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 2D sidescrolling videohry pro operační systém Windows v prostředí Unity

Vypracoval: Daniel Jírovec
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2021

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Daniel JÍROVEC**
Osobní číslo:	**E17442**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Vývoj 2D sidescrolling videohry pro operační systém Windows v prostředí Unity**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je vytvořit 2D herní aplikaci pro systém Windows s použitím herního enginu Unity. Hráč bude pomocí pohybu hlavní postavy překonávat překážky v prostředí hry typu 2D scroller, aby se dostal na konec jednotlivých úrovní. Hra bude disponovat alespoň 10 úrovněmi. Pro vývoj bude využíván sprite workflow pro možnost přidávat animované objekty a grafiku na scénu v prostředí Unity. Pro implementaci herních funkcí bude využíván jazyk C#. Pro grafiku a vizuální stránku herní aplikace mohou být využity další softwarové nástroje (např. Adobe Photoshop, Krita, Unity Asset Store).

Metodický postup:

1. Studium odborné literatury.
2. Návrh a popis vývoje a implementace výsledné aplikace.
3. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
4. Závěr a doporučení.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. DaGraca, M., & Lukosek, G. (2018). *Learning C# 7 By Developing Games with Unity 2017.* Birmingham, UK: Packt.
2. Ferrone H. (2019). *Learning C# by Developing Games with Unity 2019: Code in C# and build 3D games with Unity.* Birmingham, UK: Packt.
3. Godbold A., & Jackson S. (2016). *Mastering Unity 2D Game Development.* Birmingham, UK: Packt.
4. Harpen J. (2018). *Developing 2D Games with Unity: Independent Game Programming with C#.* New York, NY (USA): Apress.
5. Lavieri E. (2018). *Getting Started with Unity 2018: A Beginner's Guide to 2D and 3D game development with Unity.* Birmingham, UK: Packt.

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

Datum zadání bakalářské práce: **25. března 2021**
Termín odevzdání bakalářské práce: **15. dubna 2022**

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

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 25. března 2021

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b 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.

Datum: 14. 4. 2021 Podpis studenta: 

Poděkování

Chtěl bych poděkovat panu Mgr. Radimovi Remešovi, za trpělivost a cenné rady nejen při vedení této bakalářské práce, ale i v průběhu mého studia. Dále chci také poděkovat své rodině, přítelkyni, kamarádům a spolužákům za jejich ustavičnou podporu.

Obsah

1 Úvod

Cílem této bakalářské práce je vytvořit 2D videohru s pomocí herního enginu Unity. Toto téma jsem si zvolil hlavně z toho důvodu, že mě hraní videoher baví, je to koníček, který mě provází již od dětství. Vždycky mě fascinoval proces tvorby čehokoliv. Obzvláště pak vytvoření jakékoliv videohry, motivace jejich tvůrců a efekt, který výsledný produkt má na hlavní konzumenty, na hráče. Nejen, že videohry jsou hlavně zdrojem zábavy, ale dokážou v člověku vyvolat celé spektrum emocí na základě jejich žánrů, uspokojit soutěživost některých jedinců a tvůrcům mohou naplnit touhu po tvoření.

Dlouhou dobu byly videohry považovány za něco, co negativně ovlivňuje život lidí. Za posledních pár let se stigma okolo videoher pomalu rozplývá, i když je pořád přítomné. Vědecké studie prokázaly pozitivní vliv na kritické myšlení, kreativitu a reakce hráčů. Herní průmysl je na vzestupu, a to dokazuje i fakt, že jeho zisky už několik let značně převyšují zisky průmyslu filmového a hudebního dohromady. Z videoher se pomalu ale jistě stává celosvětový fenomén a dostávají se do povědomí lidí pohybujících se mimo herní komunitu. Tento fenomén penetruje jakousi bariéru mezi tradičními médii a videoherním světem, a to značí velký potenciál proto, aby se videohry dostaly na první příčku masové zábavy.

V rámci teoretické části se budu zabývat nejprve základními pojmy, jejichž pochopení je stěžejní pro porozumění komplexnímu postupu vývoje videohry. Poté se budu věnovat samotnému procesu vývoje a popíšu jeho části a průběh, oblasti vývoje a další aspekty s vývojem spojené. Dále uvedu žánry a typy her, s kterými se můžeme setkat. V následující kapitole přiblížím herní engine Unity, vysvětlím jeho podstatu a vliv na vytváření videoher. Nakonec se zaměřím na to, jak mohou hry sloužit jako nástroj pro vzdělávání a předávání informací a uvedu příklady.

