Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj herní aplikace pro PC v prostředí Unity

Vypracoval: Jiří Novák
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2019

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Ekonomická fakulta
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jiří NOVÁK**
Osobní číslo:	**E16546**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Vývoj herní aplikace pro PC v prostředí Unity**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem práce je vytvořit aplikaci pro PC pomocí herního engine Unity. Aplikace bude provedena jako 2D arkádového typu s použitím grafické techniky sprite. Ve hře bude uživatel pohybovat postavou do stran s možností skákat, v okolním prostředí budou umístěny nástrahy, kterým se bude nutné vyhnout či je jinak zneškodnit. Hra bude obsahovat alespoň 10 úrovní.

Metodický postup:

1. Studium odborné literatury.
2. Návrh a popis vývoje a implementace výsledné aplikace.
3. Zhodnocení hratelnosti aplikace ostatními uživateli.
4. Vypracování doporučení a závěrů.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **DaGraca, M., & Lukosek, G. (2018).** *Learning C# 7 By Developing Games with Unity 2017.* **(3rd ed.). Birmingham, UK: Packt.**
2. **Doran, J. P. (2017).** *Unity 2017 Mobile Game Development.* **Birmingham, UK: Packt.**
3. **Hocking, J. (2015).** *Unity in Action: Multiplatform game development in C# with Unity 5.*. **Shelter Island, NY: Manning Publications.**
4. **Lavieri, E. (2018).** *Getting Started with Unity 2018: A Beginner's Guide to 2D and 3D game development with Unity*. **(3rd ed.). Birmingham, UK: Packt.**
5. **Lintrami, T. (2018).** *Unity 2017 Game Development Essentials.* **(3rd ed). Birmingham, UK: Packt.**
6. **Price, M. J. (2017).** *C# 7.1 and .NET Core 2.0: Modern Cross-Platform Development.* **(3rd ed.). Birmingham, UK: Packt Publishing.**
7. **Sapio, F., & Ferro, L. S. (2018).** *Unity 2017 2D Game Development Projects.* **Birmingham, UK: Packt.**
8. **Thorn, A. (2017).** *Mastering Unity 2017 Game Development with C#.* **(2nd ed.). Birmingham, UK: Packt.**
9. **Troelsen, A., DaGraca, M., & Lukosek, G. (2018).** *Learning C# 7 By Developing Games with Unity 2017.* **(3rd ed.). Birmingham, UK: Packt.**

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **19. ledna 2018**

Termín odevzdání bakalářské práce: **12. dubna 2019**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

L.S.

doc. RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 28. března 2018

Prohlašuji, že svoji bakalářskou práci „Vývoj herní aplikace pro PC v prostředí Unity“ jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to - v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou - elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

08.04.2019 .....

Datum Podpis studenta

Obsah

1 Úvod

V dnešní době se stále více vývojářů přiklání k nezávislé tvorbě videoher. Může za to fakt, že se zdokonalují veřejně dostupné nástroje, které tuto tvorbu podporují a vytvoření vlastní hry se stává čím dál tím jednodušší. Tato činnost zahrnuje prvky, jako jsou programování, grafika, vyprávění příběhu nebo skládání hudby. Herní enginy usnadňují kombinování těchto disciplín a umožňují i lidem méně zdatným v těchto disciplínách, stále realizovat jejich kreativní nápady ve formě videoher. Cílem této práce je seznámit se v teoretické části s problematikou herního vývoje společně s využitím enginu Unity. Cílem praktické části je vytvoření 2D hry ve stylu platformer za pomocí informací získaných v teoretické části. Tato hra bude mít alespoň deset úrovní, které bude moct hráč pokořit pomocí pohybu hlavním charakterem v horizontálním směru a pomocí skoku ve vertikálním směru. Výsledná aplikace bude otestována ostatními uživateli, kteří následně poskytnou zpětnou vazbu.

V teoretické části se věnuji definici videoher společně s jejich historií a rozdělením podle žánrů. Poté se zde zabývám rozdělením tvorby videoher a platformami, na které je možné hry vyvíjet. A dále zde přibližuji problematiku herních enginů a rozdělení tvorby grafiky, v praktické části pak popisuji svůj postup při tvorbě jednotlivých prvků videohry. Také se zde věnuji vizuální stránce hry a softwaru, který jsem při tom využil.

2 Definice videohry

Seznámím vás s definicí videoher podle amerického designéra Chrise Crawforda, který se inspiroval od předchozích nápadů Davida Walkera a Evana Robinsona.

Zábava je kategorie činností, které nám přináší radost a uspokojení. Tento velký okruh můžeme rozdělit na dvě části, a to interaktivní příběhy a „Playthings“. Tento druhý pojem raději ponechám v originálním jazyce, ale dal by se přeložit, jako hříčka.

Interaktivní příběhy mají definovanou příběhovou linii, jejíž prostřednictvím dosáhnou předem definovaný cílový stav. Uživatel může procházet různými větvemi příběhu, ale konečný směr zápletky se nemění. Primárním záměrem zábavy zde není řešení hádanek, ale požitek z příběhu samotného.

„Playthings“ jsou systémy, které uspokojují hráče reakcí na jeho činy. Jedná se o systémy s definovaným chováním založeným na fyzikálních vlastnostech, formálních pravidlech nebo algoritmech. Chování je zábavné, protože je hráč zaujat strukturou hry a hraním samotným. Existují dvě podkategorie, a to hračky a výzvy. Rozdíl mezi oběma třídami spočívá v přítomnosti definovaného cíle. Přítomnost takového cíle poskytuje samotný hráč při hraní s hračkou. To znamená, že daná hra může být použita jedním hráčem jako hračka a později použita jiným hráčem jako výzva. Rozdíl vyplývá pouze z hráčova pohledu na hru a toto rozdělení není pro hru nezbytně nutné.

Hračky jsou „Playthings“ bez definovaných cílů. Hráč používá hračku nestrukturovaným způsobem, aniž by vysloveně vyhledával cíl. To však neznamená, že akce hráče jsou libovolné. Hráč může být stále zapojen do zkoumání hry, čímž určitým způsobem určuje chování hračky. Průzkum hráče může skutečně následovat určitou strukturu, ale tato struktura není zaměřena na uspokojení jakéhokoliv jiného cíle, než je stanovený systémem.

Výzvy jsou hračky s jasně definovanými cíli. Hráč výzvy se snaží dosáhnout určitého standardu výkonu. Výzva může mít mnoho podob: fyzická, jako je atletický sport, koordinace zraku a ruky, jako v arkádových hrách, nebo intelektuální, jako v šachu. Existují dvě podkategorie výzev, a to hádanky a konflikty, diferencované přítomností úmyslných oponentů. Úmyslný oponent nemusí být vždy lidskou bytostí. Místo toho to může být jakákoliv kolekce algoritmů, vytvářející v mysli hráče iluzi úmyslné vůle, která se snaží hráče porazit. Obtížnost je určena jak složitostí algoritmů, tak vnímavostí hráče. Dostatečně bystrý hráč nebude vnímat umělou inteligenci jako nic jiného než sbírku algoritmů, které mají být hádány. K takovému hráči neexistuje žádný cílevědomý soupeř, pouze hádanka.

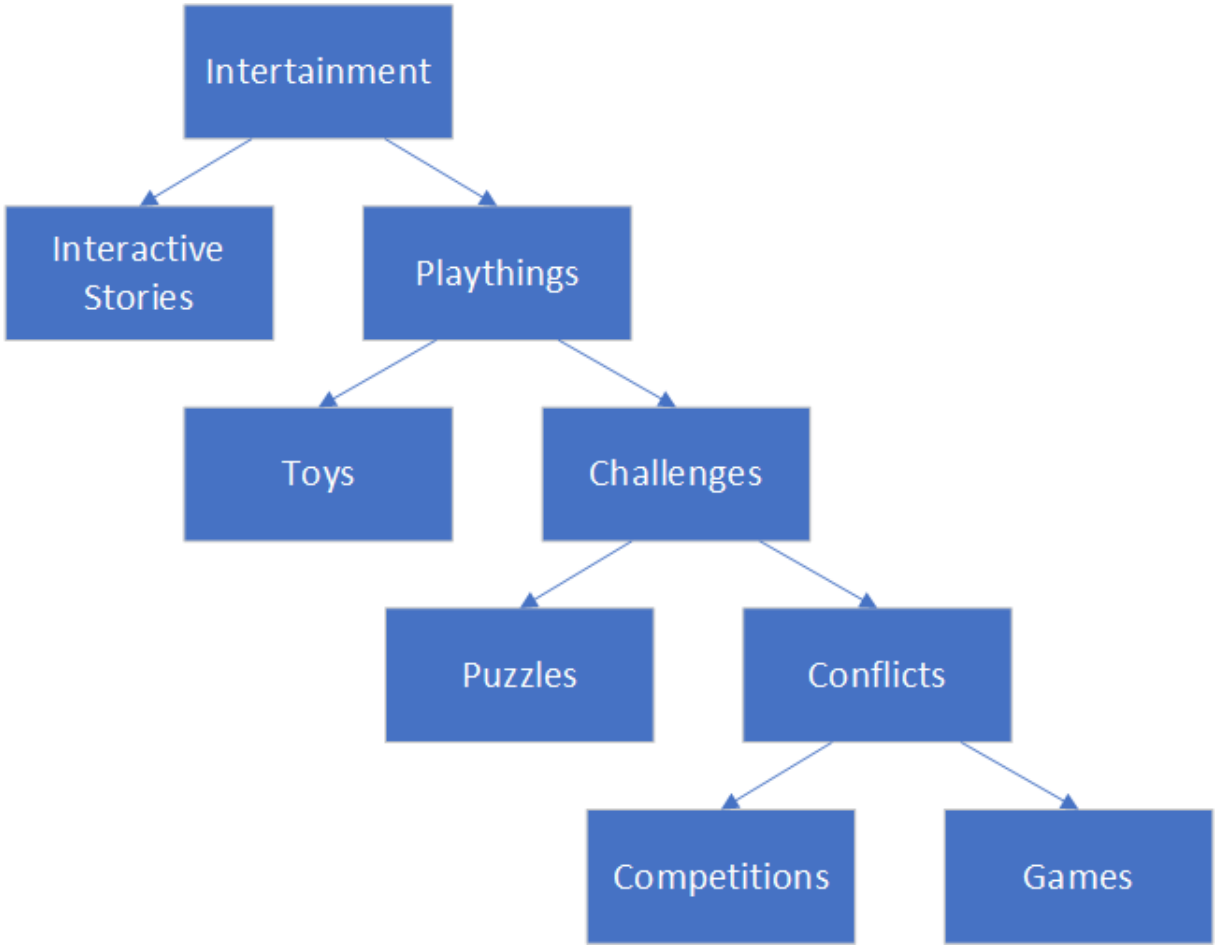
Hádanky jsou výzvy, které nemají žádné cílevědomé soupeře. Mají jasně definovaný cíl a různé překážky, které musí hráč překonat, aby dosáhl cíle. Překážky mohou být aktivní nebo dynamické a mohou mít i vlastní chování. Nicméně, pokud hráč nepřikládá jejich chování účelný pokus zmařit jeho činy, jsou to spíše překážky než soupeři a výzvou je hádanka. Většina produktů zábavního softwaru jsou hádanky.

Konflikty jsou výzvy s cílenými protivníky. Existují dvě podtřídy, a to soutěže a hry. Rozlišují se podle schopnosti oponentů bránit vzájemnému výkonu. Jsou-li soupeři omezeni v tom, aby se navzájem bránili a místo toho věnovali většinu své pozornosti maximalizací vlastního výkonu, pak je konflikt soutěží. Pokud je na druhou stranu dovoleno zranit si navzájem výkon, pak je konflikt hrou.

Soutěže jsou konflikty, aniž by se bránilo konkurenci v jejím progresu. Příkladem mohou být závody a politické kampaně. Je téměř vždy v rozporu s pravidly závodu pro závodníky, aby jednali přímo proti sobě.

To nám ponechává hry jako konflikty, ve kterých hráči komunikují takovým způsobem, že si vzájemně maří cíle. Jinými slovy, pokud můžete poškodit soupeře a soupeř vám škodí způsobem, který vás přesvědčí, že vás chce úmyslně poškodit, pak se jedná o hru. To vše lze shrnout pomocí taxonomického diagramu (Obrázek 1). (Crawford, 2011)

Obrázek 1: Diagram definice hry



(Crawford, 2011)

3 Historie videoher

První digitální počítače byly dokončeny počátkem čtyřicátých let, ale první počítačové hry se datují až o deset let později. Přesuneme-li se do této doby, můžeme říci, že počítačů bylo stále velice malé množství a byly dedikovány velmi specifickým funkcím. Bylo je obtížné přeprogramovat, tedy pokud to vůbec bylo možné a postrádaly možnost vykonávání programů uložených přímo v jejich paměti. V důsledku toho, i když by výzkumník usoudil, že simulace nebo hra by mohla být v jeho práci užitečná, měl by pouze málo příležitostí ji vytvořit. Nicméně v padesátých letech prošli počítače komercializací a staly se dostatečně sofistikovanými na plnění odlišných úkolů.

