Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Vývoj 2D hry v unity 3D

Ondřej Burda

Bakalářská práce

2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej Burda**
Osobní číslo:	**I15060**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Vývoj 2D hry v unity 3D**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

V teoretické části práce se student zaměří na problematiku počítačových her. Popíše proces vývoje počítačových her, jednotlivé fáze vývoje. V obou předešlých odvětvích bude student klást důraz na level design.
V rámci praktické části práce se student zaměří na implementaci vlastní 2D hry. Vývoj bude proveden v engine Unity3D s využitím programovacího jazyku C#. V rámci práce je přípustné použít externí assety, avšak důsledně s dodržením licenčních podmínek autorů. Důraz ve vlastní hře bude kladen na level design.

Rozsah grafických prací:

Rozsah pracovní zprávy: **35 - 45 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

OKITA, Alex. Learning C# Programming with Unity 3D. Natick: Taylor & Francis, 2014. ISBN 9781466586529.
MENARD, Michelle a Bryan WAGSTAFF. Game Development with Unity. Mason: Cengage Learning, 2014. ISBN 9781305110540.
LINOWES, Jonathan. Unity Virtual Reality Projects. Birmingham: Packt Publishing Limited, 2015. ISBN 9781783988556.

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2017**

Termín odevzdání bakalářské práce: **12. května 2018**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

Univerzita Pardubice Česká republika

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 20. března 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 5. 2018

Ondřej Burda

PODĚKOVÁNÍ

Děkuji Ing. Josefu Brožkovi, který mě přivedl k oblasti vývoje počítačových her a s vypracováním této práce mi velice pomohl. Děkuji také své rodině, která mě po dobu studia na vysoké škole výrazně podporovala a umožnila mi dělat zejména to, co mě v životě baví.

ANOTACE

Bakalářská práce zkoumá proces vývoje 2D počítačové hry v Unity 3D enginu s hlavním zaměřením na problematiku level designu. V práci jsou popsány teoretické postupy tvorby počítačové hry, práce s externími programy a zásady správného level designu. Tyto zásady jsou předvedeny v implementaci konkrétní počítačové hry.

KLÍČOVÁ SLOVA

Unity, engine, programování, hra, asset, level design, distribuce

TITLE

The 2D game development using Unity 3D

ANNOTATION

This Bachelor's thesis deals with the process of developing 2D computer game in the Unity 3D engine with the main focus on level design. The theoretical methods of computer game creation are described in the thesis, as well as working with external programs and principles of the right level of design. These principles are demonstrated in the implementation of a particular computer game.

KEY WORDS

Unity, engine, programming, game, asset, level design, distribution

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZDROJOVÝCH KÓDŮ

ÚVOD

Cílem této práce je vytvoření počítačové hry za použití herního enginu Unity ve snaze předvést a dodržet pravidla a zásady správného level designu. Vytváření počítačové hry je v dnešní době samo o sobě poměrně složité. Vývojář se musí potýkat s problémy v oblastech programování, grafiky, animací, zvuku, tvorby příběhu, psaní dialogů, marketingu a dalších. Hlavní část této práce je zaměřena na level design, zásady správného vytváření herního světa a metody dosažení ideálního herního zážitku.

Toto téma jsem si zvolil proto, že mě již delší dobu fascinují počítačové hry a herní průmysl, který již v roce 2013 překonal zisky toho filmového o téměř dvojnásobek, což značí obrovský potenciál. Zaměření na level design pro mě bylo novinkou a hlavně výzvou, protože je to jeden z nejdůležitějších aspektů při vývoji her, kterému bych se chtěl v budoucnu věnovat podrobněji.

Na začátku práce se zabývám základními pojmy a prvky, které jsou velice důležité k pochopení a docenění informací v dalších kapitolách. Následuje kapitola zaměřená na teorii kolem procesu vývoje počítačové hry. V této části představuji některé důležité pojmy, různé role uvnitř vývojového týmu a průběh vývoje. Čerpám hlavně ze zkušeností odborníků z praxe.

Další kapitola se věnuje nástrojům použitým k tvorbě praktické části, a to zejména hernímu enginu Unity, který je zde detailně představen a popsán.

Čtvrtá kapitola představuje hlavní zaměření teoretické části práce a zabývá se odvětvím level designu. Jsou zde vysvětleny základní pojmy, příklady level designu v reálných hrách a deset pravidel, jak vytvářet správný level design. Poslední kapitola se zabývá vlastní hrou, jejími alternativami a ukázkami praktických řešení a skriptů.

Celá práce je protkána mými osobními zkušenostmi s počítačovými hrami.

1 ZÁKLADNÍ POJMY

V následujících podkapitolách jsou popsány elementární pojmy, na kterých stojí základy této bakalářské práce.

1.1 Program

„*Program bychom mohli charakterizovat jako v nějakém programovacím jazyku zapsaný předpis, popisující, jak má procesor, pro nějž je program určen, splnit zadanou úlohu*.“ [1] Programy se dělí na překládané a interpretované, podle toho, jestli se napsaný program předává ke zpracování kompilátoru nebo interpreteru. Hlavní rozdíl je ten, že kompilátor překládá (kompiluje) program do kódu příslušného procesoru a zároveň musí používat instrukce vhodné pro použitý operační systém. Interpreter obdržený program postupně prochází a současně provádí, kvůli čemuž je oproti překládaným programům pomalejší. Není ale závislý na platformě, stačí, aby na daném počítači běžel potřebný interpreter.

Vedle tohoto základního rozdělení můžeme některé programy označit také jako hybridní. V tomto případě jsou programy současně překládané i interpretované, pomocí čehož se snaží sloučit výhody těchto metod. Program se nejprve zkompiluje do mezijazyka, který je určen k co nejrychlejší interpretaci speciálním interpreterem, kterému se říká virtuální stroj.

Zdroje [1], [2].

1.2 Programovací jazyk

Programovací jazyk slouží zejména k usnadnění práce programátora, který se podle zavedené syntaxe snaží popsat, jak má počítač vykonat požadovanou úlohu. Pomocí programovacího jazyka vytváří programátor výsledný program.

Dělí se zejména podle míry abstrakce na vyšší a nižší programovací jazyky. Mezi nižší patří například jazyk C nebo Assembler, protože se svojí malou mírou abstrakce velice podobá tomu, jak funguje procesor. Naopak jako vyšší programovací jazyky se označují takové, které se svou mírou abstrakce blíží k tomu, jak na problémy pohlíží lidská mysl. Mezi tyto jazyky patří Basic, C++, Java, C#, Python a další.

Zdroje [1], [2].

1.3 Počítačová hra

Počítačová hra je program určený především k zábavě, který pomocí textu, grafiky a zvuků komunikuje s uživatelem a poskytuje zpětnou vazbu na jeho činy. Hry se rozdělují podle žánrů nebo podle velikosti a náročnosti vývoje.

Mezi hlavní kategorie počítačových her patří FPS (first person shooter), RPG (role playing game), MMORPG (massively multiplayer online role playing game), sportovní hry, závodní hry, bojové hry, budovatelské strategie, RTS (real time strategy), adventury, logické hry, plošinovky, karetní hry, simulátory, hry s otevřeným světem a MOBA (multiplayer online battle arena) hry.

Dále se hry dělí na nezávislé a mainstreamové tituly. Nezávislé hry se označují jako indie (od slova independent) a jsou tvořeny převážně jednotlivci či malými týmy bez finanční podpory vydavatele. Indie hry proto většinou staví na originálním nápadu, příkladem může být velice úspěšný Minecraft nebo logická hra Limbo. Opakem indie her jsou mainstreamové tituly velkých studií, na jejichž vývoji se podílí týmy o desítkách až stovkách zaměstnanců. Za vývojem téměř vždy finančně stojí vydavatelská společnost, která ovšem určitým způsobem vývoj ovlivňuje a může bránit kreativitě a inovacím ze strany herních studií. Ty největší z mainstreamových titulů se často označují jako AAA hry. U těchto projektů se také investují velké částky do reklamy a klasické fyzické distribuce. Oproti tomu indie tituly spoléhají zejména na digitální distribuci (Steam, itch.io, GOG, Green Man Gaming).

Zdroje [3], [4], [5].

1.4 Herní engine

Jako herní engine se většinou označuje kompletní technologické řešení, pomocí kterého vývojáři vytvářejí hry. Konkrétně se tedy jedná o software, který se stará o zobrazování herní scény, zvukový systém, poskytuje skriptovací framework, stará se o simulaci fyziky (gravitace, detekce kolizí, simulace hmoty a další) a v mnoha případech zároveň disponuje sadou nástrojů pro editaci scén, import dat (assetů) a možností nastavovat různé aspekty a technické možnosti hry.

Herní engine se také stará o přijímání vstupů od uživatele (myš, klávesnice, gamepad, joystick, mikrofon, VR headset a další). Vstupy jsou pomocí mapovací tabulky napojeny na konkrétní události herního enginu.

Většina moderních enginů také nabízí mnoho komponent a předepsaných skriptů k řešení síťového připojení k serverům v případě hry více hráčů.

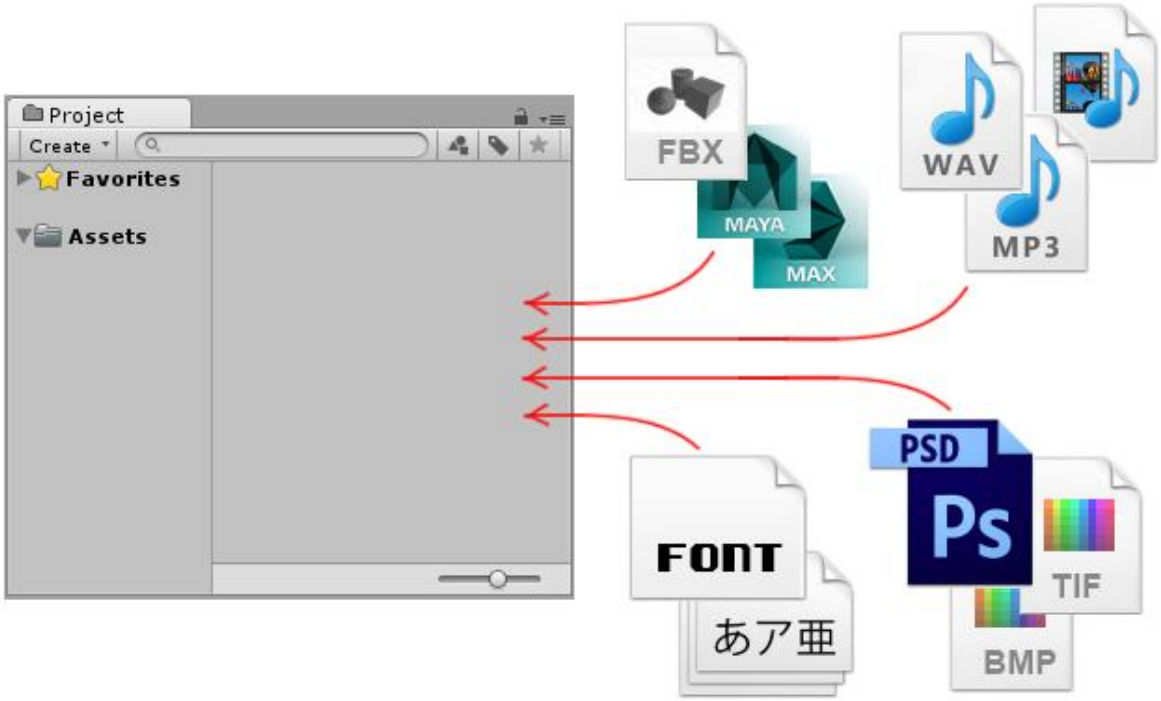
Příklady nejpopulárnějších herních enginů, které jsou volně dostupné:

- CryEngine – engine německého studia Crytek, do roku 2011 byl pouze komerční
- Unreal Engine – populární engine firmy Epic Games, vyzdvihovaný hlavně pro svoje schopnosti zpracování 3D grafiky
- Unity 3D – vyvíjen firmou Unity Technologies, vhodný pro začátečníky
- GameMaker – pouze pro tvorbu 2D her, vhodný pro začátečníky bez znalosti programování
- Lumberyard – engine od firmy Amazon, postavený na základech CryEngine
- Godot – méně známý open-source engine

Zdroje [4], [6].

1.5 Asset

Asset je obecný název pro jakýkoliv objekt používaný při vývoji hry. Může se jednat o 2D sprite (obrázek), 3D model, zvuk, animaci, tileset, materiál, texturu a podobně. Jednotlivé assety se vytvářejí v k tomu určených programech a následně se importují do Unity projektu. Pro tvorbu 2D spritu můžeme využít například Adobe Photoshop a k modelování 3D objektů Blender nebo podobný software. Některé typy assetů, jako Audio Mixer nebo Animation Controller, se dají vytvořit přímo v Unity.



