Hru The Binding of Isaac jsem totiž hrál již od svého brzkého dětství a fascinovala mě svojí jednoduchostí a zároveň hloubkou. Hra se do povědomí dostala v roce 2011, když ji Edmund McMillen a Frorian Himsle společně vydali na herní komunitu Steam. Binding of Isaac si díky nízké ceně 5 euro / dolarů zakoupilo mnoho lidí a většině lidí se hra velmi líbila. Na Steamu má aktuálně 95%ní kladné hodnocení, což znamená, že ze 100 lidí by ji 95 doporučilo. Když jsem řekl, že mě fascinovala jednoduchostí, ale i hloubkou, tak to mohlo pro někoho vyznít pravděpodobně divně, proto nyní uvedu vše na pravou míru.

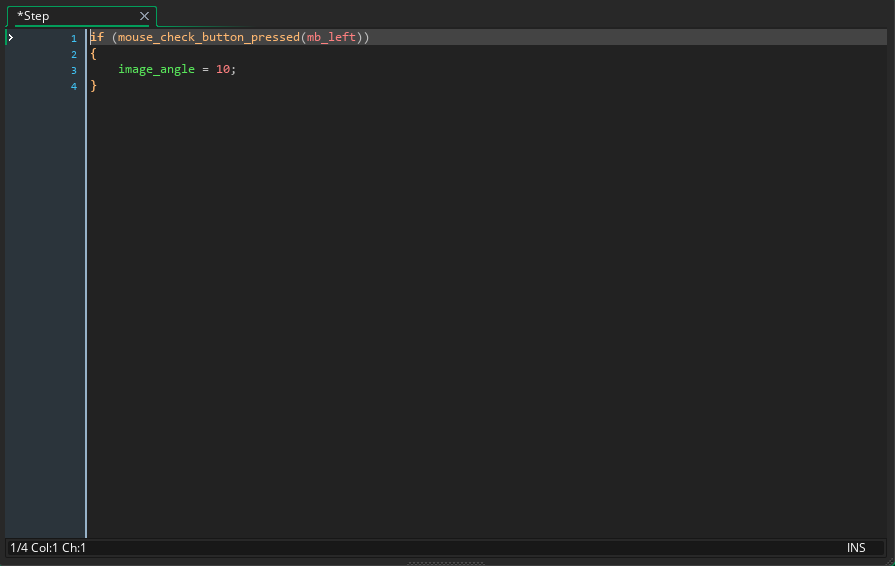
Hra na první pohled vypadá velmi jednoduše, hráč prochází skrz náhodně generované úrovně, kde vždy porazí jednoho závěrečného protivníka a pokračuje dál. Na každém poschodí může hráč najít řadu předmětů, které mu zjednoduší pokračování ve hře. Na konci hry se vždy nachází jeden těžký závěrečný protivník, kterého když hráč zabije, tak je odměněn předmětem, který může používat při opakovaných hrách. Ale na druhý pohled v sobě hra nese hlubší a malinko temnější stránku. Lze ji také vnímat jako mysl již jmenovaného Isaaca, který bojuje se svými vnitřními démony. A tady je několik důvodů, proč si to myslím: hodně nepřátel vypadá jako Isaac, Isaac bojuje pomocí svých slz, nikdy nenalezne cestu ze "sklepa" (to je místo, kde se hráč objeví, když poprvé zapne hru), ale pouze pokračuje hlouběji do neznáma, v některých místnostech lze nalézt divně vypadající monstra, která nic nedělají, ale zároveň vypadají, že jsou mrtvá. Také lze vidět několik oprátek náhodně rozmístěných po místnostech a jedním z finálních závěrečných protivníků je jeho vlastní velmi křesťansky založená matka, která si o něm myslí, že je plný hříchů a musí být očištěn. Vše tak směřuje k tomu, že Isaac je obyčejný kluk, který trpí mentální poruchou, ale toto jsou pouhé domněnky a nic není jisté.