Programům, které by se daly nazývat hrou, v té době určitě vzniklo více, ale většina z nich byla výzkumnými projekty, které nebyly zpřístupněny široké veřejnosti. Bertie Brain (Obrázek 2) byl počítač postavený na zakázku, financovaný Rogersem Majesticem a jedná se o jednu z nejstarších známých počítačových her. Na rozdíl od většiny her tohoto období, byl Bertie určen, aby zapůsobil na veřejnost, jako potenciál počítačů. (Smith, 2014)

Obrázek 2: Počítač Bertie Brain zobrazující hru Nim (1950)

(Smith, 2014)

Nim je hra pro dva hráče, která hráčům poskytuje žetony v řadě hromádek. Hráči se postupně střídají po tazích a odebírají z hromádky jeden, dva nebo tři žetony. V základní hře je cílem, vzít poslední token a v reverzní verzi se snažíte přinutit druhého hráče, aby si vzal poslední právě on. Tuto hru umožnil vyzkoušet veřejnosti počítač s názvem NIMROD. Byl vystaven na „Festival of Britain“ v květnu roku 1951. Počítač stále nezahrnoval monitor, a proto používal řadu světel jako displej, který reprezentoval jednotlivé tokeny. Žetony si hráč vybíral pomocí tlačítek umístěných na ovládacím panelu stroje. NIMROD byl sestaven australským inženýrem Johnem Bennetem a byl spíše zaměřen na demonstrování programovacích algoritmů a principů než k zábavným účelům. ("Challenge NIMROD", 2011)

Ke konci roku 1952 Arthur Samuel, který byl v tu dobu zaměstnancem společnosti IBM, vytvořil program na hraní dámy. Jedná se o jedno z prvních využití jednoduché umělé inteligence. Inspiroval se verzí dámy vytvořené Christopherem Stracheym, která byla vytvořena o rok dříve, ale nebyla zcela funkční. Samuelova verze byla vytvořena na počítači IBM 701 a její vývoj pokračoval i mnoho let poté. V roce 1955 dosáhl program významného

milníku, kdy dokázal analyzovat svou vlastní hru a byl se schopen učit ze svých chyb. (Smith, 2014)

Pokud bychom chtěli mluvit o první komerčně úspěšné arkádové hře, museli bychom se přesunout do roku 1972. V tomto roce byla založena společnost s názvem Atari, Inc., která vydala svou první hru PONG. Jednoduchý 2D tenis pro dva hráče se stal masivně populární a o dva roky později prodal přes 35000 kusů. Úspěch této hry vyvolal velkou vlnu konkurence a společnost Atari s tím nemohla nic udělat, jelikož nestihla zaregistrovat patent své technologie. Herní experti a publikace považují PONG za hru, která odstartovala průmysl her, jako lukrativní podnikovou činnost. Dnešní cena původní hry PONG přesahuje 4000 dolarů. ("The History Of Video Arcade Games", n.d.)

Zůstaneme-li u arkádových her, mohu zmínit hru Pac-Man z roku 1980. Pac-Man je univerzálně považován za videoherní klasiku a je ikonický pro pop kulturu 80. let. Pac-Man byl vyvinut během osmnáctiměsíčního období zaměstnancem společnosti Namco Torem Iwatanim vedoucím devítičlenného týmu. Originální titul byl vyslovován „Pakku-Man“, kontrakce „Paku-Paku“, která popisuje pohyb úst při zavírání a otevírání. Pac-Man představil do videoher prvek humoru, který se další návrháři snažili napodobit a apeloval na širší demografickou skupinu než pouze na náctileté chlapce. Uspěl tím, že vytvořil nový herní žánr a poprvé zde byla hra, která apelovala i na ženy. ("The History Of Video Arcade Games", n.d.)

Přesuneme-li se do poloviny července 1983, tak mluvíme o založení společnosti Nintendo Entertainment Systém, v tu dobu ještě známou jako Family Computer, zkráceně Famicom. Nintendo šlo svou vlastní cestou a v říjnu 1985 zveřejnilo na trh konzoli NES, která svým designem předčila většinu své dosavadní konkurence. Dalším silným krokem ze strany Nintenda bylo zacházení s vývojáři třetích stran, kteří museli být pro vývoj her na systém Nintendo licencování. Dále mohli vydávat pouze dvě hry ročně a také nesměly být určené pro jiné konzole než Nintendo. Licencované hry obdržely na svých krabicích jak tištěnou pečeť kvality, tak speciální 10NES chip pro „cartridge“, který byl spojený s odpovídajícím chipem konzole. V počátcích byl NES možná nespolehlivý, ale pomohl se zbavit záplavy nízko kvalitního softwaru, který zlikvidoval nejednu konkurenci. (Cunningham, 2013)

Zakončil bych to skokem do roku 1994 přesněji 23. listopadu, kdy byla vydána hra Warcraft: Orcs & Humans. Jedná se o první hru ze série Warcraft a zároveň o jednu z prvních strategií v reálném čase, která se rychle stala nejlépe prodávanou hrou. Představuje dvě rasy, a

to lidi z království Azerothu a invazní orky z Hordy. Warcraft se stal nejen klasikou, která získala mnoho ocenění, ale také stanovil standardy pro multiplayerové hry. Software byl vydán pro platformy MS-DOS 5.0 a Mac 68030/68040. (FANDOM, 2010)

4 Žánry videoher

Existuje mnoho různých druhů videoher a typicky jsou kategorizovány podle jejich vlastností nebo základních cílů. Kategorie her nebo žánry mohou mít také své podkategorie a velká většina her spadá pod více žánrů. Dnešní technologie umožňuje vývojářům vytvořit cokoliv, co si můžou představit a pro spoustu her vytvořených v posledních letech, musely být vytvořeny nové kategorie her a žánrů, jelikož hry spadaly mimo tradiční klasifikaci her. Pokusím se zde vyjmenovat a přiblížit jejich základní rozdělení a popis, ale jsem si jist, že zde nevyjmenuji zdaleka všechny. (Vince, 2018)

4.1 Action Games

Akční hry jsou hry kde hráč ovládá herní prvek a je většinou ve středu akce, která se skládá především z fyzických výzev, které musí hráč překonat. Mnoho prvních videoher jako je Donkey Kong a Galaga spadají právě do této kategorie. Vzhledem k tomu, že je jednoduché se dostat do her této kategorie, drží akční hry první místo v oblíbenosti. (Vince, 2018)

4.1.1 Shooters

Střílecí hry nechávají hráče využívat zbraně, aby se zapojili do akce, jejímž cílem je většinou odstranit nepřátele nebo protivníky. Střelci jsou dále rozděleni podle pohledu hráče. Střílečky první osoby (FPS) se hrají z hlediska hlavní postavy. Dobrými příklady jsou Call of Duty, Half-Life a Halo. U střelců z třetí osoby, jako je Fortnite a Splatoon, se akce odehrává z pohledu nad postavou a mírně za ní. (Vince, 2018)

4.1.2 Fighting games

Bojové hry jako je Mortal Kombat a Street Fighter II se zaměřují především na bojová umění a v některých případech i na boj s nestřelnými zbraněmi. Většina bojových her má stabilní herní charaktery, kteří mají svou specializaci a ovládají jedinečné schopnosti nebo styl bojování. Ve většině tradičních bojových hrách, hráč postupuje ve své cestě k vrcholu, přičemž musím porazit více a více obtížné soupeře. (Vince, 2018)

4.1.3 Survival

Hry zaměřené na přežití jsou podkategorií akčních her a jejich rozvoj proběhl teprve v nedávných letech. Cílem hry je přežít s omezeným množstvím zdrojů. Hráč se v průběhu hry dostává k efektivnějším nástrojům, které mu pomáhají s přežitím. Hry tohoto žánru se od sebe často velice liší. Některé se zaměřují na čelení proti hrozbám ze strany přírody, kdežto například Fortnite se soustředí na předem definované herní prostředí a dává hráčům přístup ke zbraním, aby se mohli bránit před ostatními. (Vince, 2018)

4.2 Adventure games

Dobrodružné hry jsou roztříděny podle stylu hratelnosti, a ne podle příběhu nebo obsahu. A zatímco dala technologie vývojářům nové způsoby, jak vyprávět příběh, tak se dobrodružné hry od jejich textových předchůdců moc nevyvinuly. V dobrodružných hrách, hráči obvykle interagují s jejich prostředím a jinými postavami. Často se zde řeší různé hádanky pomocí vodítek, které hráč nachází v průběhu hry. Kromě příležitostných mini-her zde sotva najdeme tradiční akční prvky videoher, a proto nejsou tolik populární. (Vince, 2018)

4.3 Role-Playing games

Pravděpodobně druhým nejpopulárnějším žánrem hry jsou „RPG“ (Role Playing Games). Většinou se odehrávají ve středověku nebo mají „fantasy“ podtext. To je hlavně z důvodu původu žánru, jehož kořeny sahají k deskové hře Dungeon & Dragons a jiným obdobným hrám jež se hráli za využití pera a papíru s různými charaktery. Kulturní rozdíly mají velký dopad na tento žánr a často se rozděluje na „WRPG“ se západozemní kulturou a „JRPG“ s kulturou japonska. Hráči jsou v tomto žánru vystavování volbám, které ovlivní finální výsledek hry, z čehož vyplývá, že je zde mnoho alternativních konců. (Vince, 2018)

4.4 Simulation games

Hry z žánru simulace mají jednu věc společnou. Jsou navrženy tak, aby napodobovaly skutečnou nebo smyšlenou realitu. Jedním z nejpopulárnějších simulátorů života (Life Simulation) je The Sims a zároveň se jedná o jednu z nejprodávanějších videoher všech dob. V The Sims kontrolují hráči jednotlivé aspekty uměle vytvořeného života. Simulace umožňují hráčům manipulovat s genetikou postav nebo jejich vlastnostmi. Dokonce mohou mít pod

kontrolou i jejich reakce na různé situace. Do tohoto žánru by se dala zařadit i hračka Tamagotchi, jakožto simulace domácího mazlíčka. (Vince, 2018)

4.5 Strategy games

Strategické hry vychází svou hratelností z tradičních stolních her a dávají hráči přístup k často smyšleným světům a jejich zdrojům. Hráč ovládá dění na mapě pomocí příkazů, které rozdává jednotkám pod jeho vedením. Tyto hry vyžadují, aby hráči pečlivě vyvíjeli strategie a taktiky k překonávání výzev. V posledních letech se tento typ her přiklonil spíše k systému hratelnosti v reálném čase od jeho předchůdce založeném na střídání ve vykonávání tahů. (Vince, 2018)

4.5.1 Tower defense

Ve hrách Tower Defense, aby hráči vyhráli, musí odrážet vlny nepřátel ovládané počítačem. Tito nepřátelé se často nazývají „Creeps“. Hráči mají většinou na výběr z různých budov a věží, které mohou postavit kolem předem definované cesty. Schopnosti jednotlivých věží se následně snaží zabránit, aby nepřátelé prošli. Když hráč eliminuje dostatečné množství nepřátel, dostanou měnu na nákup více věží, nebo vylepšení pro již existující budovy. (Vince, 2018)

4.6 Sports games

Sportovní hry simulují sporty jako je fotbal, basketbal, nebo třeba golf. Mohou také zahrnovat olympijské sporty jako je lyžování nebo dokonce hospodské sporty jako jsou šipky nebo kulečník. Protivníci v těchto hrách jsou často řízeni počítačem, ale mohou to být i reální lidé. Do tohoto žánru spadají i závodní hry. (Vince, 2018)

4.7 Puzzle games

Posledním žánrem základního rozdělení jsou logické hry. Hráči jsou vyzváni k řešení logických hádanek, nebo jsou odkázáni na svou intuici v různých výzvách, kterými mohou být například bludiště. Tetris by spadal právě do této kategorie. (Vince, 2018)

4.8 Platformer

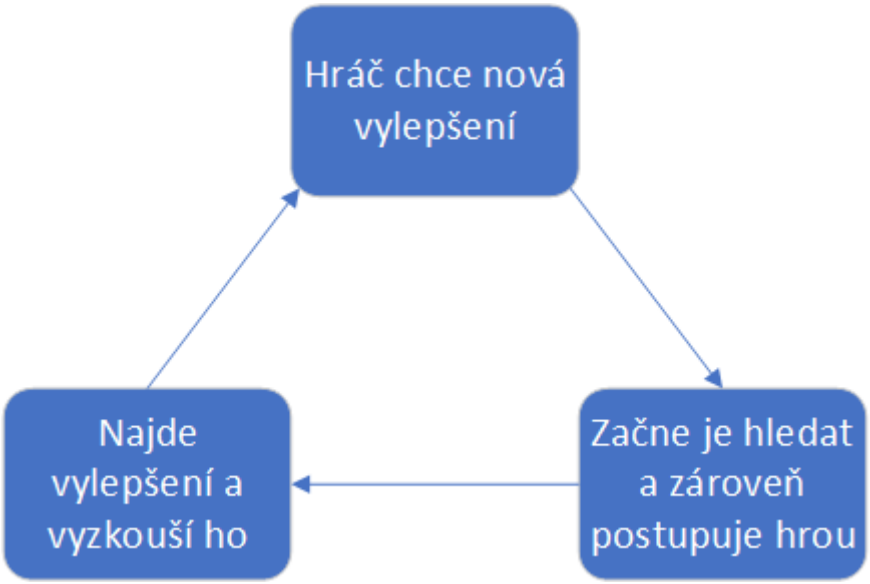
Platformové hry získali své jméno ze skutečnosti, že herní charakter interaguje s různými plošinami v průběhu hry. Většinou se jedná o úkony, jako je: běhání, skákání, chození a další jiné pohybové dovednosti. Samotné plošiny mohou nabývat různých velikostí a tvarů. Často obsahují i další vlastnosti, jako je jejich pohybování, propustnost herních objektů z omezených směrů nebo otáčení. Někdy mohou sloužit jako překážky a jindy mohou napomoci k dosažení herního cíle. Dříve se jednalo především o 2D hry, jako je „Mario“ a „Super Meat Boy“, ale v průběhu let se stále více a více vývojářů zasloužilo o spoustu 3D plošinových her. Tento žánr hry je v základu velice jednoduchý, jelikož hráče omezuje pouze v principu pohybování herního charakteru. Z toho důvodu je často adaptován ostatními žánry, jako jsou akční hry, střílecí hry nebo dobrodružné. (Vince, 2018), (Greenslade, 2006)

4.9 Metroidvania

Jedná se o jeden z mála žánrů, které byly pojmenovány na základě již existujících her. V tomto případě se jedná o hry Metroid a Castlevania. Spojením těchto titulů nám vzniká žánr spadající pod dobrodružné-akční hry. Není však lineárně založen a často vyžaduje, aby se hráči vraceli, nebo je zadržuje v postupu, dokud nezískají určitý předmět nebo speciální schopnost. Vylepšení postavy novými zbraněmi, schopnostmi a dalšími předměty, získá hráč přístup ke speciálním oblastem hry. Jednotlivé sekce jsou většinou zakončené bojem proti nepříteli, který svou silou převyšuje všechny ostatní běžné nepřátele.