Obrázek 1: Import assetů do projektu [1]

Na obrázku 1 jsou zobrazeny některé podporované typy assetů a jejich import do Unity projektu.

Zdroj [7].

1.6 Skript

Jako skript se označuje program napsaný v libovolném skriptovacím jazyce. Často se jedná o určitou rozšiřitelnou část většího softwarového projektu, která se dá měnit bez nutnosti rekompilovat hlavní spustitelný soubor.

Skriptování je základní součástí všech her. Dokonce i nejjednodušší hra potřebuje skripty, aby reagovala na vstup od hráče a zajistila, aby události ve hře probíhaly tak, jak by měly. Kromě toho mohou být skripty použity k vytváření grafických efektů, ovládání fyzického chování objektů nebo dokonce k implementaci vlastního systému umělé inteligence pro postavy ve hře.

[1] Obrázek převzat ze zdroje [7]

V Unity3D se jako scriptovací jazyk používá C# nebo JavaScript. Mezi další populární scriptovací jazyky patří například Lua (jazyk používaný českým studiem WarHorse) nebo jazyk C++, kterým disponuje Unreal Engine.

Zdroj [8].

2 PROCES VÝVOJE POČÍTAČOVÉ HRY

Tato kapitola se zabývá procesem vývoje počítačové hry od prvotního nápadu až k distribuci výsledného produktu. Popisuje hlavní témata spojená s vývojem počítačové hry a klade důraz na zkušenosti odborníků z praxe.

2.1 Game design a GDD

Game design „*definuje, o čem hra je a jak se hraje. Game design je proces vytváření cílů, kterých se hráči cítí být motivováni dosáhnout, a pravidel, kterými se hráč při pronásledování cílů musí řídit*“. [3]

Game design se zapisuje a detailně popisuje v tzv. game design dokumentu (GDD). V tomto dokumentu je možné, a většinou téměř nevyhnutelné, dělat změny a přidávat nebo upravovat herní prvky. Je důležité mít na paměti, že game design dokument se nevytváří pro hráče, ani pro vydavatele, nýbrž pro ostatní členy týmu. V průběhu vývoje pomáhá všem zaměstnancům udržet si jasnou a aktuální vizi o projektu, na kterém zrovna pracují. Obvykle obsahuje kapitoly: příběh, postavy, hratelnost, prostředí a herní svět, zvuk, ovládání a uživatelský interface, přehled assetů a v neposlední řadě i plán vývoje, milníky a různé analýzy.

„*Ve středu dobrého game designu je samotný hráč. Je důležité si uvědomit, že hra je tvořena jenom a pouze pro něj. Místo toho, abychom mu v rámci pravidel hry něco nařizovali, snažíme se jej motivovat, aby cítil chuť kráčet cestou, kterou jsme pro něj připravili. Snažíme se jej poznat, abychom tuto cestu vydláždili přesně tam, kam ho to táhne.*“ [3]

Součástí game design dokumentu by měl být i tzv. pitch, což je krátký a jasný popis hry, který se prezentuje veřejnosti, novinářům a potencionálním vydavatelům.

„*Zatímco v případě účetního programu jde pouze o jeho funkcionalitu, hry mají v hráči vytvářet zážitky, vzbuzovat pocity a zanechat hluboký dojem.*“ [3]

Jelikož se zážitek ani pocity z hraní nedají předem naplánovat žádným diagramem, musí se v průběhu vývoje nově vytvořené části hry okamžitě testovat. V praxi se game design, jeho herní prvky a vyvíjená hra budují metodou iteračního cyklu. Tento cyklus se skládá ze tří základních bodů:

1. Navrhnout
2. Vyhotovit

3. Vyzkoušet a vyhodnotit

Hra tak vzniká po jednotlivých částech, jejichž funkčnost a předpokládaný herní zážitek se ihned testují.

„*Designování her doprovází zklamání, špatná rozhodnutí, omyly a nutnost činit kroky zpět. Iterační proces je tu od toho, aby odhalil problém dříve, než na jeho základech vytvoříme další struktury hry, který by pak musely být strženy*.“ [3]

Zdroje [3], [4].

2.2 Týmové role

V početných vývojářských týmech je možné pozorovat velké množství různých týmových rolí a specializací. U menších týmů je pravděpodobné, že jedna osoba bude zastávat více rolí. Například game designer může být zároveň hlavní programátor nebo spisovatel.

Game designer je jedinec, který prezentuje komplexní uměleckou vizi a současně disponuje technickými schopnostmi a znalostmi, aby dokázal tuto vizi přeměnit ve výsledný produkt. Hlavní výsadou game designera je činit rozhodnutí, za která následně nese i zodpovědnost.

Programmer (programátor) se může věnovat různým programátorským disciplínám jako například tvorba herního enginu, programování fyziky, komplexní grafické 2D/3D renderování obrazu, umělá inteligence, skriptování levelů, hratelnost a herní prvky nebo multiplayer (hra více hráčů po lokální síti nebo internetu).

Tester (Quality Assurance), jeho úkolem je hledat ve hře chyby a následně o nich podávat zprávy programátorům. Osoba zaměřená čistě na testování hry z pohledu opravdového hráče nemá přístup k debug verzi, proto je někdy obtížné chybu ve hře objevit.

Writer (spisovatel) je osoba zodpovědná za tvorbu dialogů mezi různými postavami ve hře a další texty spojené s příběhem.

3D/2D Artist (modelář/grafik) vytváří grafické assety. Může se jednat o 3D modely, obrázky, textury a podobně. Speciálním odvětvím je tzv. concept artist, který tvoří pouze konceptuální návrhy prostředí a postav, které následně grafici a modeláři vytváří jako assety. Často bývá concept artist a grafik tatáž osoba.

Sound designer/composer má na starost skladbu hudebního podkladu hry. Pokud hudba pasuje do prostředí a situace, ve které se hráč zrovna nachází, pomáhá mu to k vnoření se do hry.

Výrazná hudba ve špatný moment může naopak hráče vyrušit ze soustředění a kazit mu herní zážitek.

Project manager je osoba, která se stará o správný chod vývojářského týmu, dohlíží na plnění cílů, činí důležitá rozhodnutí a urovnává konflikty mezi členy týmu.

Zdroje [3], [4].

2.3 Fáze vývoje

Vývoj počítačové hry prochází z pravidla třemi základními fázemi. Preprodukce, produkce a postprodukce. Všechny tyto fáze se od sebe výrazně liší, a to zejména složením vývojářského týmu a druhem práce, kterou se tým v danou chvíli zabývá.

Preprodukce

„*Úvodní fáze projektu se z větší části odehrává na papíře. Designéři připravují „bibli projektu“, tedy design dokument s popisem obsahu, průběhu a vlastnostmi hry. Inženýři dělají nezbytné technologické prototypy pro ověření, zda jsme to schopni vůbec udělat. Jednotlivé odborné skupiny testují výrobní proces a nástroje drobnějšími ukázkovými benchmarky, které zároveň nastaví šablonu pro časové kalkulace celého vývoje. A manažeři na základě designu a záměru sepisují nezbytné plány: rozpočet, alokace zdrojů, strukturu týmu, plán procesů a personální plán (kolik lidí bude potřeba na výrobu najmout a podobně).*“ [3]

Produkce

Produkční část trvá zpravidla nejdéle a v době této části čítá vývojářský tým nejvíce pracovníků. Produkce postupuje podle plánu vytvořeného v preprodukční fázi. *„Takovýto plán musí kromě všeho obsahu projektu a jednotlivých kroků k výrobě tohoto obsahu mít logicky napojené závislosti: například výroba skriptu určitého herního levelu nemůže začínat ve stejný den, jako výroba samotné geometrie té stejné úrovně…*“ [3]. Na začátku produkce se vytváří hrubý prototyp s tzv. placeholdery, což jsou objekty, které slouží pouze k testovacím a ukázkovým účelům. Prototyp by měl obsahovat hlavní herní mechaniky.

V této produkční fázi prochází vytvářená hra čtyřmi stádii vývoje. Pre-alfa, alfa, beta a release candidate. V pre-alfa fázi se proces vývoje zaměřuje hlavně na programování a tvorbu veškerých assetů. Do alfa fáze se hra dostává v momentě, kdy už je hratelná, ale ještě není dokončená. Alfa verze hry znamená, že celý herní koncept je již hotový a dále se nemění. V tento moment se v produkčním týmu začínají objevovat i testeři. Do beta verze se hra dostává

v momentě, kdy už je hotová, ale se spoustou potencionálních chyb, které je nutné opravit. Využívá se hlavně k analýze dat získaných od testerů (v tomto případě obyčejných hráčů), opravě chyb, na které při hraní narazí, a balancování různých aspektů hry. Beta verze může být dvojího druhu. Uzavřená, kde se beta testeři pečlivě vybírají na základě informací, které o sobě poskytli v přihlášce, nebo otevřená, do které má přístup kdokoli. Otevřená beta se využívá hlavně u online her, kde je nutné testovat i stabilitu serverů v herní špičce, k čemuž je zapotřebí větší počet hráčů. Další výhodou otevřené bety je, že se jedná o velice efektivní způsob jak produkt otestovat na různém hardware. V momentě, kdy je hra připravena na vydání, se označuje jako release candidate.

Postprodukce

V momentě, kdy se produkt dostává na regály obchodů a digitální distribuce, začíná fáze postprodukce. U počítačových her způsobuje většinu chyb různorodost a konfigurace hardware a software, proto vydáním vývoj hry nekončí (jako je tomu u většiny konzolových her). Z tohoto důvodu je nutné vydávat opravné patche, které se zaměřují na hlavní a nejčetnější chyby, které uživatelé hlásí. V některých případech bývá součástí opravného patche i nový herní obsah.

Zdroj [3].

2.4 Marketing

Marketing je nedílnou součástí každého projektu týkajícího se vývoje počítačové hry. Stará se o to, jak dostat produkt k zákazníkovi. Součástí projektu by proto měl být i marketingový plán se zaměřením na hlavní cíle a na to, jak těchto cílů dosáhnout.

„*Základní definice marketingu o něm hovoří jakožto o procesu, který vede k uvedení nového produktu na trh. Patří sem věci jako je cílová skupina, nacenění, místa, kde se dá produkt koupit a pak samozřejmě i ta inzerce a PR. Těmto složkám se říká marketingový mix (tzv. 4P – Product, Place, Promotion, Price).*“ [4]

Product, v našem případě hra, vytváří základ, na kterém jsou postaveny všechny ostatní marketingové náležitosti. Správná propagace a komunikace s veřejností a novináři zvyšuje pravděpodobnost, že hře bude věnována odpovídající pozornost. Je také zapotřebí správné cenové strategie, která bude hře odpovídat a osloví cílovou skupinu zákazníků.

Hlavním cílem marketingu je prodej (v našem případě produktu - hry). „*Věci jako budování komunity, budování značky, rozšiřování povědomí nebo výzkum trhu k marketingu patří, ale jako nástroje nebo strategie. Před každou aktivitou nebo kampaní je třeba si zhodnotit, zda vede potencionálně k prodeji*.“ [4]

Zdroje [3], [4], [9].

2.5 Distribuce

Distribuce je proces, kterým prochází každá hra, než se dostane od výrobce k zákazníkovi. Cesta, kterou se hra k zákazníkovi může dostat, se označuje jako distribuční kanál. V současnosti se rozlišují dva základní způsoby, jak se může hra dostat až k zákazníkovi.

Součástí distribuce je i vydavatelská společnost (publisher), která investuje do vývoje hry, stará se o reklamu, propagaci, následnou spolupráci s distributorem a uvedení hry na trh.

Fyzická distribuce

Fyzická distribuce je *„klasický model, který je pro fyzický prodej her v kamenných obchodech nejtypičtější. Distributor zde figuruje jako objekt, který přistupuje k některému trhu, má přímý kontakt s obchodníky i sílu produkt protlačit na poličky obchodu a postarat se o reklamu. Distributor zde má na starosti i lokální úpravy titulů, jako je například překlad do místního jazyka“*. [3]

Digitální distribuce

Digitální distribuce je proces, kterým se dostává výsledný produkt k zákazníkovi pomocí internetu. V tomto případě nedostane zákazník žádnou fyzickou kopii produktu. Ten je uložen na jeho pevném disku a je jím právně vlastněn. V tomto případě splývají distributor i obchodník v jedno a to samé, více peněz tak zůstane na straně herního vydavatele nebo vývojářského studia. Pro zákazníky to bohužel znamená, že je téměř nemožné digitální kopii hry prodat někomu jinému, jako se to v minulosti u fyzické distribuce běžně praktikovalo. Mezi nejznámější digitální distribuční kanály na PC patří Steam, Microsoft Store, EA Store a Origin. U konzolí jsou to potom portály samotných výrobců. Například Xbox Live, PlayStation Network a další.