# Návrh hry, výběr vhodných technologií

Prvním bodem maturitní práce bylo navrhnutí hry a zvolení vhodných technologií. Ze začátku jsem se rozhodoval, jakou technologii vybrat a jaké budou její plusy a mínusy. První technologie, nad kterou jsem uvažoval, byla Phaser. Phaser je open source, free a rychlý framework[[1]](#footnote-1) pro Canvas a WebGL hry za pomocí Javascriptu a HTML. Uvažoval jsem nad ním jako nad první technologií, protože jsem ho již hodně znal a je velmi jednoduchý na používání. Ale po hlubší inspekci jsem zjistil, že je to velmi nevhodný framework pro daný projekt a to proto, že Phaser neumí kolize zaznamenávat jinak než jako obdélník nebo čtverec. V mém projektu je potřeba zaznamenávat kolize i jako elipsu anebo mnohem složitější obrazce. Proto jsem poté uvažoval o frameworku QT. Tento framework pracuje na základním C++, který vylepšuje pomocí možnosti používat sloty a signály. Pracovat v tomto frameworku jsem se učil ve 3. a 4. Ročníku na této střední škole v předmětu Programování. V tomto frameworku jsem taky vytvořil už dva menší projekty, které byly také počítačové hry. Takže zkušenosti pro vytváření her jsem měl, ale řekl jsem si, že bude lepší podívat se i po dalších možnostech. Proto jsem následně prohledával další možnosti a nalezl jsem Game Maker Studio 2. Nic jsem o tohle programu na vytváření her nevěděl, ale řekl jsem si, že nebude na škodu, podívat se, co to je a co to umí. Zjistil jsem, že je to program vydaný v roce 2017 a programuje se v něm pomocí dvou možností v jazyku Game maker language. A velmi se mi zalíbil, a proto jsem si ho zvolil jako technologii, kterou chci používat. Jednou z možností je Drag and Drop. Jednoduše vysvětleno, Drag and Drop je grafické programovaní, kde uživatel tahá předefinované funkce z menu funkcí, jako je například move(x,y), která slouží k pohybu objektu. Tyto funkce pak uživatel propojuje mezi sebou a vzniká tak normální kód, jak ho všichni známe. Ale zároveň je zde další možnost, jak napsat kód, a to tak, jak to všichni známe.

Ukázka DnD kódování v GML.

   
Ukázka běžného kódování v GML.



# Co je to Game Maker Studio 2

Programovací jazyk v Game Maker Studiu je Game maker language. Game maker language vytvořil Markus Hendrik "Mark" Overmars. "Mark" Overmars je holandský programátor a učitel programování pro jazyk, který vymyslel. Má syntaxi velmi podobnou programovacímu jazyku C s možností použití syntaxe Pascalu. Game maker language může, ale nemusí, mít řádky kódu ukončeny středníkem. Proměnné není nutno deklarovat, což znamená, že uživatel může přiřadit ke každé proměnné jakoukoliv hodnotu bez ohledu na to, jaký je to datový typ. Jsou tu pouze dva datové typy, a to řetězce a reálná čísla. Maximální velikost řetězce je 4 294 967 296. Game maker language nepoužívá boolean tak, jak ho známe. Pokud uživatel definuje proměnnou true nebo false, tak ji nastaví buď na 1, nebo na 0. Dále se zde nacházejí již předdefinované proměnné jako je x nebo y, které slouží pro nastavení pozice. Pokud by uživatel chtěl vytvořit proměnnou, ke které by chtěl přistoupit kdekoliv v kódu, tak ji musí definovat jako globalvar nebo global. Jediný rozdíl mezi global a globalvar je ten, že pro přístup ke global proměnné je potřeba napsat před proměnnou pokaždé global. Zatímco pro přístup ke globalvar není potřeba pokaždé napsat globalvar. Je možnost deklarovat jedno i dvourozměrné pole. Další datové struktury jsou zásobník, fronta, seznam a slovník. Game maker language automaticky alokuje paměť při prvním použití a používá dynamické typování. Uživatel nemusí uvolňovat paměť, kterou používají proměnné, protože se paměť uvolní automaticky po tom, co skončí skript nebo instance. Game maker language nepodporuje ukazatele, takže uživatel se musí odkazovat na instance pomocí ID, které jde vždy získat pomocí funkce id().