Existují určité prvky pro vytvoření úspěšné Metroidvania hry. Příkladem může být intuitivní a zábavné ovládání hlavního charakteru. Pokud je zábavné samotné pohybování v herním prostředí, je zábavná i celá hra. Tento aspekt lze podpořit i přidáváním nových herních mechanik, které vyvolají v hráči další zájem zkoušet nové metody, jak překonávat překážky. Toto chování hráče se dá shrnout do určitého cyklu (Obrázek 3). Hráč se začne nudit a bude požadovat nová vylepšení nebo předměty, z toho vyplývá, že je začne hledat a přirozeně postupuje hrou. Po nalezení vylepšení ho vyzkouší a přesouvá se do prvního stádia cyklu. (Igarashi, 2016)

Obrázek 3: Metroidvania, cyklus chování hráče



(Zdroj: Autor, 2019)

Dalším prvkem je prozkoumávání herního prostředí. Hra by měla obsahovat hlavní cestu, která vede ke splnění cíle hry. Je však důležité, aby obsahovala i vedlejší odbočky a skrytá tajemství, která nemusí zcela nutně souviset se samotným dohráním hry. Je to obzvláště důležité v tu chvíli, kdy se hráč vrací, aby našel nějakou stopu, kterou předtím přehlédl. S objevováním těchto skrytých cest, přijde hráči, že dělá ve hře stále něco nového, i když se nachází v již prozkoumaném prostředí. Prvek objevování souvisí s předchozím, jelikož nově nalezená vylepšení často vedou k získání dovedností, která pomohou hráči dostat se do dříve nedostupných míst.

Při vytváření hádanek nebo záhad nutných pro dohrání hry, nesmí vývojář zapomenout, že hráč nezná veškeré lokace úrovní stejně jako ten, kdo všechny úrovně vytvářel a ví přesně kde se co nachází. Hráč musí být schopen intuitivně postupovat a žádná část hry by neměla být obzvláště matoucí. Co může být jednoznačné pro vývojáře, nemusí být vždy jednoznačné pro hráče. Proto je důležité získat co nejvíce zpětné vazby, aby byl hra jednoduchá na pochopení pro každého. (Igarashi, 2016)

5 Tvorba her

Game development neboli tvorba videoher, je proces, kdy jednotlivci nebo týmy vytváří hru. Při tomto procesu je zapotřebí přemýšlet nad nápadem hry, příběhem, vizuální stránkou, programováním, testováním, produkování, marketingem a spoustou dalších věcí, které na ně navazují. Tvorba videoher se dá rozdělit na dva základní vývojové přístupy. Výsledné produkty těchto přístupů se pak nazývají „Triple-A“ hry a „Indie“ hry.

5.1 Vývoj triple-A her

Triple-A hra (AAA) je obecně titul vyvinutý velkým studiem, které je financováno masivním rozpočtem. Tvorba takovéto hry je obdobná jako tvorba filmového trháku. Náklady na vývoj jsou obrovské, ale tržby za prodej konečného produktu za to stojí. Aby bylo možné tyto počáteční náklady získat zpět, tak vyvíjí převážně pro hlavní platformy jako jsou Xbox nebo Sony, aby maximalizovali zisky. Triple-A hry jsou většinou navrženy kolem toho, co již fungovalo v minulosti, což často omezuje přístup hry k širšímu okruhu uživatelů. (Schultz, 2018)

Výhody vývoje triple-A her:

- Jasně definovaný cíl
- Finanční stabilita
- Technologie přístupné ve velkých studiích
- Právní podpora

Nevýhody:

- Restrikce ze strany vedení
- Méně svobody kreativní tvorby
- Stresové situace během krize
- Lidé jsou jednoduše zaměnitelní

(Stenquist, 2016)

5.2 Nezávislý vývoj her

Indie vývoj neboli nezávislý vývoj, je proces tvorby videoher, kde je konečný produkt vyvíjen jednotlivci, nebo malými týmy, bez podpory oficiálních vydavatelských firem a významných finančních prostředků. Velký rozmach nezávislých vývojářů nastal kolem roku 2000 a každým rokem nabývá na oblíbenosti. V dnešní době se může skoro kdokoliv stát

vývojářem vlastní hry a bez větších problémů se ji může pokusit i zpeněžit. Existují tedy dva základní rysy pro nezávislé vývojářství, a to absence finanční podpory a časová nebo nápadová nezávislost. To znamená že částka vynaložená na projekt patří vývojářům nebo takzvanému davovému financování a při samotné tvorbě nejsou omezováni žádnými požadavky nebo rozhodnutími třetích stran, jako je marketingové oddělení. Vývoj závisí pouze na samotných vývojářích a může trvat od jednoho týdne až po několik let.

Výhody spojené s nezávislým vývojem mohou být:

- Volná pracovní doba
- Kreativní nezávislost
- Rozšiřování svých dovedností v závislosti překonávání různých překážek
- Nespočet dostupných nástrojů pro vývoj her
- Šance získání peněžního profitu za dokončený produkt.

Nesmíme však opomenout nevýhody, které nezávislý vývoj skýtá:

- Časová náročnost hledání financí
- Omezené zdroje a nástroje, které jsou běžně dostupné velkým týmům
- Větší šance na vznik drobných chyb
- Častá nutnost spolupráce
- Méně příležitostí pro vytvoření nových herních iterací

(Kir, 2016)

5.2.1 Týmové role

Jednotliví členové nezávislých týmů se často specializují na určitý úsek ve tvorbě videoher, jelikož ne všichni dokážou stejně dobře programovat jako kreslit nebo skládat hudbu. Z toho důvodu vznikají týmové role, které musí někdo zastávat. Těmito rolemi mohou být návrháři, pisatelé, producenti, testeři, projektoví manažeři, výtvarníci, hudební skladatelé a v poslední řadě programátoři. Ne vždy musí mít člen pouze jednu roli. Záleží to na typu vyvíjené hry a schopnostech jednotlivců. (NYCfilmAcademy, 2014)

Programátoři

Člověk může mít sebelepší nápad na videohru, ale bez programátorů žádnou kvalitní hru nevytvoří. Programátoři jsou ti, kteří většinu svého času tráví vývojem kódu a nástrojů pro hru. Obvykle dělají více než jen kódování. Rovněž přebírají odpovědnost za ladění a „beta“ testování, aby byla hra perfektní a bez zbytečných chyb. Dá se říci, že programátoři jsou nejdůležitější členové týmu. (NYCfilmAcademy, 2014)

Návrháři

Nezávislé týmy často nemají člena, který by se věnoval pouze návrhu. Obvykle se návrhářem stává někdo, kdo již zastává jinou roli. Mohou to být programátoři, umělci nebo dokonce pisatelé. Samozřejmě je dobré mít definovaného návrháře, který má konečné rozhodnutí a udává směr původní nápadu. (NYCfilmAcademy, 2014)

Výtvarníci

Umělci jsou rolí, kterou často najdete v nejednom nezávislém týmu. Je to proto, že tvorba grafické stránky hry zabírá spoustu času a je to první věc, co hráč vidí při hraní vytvořené hry, což je velice důležitým prvkem projektu. Výtvarníci často úzce spolupracují s programátory, návrháři a pisateli. Začínají konceptovým návrhem, ze kterého po schválení vytváří veškeré textury, sprity a animace. Počet výtvarníků pro projekt se může lišit. Pro 2D hru není potřeba velké množství umělců, ale pro 3D textury, jejichž vytvoření trvá mnohem déle, je dobré zamyslet se nad výhodností dalšího člena. (NYCfilmAcademy, 2014)

Zvukoví skladatelé

Vzhledem k tomu, že je zvuk pro většinu her neuvěřitelně důležitý, je dobré mít někoho kdo ví co dělá. Často se stává, že lidé umí dobře jednu činnost, přičemž zvládají i jinou na dostatečné úrovni, ale designování zvukových efektů mezi ně většinou nepatří. Lidí, kteří umí vytvářet krásné zvukové efekty a hudbu je v této oblasti výrazně méně nežli lidí zastávající jiné role. Zvuk je velice důležitým prvkem pro navození atmosféry ve hrách, a i když se přímo nejedná o hororovou hru, kde je zvuk neodmyslitelný, tak je stále dobré mít kvalitní hudbu i u her, jako jsou strategické a logické hry. (NYCfilmAcademy, 2014)

Testeři

V ideálním případě by měl být testerem každý v týmu. A to z toho důvodu, že při postupném vývoji každý testuje svou odpovědnou část a může odhalit chyby ostatních. Zároveň není špatný nápad mít někoho, kdo se přímo zaměří na hledání chyb. Mohou to být přátelé nebo lidé co se podíleli na financování. Je dobré zmínit, že by měli testovat i lidé co se nezabývají programováním, jelikož mají lepší nadhled než lidé, co strávili nad možná nefunkčním kódem hodiny svého času a jsou více náchylní k přehlížení chyb. (NYCfilmAcademy, 2014)

Pisatelé

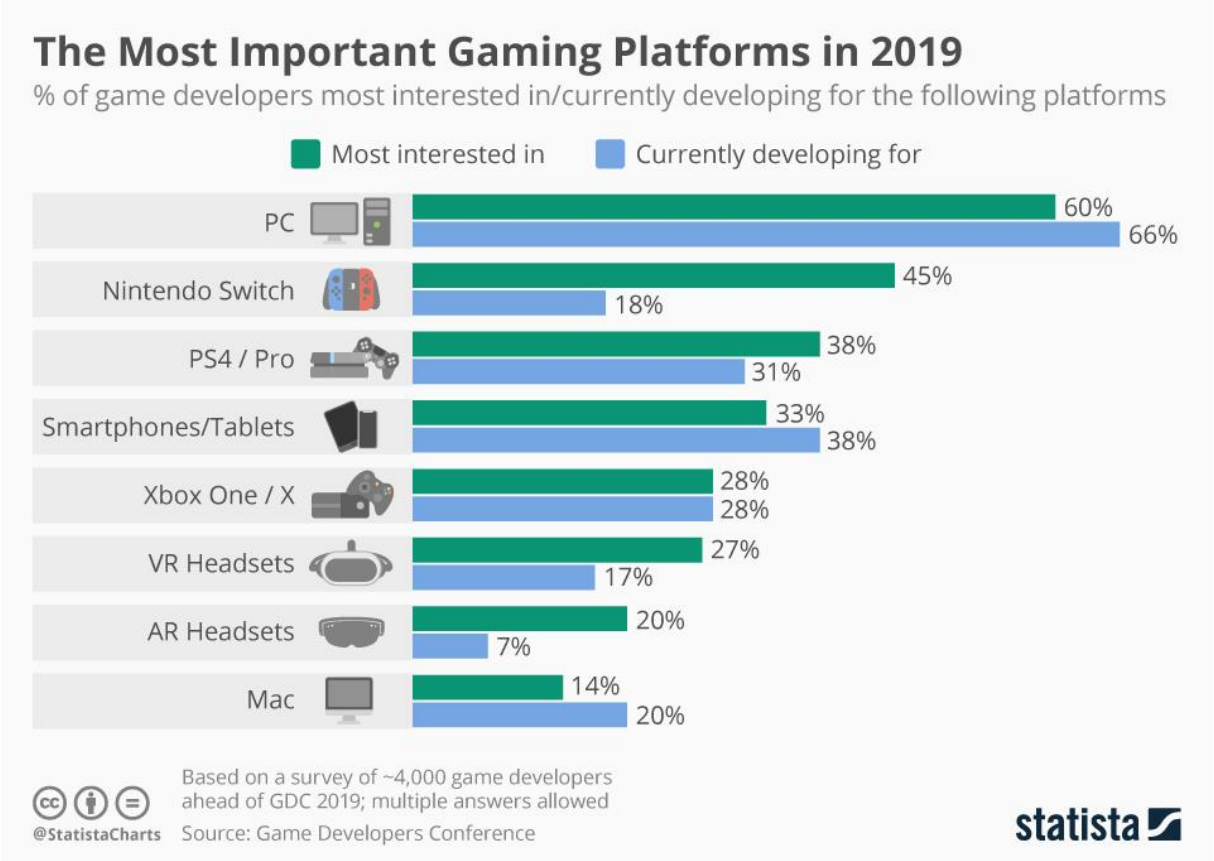
Pisatelé jsou odpovědni za vymýšlení příběhů a dějových linií. Často se tato činnost nazývá „Story Telling“, neboli vyprávění příběhu. K příběhu mohou přispívat i ostatní členové týmu, ale primárně by se tomu měla věnovat pouze jedna osoba, která vytváří zápletky, vlastnosti jednotlivých postav a jejich dialogy. Příběh se dá sdělit i jinak než jen

pomocí textu nebo zvuku, a proto by měli být pisatelé v úzkém spojení s návrháři a výtvarníky, aby vytvořili prostředí, které bude vyprávět příběh stejně dobře, jako samotný text. (NYCfilmAcademy, 2014)