Z výzkumu americké společnosti Statista, provedeného mezi roky 2009 a 2016, se poměr prodaných kusů videoher na půdě Spojených Států Amerických téměř obrátil. Oproti minulosti, kdy digitální distribuci bránilo zejména pomalejší připojení k internetu většiny zákazníků, se

v současnosti karta otočila a digitální distribuce dominuje nad klasickým fyzickým modelem. Na počátku zmiňovaného statistického výzkumu v roce 2009 se 80% veškerých prodaných videoher distribuovalo skrze obchody jako fyzické kopie. V roce 2016 už je toto číslo pouze 26%. Nejen tento výzkum je důkazem, že budoucnost patří digitální distribuci, protože model fyzické distribuce se stane vedlejším a využívaným zejména sběrateli.

Zdroje [3], [4], [10].

3 POUŽITÉ NÁSTROJE

V této kapitole jsou popsány nástroje a programy použité při tvorbě praktické části práce. Při výběru nástrojů bylo nahlíženo zejména na pořizovací náklady a možnosti nástrojů vzhledem k typu a potřebám projektu.

3.1 Unity3D

Unity3D je, jak již bylo zmíněno v kapitole 1.5, multiplatformní herní engine vyvíjený firmou Unity Technologies. V aktuální verzi 2017.3 nabízí možnost vývoje až pro 27 různých platforem, mezi které patří například Windows, iOS, Android, Linux, PlayStation, Xbox, Wii U, Nintendo 3DS, Steam VR, Oculus Rift a další. Kromě základní Personal verze, která je zdarma a disponuje všemi možnostmi enginu, nabízí Unity také dvě placené verze. Verzi Plus, která cílí na zkušenější vývojáře – jednotlivce, a verzi Pro, jejíž cílová skupina jsou profesionálové a herní studia.

Obrázek 2: Logo Unity enginu [2]

Unity (obrázek 2) se těší početné a ochotné komunitě vývojářů, má k dispozici rozsáhlou dokumentaci a množství vlastních tutoriálů. Další předností Unity je internetový obchod s assety. Zde je možné si zakoupit různé druhy assetů od 2D spritů, kolekcí 3D modelů, zvuků, efektů, skriptů až po celé scény s vytvořeným herním světem nebo různá rozšíření editoru. Někteří umělci zde dokonce nabízejí část svých produktů zdarma. Vývojář se díky tomuto systému může soustředit na práci, která mu jde nejlépe, a zbytek si dokoupit.

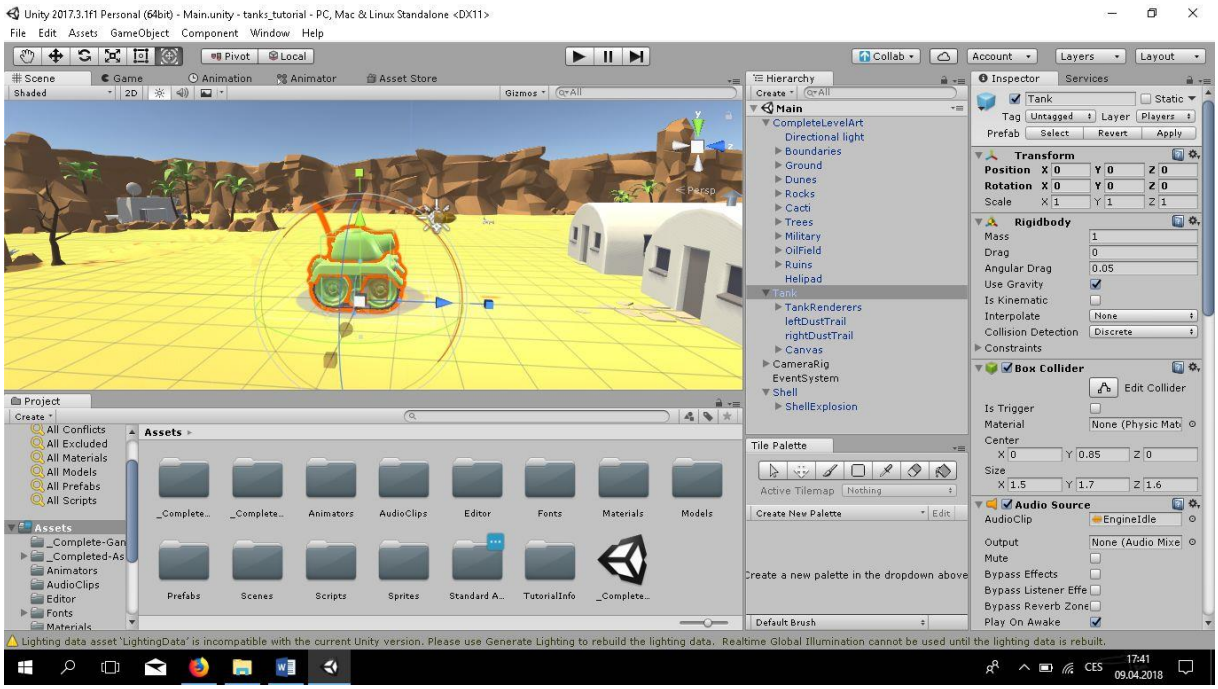
Zdroje [11], [12].

[2] Obrázek převzat ze zdroje [12]

3.1.1 Editor

Hlavní okno Unity editoru (obrázek 3) se skládá z několika různých panelů, se kterými může uživatel libovolně pohybovat a každý z nich slouží ke specifické činnosti. Pokud si uživatel vytvoří z panelů určitý layout, může si ho také uložit. Toto umožňuje používat pro různé projekty i různé přednastavené layouty panelů v editoru, vhodné například pro střídání mezi 2D a 3D projekty, kde se občas používají rozdílné panely a nastavení.

Jedním z nejzákladnějších panelů je prohlížeč Project. Zde můžeme procházet a spravovat veškeré assety, které k projektu patří. Levý panel prohlížeče zobrazuje strukturu složek projektu jako hierarchický seznam. Pokud je ze seznamu vybrána složka, její obsah se zobrazí v panelu napravo. Jednotlivé položky jsou zobrazeny v pravém panelu jako ikony označující jejich typ (skript, materiál, scéna, podadresář a další).



Obrázek 3: Unity editor

Další základní a velice důležitý panel se nazývá Hierarchy. Obsahuje všechny objekty, které se nacházejí v aktuální otevřené scéně. Mohou to být přímé instance assetů nebo tzv. prefaby, což jsou uživatelské šablony objektů, které tak umožňují ukládat vytvořené objekty společně s jejich nastavením a vlastnostmi. Výhodou prefabů je, že pokud se změní šablona, změny se provedou u všech objektů z prefabu vytvořených.

Game panel reprezentuje výslednou hru a aktivuje se automaticky po spuštění módu play. Jedná se o vyrenderovaný pohled z kamery do herního světa a reprezentuje výslednou podobu produktu. Každá hra musí obsahovat alespoň jeden objekt typu Camera, pomocí kterého vývojář určuje, co hráč v určitou chvíli vidí. Play mód zároveň umožňuje sledovat určité statistiky, jako například počet snímků za vteřinu nebo vytížení procesoru.

Poslední ze základních panelů je Inspector. Hry v Unity enginu jsou tvořeny z různých objektů, které se skládají z mnoha komponent, jako jsou collidery, meshe, skripty, materiály a další. Inspector umožňuje zobrazit informace o aktuálně vybraném objektu a jeho komponentách, měnit jejich vlastnosti, hodnoty a celkově upravovat zvolený objekt. Jakákoliv vlastnost, která je v inspektoru zobrazena, může být přímo upravena a změněna. Dokonce i proměnné skriptu lze měnit bez úpravy samotného skriptu. Stačí ve skriptu definovat proměnnou jako veřejnou. Pokud se jedná o objektový typ, stačí v inspektoru přetáhnout zvolený objekt do příslušného políčka reprezentujícího proměnnou. Vývojář si musí dávat pozor na fakt, že hodnota nastavená v inspektoru vždy přepíše hodnotu ve skriptu. Inspektor se tak dá použít i pro testování a experimentování s hodnotami proměnných skriptů.

Zdroj [13].

3.1.2 Scéna

Panel scéna je jedna z nejdůležitějších částí editoru, protože představuje herní svět, do kterého vývojář umísťuje veškeré objekty a hráč se v něm následně pohybuje či s ním jakkoliv interaguje. I v případě 2D projektů je scéna a herní svět vždy trojrozměrný, a to zejména kvůli zajištění hloubky a pořadí vykreslování objektů na obrazovku.

Obrázek 4: Scéna s vloženým objektem [3]

V pravém horním rohu scény je vidět aktuální orientace kamery ve scéně a umožňuje rychle upravit úhel pohledu. Každý z barevných kuželů reprezentuje jednu z os X, Y a Z. Kliknutí na kterýkoli z nich nastaví kameru na ortografický (bez perspektivy) pohled podél odpovídající osy. Ovládací lišta panelu umožňuje vybrat různé možnosti zobrazení, například vypnout/zapnout osvětlení, zvuky, 2D režim a také ovládat zobrazení některých efektů a objektů.

Zdroj [14].

3.1.3 Základní komponenty

Komponenty jsou funkčními částmi všech herních objektů a chování hry. Objekt ve hře je tedy pouhý kontejner pro mnoho různých komponent. Ve výchozím nastavení mají všechny tyto

[3] Obrázek převzat ze zdroje [14]

objekty automaticky přiřazenou komponentu Transform. Jako komponenty se považují i skripty (kapitola 1.6). Následující popis jednotlivých komponent vychází z oficiální dokumentace Unity.

Transform

Komponenta Transform slouží k orientaci objektu v 3D či 2D prostoru. Umožňuje měnit pozici objektu, rotovat a škálovat jeho velikost.

Collider

Komponenta, která definuje fyzikální tvar objektu a stará se o detekci kolize, se nazývá Collider. Mezi základní 3D typy patří Box Collider, Sphere Collider, Capsule Collider a Mesh Collider. V projektech zaměřených na 2D svět se nejčastěji používají Box Collider 2D, Circle Collider, Polygon Collider a Edge Collider. Jednotlivým colliderům je možné přiřadit vlastní fyzikální materiál, který může sloužit například k určení míry tření mezi jednotlivými objekty. Collider nastavený jako trigger (pomocí vlastnosti Is Trigger) se nechová jako pevný neprůchozí objekt, ale pouze volá metodu OnTriggerEnter na skriptech objektu, kterou musí vývojář doplnit o vlastní kód.

Rigidbody

V případě 2D i 3D projektů je Rigidbody hlavní komponenta, která umožňuje objektu chovat se podle fyzikálních zákonů. Pokud je k objektu připojena tato komponenta v režimu dynamic, okamžitě reaguje na gravitaci. Pokud je k objektu zároveň připojena jedna nebo více komponent typu Collider a jsou správně nastaveny některé vlastnosti, jako například hmotnost, tak kolize zapříčiní pohyb objektu. V případě, že chce vývojář ovládat objekt pouze pomocí skriptu, nikoliv fyziky, a z nějakého důvodu potřebuje zachovat komponentu rigidbody, může využít režim kinematic.

Animator

Komponenta Animator slouží k přiřazení a spouštění animací objektu. Vyžaduje odkaz na Animator Controller (asset, jenž se dá vytvořit v Unity), který definuje, které animační klipy použít, a řídí, kdy a jak se mezi nimi přechází.

Camera

Jak již název napovídá, Camera je komponenta sloužící k zachycování a následné prezentaci herního světa na obrazovku pro hráče. Může být také libovolně upravována a využívat skriptů k dosažení rozmanitých efektů. Pro různé herní žánry se hodí jinak zpracované kamery.

V případě střílečky z první osoby se kamera umísťuje do objektu hráče na úroveň očí, u strategií zabírá kamera pohled shora na herní svět a v případě závodních her většinou kamera následuje v těsné blízkosti a v plynulém přirozeném pohybu hráčovo vozidlo.

Particle System

Particle System je komponenta, která se používá k simulaci tekutiny, mraků, ohně nebo třeba mlhy, a to tím, že generuje a animuje velké množství malých částic ve scéně. Jako částice mohou sloužit 2D obrázky nebo 3D objekty. Každá částice poté reprezentuje pouze zlomek výsledku a má určenou dobu vlastní životnosti, po kterou může projít mnoha změnami. Na částice mohou také působit okolní fyzikální síly, čímž se dá docílit například realistického počasí a pohybu mraků.