Kdybych měl shrnout Game maker language, řekl bych, že je velmi podobný pro mě již dobře známému C++. Navíc Game maker language podporuje takovou jednodušší syntaxi než C++. Například je možné vynechávat středník na konci jednotlivých řádek a nebude to bráno jako chyba syntaxe. Tuto možnost jsem ze začátku vůbec nevyužíval, ale v pozdějších částech projektu mi to přišlo velmi výhodné, protože se vyskytla možnost psát kód rychleji. Také zde je možnost psát podmínku if zkráceně, a to jak bez kulatých závorek pro podmínku, tak i bez složených závorek pro tělo podmínky. Také by stálo za to, abych zmínil, že neexistuje něco jako hlavičkový nebo cpp pro vše, co uživatel naprogramuje, ale jednotlivé prvky se dělí do skupin podle toho, co jsou. Uživatel může vytvořený prvek zařadit do jedné z následujících skupin: Sprites, Tile Sets, Sounds, Paths, Scripts, Shaders, Fonts, Timelines, Objects, Rooms, Notes, Included Files, Extensions, Options, Configurations, které se nyní pokusím popsat.

## **Sprites**

Tato skupina slouží k vytvoření Sprite[[2]](#footnote-2) pro objekt. Nachází se zde i editor obrázků, ve kterém se dá dělat mnoho různých věcí. Buď může uživatel importovat svůj image z jakéhokoliv formátu (png, jpg apod.), anebo si může i vlastní image vytvořit v zabudovaném image editoru. Image editor je na úrovni složitosti mezi malováním a programem Photoshop. Pracuje se tu ve vrstvách, takže uživatel může skládat obrázky na sebe. Nachází se zde mnoho mocných nástrojů, ať už od obyčejné tužky, kterou zná snad úplně každý, až po něco, jako je magická tužka, která vybere pouze část barevného spektra pro editaci části image. Taky je zde několik možnost vytvářet animace, a jednou z nich je ta, že si uživatel vytváří jednotlivé snímky pixel po pixelu. Tato možnost je velice zdlouhavá, a použil jsem ji jenom párkrát. Dále se tu nachází i speciální funkce, která se jmenuje Animate and Draw. Tato funkce slouží k vytváření animací za pochodu, což znamená, že uživateli běží celý animační cyklus a on v něm maluje. Toto je například vhodné pro dělání animace plamenu ohně. Tuto funkci jsem osobně ani jednou nevyužil, protože většinu animací, které bylo nutno udělat, jsem byl schopný najít ze stránek, které poskytují různé, již hotové Sprite animace zadarmo.

Game maker studio 2 dovoluje i další možnost na dělání animací, a sice že si uživatel naimportuje image strip[[3]](#footnote-3), ve kterém si nadefinuje, jak mají být jednotlivé obrázky velké, jak velký mají mít offset apod., a Game make studio 2 za něho vše, ohledně animací, vyřeší. Tuto funkci jsem využíval velmi hodně, hlavně u dělání animace hráče. Uživatel si musí dále uvědomit, že importovat image do Sprite lze pouze jednou, protože kdykoliv když se naimportuje nový image do Sprite, tak se všechny ostatní snímky daného Sprite smažou. Já jsem se toto naučil tou špatnou cestou, když jsem chtěl dokončit jednu z animací dalším obrázkem, který když byl naimportován, tak smazal všechnu mojí předchozí práci v daném Sprite. Takže, pokud by měl uživatel animaci ve více souborech, tak nejdříve musí všechny soubory vložit do jednoho souboru, odkud pak naimportuje image strip do Sprite. Také by stálo za zmínku, že kolize se definují a vytvářejí zde, a ne v jednotlivých Objektech.

## Tilesets

Tato skupina slouží pro importováni tilesetů (česky seznamu dlaždic), které lze využít pro jednoduché vytváření grafiky jednotlivých místností. S tilesety jsem si hrál, když jsem poprvé začal dělat v Game maker studiu 2. Tilesety jako takové ve finální verzi hry vůbec nevyužívám, protože mi přišlo jednodušší si grafiku vytvářet pomocí objektů s Spritem. Je pravda, že hezké tilesety můžou hru velmi vylepšit po grafické stránce, ale pracovat s nimi pro generování náhodného prostředí mi přišlo zbytečně obtížné.

## Sounds

Tato skupina, jak již název napovídá, slouží pro zvuky. Zde si uživatel importuje všechny zvuky potřebné pro hru. Je zde možnost trochu editovat a upravovat jednotlivé zvuky, ale tento editor je velmi omezený v porovnání s ostatními editory zvuku, jako je například Audacity. Není zde možnost editovat délku zvuku, ani hloubku nebo výšku. Ale je zde možnost editovat hlasitost jednotlivých zvuků a také je zde možnost mixovat zvuky dohromady, což znamená vytvářet zvuky, které se skládají z více zvuků.