6 Herní platformy

Videohry se hrají pomocí různých zařízení, která podporují spouštění herních aplikací. Obsahují uživatelské prostředí, díky kterému hráč ovládá herní prvky za pomocí určitého hardwaru. Může se například jednat o herní ovladače, dotykové obrazovky, klávesnice či pohybové senzory. Platformy, kterými se hrají videohry, se vyvíjely již od počátku. Od jednoduchých hracích automatů až po arkádové stroje. Dnes je k dispozici několik platforem pro hraní videoher. Těmi oblíbenými jsou mobilní zařízení, konzole a osobní počítače. (Richter, 2019)

Obrázek 4: Graf významnosti herních platforem



(Richter, 2019)

Na grafu (Obrázek 4) je vidět rozdělení herních platforem podle počtu vyvíjených herních aplikací a zájmu vývojářů o danou platformu. Nejzajímavější je asi poměr výše zájmu o platformu Nintendo Switch oproti stávající výši vývojářů. Dalo by se z toho usoudit, že v budoucích letech uvidíme nárůst vytvářených aplikací pro Switch, ke kterému může přispět i nová nahlášená verze Switche, která by měla být cenově dostupnější. (Richter, 2019)

6.1 Konzole

Konzole pro videohry jsou zařízení, která jsou speciálně vytvořena pouze pro hraní videoher. Obvykle jsou dodávány se vstupními zařízeními, jako je joystick a herní ovladač, které jsou propojeny s hlavní jednotkou provádějící veškeré výpočetní úkony. Jako vizuální výstupní zařízení slouží často televize, která se také zapojuje do hlavní jednotky. V současné době je na trhu k dispozici několik typů konzolí. Mezi oblíbené patří například PlayStation 4 nebo Xbox One. Existují také ruční konzole, které jsou malé, přenosné a mají vlastní obrazovku s napájením. Příkladem takových konzolí může být PSP Vita, nebo dnes velice populární Nintendo Switch. (Williams, 2015), ("Types of Video Game Platforms", 2016)

6.2 Mobilní telefony

Stále více a více populárním se stává hraní her na mobilních zařízeních. Může za to fakt, že střední třídy telefonů, které jsou více než dostatečné na hraní občasných her, klesly svou cenou na takovou úroveň, že si je může dovolit čím dál tím větší počet lidí. Někteří výrobci dokonce vyrábí speciální mobilní telefony zaměřené na hraní videoher. Tyto telefony mají často větší obnovovací frekvenci a silnější procesorovou řadu. Co se týče hraní, nemusí se jednat pouze o telefony ale i o tablety. Toto odvětví však poslední dobou upadá, jelikož se nové řady telefonů se svými poměry displeje ku tělu velikostně přibližují menším tabletům a uživatelé preferují vlastnit pouze jedno zařízení. Mobilní hry jsou obvykle vytvořeny pro mobilní operační systémy. V dnešní době se jedná převážně o různé verze Androidu a iOS pro IPhone. Existují i další operační systémy jako je Windows 10 Mobile nebo BlackBerry 10, ale ty jsou spíše zanedbatelné. (Williams, 2015), ("Types of Video Game Platforms", 2016)

6.3 Virtuální realita

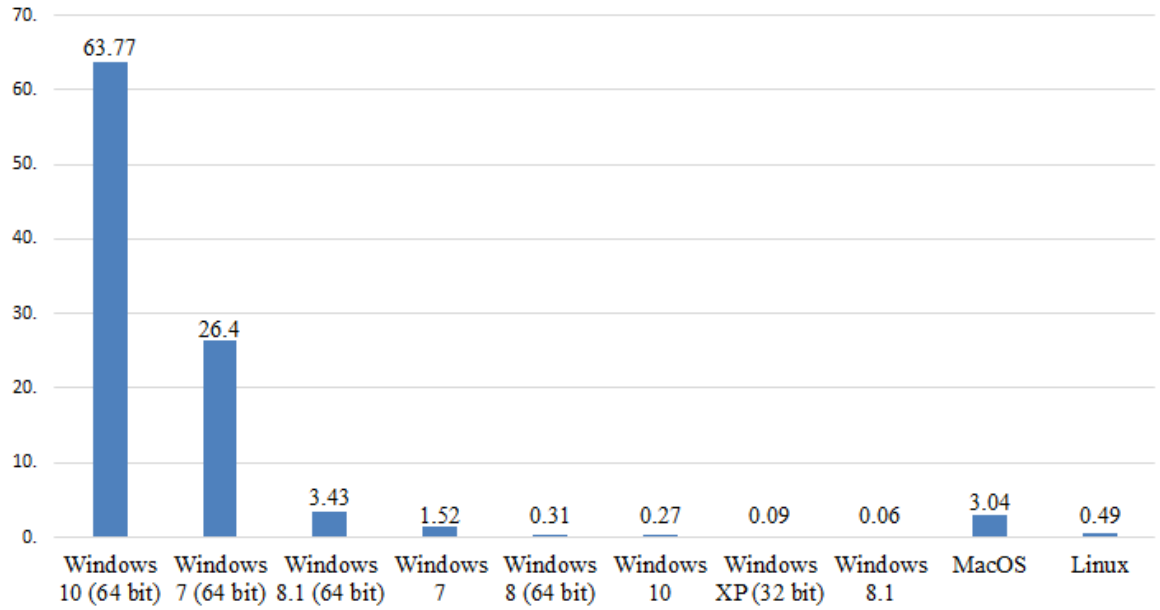
Jednou z nejnovějších platforem, která se pomalu dostává do povědomí lidí, je Virtuální Realita, zkráceně VR. Virtuální realita využívá počítačové technologie k vytvoření simulovaného prostředí. Na rozdíl od tradičního uživatelského prostředí jsou uživatelé umístěni přímo do grafické simulace. Namísto prohlížení obrazovky před nimi mohou interagovat přímo s 3D prostředím. Simulováním smyslů, jako je vidění, sluch, dotek nebo zápach, se počítač promění na bránu do uměle vytvořeného světa. Limitem pro téměř reálné zkušenosti Virtuální Reality je dostupnost podporovaných aplikací a levný hardware pro výpočetní výkon. Často se zaměňuje pojem virtuální a rozšířené reality (AR). Rozšířená

realita využívá senzory a algoritmy k určení polohy a orientace kamery. AR pak vykresluje 3D objekty do reálného světa. Ve virtuální realitě využívá počítač podobné senzory a výpočty, ale místo umístění skutečné kamery ve fyzickém prostředí, jsou oči uživatele umístěny v simulovaném prostředí. Namísto vytváření virtuálních objektů v reálné scéně, vytváří Virtuální Realita přesvědčivý, interaktivní svět. (Williams, 2015), (Bardi, 2019)

6.4 Osobní počítače

Osobní počítače jsou jedním z nejpraktičtějších typů herních platforem, které jsou dnes k dispozici. Mohou to být stolní počítače nebo notebooky se speciální hardwarovou výbavou, která napomáhá spustit hry, náročnější na výpočetní výkon. Ať už se jedná o notebook nebo stolní počítač, obě tyto možnosti musí mít dostatečně výkonný procesor a grafickou kartu společně s hlavní pamětí. Procesor neboli CPU se stará o věci, jako je detekce kolizí, pohyb objektů nebo transformování objektů. Grafický čip (GPU) se na druhou stranu stará o veškerou vizuální stránku hry, jako je generování stínů, vykreslování hlavní scény nebo vykreslování částic při explozích. Zároveň není možné, mít výkonnou pouze jednu z těchto procesních jednotek, jelikož by hra běžela na nejvyšší možnosti nejslabšího článku a zbrzďovala tu druhou. Výhodou stolních počítačů od herním konzolím je ta, že jednotlivé komponenty lze snadno vyměnit za nové. Výměna je limitována pouze podporou nových komponentů na základové desce. Často se jedná o typ patice procesoru nebo nové generaci pamětí a disků. Dalším důležitým faktorem, který ovlivňuje hratelnost videoher je počítačový operační systém. Když budeme vycházet ze statistiky pořízené v lednu 2019 distributorskou platformou Steam (Obrázek 5), na první pohled uvidíme jasného vítěze v této kategorii. Kdybychom sečetly všechny verze operačního systému Windows dostaneme se na necelých 96% všech využívaných operačních systémů, z čehož 63,77% tvoří Windows 10 (64 bit) a 26,4% Windows 7 (64 bit). (Williams, 2015), ("Types of Video Game Platforms", 2016), (Statista, 2019)

Obrázek 5: Využití Operačních systémů, Steam (2019)



(Statista, 2019)

7 Grafika

7.1 3D grafika

3D objekt neboli trojrozměrný objekt, je reprezentován ve třech dimenzích a to výškou, šířkou a hloubkou. Tvorba 3D objektu se dá rozdělit do tří fází. Teselace, kde se model vytvoří z jednotlivých objektů pomocí propojených bodů. Geometrie, kde se různými způsoby transformují vytvořené polygony a aplikují se světelné efekty. A v poslední řadě renderování, kde je 3D model vykreslen s velmi jemnými detaily. (Rouse, 2016)

7.2 2D grafika

2D objekt, zkráceně pro dvourozměrný objekt, nemá třetí dimenzi v podobě hloubky. Pokud by se jednalo o obrázek ve 2D, můžeme ho pozorovat pouze z jednoho směru, na rozdíl od 3D objektů, které lze prohlížet z libovolných úhlů. 2D grafika je často vytvářena v aplikacích, které byly vyvinuty na základě tradičních technologií tisku a kreslení. Příklady oborů využívající dvourozměrnou grafiku, mohou být typografie, kartografie, technické kreslení nebo reklama. Úkony prováděné nad dvourozměrnými objekty, mohou být otáčení v jedné rovině, zvětšování nebo transformování. Funkcí, která 2D objektům chybí, je automatické generování světelných efektů. Veškeré stíny ve 2D grafice musí být tedy vytvářeny ručně. (Computerlanguage, 2018), (ComputerHope, 2018)

7.2.1 Vektorová grafika

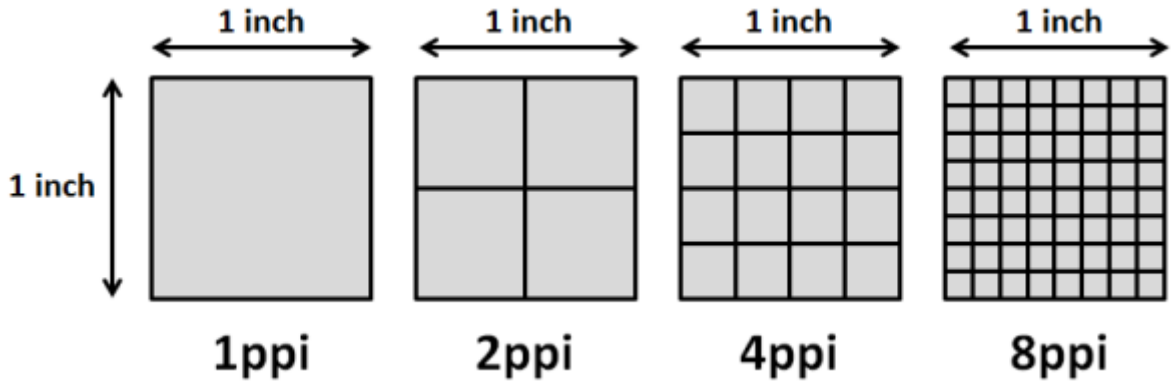
Vektorová grafika se skládá z jednotlivých cest a každá cesta obsahuje matematický popis ve formě vektorů, které definují, jaký má cesta tvar. Dále jsou také popsané barvou, kterou jsou buď ohraničeny nebo vyplněny. Vzhledem k tomu, že je způsob vykreslování obrazu závislý na matematických vzorcích, uchovává si obraz svou velikost nehledě na jeho zvětšení nebo zmenšení. Z toho důvodu je vektorová grafika dobrou volbou pro tvorbu loga a jiných grafických výtvorů náročných na velikosti. Vektorovou grafiku lze vytvářet a upravovat v programech jako jsou Illustrator, CorelDraw nebo InkScape. (PSprint, n.d.)

7.2.2 Rastrová grafika

Rastrové obrázky, také známé jako bitmapové, jsou obrázky složené z individuálních barevných pixelů. Každý barevný pixel se samotně chová pouze jako barevný čtverec, ale

dohromady tvoří celkový detailní obraz. Hlavní výhodou rastrové grafiky je ten, že nám umožňuje upravovat pixel po pixelu. Další výhoda, která z toho vychází, je tvorba barevných přechodů, které nejsou možné vytvořit pomocí vektorové grafiky, jelikož má každý tvar pouze jednu barvu. Kvalita rastru se často určuje podle množství pixelů na jeden palec čtvereční a označuje se jednotkou PPI (Obrázek 6). Aplikujeme-li rastrovou grafiku pro tvorbu videoher, můžeme jednotlivé obrázky nazývat „sprity“. Rastrovou grafiku lze vytvářet a upravovat v programech jako je Photoshop, MS Paint nebo Krita. (PSprint, n.d.)

Obrázek 6: Pixel Per Inch



(Kenlo, 2013)

7.2.2.1 Pixel Art

Pixel Art je typem rastrové grafiky. Rozdílem oproti klasické rastrové grafice je ten, že se nesnaží, aby člověk neviděl jednotlivé pixely. Zde je to přímo žádoucí a z toho důvodu mají výsledné obrázky velice malé rozlišení a zobrazují se ve velkém zvětšení. Jelikož mají lidé vytvářející Pixel Art mnohem méně možností pro vybarvení celého obrazu, tak je významnost každého pixelu mnohem větší než u rastrových obrazů s velkým rozlišením, kde je změna jednoho pixelu zanedbatelná. Pixel Art byl dříve využíván z důvodu nedostatku paměti.