Ve zpopulárnění herního průmyslu osobně vidím velkou příležitost právě pro nástup videoher se vzdělávacím účelem a pro zábavné a interaktivní předávání informací. I proto jsem se rozhodl vytvořit 2D videohru typu sidescroller, která svým poselstvím apeluje především na děti a zábavným způsobem je poučuje o problematice plastového znečistění v přírodě a v našem okolí. V praktické části práce popisuji vývojový proces mnou vytvořené videohry.

2 Základní pojmy

2.1 Videohra

Videohra je v základu složení dvou slov, a to video a hra. Hra reprezentuje nějakou činnost člověka, jejímž účelem je především zábava. Video reprezentuje technologii pro zobrazování pohyblivého obrazu. Spojením tedy dostáváme slovo videohra a pod ním si můžeme představit hru, kterou hrajeme pomocí nějakých audiovizuálních prostředků. Synonymními výrazy mohou být také spojení počítačová hra, digitální hra nebo elektronická hra. Lze mezi nimi ale nalézt rozdíly a pojem videohra je mnohem obecnější a svým způsobem zastřešuje všechny zmíněné spojení. (Tavinor, 2008)

Stejně jako videohra i samotný pojem hra má několik různých definicí. Existují ale prvky, které všechny tyto definice spojují a nalezneme je v podstatě ve všech hrách a to jsou:

- konflikt – s oponentem nebo okolnostmi
- pravidla – stanovují, co je a není povoleno
- schopnosti hráče – dovednosti, strategie
- výsledek – výhra nebo prohra, nejvyšší skóre, splnění úkolu

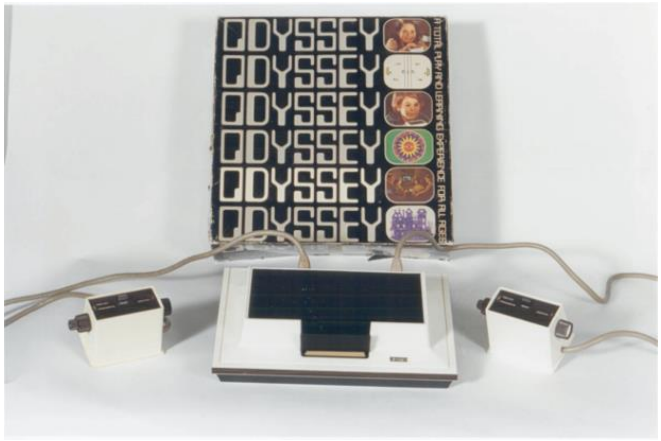
Všechny tyto prvky se ve valné většině případů vyskytují i ve videohrách, ale samozřejmě se mohou do určité míry lišit. Ve videohrách tedy z pohledu toho, co dělá hru hrou, vzniká zvláštní situace, kdy počítač slouží jako soudce toho, zda se hraje podle stanovených pravidel. Současně může ale kontrolovat i hráčova oponenta. (Wolf, 2007)

Dalším důležitým prvkem videohry je samotná identita počítače jako hráče nebo jak uvádí Keith Feinstein (majitel muzea videoher Videotopia), tzv. emocionální prvek. Musí být dosaženo pocitu, jako by hráč soupeřil s podobně schopným člověkem. Počítač musí být více než jen rozhodčí, který ovládá svět videohry, ale i aktivní soupeř. Tímto získáním identity a vytvořením tzv. „jeden na jednoho“ situace, se hráč do hry emotivně ponoří a pociťuje konkurenci, stejně jako by k tomu mohlo dojít ve hře pro dva hráče, kde proti sobě soutěží dva lidé. (Wolf, 2007)

Zatímco pro určení toho, co je a není hra existuje mnoho odlišných kritérií, striktní definice pojmu „video“ je jasně daná použitím analogového signálu, který byl zobrazen na katodové trubici. Tato technologie zvaná CRT, byla typická pro televize nebo počítačové obrazovky a produkovala rastrový obraz. Podle této definice je tedy Ralph Baer prvním člověkem, který použil televizi jako zobrazovací zařízení pro videohru a tvůrcem prvního

domácího herního systému nazvaného Magnavox Odyssey (Obrázek 1), který se na trhu objevil v roce 1972. Populární termín videohra ale nezůstal vázán pouze ke své technické definici. Později se rozšířil i pro odlišné způsoby zobrazování her na obrazovkách a stal se z něj spíše konceptuální pojem. (Wolf, 2007)