## Paths

Je skupina, kde lze nadefinovat specifické cesty pro jednotlivé objekty. Toto je velmi mocný nástroj, pokud uživatel potřebuje, aby se některé objekty pohybovaly po speciálních trasách a nikde jinde. Já osobně jsem toto nevyužil, protože mi to přišlo v mojí hře zbytečné. Jediný pokročilý pohyb, který jsem potřeboval, bylo obcházení nepřátel sama sebe, protože když se potkali dva nepřátelé, tak se sami do sebe zasekli. Toto jsem ale vyřešil pomocí zabudované funkce place\_empty(). Tato funkce se vždy zeptá, jestli je místo volné a vrátí true nebo false. A já na základě této odpovědi buď nepřítele posunu, nebo ne.

## Scripts

Slouží k vytváření skriptů. Tyto skripty se využívají, pokud uživatel chce vytvořit vlastní funkce. Mě osobně používání skriptů přišlo zbytečné, proto jsem je nikde nevyužil. Jediné místo, kde bych je mohl využít, jsou jednotlivé předměty, které hráč může sbírat. Ale tyto předměty jsem vytvářel tak, že jsem duplikoval celý předmět. To znamená, že se vždy celý kód zkopíroval a já poupravil jen pár řádku kódu, jako je například Sprite, jméno objektu, nebo co hráč dostane, když ho sebere. Generování mapy by šlo dělat pomocí skriptu, ale taky pomocí vloženého kódu do objektu.

## Shaders

Tento mocný nástroj lze použít pro úpravu způsobu vykreslování na obrazovku. Tyto malinké programy jsou spuštěny na grafické kartě. A tím se urychluje chod hry, jelikož se uvolňuje CPU.

## Fonts

Základním fontem pro výpis je Arial o velikosti 12. Pokud by ale uživatel chtěl použít jiný font, tak lze vytvářet i speciální fonty, které se dají využít pro vypsání textu v Game maker language.

## Timelines

Toto je podobné jako třídě QTimer v QT/C++ a alarmu v GML. Používá se pro vykonání kódu v přesný tik od vytvoření. Chápu, že pro někoho je toto velmi používaný nástroj, ale já v tom nevidím nic navíc, co bych nemohl udělat pomocí alarmu v objektech. Také je důležité zmínit, že lze vytvořit momenty pro negativní tik, ale je to úplně zbytečné, protože se nikdy nevykonají.

## Objects

Toto je nejpoužívanější skupina. Zde může uživatel vytvořit objekty. Pokud by uživatel chtěl vložit objekt do své hry, tak se mu to nepovede. Objekty jako takové slouží jenom jako šablona pro vytvoření kopie. Tato kopie se pak vloží do hry. Tyto kopie nazýváme instance. Každá instance má své unikátní ID. Díky tomu, že se do hry vloží instance a ne objekt, tak se může vytvořit neomezeně instancí (avšak jen do té doby, než dojde paměť) a s každou instancí se může pracovat zvlášť, a to díky ID. Objekt je v základu bez velikosti a bez Sprite. Když uživatel dosadí Sprite do objekt, tak se automaticky nastaví velikost objekt na velikost Sprite. Tato velikost lze upravovat v objektu pomocí předdefinovaných proměnných image\_xscale a image\_yscale. Těmto dvěma proměnným lze nastavit i negativní číslo, poté se celý obrázek převrátí po ose. Pokud uživatel chce, aby objekt dělal něco, tak musí vytvořit jeden z mnoha možných událostí. Události slouží k vykonání kódu za speciálních podmínek, jako je např. vytvoření objektu, každý herní tik, zničení objektu, kolize objektu s dalším objektem, zmáčknutí nějaké klávesy a mnohé další. Objektu bez Sprite s kódem v události Create funguje hodně podobně, jak by fungoval skript.

## **Rooms**

Slouží k vytváření jednotlivých herních místností. Každá hra musí mít alespoň jednu místnost, aby ji bylo možno zapnout. Pro pohyb mezi místnostmi se používají hlavně dvě funkce, a to jsou room\_goto\_next() a room\_goto(x). První funkce slouží pro načtení další místnosti. Další místnost je taková, která je umístěna hned za touto. Druhá funkce slouží pro přemístění do specifické místnosti. Jako vstupní proměnou lze použít jak celé číslo, tak lze použít i jméno specifické místnosti.