Light

Komponenta Light je jednou ze základních částí grafického vykreslování, protože určuje stínování objektu a stíny, které vrhá. Světla jsou skvělým způsobem, jak do scény přidat mnoho různorodosti, a slouží zejména k budování atmosféry a prostředí. Unity disponuje čtyřmi druhy světla. Point Light reprezentuje bod v prostoru a vysílá světlo rovnoměrně do všech směrů a nastavitelné vzdálenosti. Spot Light se používá u objektů jako je baterka, protože svítí pouze v určitém úhlu a do určité vzdálenosti. Třetím typem je Directional Light, který nemá pevný počáteční bod, ale pouze směr, ve kterém světlo dopadá. Posledním druhem světla v Unity je Area Light, který se využívá například pro realistické zobrazení interiéru domu.

Audio Source

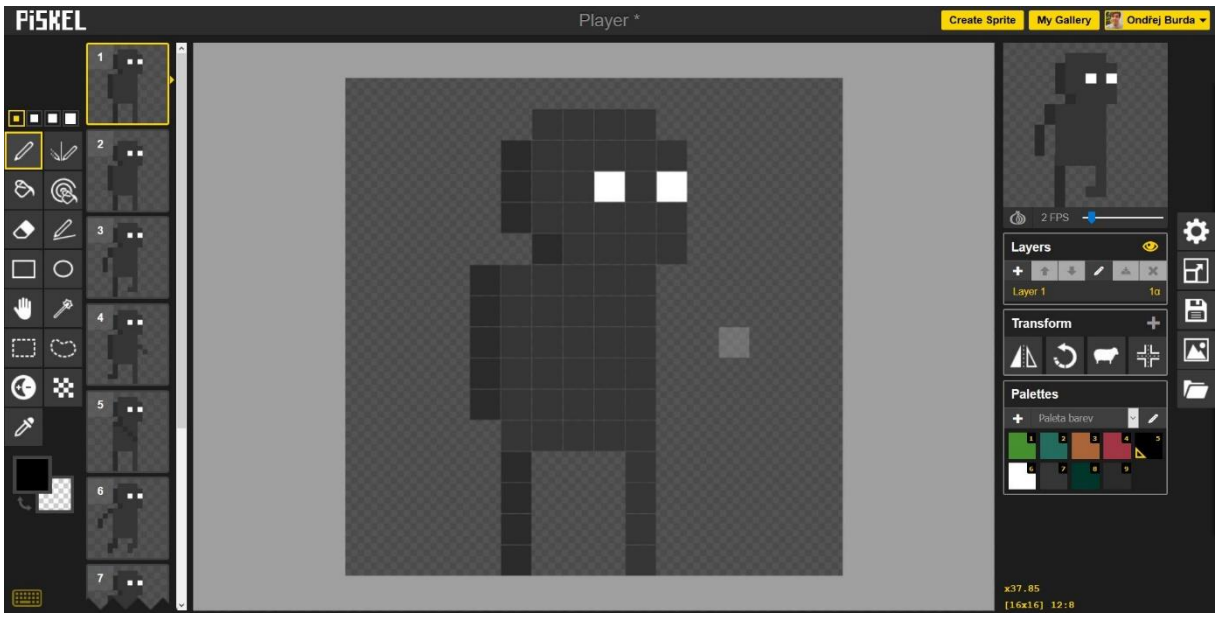
Audio Source slouží k přehrávání zvuku v herním světě a využívá k tomu assetu Audio Clip, který vzniká po importu nějakého podporovaného zvukového souboru (Unity podporuje formáty aif, wav, mp3 a ogg). Změnou vlastnosti Rolloff se dá pomocí křivky nastavit postupné klesání či stoupání hlasitosti v závislosti na vzdálenosti.

Zdroj [15].

3.2 Piskel

Piskel (obrázek 5) je bezplatný grafický editor pro tvorbu animovaných obrázků a pixel artu. Jedná se o open source projekt postavený čistě na technologiích JavaScript, HTML a CSS.

Program je k dispozici v online verzi i jako desktopová aplikace pro Windows, OSX a Linux, a je velice jednoduchý a uživatelsky příjemný.



Obrázek 5: Piskel

Editor nabízí základní funkcionality, jako je například kreslení tužkou, čtyřúhelníkem, elipsou, tužkou se zrcadlením, přímkou, vymalování všech pixelů stejné barvy a gumování. Umožňuje také vybrat určitý tvar, smazat ho, přesunout nebo na místo výběru vložit.

V případě vytváření animace umožňuje i okamžitý náhled s měnitelným počtem snímků za vteřinu.

Dalšími z funkcionalit, které Piskel nabízí, jsou možnost práce s libovolným množstvím vrstev, transformace obrázků nebo vytváření barevných palet.

Jednotlivé spritesheety (listy obrázků) se dají uložit do místní galerie, stáhnout ve formátu piskel nebo exportovat jako gif či png.

Zdroj [16].

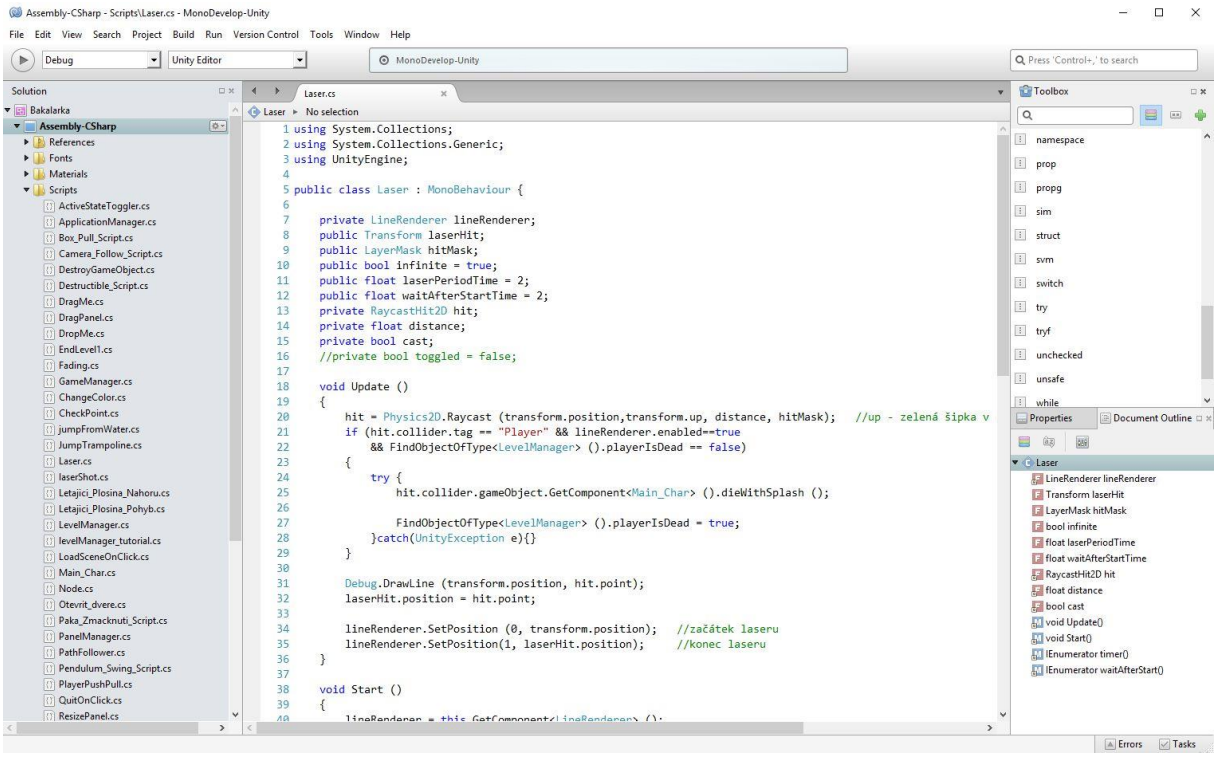
V rámci praktické části byl tento program použit k tvorbě veškerých spritů a animací.

3.3 MonoDevelop

MonoDevelop, také známý jako Xamarin Studio, je open source vývojové prostředí pro Linux, MacOS a Windows. Jeho primárním zaměřením je vývoj projektů, které používají rámce Mono a .NET. MonoDevelop integruje funkce podobné funkcím NetBeans a Microsoft Visual Studio,

jako je automatické dokončení kódu, grafické uživatelské rozhraní (GUI) a návrhář webu. Podporuje C, C++, C#, CIL, F#, Boo, Java a další.

MonoDevelop lze použít v systémech Windows, MacOS a Linux. Mezi oficiálně podporované linuxové distribuce patří CentOS, Debian, Fedora, openSUSE, SUSE Linux Enterprise, Red Hat Enterprise Linux a Ubuntu, ale je kompatibilní i s mnoha dalšími distribucemi, které poskytují své neoficiální sestavy MonoDevelopu ve svých úložištích.



Obrázek 6: MonoDevelop IDE

Upravenou verzi MonoDevelop je možné stáhnout přímo při instalaci Unity. Tato verze podporuje pokročilé skriptování v jazyce C#.

Zdroje [17], [18].

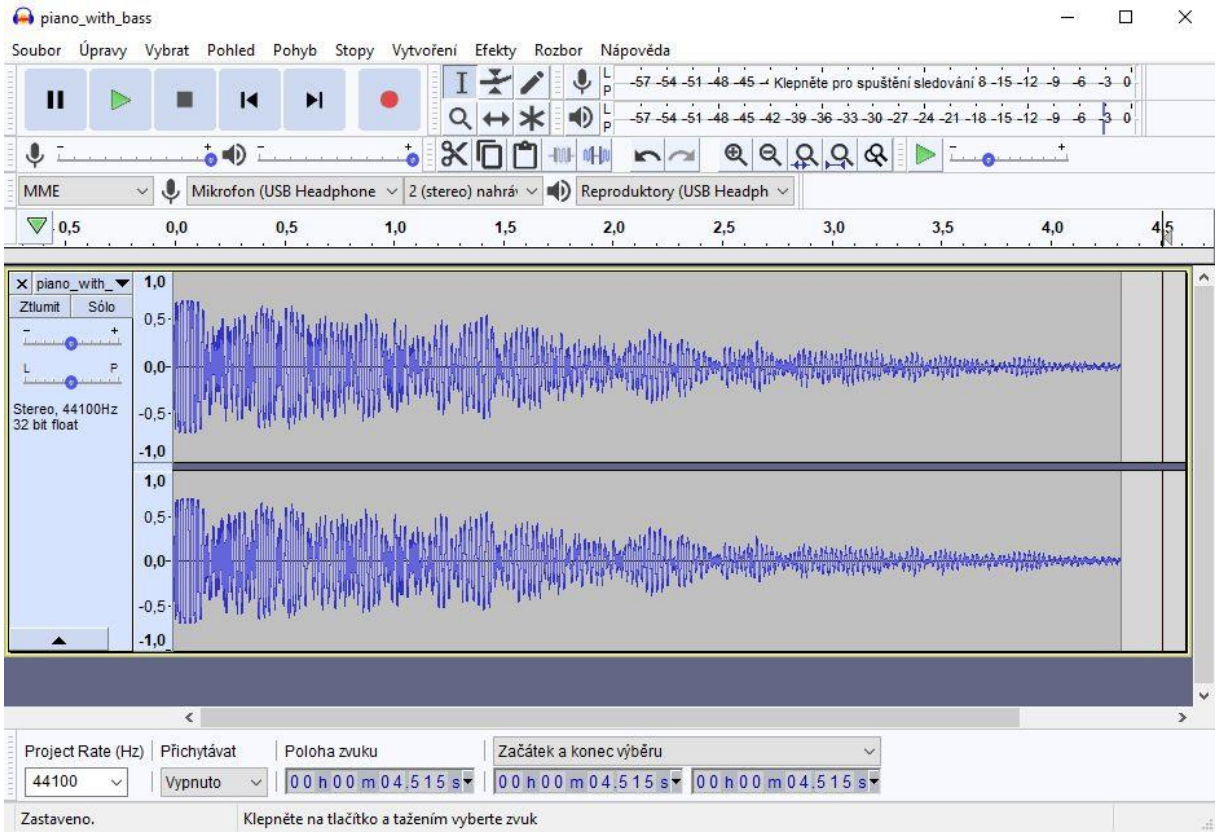
V rámci praktické části této práce bylo vývojové prostředí MonoDevelop (obrázek 6) použito pro tvorbu a úpravu skriptů použitých ve hře.

3.4 Audacity

Audacity (obrázek 7) je open source multiplatformní program pro nahrávání a editaci zvuku, který je k dispozici zdarma pod licencí GNU (General Public License). Podporuje formáty WAV, AIFF, MP3, FLAC a další. Mezi hlavní vlastnosti programu patří stříhání, rozdělování

nebo mixování zvukových stop, odstraňování šumu, přidávání různých efektů včetně změny rychlosti. Další výhodou je možnost tvorby vlastních pluginů pomocí programovacího jazyku Nyquist, který je určený pro analýzu zvuku a je vyvinutý na základech jazyka Lisp.

Zdroj [19].



Obrázek 7: Audacity

V praktické části byl program využit k úpravě a střihu zvukových stop, které vyzdvihují a podporují atmosféru výsledné hry.