8 Game engine

Game engine je software, který se chová jako platforma, poskytující nástroje pro tvorbu videoher. Spousta lidí často zaměňuje game engine za koncový produkt, jako je Unreal nebo Game Maker, které jsou založené na bázi předplatného, nebo jiného druhu zpeněžení. Pravdou však je, že drtivá většina enginů je vyvinuta velkými společnostmi, jako je Activision Blizzard, EA nebo Ubisoft a jedná se o platformy určené pro samotné firmy, ke kterým mají přístup jen interní zaměstnanci. Většinou se tedy nejedná o veřejný software, ale o prostředí, které společnosti využívají k tvorbě videoher.

Výsledným produktem vytvořeným v nějakém z těchto enginů, nemusí být zcela nutně hrou. Mohou být využity pro tvorbu jakýchkoliv aplikací, jako je software pro virtuální realitu nebo tvorbu simulací. Samozřejmě jsou primárně uzpůsobeny herní tématice a jiné projekty by vyžadovaly spoustu dodatečného kódu, ale pokud chce člověk vytvořit jakoukoliv aplikaci zobrazující grafický výstup, je to zcela možné s většinou veřejných game enginů. Pokud bychom rozebrali game engine do jednotlivých funkcí, tak je jeho základním cílem transformovat data z jednoho formátu do druhého. Data uložená na disku v počítači přeměňuje na vizuální výstupy zobrazující na monitoru a zároveň s nimi umožňuje manipulovat. Využívané soubory nebo balíčky dat se zde nazývají „assets“ a slouží jako vstupy pro game engine. Výtvarníci, kteří tyto vstupy poskytují, jsou zodpovědní za jejich kompatibilitu s enginem se kterým zrovna pracují. Je to z toho důvodu, že poskytovaná data nemusí být vždy v unifikovaných formátech, jako je „PNG“ nebo „JPG“, ale mohou obsahovat dodatečné informace o pozicích objektů, vrstvách nebo vlastnostech vrstvy. (Chernikov, 2018)

8.1 Druhy Game enginů

Pokud budu srovnávat game enginy na základě tvorby začátečníků a nezávislých vývojářů, mohu vycházet z dat poskytujících stránkou itch.io (Tabulka 1). Jedná se o stránku provozující trh pro nezávislé tvůrce, zaměřující se na indie videohry. Je to platforma, která umožňuje komukoli prodat obsah, který vytvořili. Prodejci si zde sami mohou nastavit cenu, za kterou se bude jejich hra prodávat a mohou si i navrhnout vzhled své stránky.

Tabulka 1: Využívané Game enginy na webu Itch.io

Game Engine	Počet projektů	Počet projektů za týden	Informace o Enginu
Unity	27500	210	Multi-platformový herní engine s podporou 3D a 2D grafiky
Construct	6715	24	Nástroj pro tvorbu převážně 2D her pro HTML5
GameMaker: Studio	6012	25	Multi-platformový vývojový software zaměřený na 2D grafiku
Twine	3444	21	Open-source nástroj pro vyprávění interaktivních příběhů
RPG Maker	2090	21	Desktopová aplikace zaměřená na tvorbu RPG her
Bitsy	1890	9	Jednoduchý editor pro tvorbu malých her
Unreal Engine	1626	15	Sada nástrojů pro vytváření her zaměřených na realistickou grafiku
PICO-8	1604	11	Software pro tvorbu malých her a počítačových programů
Godot	1518	7	Multi-platformový open-source software pro tvorbu 2D a 3D grafiky
Ren'Py	1109	28	Software pro tvorbu vizuálních novel

("indie game hosting marketplace", 2019)

8.2 Unity

Unity je herní engine, který poskytuje spoustu užitečných nástrojů pro tvorbu videoher. Unity podporuje tvorbu 2D i 3D projektů. Obsahuje vlastní editor pro vytváření a upravování grafických nebo logických prvků hry. Podporuje spoustu funkcí, jako je „Prefab“, jenž duplikuje nastavení často používaných objektů a zrychluje tím pracovní postup nebo umělou inteligenci pro pohyb nepřátel prostředím. Seznam výčtu komponent naleznete v této tabulce (Tabulka 2). Výstupný produkt je možné vyvíjet pro více než 25 platforem zahrnující telefony, desktopy, konzole, televize, virtuální realitu a další.

Tabulka 2: Komponenty poskytované enginem Unity

Název komponent	Typ komponent	Popis
Rigidbody	Třída v Unity	Přiřazením komponenty Rigidbody k objektu, předává ovládání svého pohybu pod fyzickou správu Unity.
Collider	Třída v Unity	Přiřazuje hranice kolizí danému objektu. Je potřeba Rigidbody aby správně fungoval.
GetComponent	Funkce	Vrací komponent předaného typu a pokud není přiřazen, vrací hodnotu null.

Input	Třída v Unity	Rozhraní pro správu veškerých vstupů ze strany hráče.
GetAxis	Funkce	Vrací hodnotu virtuální osy podle předané hodnoty string.
Vector2	Struktura v Unity	Reprezentuje pozice, vektory a body objektů.
velocity	Hodnota	Udává změnu rychlosti u vektorů.
GetKeyDown	Funkce	Vrací hodnotu true během snímku, kdy hráč stiskl klávesu, předanou v parametru.
Update	Funkce	V každém snímku vykonává kód ve svém bloku.
Start	Funkce	Je volána na začátku snímku, při uvedení skriptu do provozu, ještě před funkcí Update.
KeyCode	Enumerace	Vrací hodnotu stisknuté fyzické klávesy na klávesnici
Time	Třída v Unity	Rozhraní pro získání informace o čase v Unity
deltaTime	Hodnota	Hodnota obsahující čas mezi současným a předchozím snímkem
SceneManager	Třída v Unity	Spravuje jednotlivé scény v průběhu hry
LoadScene	Funkce	Načte scénu specifikovanou jménem nebo indexem
Is Trigger	Hodnota	Určuje, zda objekt posílá OnTriggerEnter, Exit, Stay
OnTriggerEnter2D	Funkce	Spouští se v případě kolize s jiným colliderem
transform	Třída v Unity	Spravuje pozici, rotaci a velikost objektu
Translate	Funkce	Pohybuje s transformem předaným směrem a vzdáleností
eulerAngles	Funkce	Představuje rotaci pomocí X a Y ve stupních
Physics2D	Třída v Unity	Globální rozhraní pro nastavení 2D fyziky
Raycast	Funkce	Vyšle neviditelný paprsek daným směrem a délkou

Random	Třída v Unity	Třída pro generování náhodných dat
Range	Funkce	Vrací náhodné číslo mezi předaným hodnotami
OverlapCircle	Funkce	Kontroluje, zda spadá collider do určeného kruhu
Coroutine	Třída v Unity	Funkce, která dokáže pozastavit chod kódu, dokud se nevykoná yield instrukce
StartCoroutine	Funkce	Zavolá danou Coroutine
WaitForSeconds	Třída v Unity	Pozastaví průchod Coroutine na specifikovanou dobu
FindGameObject	Funkce	Vrací aktivní herní objekt, odpovídající předanému jménu
MoveTowards	Funkce	Vypočítává pozici mezi dvěma body

("Unity - Scripting API", 2018)

9 Tvorba aplikace – praktická část

9.1 Nápad hry

Hra je na styl Metroidvania a společně spadá pod žánr Platformer. Z toho vyplývá, že je zde hlavní charakter, se kterým může hráč pohybovat. V našem případě se jedná o void goblina s malým tělem a velkou hlavou. Gravitace je směřovaná směrem dolů a hráč koliduje se zemí v podobě plošin. Hráč tedy stojí na plošinách a může se pohybovat do stran. Pohybem se postupně vyrovnává i pohled na hráče, aby se nestalo, že nebude na obrazovce vidět. Dalším pohybem, který může hráč vykonat je skok. Výška skoku odpovídá pohybu hráče po obrácené parabole a při současném pohybu do strany ji přesně opisuje. Při skoku na vertikální hranu plošiny se považuje, že se hráč drží a pomalu klouže dolů. Při skluzu se může hráč svévolně pustit, aby padal normální rychlostí. Další vlastností hráče je rychlý posun do strany, který trvá pouze zlomek času a často se označuje, jako „dash“. Skok i „dash“ mají omezenou iteraci používání a jsou dále ovlivněny prostředím. Poslední dovedností hráče je útok, v podobě švihu zbraní.

Přesuneme-li se k prostředí, skládá se z primárních plošin, nebo jakýchkoliv jiných objektů poskládaných ze základních plošin. Některé plošiny se mohou pohybovat do stran. Dále jsou zde překážky ve formě ostnů a pohybujících se bloků. Při dotyku hráče s překážkou se hráči ubírají životy. Dále jsou v prostředí umístěny místa, kde se hráč zrodí po spuštění levelu nebo při klesnutí hráčovo životů na nulu. Tyto místa se nazývají „spawn point“. Podobným místem je „check point“, který zrodí hráče pouze po klesnutí životů na nulu a je získán po dosažení určitého posunu ve hře, aby hráč nemusel opakovat již pokořené překážky. Dále jsou ve hře objekty, které se dají „sbírat“. Jedná se o ozubené kolo, bonusový skok a bonusový dash. Při sebrání ozubeného kola se hráčův indikátor bodů zvýší o jedna. Bonusový skok přidá dočasně hráči možnost použít jeden skok navíc. Bonusový dash se chová obdobně a přidává hráči možnost použít jeden dash navíc. V prostředí se také nachází teleport, který přesune hráče do jiné lokace nebo na jiné souřadnice.

Přesuneme-li se k nepřátelům, jsou zde celkem tři druhy. Prvním nepřítelem je pistolník, který se posouvá horizontálním směrem a mění směr pouze tehdy, když narazí na zeď nebo okraj platformy. Druhým nepřítelem je sliz. Je ovlivňován gravitací a pohybuje se pomocí skoků horizontálním směrem opisující obrácenou parabolu. Směr skoku závisí na poloze hráče. Třetím nepřítelem je sršeň, který se pohybuje vzduchem a pomalu se přibližuje

ke hráči. Při doteku hráče a nepřátel, jsou hráčovi ubrány životy. Při doteku hráčova útoku a nepřátel, jsou nepříteli ubrány životy.

Cílem hráče je postupovat předem vytvořeným prostředím a sbírat ozubená kola. Současně se musí vyhýbat přednastaveným nástrahám a nepřátelům, aby neklesl svým počtem životů na nulu. Při dokončení všech úrovní hráč vyhrává a hra končí.

9.2 Vývojové prostředí

Jakožto vývojový engine jsem si zvolil Unity, jelikož je intuitivní na ovládání, průběžně dostává nové verze odstraňující chyby a obsahuje rozsáhlou dokumentaci, která je napsaná tak, že jí porozumí i začátečníci v programování. Další velkou výhodou je množství lidí, kteří s Unity pracují. Čím více lidí se věnuje právě tomuto enginu, tím více bude existovat návodů. Samotné Unity spravuje svůj YouTube kanál, kde zveřejňuje návody na často řešené problematiky. Hodně nápomocní jsou i lidé na fórech, zaměřující se na problematiku Unity.

Jakožto programovací jazyk jsem si zvolil „C#“, což je univerzální objektově orientovaný programovací jazyk, jehož cílem je zvýšení produktivity programátorů. Za tímto účelem vyvažuje jednoduchost, expresivitu a výkon. Hlavním architektem jazyka je od jeho první verze Anders Hejlsberg tvůrce Turbo Pascalu a architekt Delphi. Jazyk C# je platformově neutrální, ale byl navrhnut, aby dobře pracoval na rozhraní Microsoft .NET Framework. C# obsahuje bohatou implementaci objektové orientace, která zahrnuje zapouzdření, dědičnost a polymorfismus. Zapouzdření se dá vysvětlit, jako vytvoření hranic objektu, které odděluje jeho vnější veřejné chování od jeho vnitřního chování. Výhodami objektově orientované perspektivy C# je unifikovaný typový systém, rozhraní, vlastnosti funkcí, metody a „eventy“. (O'Reilly, 2012)

Jako vývojové prostředí pro psaní kódu jsem si zvolil Visual Studio verzi 2017. Důvody, proč jsem si ho zvolil jsou jednoduché. Už jsem zvyklý na jeho používání a vyhovují mi „snippety“, neboli zkratky, které jsou zde implicitně nastaveny. Avšak tím nejdůležitějším důvodem je možnost nastavení „IntelliSense“ neboli našeptávače, který podporuje funkce v Unity a pomáhá dokončovat rozepsaný kód.

9.3 Kód hlavního charakteru

9.3.1 Objekt hlavního charakteru

Chceme-li rozpohybovat hlavní charakter, potřebujeme nejdříve vytvořit prázdný objekt. To lze uskutečnit v okně hierarchie vrstev, pomocí nabídky vytvořit nový prázdný objekt. Pro vizualizaci nám zatím postačí jakýkoliv obrázek, který objektu přiřadíme, abychom viděli, kde se objekt pohybuje. Toho můžeme dosáhnout dvěma způsoby. Buď můžeme pomocí funkce „drag and drop“ přenést z našeho pracovního adresáře požadovaný obrázek na cílový objekt, nebo můžeme v inspektoru přidat komponent „Sprite renderer“, ve kterém následně vybereme náš obrázek pomocí výběrové funkce. Dále přidáme našemu objektu tag „Player“ pro pozdější využití. Na náš nově vytvořený objekt zatím nepůsobí žádná síla a kdybychom dali spustit simulaci hry, tak se nic nestane. To zajistíme přidáním komponentu Rigidbody 2D, s tím že zaškrtneme typ těla jako dynamický. Tím zajistíme samovolný pád objektu, jelikož na něj začíná působit gravitace, kterou můžeme ovlivnit pomocí velikosti „Gravity scale“, jež se také nachází v nabídce Rigidbody. Tato možnost ovlivňuje, jak moc působí gravitace na daný objekt.