## Notes

Skupina notes slouží k vytváření textových dokumentů. Tyto dokumenty žádným způsobem neovlivňují průběh hry. Slouží pouze pro potencionální dokumentaci projektu nebo pro poznámky uživatele. Mě osobně tato skupina přijde úplně zbytečná, protože to, co poskytuje tato skupina lze dosáhnout pomocí použití komentářů.

## **Included files**

Tato skupina je velmi užitečná. Jdou do ní vkládat soubory, které chceme, aby vyšly společně se závěrečným exe souborem. Lze sem vložit např. ovládání hry nebo textový soubor typu READ ME.

## **Extensions**

Většinou jsou to rozšíření vydaná třetí stranou. Lze je získat na Game maker studio marketu. Jedná se jak o placené, tak i o neplacené položky. Tato rozšíření zjednodušují práci v Game maker studiu 2 tím, že přidávají funkce, které v Game maker studiu nejsou. Jednou z mnoha je konvertor xml, který slouží pro čtení dat z xml.

## **Options**

Slouží pro nastavení výstupu programu. Lze zde definovat mnoho věcí. Například, když chci hru vydat ve Windows, tak můžu nastavit: jaká společnost ji vydala, jak se ten produkt jmenuje, jestli má copyright a další.

## Configurations

Zde lze nastavovat konfigurace pro jiné platformy, když vytváříme na více platforem najednou.

V tomto bodě jsem si také rozmyslel, jak by moje hra měla vůbec vypadat. Řekl jsem si, že chci, aby byla pokaždé náhodně generovaná a aby hráč nikdy nehrál stejný úroveň vícekrát. A tak jsem si začal rozmýšlet, jak chci, aby se hra vytvářela. Ale jelikož tento bod jsem měl dělat až později, tak jsem si tím zatím nelámal hlavu a šel jsem na krok.

# Základní implementace hráče

Jelikož vytvářím hru, která je hodně podobná hře Binding of Isaac, tak jsem chtěl vytvořit úplně stejnou postavu v mé hře, jako je v originálu. Myslím, že finální verze je hodně podobná, ba i identická. Když jsem poprvé začal plnit tento bod, tak jsem si vytvořil pouze jeden objekt s Sprite celého těla Isaaca. Ale hned bylo vidět, co je špatně. V originálním Binding of Isaac lze běžet jedním směrem a útočit jiným směrem. U mě to šlo také, ale jelikož jsem měl pouze jeden sprite, tak bylo nemožné, aby se hlava hráče koukala jinam než tělo hráče. Proto jsem musel hráče vytvořit úplně jiným způsobem. Tak mě napadlo, že hráče vytvořím tak, že udělám dva objekty. Jeden objekt by ovládal otáčení těla hráče, pohyb a vlastně všechno, co hráč musí umět, a druhý objekt by se staral jen o hlavu a její následování těla. Tak jsem se dal do práce. Začal jsem děláním pohybu. Chtěl jsem obstarat všechny možnosti, co hráč ohledně pohybu může udělat. Tím myslím, aby objekt věděl, co dělat, když hráč zmáčkne W, A, S, D najednou. Proto jsem si vypsal všechny kombinace kláves v podmínkách if, které lze zmáčknout a postupně jim nastavoval hodnoty. Celkem tato velká kombinace podmínek if a else if má přes 100 řádků. Když se zpátky koukám na tento kód, tak si říkám, že by to šlo napsat o dost jednodušeji, kdybych udělal pouze 4 podmínky if pro každou klávesu a vždy, pokud by byla nějaká zmáčklá, tak bych nastavil směr. Tímto bych si o hodně zkrátil řešení pohybu hráče, ale zase bych musel nějakým způsobem řešit animace a Sprite pro hlavu. Proto jsem to nechal tak, jak to je zatím napsáno. Tady se pokusím vysvětlit jednu z těchto podmínek, a proč to dělám tak, jak to dělám:

*if (keyboard\_check(ord("D")))*

*{*

*sprite\_index = spr\_player\_left;  
image\_xscale = 2.5;  
obj\_player\_head.image\_index = 2;  
x = x + rychlost;*