4 LEVEL DESIGN

Jako level design se označuje proces vytváření herního prostředí, se kterým hráč interaguje. Takováto definice je bohužel příliš jednoduchá a neúplná, protože level design je značně rozsáhlá disciplína, která se neomezuje pouze na určitou oblast herního vývoje. Proto se tato kapitola bude level designu věnovat velice podrobně.

Od počátku osmdesátých let minulého století se pod pojmem level design rozumělo umístění překážek, dlaždic, vylepšení a nepřátel. Příkladem mohou být hry Mario Bros nebo Sonic the Hedgehog. S příchodem 3D her, které se začaly hojně vyskytovat v devadesátých letech minulého století, se level design rozšířil. Zahrnoval zejména vytváření bludišť a map k prozkoumávání a přidělování nepřátel a objektů na příslušné pozice. V této době vznikly přelomové hry jako Doom, Quake, Diablo nebo Fallout, které určily nový směr herního průmyslu, a zejména level designu.

V dnešní době se level design rozumí jako něco mnohem komplexnějšího a důležitějšího. S rostoucí velikostí moderních her level design obvykle zahrnuje vytváření obrovských, rozsáhlých prostředí, ve kterých je detailně zkonstruováno všechno, od oken, dveří, budov, krajních prostředí a podobně. Kromě toho zahrnuje nejen umístění předmětů a nepřátel, ale i skriptování jejich chování v reakci na události v herním světě nebo akce uživatelů. Součástí level designu je také analýza toho, jak se hráči chovají ve fiktivním prostředí, a manipulace s nimi za účelem vytvoření herního zážitku. Kombinuje dovednosti z programování, skriptování, architektury, umění, psychologie a spousty dalších odvětví.

Zdroje [20], [21].

4.1 Rozdíl mezi level designérem a game designérem

Pro hlubší pochopení hlavního tématu této práce je důležité si uvědomit rozdíl mezi level designem a game designem. Tyto termíny se na pohled zdají snadno zaměnitelné.

Game designér je člověk, který se stará o celý koncept hry a vymýšlí jednotlivé její elementy, mechaniky a vlastnosti. Je zodpovědný za veškeré netechnické aspekty hry. Level designér podle game designu realizuje herní svět a ovlivňuje hratelnost za použití vlastní fantazie a implementace. Je-li uvažován příklad 3D počítačové válečné hry s důrazem na reálnost, game designér vymyslí a společně s programátory připraví speciální a zábavný systém krytí, což bude základním stavebním kamenem této hry. Po zdlouhavém zkoumání historických událostí se

rozhodne pro zasazení hry do Belgie v době druhé světové války. Následná práce level designéra spočívá ve vytvoření a naskriptování světa nebo úrovní, které budou zmíněný systém krytí přímo podporovat a nutit hráče využívat ho bez pocitu, že by k tomu byl hrou bezprostředně přinucen. Zároveň by měl ale hráč instinktivně vědět, že bez využití krytí se v postupu hrou dál nedostane.

V menších studiích bývá většinou game designér a level designér tentýž člověk. Pokud to tak ovšem není, je důležité, aby byly všechny osoby zapojené do designu hry naladěny na stejnou vlnu uměleckého cítění a podrobně znaly všechny důležité aspekty game designu.

Speciálním zaměřením level designu je environment design, který se soustředí výhradně na tvorbu základního světa, prostředí a atmosféry. Osobě zabývající se tímto odvětvím se říká environment artist. Pro výše uvažovanou válečnou hru z prostředí Belgie by se jednalo zejména o vytvoření 3D prostředí podle skutečnosti (kopce, cesty, rokliny, hory, nížiny, řeky…), mapování textur (skála, tráva, hlína…), tvorbu přírodních živlů (déšť, vichřice…) a osazení základními objekty (stromy, vegetace, budovy…).

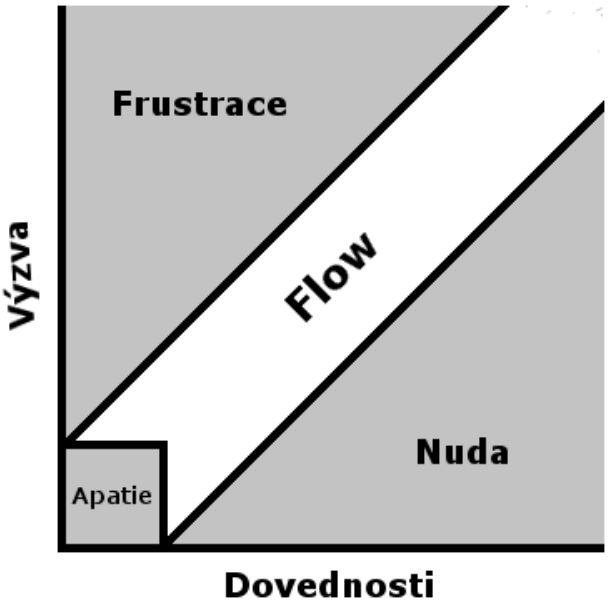
Zdroje [3], [4], [21].

4.2 Flow

Pojem flow zavedl maďarský vědec a psycholog, Mihaly Csikszentmihalyi, ve svém díle Flow: The Psychology of Optimal Experience. V této publikaci popsal flow jako stav, kdy je osoba ponořena do určité činnosti, při níž se její tělo nebo mysl vzepne k hranicím svých možností ve vědomé snaze dosáhnout něčeho obtížného, co jí za to stojí. Do češtiny je flow často překládáno jako „stav plynutí“.

Každý herní vývojář chce docílit toho, aby jeho produkt vyvolal v hráči právě flow.

Mezi faktory, které toto udávají, patří například absolutní koncentrace hráče, oproštění se od reality, subjektivní vnímání času nebo vnitřní pocit odměny a uspokojení.



Obrázek 8: Flow

Vývojář musí proto při tvorbě hry brát v potaz i teorii flow. Zejména pokud se jedná o balancování náročnosti hry. Příliš těžké výzvy prezentované hráči, který ještě nemá adekvátní dovednosti k překonání problému, mohou vést k frustraci a úzkosti. Naopak příliš jednoduché překážky vedou k pocitu nedostatečného uspokojení a nudy, viz obrázek 8.

Dosažení flow může být podporováno i různými prvky v designu hry. Pokud se ve hře Doom (2016) hráč dostane na kritickou hranici života, jeho odolnost se mírně zvýší a má tak větší šanci se na poslední chvíli zachránit. Tato metoda podporuje hráčovo vnoření do hry a probouzí v něm pozitivní pocit, že svými schopnostmi dokázal překonat situaci, která se už jevila jako téměř bezvýchodná.

Zdroje [4], [21], [22].

4.3 Zásady level designu

Následujících deset principů správného level designu prezentoval na konferenci Game Developers Conference [4] v roce 2013 veterán herního průmyslu Dan Taylor ze studia Square Enix [5].

Správný level design je podle Taylora takový, který:

- **Je zábava prozkoumávat -** jasným vizuálním způsobem vede hráče po primární herní cestě a zároveň v nich budí zájem prozkoumávat sekundární cesty, skryté lokace a různá bludiště.

- **Nespoléhá pouze na slova k vyprávění příběhu** - kromě explicitního vyprávění líčeného příběhem a herními cíli poskytuje také implicitní vyprávění skrze prostředí a nabízí hráčům možnost svými rozhodnutími probíhající příběh ovlivňovat a vytvářet si tak vlastní.

- **Říká hráči, co má dělat, ale ne jak to má dělat** - zajišťuje jasné sdělení jednotlivých cílů, ale umožňuje hráčům, aby je dokončili, jakýmkoli způsobem uznají za vhodné, a, pokud je to možné, v jakémkoli pořadí.

- **Konstantně učí hráče něco nového** - udržuje hráče vnořeného do hry pomocí neustálého představování nových mechanik. Zároveň ale zabraňuje starým mechanikám, aby se postupně vytrácely, a představuje jejich modifikace nebo využití neobvyklými způsoby

- **Je překvapivý** - klasický Aristotelův tří-aktový přístup většinou není vhodný pro interaktivní médium, jako jsou počítačové hry, a také nestačí pouze použít model "horské dráhy" pro všechny levely. Dobrý level design je překvapivý a nebojí se riskovat s tempem, estetikou, lokalizací a dalšími prvky, které vytvoří čerstvý herní zážitek.

- **Zplnomocňuje hráče -** hry slouží k úniku z reality a jako takové by se měly vyvarovat banalit. Dobrý level design navíc posiluje hráčův pocit síly a kontroly tím, že mu umožní prožít důsledky svých činů, a to jak v okamžitém herním průběhu a v jeho jednotlivých, na sebe navazujících momentech, tak v dlouhodobém hledisku, skrze holistický pohled na level/hru jako celek.

- **Umožňuje hráči kontrolovat obtížnost** - hlavní herní cesta se zaměřuje na průměrné hráče, a proto je důležité pokročilejším hráčům nabídnout volitelnou výzvu s jasně definovanými riziky a s nimi související odměnou.

[4] GDC je největším každoročním setkáním profesionálních vývojářů her pořádaným v San Franciscu
[5] studio zodpovědné za tituly jako Deus Ex, Final Fantasy nebo Tomb Raider

- **Je efektivní** - zdroje nejsou neomezené, a proto je důležité využívat modularitu k budování herního světa a prezentovat volitelné bonusové cíle k vyplnění celého herního prostoru.

- **Vytváří v hráči emoce** - proces vytváření emocí začíná definováním požadované emoční odezvy a pokračuje výběrem vhodné mechaniky, příběhové části, prostředí a způsobu vyprávění, které tuto odpověď vyvolají.

- **Je řízený herními mechanikami** - správný level design především předvádí herní mechaniky prostřednictvím levelu jako media a posiluje tak jedinečnou interaktivitu videohry.

Zdroje [23], [24].

4.4 Ukázky a příklady

V této podkapitole je předveden level design v praxi a na reálných příkladech je vysvětleno, co je správně a co nikoliv. Jako zdroj informací slouží osobní zkušenosti hráčů a vývojářů.

4.4.1 Kladné příklady

Odpověď na otázku „Jak vypadá správně navržená výzva?“ nalezneme například v populární MMORPG hře World of Warcraft [6]. Zdejší souboje s bossy jsou toho ukázkovým příkladem. Hráč se k jednotlivým bossům propracovává postupně, přičemž prezentovaná obtížnost je s každým dalším bossem v určitém smyslu těžší a hráč musí uplatňovat různé zkušenosti a předměty, které hraním nasbíral. Tímto se předchází situaci, ve které by hráč věděl, že nemá šanci aktuální výzvu překonat. Pokud tedy z nějakého důvodu neuspěje, ví, že chyba se stala na jeho straně a musí pro příště něco zlepšit.

[6] Hra vznikla v roce 2004 a díky pravidelným rozšířením a úpravám je stále extrémně populární

Obrázek 9: World of Warcraft [7]

Samotný souboj s bossem (obrázek 9) správně vizuálně prezentuje hráčům jednotlivé události, na které mohou reagovat. Například se do určité barvy zbarví místo na zemi, kam za pár vteřin spadne meteorit, tudíž by na něm nikdo neměl stát. Pokud se hráč s touto výzvou setkává poprvé a instinktivně neustoupí, zemře, ale při příštím pokusu už ví, co by měl v takové situaci udělat. Ne vždy je to ovšem takto jednoduché a souboje s bossy vyžadují důkladný trénink, přípravu a spolupráci všech zúčastněných spolubojovníků. Hráč má následně pocit, že se ve hře zlepšuje, a překonání výzvy a získání odměny v něm vzbuzuje pocit radosti a uspokojení. Dochází také k utužování vztahů mezi jednotlivými hráči, kteří se na překonání výzvy společně podíleli.

Úplným opakem se může jevit kupříkladu hra Cat Mario, která je záměrně navržena tak, aby v hráči vzbuzovala pocit frustrace a hněvu. Jedná se o 2D plošinovou hru, ve které jsou jednotlivé překážky (a nepřátelé) neviditelné, a objeví se až v momentě, kdy má hráč nulovou nebo minimální šanci na ně reagovat. Při dalším průchodu už hráč o překážce, na které ztroskotal, ví, ovšem vzápětí se okamžitě objeví nová, která ho opět donutí začít znovu. Hra spoléhá na lidskou zarputilost a odhodlání dokončit i něco, co nás postupně dohání k šílenství.

[7] Obrázek převzat ze zdroje [25]

Odměnou za vytrvalost je hráči obrovský pocit radosti a uspokojení po dokončení aktuální mise či jednotlivých výzev.