Chceme-li pokračovat, musíme vytvořit platformu, na které bude moct náš objekt přistát. Vytvoříme tedy další prázdný objekt, ke kterému přiřadíme obrázek obdélníkového nebo čtvercového tvaru. Platformu umístíme pod naši hlavní postavu pomocí „move tool“ a přiřadíme jí komponent „Box Collider 2D“. Obdobně musíme přiřadit collider i naší hlavní postavě, jelikož by v tento moment pouze propadla plošinou. Máme na výběr z několika druhů colliderů, jako je kruhový, čtvercový nebo ve tvaru kapsle. Pro složitější tvary si můžeme vytvořit i vlastní pomocí polygonového collideru. Collidery se chovají jako hranice objektu, které reagují na kolize s jinými collidery. Můžeme nastavit jejich velikost a umístění pomocí funkce „Edit collider“, nebo přepisováním hodnot x a y. Máme-li přiřazené kolize u hlavní postavy i plošiny, můžeme otestovat, zdali sebou neprochází a vše funguje, jak má.

9.3.2 Chůze

Přesuneme-li se k nastavení chůze, musíme vytvořit první C# skript. Vytvoříme ho přidáním nového komponentu v podobě „New script“, který následně pojmenujeme. Každý nový skript v Unity obsahuje dvě předpřipravené funkce, a to Start a Update. Funkce Start se provádí při spuštění skriptu a chová se potažmo jako konstruktor, kdežto Update se provádí jednou během každého snímku. Abychom mohli objektem pohybovat, musíme vytvořit

proměnnou typu Rigidbody2D, která bude přistupovat k našemu přiřazenému Rigidbody. Získáme ho pomocí GetComponent<Rigidbody2D>. Dále budeme potřebovat proměnnou pro rychlost a další číselnou proměnnou float, ke které přiřadíme Input.GetAxis(„Horizontal“), jenž vrací číslo od -1 do 1 podle stisku šipek na klávesnici a přiřadíme jí název moveInput. To nám poslouží jako směr pohybu pro funkci Moove, vypadající takto (Obrázek 7).

Obrázek 7: Funkce pro pohyb hlavního charakteru

```
protected override void Moove()
{
    rb.velocity = new Vector2(moveInput * speed, rb.velocity.y);
}
```

(Zdroj: Autor, 2019)

Měníme zde pouze rychlost naší proměnné Rigidbody a přiřazujeme jí nový vektor, kde za x dosazujeme směr krát rychlost a za y necháváme vždy výchozí ypsilonovou hodnotu našeho Rigidbody. Funkce Moove je pak volaná z funkce Update.

9.3.3 Skok

Dalším pohybem na pořadí je skok. K němu potřebujeme další číselnou proměnnou, která bude uchovávat sílu skoku. Poté stačí využít Vector2.up, jako vertikální směr. Vynásobíme-li ho naší silou skoku, vzniká funkce pro skok (Obrázek 8). Tu obdobně jako chůzi, voláme z funkce Update za podmínky, že stiskneme písmeno W, čeho dosáhneme pomocí Input.GetKeyDown(KeyCode.W).

Obrázek 8: Funkce pro skok hlavního charakteru

```
void Jump()
{
    rb.velocity = Vector2.up * jumpForce;
}
```

(Zdroj: Autor, 2019)

9.3.4 Dash

Pro funkci dash budeme potřebovat proměnnou facingRight s informací, jakým směrem je hráč otočený, jestli doprava nebo doleva. Vytvoříme si funkci, která bude otáčet tuto proměnnou při změně směru. Otočení můžeme docílit přiřazením negace této proměnné a detekci toho, jestli jsme změnili směr, můžeme zjistit za pomocí moveInput využité v pohybové funkci. Pak využijme podmínky, která zjistí, jestli se naše facingRight proměnná

liší od směru zjištěného v moveInput. Pokud se liší, tak zavoláme naší otáčecí funkci (Obrázek 9).

Obrázek 9: Otáčení hlavního charakteru

```
if (facingRight == false && moveInput > 0)
{
    Flip();
}
else if (facingRight == true && moveInput < 0)
{
    Flip();
}
```

(Zdroj: Autor, 2019)

Teď tedy víme, jakým směrem je hráč otočený a můžeme začít se samotným dashem. Budeme potřebovat číselnou hodnotu startDashTime, která nám bude určovat, jak dlouho aplikovat sílu na objekt, proměnnou dashTime zobrazující uplynutý čas dashování a proměnnou dashSpeed uchovávající rychlost při dashi. Teď nám stačí nastavit podmínku, která se bude ptát, zdali je dashTime menší nebo rovno nule. Pokud ano přiřadíme hodnotě dasTime hodnotu startDashTime. Pokud ne tak začneme odečítat od dashTime hodnotu Time.deltaTime a v ten stejný moment začneme aplikovat sílu na naše Rigidbody podle toho, jakým směrem je objekt otočený (Obrázek 10). Tím docílíme tak, že se bude proměnná dashTime zmenšovat a dokud bude větší než nula, tak bude na objekt aplikována rychlost.

Obrázek 10: Dash hlavního charakteru

```
if (facingRight == true)
{
    rb.velocity = Vector2.right * dashSpeed;
}
else if (facingRight == false)
{
    rb.velocity = Vector2.left * dashSpeed;
}
```

(Zdroj: Autor, 2019)

Celou tuto funkci budeme volat pouze tehdy když hráč stiskne tlačítko shift, čeho dosáhneme pomocí Input.GetKeyDown(KeyCode.LeftShift), obdobně jako u skoku.

9.3.5 Ubírání životů

Základní nápad ubírání životů je takový, že pokud hráči klesne počet životů pod nulu, tak umírá a hra se resetuje na začátek. Musíme tedy nejdříve definovat co se stane, pokud hráč

umře pomocí funkce Die. Zde akorát znovu načteme scénu, ve které se hráč nachází. Toho docílíme pomocí SceneManager.LoadScene kterému předáme název scény. Nesmíme však zapomenout definovat, že využíváme UnityEngine.SceneManagement.

Když máme hotovou funkci Die, můžeme začít s ubíráním životů. Budeme potřebovat proměnou skladující počet životů hráče. Dále vytvoříme funkci TakeDamage (Obrázek 11) s číselným parametrem damage, kterou budeme volat při styku s nepřítelem. Jelikož se může stát, že hráč dostane větší poškození, než je jeho počet životů, využijeme funkci Maxima, porovnávající rozdíl s nulou. Poté stačí porovnat počet zbývajících životů s nulou a pokud se rovnají, tak zavoláme funkci Die.

Obrázek 11: Funkce pro ubrání životů

```
public override void TakeDamage(int damage)
{
    health = Mathf.Max(0, health - damage);
    if (health == 0)
    {
        Die();
    }
}
```

(Zdroj: Autor, 2019)

9.4 Kód nepřátel

9.4.1 Poškození

Všichni nepřátelé mají funkci udělení poškození při kolizi s hráčem. Oba objekty hráč i nepřítel musí mít přiřazený Collider a nepřítel ho musí mít nastavený na „Is Trigger“. Dále musíme vytvořit proměnnou, která bude uchovávat výši poškození. Teď stačí vytvořit funkci OnTriggerEnter2D (Obrázek 12), která má v parametru objekt se kterým došlo ke kolizi. U tohoto objektu dostaneme pomocí GetComponent script hlavního charakteru, ve kterém následně vyvoláme naši před vytvořenou funkci TakeDamage a předáme jí v parametru výši poškození.

Obrázek 12: Vyvolání funkce poškození.

```
public virtual void OnTriggerEnter2D(Collider2D hit)
{
    hit.GetComponent<Player>().TakeDamage(dmg);
}
```

(Zdroj: Autor, 2019)

9.4.2 Pistolník

Pohyb pistolníka (Obrázek 13) je poměrně jednoduchý. Pohybuje se horizontálně do jedné strany, dokud nenarazí na konec plošiny nebo zeď. Budeme tedy potřebovat proměnnou pro rychlost, kterou použijeme u transform.Translate. Dále využijeme Physics2D.Raycast, který vytvoří neviditelný paprsek kontrolující kolizi s plošinou. Paprsek potřebuje počáteční bod, proto vytvoříme prázdný objekt a přiřadíme ho proměnné Transform groundDetection. Dále vytvoříme proměnnou distance pro délku paprsku. Teď již stačí zkontrolovat, zda došlo ke kolizi s paprskem a v kladném případě můžeme měnit rotaci objektu. K tomu budeme potřebovat proměnnou movingRight obdobně, jako u hlavního charakteru. Dále využijeme podmínky, zdali se objekt pohybuje jedním směrem a pokud ano, tak ho převrátíme za pomocí transform.eulerAngles.

Obrázek 13: Pohyb pistolníka

```
transform.Translate(Vector2.right * speed * Time.deltaTime);
RaycastHit2D groundInfo = Physics2D.Raycast(groundDetection.position, Vector2.down, distance);
if (groundInfo.collider == false)
{
    if (movingRight == true)
    {
        transform.eulerAngles = new Vector3(0, -180, 0);
        movingRight = false;
    }
    else
    {
        transform.eulerAngles = new Vector3(0, 0, 0);
        movingRight = true;
    }
}
```

(Zdroj: Autor, 2019)

9.4.3 Sliz

Pohyb slizu je malinko složitější, musíme u něj na začátku najít objekt hlavního charakteru, jelikož se pohybuje v závislosti, na jaké straně od něj jsme. Toho docílíme pomocí FindGameObjectWithTag. Dále budeme potřebovat proměnnou pro sílu skoku, vzdálenost skoku, rádius testování země, masku whatIsGround a bool isGrounded. Obdobně jako u pistolníka vytvoříme prázdný objekt a přiřadíme ho do proměnné groundCheck. Tento objekt posuneme na spodní část slizu a s jeho pomocí vytvoříme OverlapCircle společně s rádiusem testování a maskou whatIsGround. Vytvořenou bool hodnotu přiřadíme hodnotě isGrounded. Dále potřebujeme porovnat souřadnici X hráče se souřadnicí X slizu (Obrázek 14).

Obrázek 14: Přiřazení směru a vzdálenosti skoku

```
if (player.position.x < rb.position.x)
{
    jumpDistance = Random.Range(-60f, -20f);
}
else
{
    jumpDistance = Random.Range(60f, 20f);
}
```

(Zdroj: Autor, 2019)

Tím zjistíme, jestli je sliz napravo nebo nalevo od hráče a můžeme přiřadit kladnou nebo zápornou sílu skoku. Také potřebujeme zjistit, zdali se objekt zrovna pohybuje doleva nebo doprava, čehož dosáhneme porovnáním rychlosti objektu na ose X vůči nule.
Teď když máme veškeré potřebné proměnné můžeme začít se samotným skokem. Vytvoříme coroutine (Obrázek 15), ve které počkáme náhodnou dobu za pomoci WaitForSeconds a poté pokud je sliz na zemi, tak přidáme Rigidbody vektor obsahující vzdálenost skoku a sílu skoku. Na konci coroutine ji zavoláme znovu pomocí StartCoroutine. Teď už nám jen zbývá prvotně zavolat coroutine ve funkci start a jsme hotovi se slizem.

Obrázek 15: Skok slizu

```
IEnumerator slizJump()
{
    yield return new WaitForSeconds(Random.Range(1, 2.5f));
    if (groundCheck)
    {
        rb.AddForce(new Vector2(jumpDistance, jumpForce));
    }
    StartCoroutine(slizJump());
}
```

(Zdroj: Autor, 2019)

9.4.4 Sršeň

Pohyb sršně je jednoduchý. Rigidbody je nastavené na typ Kinematic, tudíž není ovlivněné gravitací. Obdobně jako u slizu najdeme objekt hráče pomocí FindGameObjectWithTag. Jediné číselné proměnné, které budeme potřebovat, jsou rychlost vzdálenost a zastavovací vzdálenost. Vzdálenosti přiřadíme Vector2.Distance s pozicí Rigidbody a pozicí hráče. Dále už stačí jen zjistit, zdali je vzdálenost menší než zastavovací vzdálenost a začneme měnit pozici sršně pomocí funkce MoveTowards, ve které využijeme

pozici obou objektu a rychlost násobenou časem. Ukázku funkce pohybu sršně můžete vidět zde. (Obrázek 16)

Obrázek 16: Funkce pro pohyb sršně

```
public void srsenMoove()
{
    distance = Vector2.Distance(rb.position, player.position);
    if (distance > stoppingDistance)
    {
        transform.position = Vector2.MoveTowards(transform.position, player.position, speed * Time.deltaTime);
    }
}
```

(Zdroj: Autor, 2019)

10 Vizuální stránka

Jako styl herní grafiky jsem si zvolil Pixel Art, a to z toho důvodu, že je jednodušší na vytvoření a není nutné používat žádný přídavný kreslící hardware, jako jsou kreslící tablety. Člověk si tedy vystačí pouze s myší.