*}*

Tato podmínka je napsán pro každou kombinaci pohybových znaků. První řádek kódu slouží k nastavení animace na objektu hráče. Celkem mám 3 různé Sprite, které jsou načítány do hráče. Tento je zrovna pro pohyb těla do strany. Ještě mu můžu nastavit Sprite pro pohyb nahoru, který je úplně stejný jako pohyb dolů, nebo pouhý stojící Sprite, když hráč nic nedělá. Druhý řádek slouží ke zvětšení hráče, ale taky k převrácení hráče, pokud by se jednalo o animaci na druhou stranu. Díky tomuto mi stačilo udělat animaci jen na jednu stranu. Třetí řádek slouží k nastavení správné strany hlavy. A poslední řádek pro samotný pohyb hráče. Toto vše se nachází v události „step“, která proběhne každý tik hry. Zde ještě řeším mnoho dalších věcí. Například samotné útoky hráče. Útoky jako takové jsou zase 4, které se ptají, jestli je zmáčknuta klávesa pro útok, což jsou šipky, ale taky zároveň jestli je cooldown 0. Cooldown je pomocná proměnná, která slouží pro nastavení toho, jak rychle hráč může útočit. V útocích ještě malinko posunu hlavu a hned zase zpátky, aby to vypadalo, že hráč doopravdy útočí. Hlavně pokaždé, když se zaútočí, tak se nastaví cooldown na global\_cd. Global\_cd je proměnná, kterou zvyšuji nebo snižuju na základě předmětů, které hráč sebere a slouží určení intervalu, ve kterém hráč může útočit. Cooldown je proměnná, která slouží k zjištění, jestli hráč už může zase útočit nebo ne. Každý herní tik ji snižuji o jednu. Také v hráči řeším životy. Pokaždé když se hráči sníží život, tak nastavím proměnnou cooldown\_hp na 30. Tato proměnná se také snižuje každý tik. Ale hlavně slouží k nastavení krátkodobé nesmrtelnosti hráče. Díky této proměnné také řeším grafické problikávání hráče, aby bylo vidět, do kdy je nesmrtelný. Tato krátkodobá nesmrtelnost je velmi důležitá, protože jinak by při kolizi s nepřítelem zemřel instantně. Také zde nastavuji pozici kamery tak, aby vždy následovala hráče z místnosti do místnosti. Jako další věc zde řeším zamykání místností, pokud se v nich nachází nepřítel. Tato věc je velmi důležitá, protože jinak by hráč mohl proběhnout všechny místnosti, aniž by musel zabíjet jediného nepřítele. Řeším to tak, že se zeptám, jestli v místnosti, kde se nachází hráč, se nachází jakákoliv instance nepřítele, pokud ano, tak nastavím na proměnnou global\_door=false a pokud ne, tak nastavím global\_door=true. Také zde řeším základní herní menu, jako je například ztlumení hry nebo navrácení do hlavního menu. V poslední řadě zde také vytvořím počítadlo na aktuální počet peněz, které hráč má a může využít k nákupu předmětů v obchodě. A také počítadlo na životy hráče. Život není zobrazován číselně jako peníze, ale graficky pomocí srdíček. Srdíčka se aktualizují pokaždé, když se sníží počet života, který má hráč. Přišlo mi málo, mít jen jeden typ charakteru, za který může uživatel hrát. Tak jsem vytvořil možnost pro uživatele vybrat si vždy vybrat jeden ze tří charakterů, než začne hru. Každý z těchto charakterů má svou vlastní Sprite a své vlastní začínající hodnoty útoku, rychlosti a života.

Výběr charakterů - obrázek níže

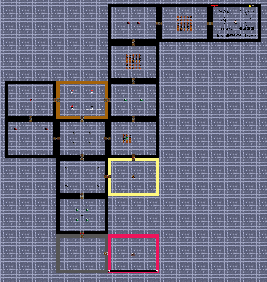
# Implementace herního prostředí, vytvoření úrovní