Další z velice povedených ukázek level designu můžeme nalézt ve hře Resident Evil 4. Úvodní sekvence zavádí hráče do vesnice, která je navržená přesně pro potřeby hráčova vnoření se do hry. Jednotlivé budovy poskytují pouze dočasný a nedostatečný úkryt před vesničany, kteří jsou schopni vyrazit dveře či prolézt do domu oknem. Hráč je tak nucen být neustále v pohybu, procházet různé budovy, hledat zbraně, náboje a hlídat blížící se nepřátele. Je to těžké, ale ne frustrující, protože díky správnému návrhu prostředí je hráč schopen účinně se vyhnout všem nepřátelům.

Speciální důraz na design jednotlivých map kladou zejména kompetitivní akční hry. V titulech jako Counter Strike nebo Call of Duty, ve kterých proti sobě stojí dva pětičlenné týmy na jedné mapě, je level design jeden z nejdůležitějších aspektů hry. Každá mapa musí být dokonale navržena, aby výrazně nezvýhodňovala ani jednu stranu, poskytovala možnost mnoha variabilních scénářů, herních strategií, taktik, produkovala divácky atraktivní momenty a pro hráče byla zábavná.

4.4.2 Záporné příklady

RPG hry Skyrim se ve světě prodalo přes 30 milionů kopií a jedná se o nesmírně úspěšný a mezi hráči oblíbený titul. I tady můžeme ale nalézt příklad špatně provedeného level designu. Hra se pyšní zejména svým otevřeným světem, příběhem a hratelností. Spousta hráčů si ovšem stěžuje na nedostatečnou variabilitu většiny podzemních lokací (tzv. dungeonů), které jsou příliš lineární, postrádají kreativní myšlenku a často se opakují. Vývojáři se v tomto případě zaměřili spíše na kvantitu na úkor kvality. Ve výsledku se jedná pouze o malý prohřešek, který ovšem může u některých hráčů překazit vnoření do hry, či dokonce zkazit jejich chuť titul dokončit.

Dalším příkladem, který se objevuje ve spoustě her, jsou mise a úkoly s cílem eskortovat NPC (Non Player Character) z bodu A do bodu B a přitom nedopustit jeho smrt. Problémů s takovou misí je hned několik. Hráč ztrácí určitou kontrolu nad průběhem mise a musí se přizpůsobovat tempu eskortované postavy. Hlavním problémem bývá převážně umělá inteligence NPC, která se většinou bezmyšlenkovitě vrhá vstříc nebezpečí a dostává tak hráče do nepříjemných situací, nebo se naopak pohybuje příliš pomalu a hráč se chvílemi nudí. Nejhorší provedení těchto misí nalezneme ve většině MMO her, kde pro programování jakékoliv umělé inteligence nejsou

vyhrazeny zdroje a eskortovaná postava jedná čistě podle předskriptovaného plánu, což může být pro hráče velice frustrující.

Dalším z level designerských hříchů jsou neviditelné překážky a nesmyslné hranice herního prostoru. V mnoha hrách se objevují případy, kdy je hratelná zóna jednoduše ukončena neviditelnou zdí nebo překážkou, kterou by hráč kdekoliv jinde v herním světě dokázal překonat – jako příklad si můžeme představit nízký plot ve hře Call of Duty 2, který nelze přeskočit i přes fakt, že hráčova postava dokáže vyskočit mnohem výše, než je nejvyšší viditelný bod plotu. Hráč si v takovém případě okamžitě uvědomí, co se děje, a jeho herní zážitek může být lehce poškozen.

5 VLASTNÍ HRA A ŘEŠENÉ PROBLÉMY

Tato kapitola se zabývá praktickou částí této práce a popisuje hru, která jako její součást vznikla. Obsahuje také ukázky zdrojových kódu a řešených problémů.

5.1 Pitch

Jedná se o poctivou 2D plošinovou skákací hru ze scifi prostředí. Grafické zpracování se řadí do kategorie pixel art a navozuje zajímavou herní atmosféru. Příběh zavádí hlavního hrdinu na strastiplnou cestu s cílem osvobodit svojí lásku ze spárů neznámého tvora. Během svého putování se potýká s nelehkými překážkami a hlavolamy.

5.2 Jednotlivé levely

Hra disponuje pěti levely, včetně tutorialu, kterými hráč postupně prochází. Výzvy před něj předkládané jsou s každým následujícím levelem složitější. Grafická podoba prostředí a navozená atmosféra se po celou dobu hry nemění.

Tutorial

Jako tutorial se označuje nultý level, kde se hráč seznamuje s ovládáním, stylizací hry a jejím příběhem. Před hráče jsou v tomto levelu postaveny jednoduché výzvy, které souvisí pouze se základním ovládáním a snaží se ho naučit ovládat herní postavu. Na začátku levelu se hráč může seznámit i s vlastní smrtelností. Pokud se mu nepodaří přeskočit propast, spadne do vody, utopí se a musí začít znovu.

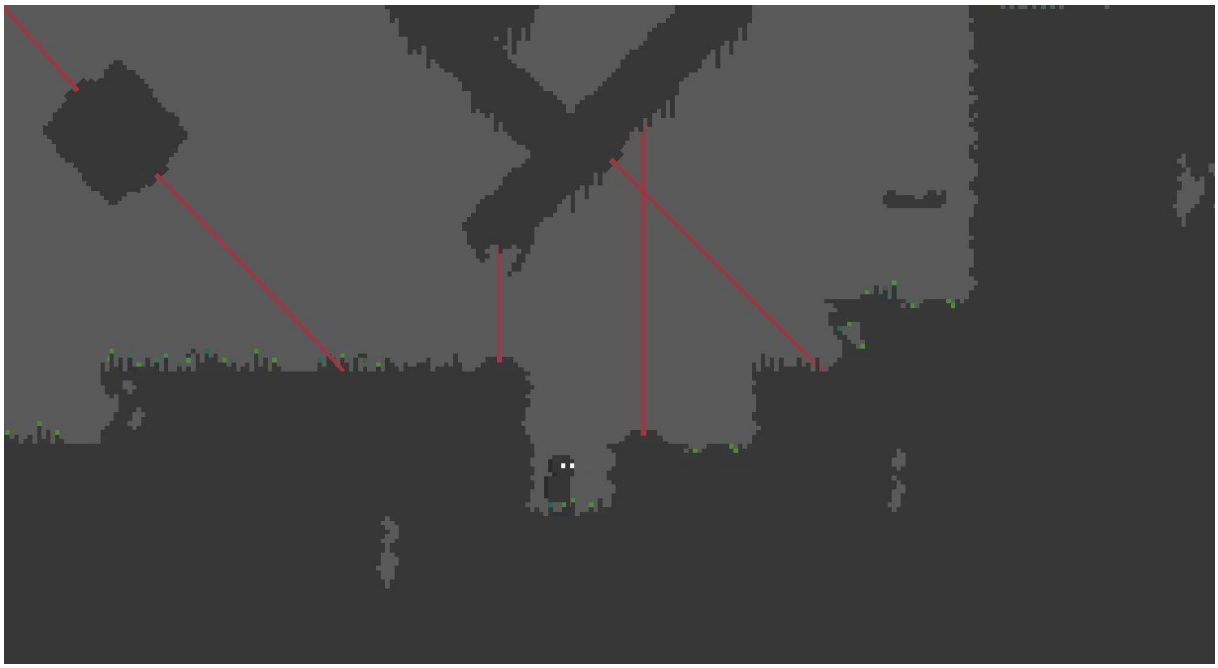
Level 1

V prvním levelu jsou hráči představeny další základní mechaniky, které musí využívat ke zdárnému průchodu hrou i v dalších levelech. V prostředí se objevují první interaktivní objekty a pohyblivé objekty. V návaznosti na tutorial tento level předpokládá hráčovu schopnost se v herním světě úspěšně pohybovat a vybízí ho, aby se v ní ještě zlepšil, prostřednictvím pohyblivých a rotujících platforem.

Level 2

Druhý level se zaměřuje na hráčovu smrtelnost. Nepřítelem jsou mu v tomto případě lasery (obrázek 10), se kterými se setkává poprvé. Svojí červenou barvou v hráči budí podezření. Variabilita laserů dává tomuto levelu dynamický rozměr a hráče nutí být neustále ve střehu a

rychle plánovat. Poprvé se zde objevuje tzv. checkpoint, což je místo, ke kterému když se hráč dostane, uloží se jeho pozice a pokud následně zemře, neobjeví se na začátku, nýbrž na posledním dosaženém checkpointu. Tento systém se využívá u delších levelů, kde je velká pravděpodobnost, že hráč v určitý moment zemře, a začínat stále od začátku by v něm mohlo způsobit nechuť k dalšímu postupu hrou a dokončení titulu.



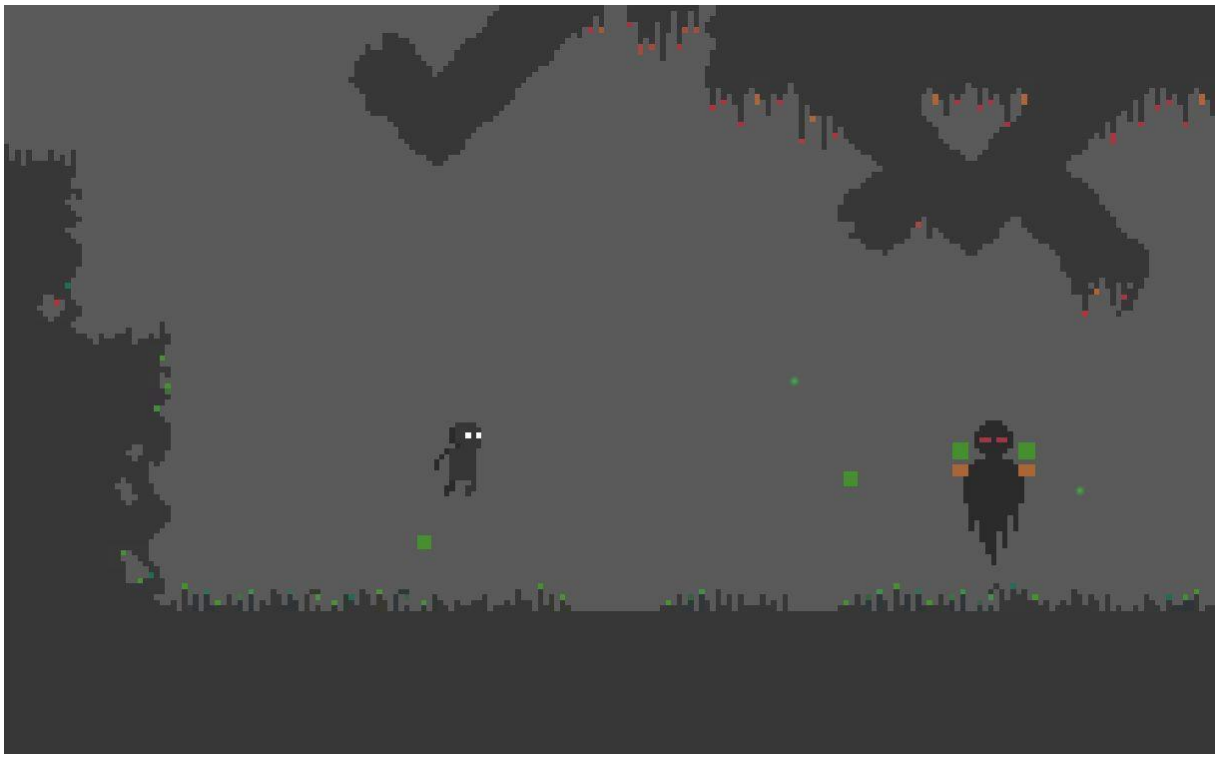
Obrázek 10: Snímek ze hry

Level 3

Předposlední level kombinuje prvky předchozích a navíc přidává další element, před kterým se hráč musí mít na pozoru. Nový druh laseru, který od určité vzdálenosti sleduje hráče a střílí po něm. Dále je zapotřebí vyřešit hlavolam a aktivovat tři objekty, které společně spustí vír, který hráči umožní další postup.

Level 4

Poslední level přináší hráči finální rozuzlení a zakončuje herní zážitek. Kombinuje herní prvky z předchozích levelů a předpokládá precizní dovednost v ovládání hlavní postavy. Na konci tohoto levelu se hráč utká s bossem (obrázek 11), který představuje herní klimax a finální výzvu, kterou je třeba překonat pro zdárné dokončení hry a završení příběhu. Pokud boss v některé ze svých fází hráče usmrtí, při dalším pokusu o jeho zdolání se začíná opět první fází. Boss disponuje čtyřmi fázemi, mezi kterými postupně přechází. V první fázi vysílá na hráče střely v rozmezí 2 vteřin. Hráč má tak dostatek času zareagovat a střele se vyhnout nebo jí přeskočit. V druhé fázi boss vzletí do vzduchu nad hráče a vysílá střely postupně v úhlu mezi 180 stupni až 360 stupni. Hráč si tak musí najíst místo, kde ho střela nezasáhne, a nebezpečí se tak vyhnout. V třetí fázi se hráč musí v pravidelném intervalu vyhýbat padajícím meteoritům, které boss sesílá ze vzduchu na zem. V poslední fázi se boss stává zranitelným a hráč ho musí zneškodnit. Po bossově usmrcení se otevřou dveře do místnosti, kde se hlavní hrdina znovu shledává se svojí láskou, a příběh šťastně končí.