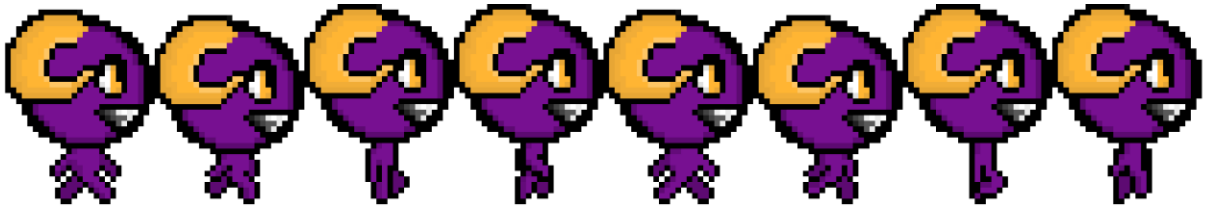
Další otázkou je, jaký použít software pro tvorbu 2D grafiky. Hledal jsem software, který by byl svou funkcionalitou podobný softwaru Adobe Photoshop, jelikož jsem s ním byl již seznámen v předchozím studiu. Ale těmi nejdůležitějšími kritérii byla volná dostupnost softwaru a dobrá podpora pixelové kresby. Nakonec jsem se uchýlil k softwaru s názvem Krita. Krita je Open source čili se za ní neplatí a podpora pixelových nástrojů byl s výjimkami více než dostatečná. Intuitivností vynikala nad ostatními programy a co se týče ovládání, tak se veškeré zkratky daly nastavit podle uvážení. A zkratky nejsou zcela tím jediným, co se zde dá nastavit. Veškeré rozhraní je interaktivní a jednotlivé doky s nástroji se dají přesouvat, zmenšovat, přidávat i odebírat. Tento software podporuje i tvorbu animace, jejíž ovladatelnost je někdy dost matoucí, ale její přítomnost je příjemným bonusem. Člověk si zde může před vytvořit i své štětce, které se dají rozčlenit do skupin pro lepší přístupnost.

10.1 Grafika hlavního charakteru

10.1.1 Pohyb

Grafika hlavního charakteru (Obrázek 17) je inspirována hrou The Binding of Isaac, kde mají postavičky malé tělo a velkou hlavu. Jedná se o goblina fialové barvy, který má zahnuté zlaté rohy. Velikost postavy je 40 na 40 pixelů. Animace cyklu chůze obsahuje osm snímků. Jedná se o první animaci, kterou jsem kdy vytvořil a myslím si, že koncový výsledek animace je natolik dostatečný, aby si hráč nedostatků během hraní nevšímal.

Obrázek 17: Cyklus chůze hlavního charakteru



(Zdroj: Autor, 2019)

10.1.2 Skok

O skok postavy se stará animace o dvou snímcích. Počet snímků je dostatečný, jelikož stačí snímek pro odraz postavy, následující pózou letu, která by měla být dynamická, aby si hráč užíval každý skok. Průběh skoku je pak řešený kódem, který posouvá postavu v prostoru. Není proto nutné animovat samotný pohyb během letu, což ušetří spoustu času. Konec skoku, kdy charakter padá směrem dolů, je stejný, jako když padá z jakéhokoliv jiného místa. Pád je tedy řešený dedikovaným snímkem, který je následován snímkem pro dopad, pokud se hráč dotkne nějaké plošiny.

10.1.3 Útok

Dále má hráč možnost útoku. Útok je ve formě švihu energie zlaté barvy (Obrázek 18). Švih je řešen animací obsahující osm snímků. Animace útoku nesmí být moc pomalá, aby hráči nepřišlo, že ho zdržuje při hraní. Postoj hráče během útoku je zabudovaný do animace švihu a je recyklovaný ze třech snímků animace chůze.

Obrázek 18: Cyklus útoku hlavního charakteru

(Zdroj: Autor, 2019)

10.1.4 Dash

Dash hráče je vyřešen jedním dedikovaným snímkem, který je vyvolaný při rapidním posunu dopředu. Vzhledově se jedná o končetiny rádoby plandající za tělem. Dokončením této animace máme hotovou vizuální stránku hráče a můžeme se pustit na grafiku prostředí.

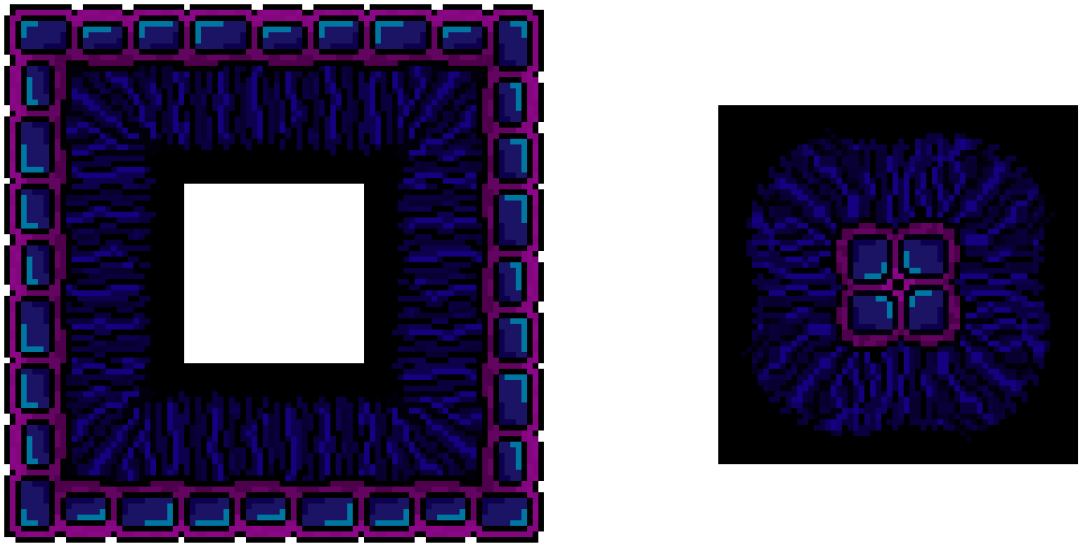
10.2 Grafika prostředí

10.2.1 Plošiny

Do prostředí spadají hlavně plošiny (Obrázek 19) a překážky. Veškeré plošiny i překážky jsou složeny ze spritů čtvercového tvaru. Aby se daly opakovaně využít za sebou, nesmí zde být vidět přechod mezi čtverci. Pro hlavní styl plošin jsem zvolil kamenný okraj

postupně přecházející do černoty. Velikost jednoho čtverce je 32 na 32 pixelů. Počet spritů se odvíjí od požadovaných tvarů, které z nich chceme vytvořit. Já jsem zvolil dvanáct pro jeden styl plošin. Jedná se o horní, spodní, levý a pravý vnější okraj a obdobně všechny čtyři vnější rohy. Zbývající čtyři jsou vnitřní rohy.

Obrázek 19: Sprity pro skládání platforem

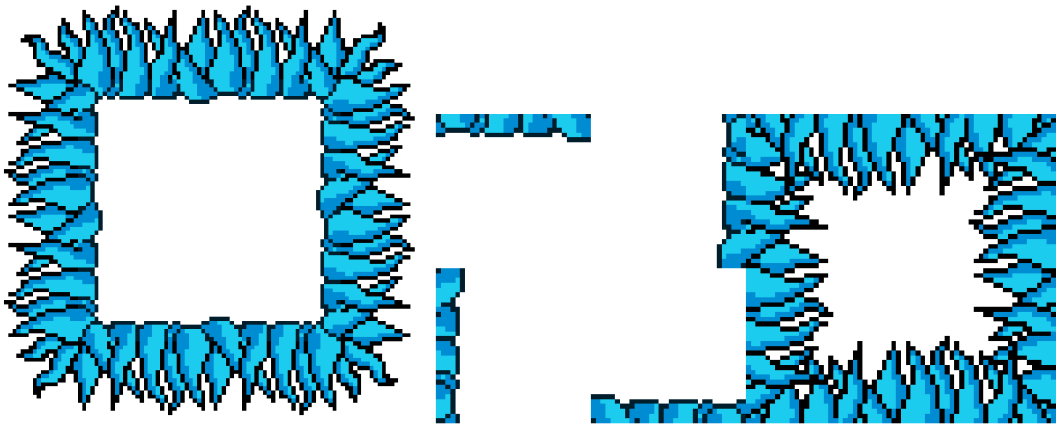


(Zdroj: Autor, 2019)

10.2.2 Překážky

Do překážek spadají dva typy, a to ledové ostny a pohybující se kvádry. Co se týče ledových ostnů (Obrázek 20), vycházel jsem z inspirace rampouchů a krápníků. Obdobně jako u plošin jsem se snažil, aby nebyl poznat rozdíl v přechodu mezi jednotlivými ostny a celkový obraz tak vypadal nenarušeně. Šířka ledových ostnů je stejná čili 32 pixelů, ale výška dosahuje něco málo přes polovinu čtverce. Co se týče počtu spritů s ostny, je zde více než u plošin, a to osm ostrých vnějších ostnů, čtyři ostré vnitřní ostny, osm tupých vnějších okrajů a čtyři vnitřní tupé okraje. Dohromady tedy 24 spritů. Čím více spritů člověk vyrobí tím lépe pak prostředí vypadá.

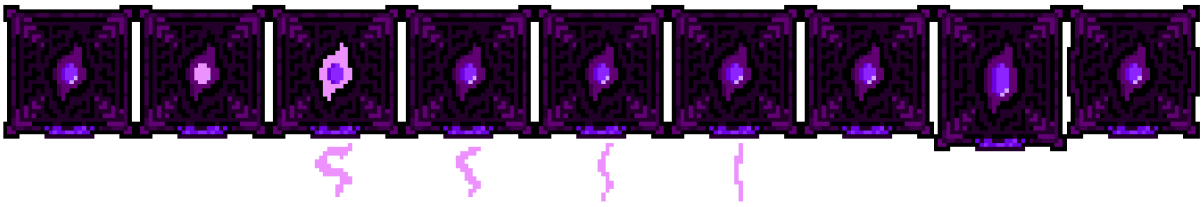
Obrázek 20: Sprity pro skládání ostnů



(Zdroj: Autor, 2019)

U pohyblivých kvádrů (Obrázek 21), jsem vyřešil vzhled pomocí animace, jelikož je potřeba upozornit hráče, než dojde k posunu. Překážka se posunuje rapidní rychlostí jedním směrem, dokud nenarazí na povrch, kde se zastaví a pomalu se posune zpět. Velikost překážky je obdobně, jako u plošin 32 na 32 pixelů. Animace je statická, tudíž se nemění pozice překážky, ale pouze vlastnosti kvádru. Animace začíná rozsvícením krystalu uprostřed bloku, následující rozsvícením prostoru pod překážkou. Po obou indikacích následují snímky s prodloužením kvádru, které reprezentují let vzduchem. Předposlední snímek naznačuje zmenšeným kvádrem náraz do povrchu a posledním snímkem se vrací výchozí vzhled kvádru do své původní pozice.

Obrázek 21: Cyklus pohybující se překážky



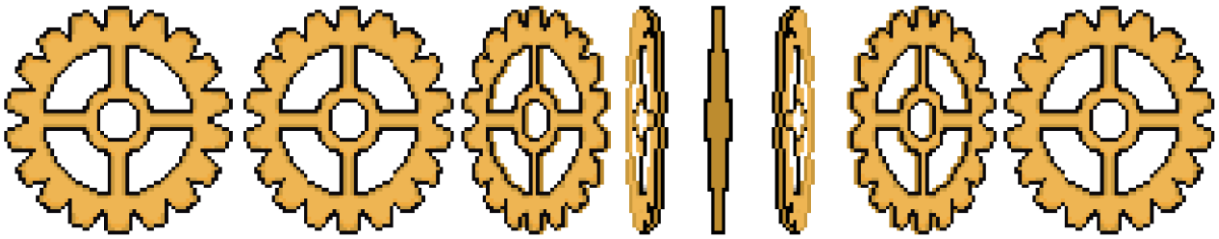
(Zdroj: Autor, 2019)

10.3 Sbírané předměty

10.3.1 Ozubené kolo

Ozubené kolo je indikátorem skóre. Jedná se o otáčející se objekt a jeho vzhled je vyřešen animací o osmi snímcích (Obrázek 22). Tvorba jednotlivých snímků animace je mnohem náročnější než rotování objektu pomocí kódu. Kolo má zlatou barvu a reprezentuje ztracené součástky, které se hlavní charakter snaží posbírat.

Obrázek 22: Ozubené kolo reprezentující skóre



(Zdroj: Autor, 2019)

10.3.2 Bonusový dash

Bonusový dash vzhledově vypadá jako fialová mince (Obrázek 23), která má dva fialové trojúhelníky ve svém středu, reprezentující zrychlení. Animaci tohoto objektu jsem vytvářel pomocí klíčových bodů, přímo v animačním nástroji Unity. Samotný kruh objektu se otáčí kolem své Y osy, obdobně jako u ozubeného kola, ale je to dosaženo jinou technikou. Trojúhelníky se otáčí v opačném směru a evokují rotaci kolem středu svým zmenšováním a zvětšováním. Aby to vypadalo věrohodně mění se během animace jejich pořadí na ose Z.

Obrázek 23: Sprite Bonusového dashe

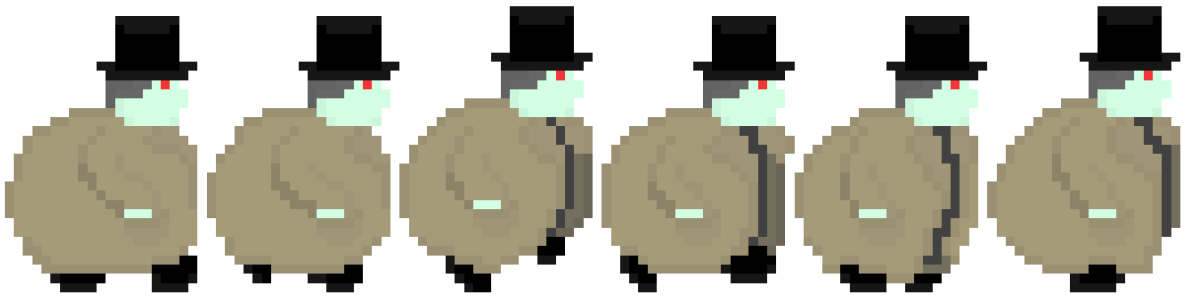
(Zdroj: Autor, 2019)

10.4 Grafika Nepřátel

10.4.1 Pistolník

Pistolník je řešený animací o šesti snímcích (Obrázek 24). Jeho pohyb je inspirovaný kolébáním tučňáků. Pohyb této animace je nejvíce unikátní a trval nejvíce času, i když má o dva snímky méně než animace hráče. Animace je statická a pohyb je řešen kódem z kapitoly kód nepřátel. Velikost jednoho snímku je 32 na 36 pixelů. Nepřítel má představovat posmrtného pochůzkáře, který uvízl v časové smyčce.