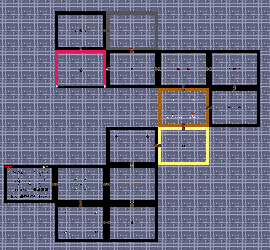
Běžně se používají místnosti pro každou herní místnost v Game maker studiu 2. Já jsem v celé hře použil pouze 5 místností. I když místností, kterýma hráč může projít, je víc, tak se skoro všechno děje ve 4té místnosti. Všechny místnosti mají velikost 1024 pixelů na 768 pixelů, kromě čtvrté, ta má 20480 pixelů na 15360 pixelů. První místnost hráč ani neuvidí, slouží pouze k nastavení několika globálních proměnných. Jako je počítadlo úrovní, ve kterém se hráč nachází. Jaký charakter se má vytvářet. Jestli je zvuk zapnutý nebo ne. A jako poslední proměnou, jestli se má charakter po vytvoření resetovat nebo ne. Protože tahle proměnná může někomu přijít divná, tak ji krátce vysvětlím. Vždy, když začne hra, tak se charakter musí resetovat. To znamená, odstranit všechny předměty z předchozí hry a nastavit životy na základní. Ale pokud bych tu neměl tuto proměnnou, tak by se charakter resetoval i po postupu do další úrovně. Druhá místnost slouží k hlavnímu hernímu menu, kde uživatel může vykonat jednu z následujících možností: spuštění hry, nastavení charakteru nebo vypnutí hry. Toto je místnost, kterou hráč uvidí hned, když zapne hru. Další místnost slouží pro nastavení charakteru, za který chce hráč hrát. Čtvrtá místnost je hlavní místností. Celá hra se odehrává jen v této místnosti. Poslední místnost je místo, kam se hráč dostane, když zemře. Čtvrtá místnost je takhle velká proto, že obsahuje všechny hrací místnosti, kterýma hráč prochází. Celé herní prostředí negeneruji jako normální místnosti, které bych nějak poskládal, ale jako hromadu objektů, které jen mezi sebou poskládám a propojím pomocí dalších objektů. Moje objekty pro tvoření místností se skládají z pěti druhů. První z nich je obyčejná místnost, která obsahuje buď nepřítele nebo srdíčko, které lze sebrat, a podobně. Vždy vygeneruji alespoň jednu normální místnost, kde vložím klíč, kterým lze odemknout cestu k závěrečnému protivníkovi úrovně. Další druh místnosti je místnost s předmětem zadarmo. Předmět v této místnosti je vždy náhodně vybraný z kolekce všech mnou vytvořených předmětů. Tyto předměty se skládají jak z předmětů na zvýšení hodnot zdraví apod., tak i ze speciálních, jako je například blokování nepřátelských střel nebo dvojitý útok. Taky existuje obchod neboli obchod. V obchodě může hráč nakupovat buď srdíčka, pokud má málo života, nebo předměty na zvýšení hodnot zdraví apod. Jelikož se v obchodě objeví vždy dva rozdílné předměty, tak vytvářím pouze předměty na zvýšení hodnot zdraví apod. Kdybych vytvářel i předměty lepší, tak by hráč vždy koupil pouze nejsilnější předmět a velice by mu to snížilo obtížnost hry. Dále tu je druh, který je velmi podobný místnosti s předmětem. V této místnosti může hráč stoupnout na bodáky, které mu seberou jeden život, ale dostane za to předmět. Posledním druhem je místnost závěrečného protivníka. Do této místnosti se může hráč dostat pouze tehdy, co sebere klíč na odemknutí. Z této místnosti se nelze dostat zpátky. Hráč proto musí zabít závěrečného protivníka, který po svojí smrti vytvoří průchod do dalšího úrovně.