Obrázek 11: Snímek souboje s bossem

5.3 Historie žánru

Počátky žánru plošinových 2D her sahají až k počátku osmdesátých let minulého století, kdy se první hry tohoto typu začaly vyskytovat na automatech a postupně získávaly na popularitě. Průkopníkem žánru se stala hra Donkey Kong z roku 1981, která jako první umožnila hráči přeskakovat překážky a skákat z jedné platformy na druhou. O 4 roky později spatřila světlo světa hra, která zpopularizovala žánr a stala se jednou z nejprodávanějších videoher vůbec. Jedná se o titul Super Mario Bros pro konzoli NES (Nintendo Entertaining System), který měl podle Guinessovy knihy rekordů v roce 1999 na kontě přes 40 milionů prodaných kopií. V začátcích devadesátých let minulého století byly plošinové hry považovány za nejžhavější zboží a vznikalo mnoho kvalitních her tohoto žánru. Za zmínku stojí i další populární a mezi hráči oblíbené značky: Sonic the Hedgehog, Prince of Persia a Jazz Jackrabbit. S příchodem stále se zlepšující 3D technologie začal zájem o 2D plošinové hry opadat. Naštěstí pro tento žánr se spoustě titulů podařilo adaptovat a zasadit základní prvky do trojrozměrného světa. Jako příklad můžeme uvést velice populární hru studia Naughty Dog z roku 1996, Crash Bandicoot.

Obrázek 12: Super Mario Bros [8]

V současnosti je na herním trhu největší poptávka po akčních 3D titulech, a proto je tento žánr oblíbený hlavně mezi nezávislými vývojáři, kteří se nebojí experimentovat a přinášet nové a unikátní nápady. Velice často také vznikají nová pokračování slavných her z minulosti, například Rayman Origins, New Super Mario Bros, Donkey Kong Country Returns a další, zejména pro moderní herní konzole (Wii U, 3DS…).

Zdroj [26].

5.4 Inspirace

V této podkapitole jsou uvedeny některé hry, které sloužily jako inspirace k vytvoření této práce, a zejména její praktické části.

Limbo je 2D logická plošinová hra z roku 2011. Vyniká zejména svým originálním grafickým provedením (obrázek 13). Ponurý temný svět v kombinaci se zajímavou ambientní hudbou na pozadí navozuje unikátní atmosféru a logické hádanky zaměstnají nejednoho zkušeného hráče.

[8] Obrázek převzat ze zdroje [27]

Obrázek 13: Limbo [9]

Super Meat Boy je populární obtížná plošinová hra, která staví na základních prvcích tohoto žánru – běh a skok. Obsahuje přes 300 levelů a poskytuje tak velké množství kvalitních a detailně promyšlených výzev.

Fez je originální titul, ve kterém je herní 2D svět vyobrazen jako jedna ze čtyř stran 3D světa. Hráč otáčí mezi těmito čtyřmi 2D pohledy (obrázek 14), což různě upravuje platformy a umožňuje mu tak vyřešit herní hádanky a postupovat dál. Cílem hry je záchrana vesmíru.

[9] Snímek ze hry

Obrázek 14: Fez [10]

Ori and the Blind Forest z roku 2015 je v jádru poctivá a obtížná plošinová hra vytvořená v Unity enginu. Zaujme zejména svojí audiovizuální stránkou a propracovaným emotivním příběhem. Na prestižním udílení herních ocenění The Game Awards získala hra cenu za nejlepší umělecký směr.

5.5 Profilace

Vytvořená hra se od ostatních odlišuje hlavně svojí vizuální prezentací a jednoduchostí. Společné prvky jako je běh, skok a řešení hádek jsou základními stavebními kameny žánru, na kterých budují všechny podobné hry. Odlišností může být absence bodů života, které se ve většině her tohoto žánru objevují.

5.6 Technické řešení

V této podkapitole je předvedeno a vysvětleno několik částí různých skriptů. Vlastní třídy a funkce, které nejsou součástí enginu, jsou v textu označeny kurzívou.

[10] Obrázek převzat ze zdroje [28]

V následující ukázce zdrojového kódu 1 je vyobrazena funkce *Update* třídy *Laser*, která je volána při každém snímku. Důležitým prvkem je třída Physics2D a její funkce Raycast, kterou můžeme chápat jako paprsek vyslaný z určitého bodu libovolným směrem. Jakýkoliv objekt, který se s paprskem střetne, může být detekován a zaznamenán. Metoda takový objekt vrací jako instanci třídy RaycastHit2D. Pokud je objekt zasažen paprskem a zároveň se jedná o objekt s tagem Player, přičemž splňuje i další podmínky, je usmrcen metodou *DieWithSplash* a jeho stav se nastaví jako neaktivní.

```
void Update ()
{
    hit = Physics2D.Raycast (transform.position,transform.up,
          distance, hitMask);
    if (hit.collider.tag == "Player" && lineRenderer.enabled==true
        && FindObjectOfType<LevelManager> ().playerIsDead == false){
        try{
            hit.collider.gameObject.GetComponent<Main_Char> ()
            .DieWithSplash ();
            FindObjectOfType<LevelManager> ().playerIsDead = true;
        }catch(UnityException e){}
    }

    laserHit.position = hit.point;
    lineRenderer.SetPosition (0, transform.position);
    lineRenderer.SetPosition(1, laserHit.position);
}
```

Zdrojový kód 1: Funkce Update třídy Laser

Na konci funkce probíhá vykreslování laseru do herního světa, pomocí komponenty třídy Line Renderer. Metoda SetPosition má v tomto případě dva parametry, index bodu a jeho pozici v prostoru.

Ve zdrojovém kódu 2 je zobrazena funkce *OnTriggerEnter2D*, která je volána v případě kolize objektu s jiným objektem, který disponuje komponentou Collider se zapnutou vlastností Trigger. V takovém případě se objekt předává v parametru odkazem na jeho komponentu třídy Collider2D. Pokud je kolidující objekt hráč, tak funkce ve hře způsobí vystřelení hráčovy postavy do vzduchu předem stanovenou silou, využívajíc funkce *PlayerJump* třídy *Main_Char*, a zároveň spustí animaci trampolíny skrze komponentu Animator a její instanci pomocí metody SetTrigger.

```
void OnTriggerEnter2D(Collider2D col)
{
   if (col.tag == "Player"){
       col.gameObject.GetComponent<Main_Char>().PlayerJump(jumpStrenght);
       anim.SetTrigger("Trigger");
   }
}
```

Zdrojový kód 2: Funkce OnTriggerEnter2D ve třídě JumpTrampoline

Zdrojový kód 3 obsahuje funkci *FixedUpdate*, která je volána pokaždé, když fyzikální engine provede nějakou akci, například pohyb objektu. V tomto případě se jedná o funkci třídy *CameraFollow*, která slouží k plynulému pohybu kamery a jejímu zaměření na hráčovu postavu. Důležité je využití struktury Vector3, která slouží k reprezentaci 3D pozice, směru a obsahuje funkce pro základní vektorové operace. Na konci metody je vidět použití její statické metody Lerp, která lineárně interpoluje mezi dvěma vektory a slouží k pohybu objektu postupně mezi těmito dva body. Důležitá je také vlastnost normalized objektu struktury Vector3, která vrací vektor o stejném směru, ale délce 1.

```
void FixedUpdate()
{
    if (target)
    {
        Vector3 posNoZ = transform.position;
        posNoZ.z = target.transform.position.z;

        Vector3 targetDirection=(target.transform.position - posNoZ);

        interpVelocity = targetDirection.magnitude * 5f;

        targetPos = transform.position +
        (targetDirection.normalized * interpVelocity * Time.deltaTime);
        targetPos.y += YoffsetUp;

        transform.position = Vector3.Lerp(transform.position,
                                          targetPos + offset, 0.25f);
    }
}
```

Zdrojový kód 3: Funkce FixedUpdate třídy CameraFollow

Výsledkem je plynulý pohyb kamery s malým odsazením směrem vzhůru po ose Y. Kdykoliv se hráč rozeběhne nebo vyskočí, kamera se vydává za ním po jeho trajektorii.

```
public void DieWithSplash()
{
    this.GetComponent<PlayerPushPull> ().DropOnDeath ();
    this.isPushing = false;
    this.GetComponent<SpriteRenderer> ().enabled = false;
    this.gameObject.tag = "Untagged";
    this.gameObject.layer = LayerMask.NameToLayer ("Default");
    Instantiate (deathSplash, transform.position,
                              Quaternion.identity);
    StartCoroutine (levelManager.loadLastCheckpoint());
}
```

Zdrojový kód 4: Funkce DieWithSplash sloužící k usmrcení hráčovy postavy

V ukázce zdrojového kódu 4 je zobrazena funkce *DieWithSplash* třídy *Main_Char*, která slouží k usmrcení hráčovy postavy v případě, že je zabit jinak, než utopením. V případě, že hráč zrovna táhne nějaký objekt, volá se metoda *DropOnDeath* a nastavují se příslušné hodnoty pro usmrcení hráče. Důležitá je metoda Instantiate, která slouží k vytvoření instance objektu, kterým je v tomto případě prefab *deathSplash*, který se skládá z komponenty Transform, skriptu pro zničení objektu po t čase a komponenty Particle System, která vytvoří několik červených částic a vystřelí je do různých směrů a simuluje tak efekt rozstřelení hráče na kusy. Jako parametr se předává objekt, ze kterého chceme vytvořit kopii v herním světě, pozice na které se má objevit a rotace objektu, reprezentovaná strukturou Quaternion. Statická vlastnost identity odpovídá žádné rotaci. To znamená, že objekt bude sladěn s osami nadřazeného objektu nebo herního světa.

Zdrojový kód 5 představuje funkci *Update* třídy *ShootingLaser*. Opět je zde použita funkce Raycast třídy Physics2D. V tomto případě je paprsek zaměřen přímo na hráče, ale je omezen volitelným dosahem. Pokud je hráč v dosahu začátku laseru a objekt, který střílí má na hráče přímou viditelnost, vytvoří se objekt podle předem připraveného prefabu a udělí se mu rychlost a směr na hráčovu postavu. Pokud objekt koliduje s hráčovou postavou, usmrtí ho. Není zde použit normalizovaný vektor a to z důvodu, aby měl hráč vždy stejnou šanci reagovat na blížící se střelu. Pokud je tedy od bodu výstřelu v maximální vzdálenosti, střela bude mít velkou rychlost, ale hráč ji uvidí včas a má dostatek možností jak zareagovat. V případě, že je hráč v bezprostřední blízkosti objektu, který laserové střely vytváří, je rychlost střely poměrně malá a hráč tak má stále možnost zareagovat a nebezpečí se vyhnout. Po výstřelu se spouští metoda

StartCoroutine. Funkce *CanShoot* je takzvaná coroutine funkce, jejíž hlavní předností je, že může pozastavit sebe sama a zde slouží k přepínání proměnné shoot v předem daném intervalu.

```
void Update ()
    {
    hit = Physics2D.Raycast(new Vector2
(startLaser.transform.position.x,startLaser.transform.position.y),
 player.transform.position - startLaser.transform.position, range);
    if (hit != null && hit.collider != null &&
        hit.collider.tag == "Player")
    {
        isVisibleAndInRange = true;
    } else {
        isVisibleAndInRange = false;
    }

    if (isVisibleAndInRange && shoot)
    {
        GameObject projectile = Instantiate(projectilePrefab,
            (Vector2)startLaser.transform.position *
            startLaser.transform.localScale.x, Quaternion.identity);

        projectile.GetComponent<Rigidbody2D> ().velocity =
        new Vector2(shotVelocity*
        (player.transform.position.x - startLaser.transform.position.x),
        shotVelocity *  (player.transform.position.y –
        startLaser.transform.position.y));

        StartCoroutine(CanShoot());
    }
    Debug.DrawLine (startLaser.transform.position, hit.point);
}
```