Obrázek 24: Cyklus chůze pistolníka



(Zdroj: Autor, 2019)

10.4.2 Sliz

Sliz vypadá jako zvětšená kapka vody ve fialové barvě. Jeho pohyb je řešený animací o devíti snímcích (Obrázek 25). Obdobně jako u ostatních, je jeho animace statická a pohyb je řešen pomocí kódu z kapitoly kód nepřátel. Rozdíl je v tom, že se nepohybuje jen horizontálně ale i vertikálně a opisuje obrácenou parabolu. Velikost jednoho snímku je 16 na 12 pixelů.

Obrázek 25: Cyklus skoku slizu

(Zdroj: Autor, 2019)

10.4.3 Sršeň

Sršeň jako jediný z nepřátel lítá a jeho animace není vytvořena pomocí jednotlivých snímků (Obrázek 26), ale pomocí klíčových bodů v animačním editoru Unity. Jeho tělo je rozděleno na jednotlivé části a každá část je posouvaná podle jednotlivých milníků animace. Animace je tedy řízena kódem generovaným editorem Unity. Pohyb samotného sršně v prostředí je řešen pomocí kódu z kapitoly kód nepřátel.

Obrázek 26: Sprite sršně

(Zdroj: Autor, 2019)

11 Závěr

Na základě této práce jsem se pokusil vytvořit hru na styl platformer a zároveň pro ni vytvořil grafiku. Co se týče kódu, tak jsem použil nástroje Visual Studio 2017 s Unity 2018.3.1 a pro kreslení a animaci jsem využil Kritu 4.1.7. Již dříve jsem se pokusil vytvořit triviální hru pouze s pomocí Visual Studia, ale stačil mi jeden den používání nástrojů poskytovaných enginem Unity a dostal jsem se mnohonásobně dál. Jakožto programovací jazyk jsem využíval C#, a i se základními znalostmi v této problematice jsem byl rychle schopen využívat poskytované funkce a třídy. Díky nespočetnému množství návodů pokrývajících všemožná témata, jsem se rychle zorientoval a postupoval jsem ve vývoji hry mnohem dál, než bych z počátku očekával. Dostal jsem se do fáze, kdy mi návody nestačily a musel jsem dané problematiky řešit vlastní vynalézavostí, maximálně s pomocí příspěvků na fórech. Rozhodně je skvělý pocit, vyřešit nějaký specifický problém, i když o něm na internetu není ani zmínky.

Přesunu-li se ke tvorbě grafiky pro tuto hru, musím uznat, že se jedná o opravdu pomalý proces. Obzvlášť pokud jsem ho provozoval já. Pouštěl jsem se do toho s minimální znalostí kreslení nebo animování. Styl Pixel Art, který jsem si pro tento projekt zvolil je hodně specifický a nejsem si vědom, že by se v dnešní době někde vyučoval. Z toho důvodu existují převážně elektronické návody a jejich množství je celkem limitované. I přes veškeré překážky jsem byl schopen vytvořit více než dostačující sprity a animace pro mé objekty ve hře.

Shrnu-li veškerou práci, tak mohu říct, že engine Unity je mocným nástrojem, jak v rukou pokročilých, tak začátečníků. Je nutná částečná znalost programovacího jazyka, ale i tato překážka se dá překonat studováním návodů a lidsky napsané dokumentace. Komunita lidí věnující se vyvíjením her je přívětivá, co se týče začátečníků a vždy rádi pomohou.

Co se týče vytvořené hry samotné, tak jsem s jejím výsledek více než spokojený. Požadované cíle hry, jako je pohyb, skok a jednotlivé úrovně, jsem rozhodně splnil a mám v plánu na této hře pracovat i do budoucna a zlepšovat prvky, které jsou v současné době na horší úrovni. Zpětná vazba lidí, kteří tuto hru vyzkoušeli, je pozitivní až na drobné připomínky, které se spíše týkají celkového vnímání hry než jakýchkoliv chyb. Příkladem může být špatná orientace při hře a neznalost správného směru. Celý projekt tedy hodnotím pozitivně, a nejen z hlediska praktického výstupu, ale i z hlediska zkušeností, které mi přinesl.

12 Sumary

The main goal of this project is to create an application in form of a PC game. The project focuses on game engine Unity and programming language called „C sharp“. Using game engine is a simpler way of coding games for multiplatform environment. It implements commonly used game functions that take care of basic game objects behaviour. Game of this project is 2D and uses sprites for graphics and animations. Game graphics takes a big part in game creation. In this case technique called Pixel art is used. Genre of this game is a platformer slightly leaning to „Metroidvania“. A player character can move vertically with the option to jump horizontally. Environment contains traps and enemies which the player tries to avoid or eliminate. During the game progress can player reach different bonuses and score points. The game includes multiple levels with different enemies and difficulties.

Key words: application, game engine, Unity, C#, coding, object, 2D, sprites, Pixel art, platformer, Metroidvania

13 Seznam zdrojů

Chris Crawford, Definition of Game [Online]. Retrieved April 01, 2019, from http://www.erasmatazz.com/library/the-journal-of-computer/jcgd-volume-4/my-definition-of-game.html

Alexander Smith, Computer Games in the 1950s [Online]. (2014). Retrieved April 02, 2019, from https://videogamehistorian.wordpress.com/2014/01/22/the-priesthood-at-play-computer-games-in-the-1950s/

Challenge NIMROD [Online]. (2011). Retrieved April 06, 2019, from http://www.goodeveca.net/nimrod/GAME/index.html

Cunningham Andrew, The NES turns 30 [Online]. (2013). Retrieved April 06, 2019, from https://arstechnica.com/gaming/2013/07/time-to-feel-old-inside-the-nes-on-its-30th-birthday/

Warcraft: Orcs & Humans [Online]. (2010). Retrieved April 06, 2019, from https://wowwiki.fandom.com/wiki/Warcraft:_Orcs_%26_Humans

The History of Video Arcade Games [Online]. Retrieved April 06, 2019, from https://www.bmigaming.com/videogamehistory.htm

Vince Matthews, List of Different Types of Video Games [Online]. (2018). Retrieved April 06, 2019, from https://www.idtech.com/blog/different-types-of-video-game-genres

Koji Igarashi, YouTube - Metroid Vania Tale [Online]. (2016). Retrieved April 06, 2019, from https://www.youtube.com/watch?v=gLyjAWYK2Kg

Amanda Greenslade, A glossary of Gaming Terms [Online]. (2006). Retrieved April 06, 2019, from https://web.archive.org/web/20070219082328/http://www.specusphere.com/joomla/index.php?option=com_content&task=view&id=232&Itemid=32

Types of Video Game Platforms [Online]. (2016). Retrieved April 06, 2019, from https://pcdreams.com.sg/different-types-of-video-game-platforms-popular-today/

Georgia Williams, Understand game platform types [Online]. (2015). Retrieved April 06, 2019, from https://georgiawilliams5.wordpress.com/2015/01/12/understand-game-platform-types/

Joe Bardi, What is Virtual Reality [Online]. (2019). Retrieved April 06, 2019, from https://www.marxentlabs.com/what-is-virtual-reality/

Richter Felix, The Most Important Gaming Platforms [Online]. (2019). Retrieved April 06, 2019, from https://www.statista.com/chart/4527/game-developers-platform-preferences/

Tatiana Kir, What is indie game development [Online]. (2016). Retrieved April 06, 2019, from https://vironit.com/what-is-indie-game-development/

Form an Indie Game Development Team [Online]. (2014). Retrieved April 06, 2019, from https://www.nyfa.edu/student-resources/forming-solid-indie-game-development-team/

Yan Chernikov, YouTube - What is a GAME ENGINE [Online]. (2018). Retrieved April 06, 2019, from https://www.youtube.com/watch?v=vtWdgtMo1T4

Statista, Operating system used 2019 [Online]. (2019). Retrieved April 06, 2019, from https://www.statista.com/statistics/265033/proportion-of-operating-systems-used-on-the-online-gaming-platform-steam/

ComputerHope, What is 2-D [Online]. (2018). Retrieved April 06, 2019, from https://www.computerhope.com/jargon/num/2d.htm

Computerlanguage, 2D graphics Definition [Online]. (2018). Retrieved April 06, 2019, from https://www.pcmag.com/encyclopedia/term/36990/2d-graphics

Margaret Rouse, What is 3-D [Online]. (2016). Retrieved April 06, 2019, from https://whatis.techtarget.com/definition/3-D-three-dimensions-or-three-dimensional

Schultz Warren, What Is a AAA Video Game [Online]. (2018). Retrieved April 06, 2019, from https://www.thoughtco.com/what-is-aaa-game-1393920

Matthew S. Stenquist, Differences AAA vs Indie [Online]. (2016). Retrieved April 06, 2019, from https://www.quora.com/What-are-the-differences-in-working-as-a-game-developer-in-AAA-studio-and-as-indie-developer

Indie game hosting marketplace [Online]. (2019). Retrieved April 06, 2019, from https://itch.io/game-development/engines/most-projects

About itch.io [Online]. Retrieved April 06, 2019, from https://itch.io/docs/general/about

Products - Unity [Online]. (2018). Retrieved April 06, 2019, from https://unity3d.com/unity?_ga=2.209030674.1664188563.1554381559-66434191.1533641010

Unity - Scripting API [Online]. (2018). Retrieved April 08, 2019, from https://docs.unity3d.com/ScriptReference/

(c2012). In C# 5.0 in a nutshell (5th ed). Sebastopol: O'Reilly.

Kenlo, Pixel Density: Pixels Per Inch [Online]. (2013). Retrieved April 08, 2019, from https://community.giffgaff.com/t5/Blog/Pixel-Density-Pixels-Per-Inch-PPI-Explained/ba-p/9252950

PSprint, Difference Between Raster and Vector [Online]. Retrieved April 08, 2019, from https://www.psprint.com/resources/difference-between-raster-vector/

14 Seznam Obrázků

15 Seznam Tabulek

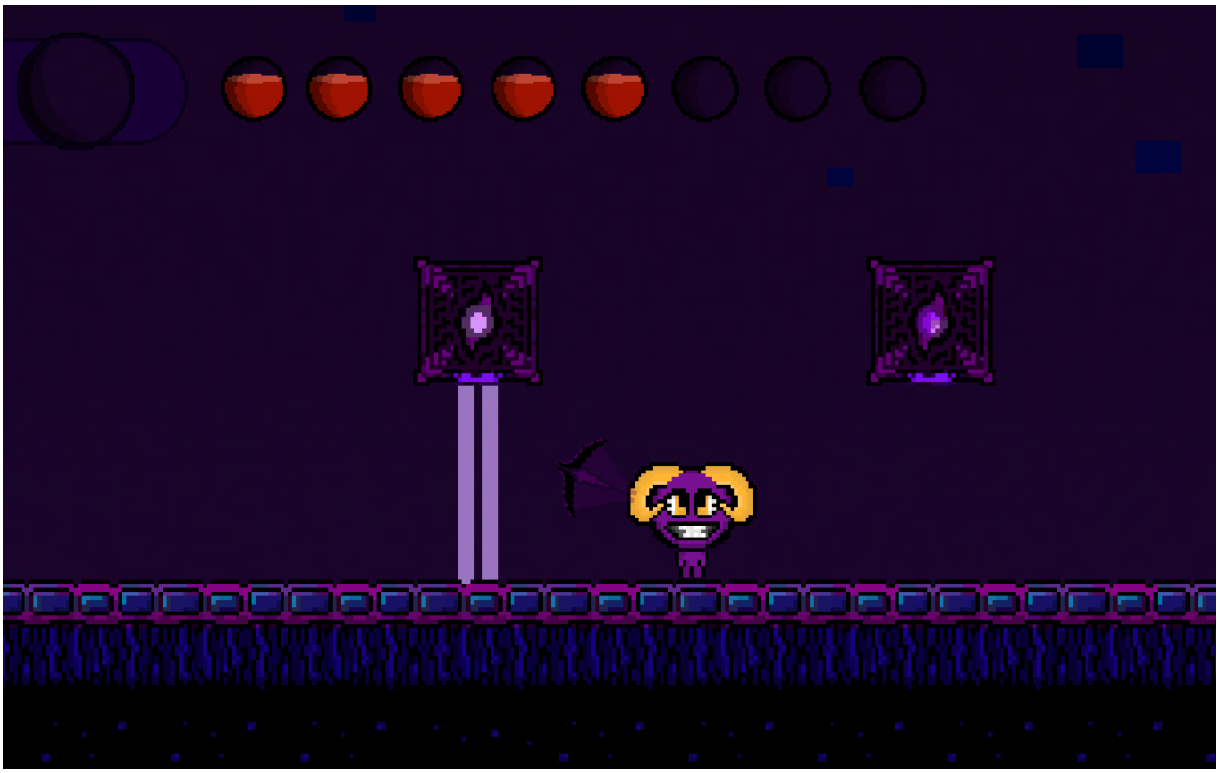
16 Seznam Příloh

17 Přílohy

Příloha 1: Úvodní obrazovka hry

(Zdroj: Autor, 2019)

Příloha 2: Pohyblivé překážky



(Zdroj: Autor, 2019)

Příloha 3: Ukázka z druhého levelu

(Zdroj: Autor, 2019)

Příloha 4: Ukázka z třetího levelu

(Zdroj: Autor, 2019)

Příloha 5: Ukázka skoku

(Zdroj: Autor, 2019)