# Vytvoření systému pro náhodné generování

Celé náhodné generování řeším v jednom objektu. Začnu tím, že si vytvořím dvoudimenzionální pole celočíselných čísel a do každého načtu 0. Následně si určím první místnost doprostřed tady toho pole a dám mu hodnotu 5. Z této pozice následně generuju náhodný směr, kde bude další místnost. Když vygeneruju směr, tak se zeptám, jestli tam už náhodou místnost není, nebo jestli se nenacházím na hraně pole. Pokud na obě otázky odpovím ne, tak tímto směrem vytvořím místnost, ale pokud na jednu z nich odpovím ano, tak vytvářím nový směr. Tento cyklus provedu 14 krát. Na poslední prvek v tomto cyklu dosadím číslo 3. To slouží pro vygenerování místnosti se závěrečným protivníkem. Až tento cyklus proběhne, tak budu mít vygenerovaný základní layout mapy. Poté si vyberu několik náhodných prvků z tohoto cyklu a dosadím do nich tato čísla: 2, 4, 6, 7. Tato čísla, společně se všemi předchozími, slouží pro generování určitých místností. Nyní si vytvořím dva for cykly, kterými projdu znova celé pole. Pokud for cykly narazí na číslo 1, tak vytvoří obyčejnou místnost. Tato místnost si vygeneruje náhodný celý číslo od 1 po 9. Na základě tohoto čísla pak vygeneruje jednu z devíti možností pro normální místnost. Když nalezne číslo 2, tak vytvoří místnost s předmětem. Když najde číslo 3, tak vytvoří místnost s náhodným závěrečným protivníkem. Pro číslo 4 vytvořím obchod. V obchodě si taky definuji proměnnou pro předchozí vygenerovaný předmět, abych zamezil duplikaci předmětů. Pomocí čísla 5 vygeneruji startovací místnost. V této místnosti taky vytvořím objekt hráče, života a peněz. V této místnosti se také objeví základní ovládání pro hráče, aby věděl, jak hru hrát. Číslem 6 se vygeneruje místnost s klíčem pro odemčení místnosti závěrečného protivníka. A naposledy se pomocí čísla 7 vygeneruje místnost s bodáky. Ještě by chtělo podotknout, že se pro každou místnost generují specifické dveře, aby hráč věděl, co se za danými dveřmi skrývá a aby nevstoupil do prázdna. Každá z místností má svůj vlastní typ dveří, které se skládají ze 4 druhů. Což znamená, že jsem celkem musel vytvořit 25 objektů pro dveře včetně poklopu z místnosti závěrečného protivníka.

Níže uvedené ukázky finálního náhodného generování mapy





# Závěr

Závěrem bych chtěl říci, že pro mě byla tato práce velice prospěšná. Hrát hry mě baví a vytvářet hry mě baví ještě víc. Za jedno z velkých plusů práce na tomto zadání považuji naučení se základům nového jazyka Game maker language a praktické používání tohoto jazyka v Game maker studiu 2. Určitě se budu v budoucnu vracet k vytvoření dalších her. Ale ještě než začnu dělat další hru, tak bych chtěl dovést do úplného konce tento projekt. Už mi chybí dokončit jenom pár věcí, které nemají vliv na vlastní chod hry, jako je udělat více nepřátel a více závěrečných protivníků, dále zlepšit grafiku místností, aby to nebyly jednobarevné zdi a nakonec ještě vytvořit více zajímavých předmětů.

Zároveň bych tímto chtěl poděkovat vedoucímu mé dlouhodobé maturitní práce panu učiteli Ondřeji Kolínovi za pomoc při vypracování.

# Přílohy

## Zdroje

*Sprites* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs2.yoyogames.com/source/_build/3_scripting/4_gml_reference/sprites/index.html>

*Tiles* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/more%20about%20rooms/tiles.html>

*Audio* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/002_reference/game%20assets/sounds/index.html>

*Paths* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/004_paths.html>

*Scripts* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/006_scripts.html>

Shaders [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs2.yoyogames.com/source/_build/3_scripting/4_gml_reference/shaders/index.htm>

*Fonts* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/003_fonts.html>

*Timelines* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/005_time%20lines.html>

*Objects* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/002_reference/objects%20and%20instances/index.html>

*Rooms* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/002_reference/rooms/index.html>

*Notes* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs2.yoyogames.com/source/_build/2_interface/1_editors/notes.html>

*Files* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/002_including%20files.html>

*Extensions* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://marketplace.yoyogames.com/category/35/extensions>

*Options* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs2.yoyogames.com/source/_build/2_interface/3_settings/game_options/index.html>

*Configurations* [online]. Dundee: YoYo Games, 2017 [cit. 2018-03-26]. Dostupné z: <https://docs.yoyogames.com/source/dadiospice/001_advanced%20use/007_configurations.html>

YoYo Games. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-03-26]. Dostupné z: <https://en.wikipedia.org/wiki/YoYo_Games>

Mark Overmars. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-03-26]. Dostupné z: <https://en.wikipedia.org/wiki/Mark_Overmars>

Game Maker Studio. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-03-26]. Dostupné z: <https://en.wikipedia.org/wiki/GameMaker_Studio>

The Binding of Isaac. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-03-26]. Dostupné z: <https://en.wikipedia.org/wiki/The_Binding_of_Isaac_(video_game)>

1. Rozhraní [↑](#footnote-ref-1)
2. Sprite – obrázek, který slouží jako viditelné „tělo“ objektu [↑](#footnote-ref-2)
3. Image strip – obrázek, který je rozdělen na menší obrázky [↑](#footnote-ref-3)