Zdrojový kód 5: Funkce Update třídy ShootingLaser

V ukázce zdrojového kódu 6 je vyobrazena část *FixedUpdate* funkce třídy *PlayerPushPull*, která se stará o tahání a tlačení objektu hráčem. Cíleného efektu je dosaženo pomocí komponenty třídy FixedJoint2D, která spojuje dvě komponenty třídy Rigidbody2D dohromady ve svých kotvících bodech pomocí konfigurovatelné pružiny. Pro táhnutí nebo tlačení musí být hráč v dostatečné blízkosti a držet stisknutou klávesu E. Pokud ji pustí, komponenta FixedJoint2D se nastaví jako neaktivní, což způsobí odpojení objektu. Pouze objekty s tagem "Pushable" se dají hráčem přesouvat.

```
void FixedUpdate ()
{
    RaycastHit2D hit = Physics2D.Raycast (ruka.transform.position,
        Vector2.right * transform.localScale.x, distance,
        boxMask);
if (hit.collider != null && hit.collider.tag=="Pushable" &&
    Input.GetKeyDown(KeyCode.E) && !pushing) {
        box = hit.collider.gameObject;
        box.GetComponent<Rigidbody2D> ().mass = 2;
        box.GetComponent<FixedJoint2D> ().enabled = true;
        box.GetComponent<FixedJoint2D> ().connectedBody =
        this.GetComponent<Rigidbody2D> ();
        setBoxBeeingPushed (true);
        pushing = true;
        this.GetComponent<Main_Char> ().IsPushing = true;
    }
else if(Input.GetKeyUp(KeyCode.E) && pushing) {
        box.GetComponent<FixedJoint2D>().enabled = false;
        setBoxBeeingPushed (false);
        box.GetComponent<Rigidbody2D> ().mass = 100;
        pushing = false;
        this.GetComponent<Main_Char> ().IsPushing = false;
    }
}
```

Zdrojový kód 6: Způsob tahání objektu

Ukázka zdrojového kódu 7 se zaměřuje na část skriptu *OpenDoor*, který slouží k otevření dveří, v tomto případě posunem dveří směrem vzhůru a umožnění tak průchodu do další části levelu.

```
if (start)
{
     if (door.transform.position.y < door.transform.position.y + posun){
          door.GetComponent<Rigidbody2D> ().velocity = new Vector2
               (0f, posun * rychlostOtvirani * Time.deltaTime);
     }
     if (door.transform.position.y >= startPositionY + posun){
          door.GetComponent<Rigidbody2D> ().velocity =
          new Vector2 (0f, 0f);
          start = false;
     }
}
```

Zdrojový kód 7: Otevření dveří

Pokud je splněna podmínka, dveře se otevřou. Splnění podmínky záleží čistě na hráčově interakci, v tomto případě na stlačení příslušné klávesy v blízkosti objektu (graficky reprezentovaného jako páka).

V následující ukázce zdrojového kódu 8 je k vidění část coroutine funkce *BossFight*, která reprezentuje předpřipravený scénář souboje s bossem ve finále hry. Na začátku se boss přesune na svoji první pozici, animace se nastaví do režimu útoku a počká dvě vteřiny, aby měl hráč šanci se rozkoukat a zjistit, co se děje. Následně cyklicky střílí po hráči, který musí rychle reagovat a střelám se vyhýbat. Použit je zde normalizovaný vektor, a to z důvodu větší obtížnosti.

```
while (transform.position.x != spots [0].position.x) {
    transform.position = Vector2.MoveTowards(transform.position,
         new Vector2(spots[0].position.x, transform.position.y), speed);
    yield return null;
}

this.GetComponent<Animator> ().SetBool ("Attack",true);
this.GetComponent<Animator> ().SetBool ("Idle",false);

yield return new WaitForSeconds (2f);

int i = 0;
while (i < firstPhaseShots){
    GameObject shot = Instantiate (projectile,
        hands[Random.Range(0,2)].transform.position,Quaternion.identity);

    Vector2 direction = PlayerTarget.transform.position –
                            this.transform.position;
    direction.Normalize ();
    shot.GetComponent<Rigidbody2D> ().velocity =
                             direction * shotSpeed;

    i++;
    yield return new WaitForSeconds (1f);
}

yield return new WaitForSeconds (2f);

this.GetComponent<Animator> ().SetBool ("Attack",false);
this.GetComponent<Animator> ().SetBool ("Idle",true);


yield return new WaitForSeconds (2f);
```

Zdrojový kód 8: První fáze finálního souboje s bossem

Mezi jednotlivými střelami je prodleva jedna vteřina. Po dokončení střeleckého cyklu se po dvou vteřinách opět mění animace do klidného stavu před začátkem druhé fáze souboje.

ZÁVĚR

Za využití herního enginu Unity jsem vytvořil 2D plošinovou hru, která se v rámci vlastních možností drží zásad správného level designu. Hra je určena pro jednoho hráče a je cílena na PC platformu. Práce by mohla sloužit jako úvodní materiál pro začínající vývojáře bez zkušeností s Unity enginem a vývojem her jako takovým.

Při práci na tomto projektu jsem se potýkal se spoustou problémů, a to zejména z důvodu, že jsem měl s Unity enginem minimální zkušenosti. Proto jsem mnohokrát učinil špatné rozhodnutí a vehnal sám sebe do slepé ulice. Během vývoje jsem řešením problémů získal spoustu nových znalostí, zejména v oblasti fungování Unity enginu, množství jeho funkcí, způsobu fungování nejrůznějších komponent. Také jsem získal cenné zkušenosti pro budoucí projekty.

Pro správné shrnutí teorie ohledně vývoje počítačové hry jsem musel přečíst několik odborně zaměřených knih, které mi rozšířily obzory v oboru tvorby her a utvrdily mě v mém odhodlání profesního rozvoje v tomto odvětví.

Při tvorbě skriptů jsem uplatnil znalosti získané při studiu na škole, zejména z předmětů objektově orientovaného programování, programovacích technik a programování v .NET a jazyce C#, ve kterém jsou veškeré skripty napsány. Na druhou stranu bylo nutné podrobně nastudovat Unity manuál, zejména jednotlivé třídy, jejich funkce a způsoby možných využití.

Na rozšíření stávající hry hodlám nadále pracovat, protože i přesto, že je hra funkční a hotová, délka její hrací doby neodpovídá zavedeným standardům. Malý počet úrovní je dán zejména časovou náročností na jejich tvorbu a v mnoha případech se vytvořený level nemusí herně osvědčit nebo nedosahuje kvalit nastavených ostatními levely. Poté musí být celý, nebo jeho část, předělán. Do hry mám v úmyslu přidat další detailně promyšlené zábavné levely plné logických hádanek a čím dál tím složitějších výzev v podobě nových překážek, pastí, nepřátel s důmyslnou umělou inteligencí a několika unikátních scénářů soubojů s bossy.

POUŽITÁ LITERATURA

[1] PECINOVSKÝ, Rudolf. *Java 7: učebnice objektové architektury pro začátečníky*. Praha: Grada, 2012. Knihovna programátora (Grada). ISBN 978-80-247-3665-5.

[2] PECINOVSKÝ, Rudolf. *Java 8: úvod do objektové architektury pro mírně pokročilé*. Praha: Grada Publishing, 2014. Knihovna programátora (Grada). ISBN 978-80-247-4638-8.

[3] JIRKOVSKÝ, Jan. *Game industry: vývoj počítačových her a kapitoly z herního průmyslu*. Praha: D.A.M.O., 2011. ISBN 978-809-0438-712.

[4] JIRKOVSKÝ, Jan. *Game industry 2*. Praha: D.A.M.O., 2012. ISBN 978-80-904387-3-6.

[5] List of video game genres. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 26 April 2018 [cit. 2018-05-05]. Dostupné z: https://en.wikipedia.org/wiki/List_of_video_game_genres

[6] ENGER, Michael. Game Engines: How do they work?. In: *Giant Bomb* [online]. 20.6.2013 [cit. 2018-05-01]. Dostupné z: https://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529

[7] Asset Workflow. *Unity - Manual* [online]. [cit. 2018-05-01]. Dostupné z: https://docs.unity3d.com/Manual/AssetWorkflow.html

[8] Scripting. *Unity - Manual* [online]. [cit. 2018-05-01]. Dostupné z: https://docs.unity3d.com/Manual/ScriptingSection.html

[9] The Marketing Guide For Indie Game Developers. *PixelProspector - the indie goldmine* [online]. August 30, 2014 [cit. 2018-05-01]. Dostupné z: http://www.pixelprospector.com/the-marketing-guide-for-game-developers/

[10] U.S. computer and video game sales - digital vs. physical 2016. *Statista: The Statistics Portal for Market Data* [online]. [cit. 2018-05-01]. Dostupné z: https://www.statista.com/statistics/190225/digital-and-physical-game-sales-in-the-us-since-2009/

[11] Unity Overview. *Unity - Manual* [online]. [cit. 2018-05-05]. Dostupné z: https://docs.unity3d.com/510/Documentation/Manual/UnityOverview.html

[12] Unity (game engine). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 4 May 2018 [cit. 2018-05-05]. Dostupné z: https://en.wikipedia.org/wiki/Unity_(game_engine)

[13] Learning the Interface. *Unity - Manual* [online]. [cit. 2018-05-05]. Dostupné z: https://docs.unity3d.com/510/Documentation/Manual/LearningtheInterface.html

[14] Scene View. *Unity - Manual* [online]. [cit. 2018-05-05]. Dostupné z: https://docs.unity3d.com/510/Documentation/Manual/SceneView.html

[15] *Unity - Manual* [online]. [cit. 2018-05-05]. Dostupné z: https://docs.unity3d.com/Manual/index.html

[16] *Piskel - Free online sprite editor* [online]. [cit. 2018-05-05]. Dostupné z: https://www.piskelapp.com/

[17] Feature List. *MonoDevelop* [online]. [cit. 2018-05-05]. Dostupné z: https://www.monodevelop.com/documentation/feature-list/

[18] MonoDevelop. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 27 April 2018 [cit. 2018-05-05]. Dostupné z: https://en.wikipedia.org/wiki/MonoDevelop

[19] About. *Audacity* [online]. [cit. 2018-05-05]. Dostupné z: https://www.audacityteam.org/about/

[20] JOHNSTO, David. What is Level Design?. *Johnsto.co.uk* [online]. 30 January 2003 [cit. 2018-05-05]. Dostupné z: https://www.johnsto.co.uk/design/level-design/

[21] KREMERS, Rudolf. *Level design: concept, theory, and practice*. Wellesley, MA: A.K. Peters, 2009. ISBN 978-156-8813-387.

[22] CSIKSZENTMIHALYI, Mihaly. *Flow: o štěstí a smyslu života*. Praha: Portál, 2015. ISBN 978-80-262-0918-8.

[23] TAYLOR, Dan. Ten Principles of Good Level Design (Part 1). In: *Gamasutra* [online]. 09/29/13 [cit. 2018-05-05]. Dostupné z: https://www.gamasutra.com/blogs/DanTaylor/20130929/196791/Ten_Principles_of_Good_Level_Design_Part_1.php

[24] TAYLOR, Dan. Ten Principles of Good Level Design (Part 2). In: *Gamasutra* [online]. 10/06/13 [cit. 2018-05-05]. Dostupné z: http://www.gamasutra.com/blogs/DanTaylor/20131006/197209/Ten_Principles_of_Good_Level_Design_Part_2.php

[25] Illidan, the Betrayer versus Alliance. In: *Dalaran WoW* [online]. 2013-11-02 [cit. 2018-05-05]. Dostupné z: http://www.dalaran-wow.com/media/the-burning-crusade-raids/images/168/illidan-the-betrayer-versus-alliance

[26] Platform game. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 24 April 2018 [cit. 2018-05-05]. Dostupné z: https://en.wikipedia.org/wiki/Platform_game

[27] TROJANOVÁ, Alžběta. Sledujte, jak se dělá světový rekord v Super Mario Bros. In: *Booom.cz* [online]. 2013-11-02 [cit. 2018-05-05]. Dostupné z: https://booom.tiscali.cz/sledujte-jak-se-dela-svetovy-rekord-v-super-mario-bros-je-to-nervy-drasajici-274525

[28] KAHN, Justin. The stellar puzzle-platfomer Fez... In: *9to5Toys* [online]. Dec. 14th 2017 [cit. 2018-05-05]. Dostupné z: https://9to5toys.com/2017/12/14/fez-puzzle-platfomer-ios/

[29] EROKIA. Piano Stab with Bass. In: *Freesound* [online]. April 10th, 2018 [cit. 2018-05-05]. Dostupné z: https://freesound.org/people/Erokia/sounds/424859/

[30] SHADY, Dave. My love (piano loop). In: *Freesound* [online]. October 22nd, 2015 [cit. 2018-05-05]. Dostupné z: https://freesound.org/people/ShadyDave/sounds/325611/

[31] ROSES. I Always Love You. In: *Freesound* [online]. April 7th, 2018 [cit. 2018-05-05]. Dostupné z: https://freesound.org/people/Roses1401/sounds/424427/

[32] UNITY TECHNOLOGIES. Unity Samples: UI. In: *Asset Store* [online]. Sep 9, 2016 [cit. 2018-05-05]. Dostupné z: https://assetstore.unity.com/packages/essentials/unity-samples-ui-25468