Obrázek 1: *První domácí herní systém Magnavox Odyssey*



(Zdroj: Retro Gamer, 2014)

Jak jsem již zmínil, pro označení her, kde hráč provádí interakci s tím, co se zobrazuje na obrazovce či displeji, se někdy možná i častěji používá spojení počítačové hry. Toto označení vzniklo na základě toho, že videohry využívají mikroprocesor, tedy počítač, k jejich fungování a označení je tak výstižné. Ne všechny počítačové hry jsou ale zároveň i videohry. Například desková hra s názvem „Operace“ využívá mikroprocesor k aktivování spínačů a je to tedy také počítačová hra, nicméně ne ve smyslu, ve kterém by si ji většina z nás představila.

2.2 Operační systém

Operační systém je typ softwaru, který spravuje a řídí počítač a poskytuje uživateli logické rozhraní pro přístup k hardwaru a zpracovává úlohy zadané uživatelem. Má mnoho funkcí, které se liší podle typu operačního systému, avšak některé funkce jsou nutné a shodují se v podstatě pro všechny operační systémy. (Lavrinčík, 2018) Mezi hlavní funkce patří:

- Správa paměti – vede evidenci paměti, přiděluje paměti různým procesům a řeší situaci při nedostatku paměti
- Správa procesů – vede evidenci spuštěných procesů, sleduje jejich stav, zajišťuje komunikaci mezi procesy a plánuje přidělování procesoru

- Správa periférií – vytváří rozhraní mezi vstupními a výstupními zařízeními a procesy, sleduje stav zařízení, přiděluje zařízení procesům
- Správa systému – rozlišují se různé režimy práce systému, a to alespoň uživatelský – běžné činnosti, a privilegovaný – údržba, instalace, konfigurace
- Správa souborů – udržuje informace o struktuře souborů na disku, kontroluje přístupová práva procesů k souborům, vytváří rozhraní pro přístup procesů k souborům
- Správa uživatelů – vede evidenci o uživatelích systému a jejich činnostech, přihlašuje a odhlašuje uživatele
- Uživatelské rozhraní - sada programů, které slouží ke komunikaci mezi uživatelem a operačním systémem
- Programové rozhraní – rozhraní mezi programy a operačním systémem, které se označuje jako API a je obvykle tvořeno knihovnami, jež může program využívat pro svou činnost (dialogová okna, grafické prvky rozhraní atd.) (Lavrinčík, 2018)

Operačních systémů je opravdu mnoho. Nejběžněji se s nimi setkáme v osobních počítačích nebo mobilních zařízeních, potažmo také ve spotřební elektronice, serverech nebo mainframe počítačích, což jsou střediskové počítače zpracovávající obrovské množství dat.

Mezi nejznámější operační systémy osobních počítačů patří Windows od společnosti Microsoft a v dnešní době je společně s operačním systémem Android, což je operační systém pro mobilní telefony, nejpoužívanějším operačním systémem vůbec. Mezi další operační systémy patří například Unix a unixové systémy, které jsou na Unixu založeny jako otevřený operační systém Linux a jeho distribuce nebo macOS od společnosti Apple.

2.3 Program a programovací jazyk

Pro pochopení spojení programovací jazyk je nejprve třeba definovat pojem program a algoritmus. Neexistuje jedna jednoznačná, společensky uznávaná definice algoritmu, ale jednoduše lze popsat jako nějaký návod nebo postup pro řešení určitého problému. Tento postup musí disponovat několika vlastnostmi, a to hlavně tím, aby byl jednoznačný a nesmí také umožňovat více výkladů. Příkladem algoritmu v reálném světě může být například recept v kuchařce. Definuje postup přípravy pokrmu a měl by být jednoznačně definovaný a neumožňovat více způsobů výkladů toho, jak tento daný pokrm realizovat. (Bory, 2016)

Pokud si takto určíme definici algoritmu, počítačový program potom můžeme popsat jako zápis algoritmu ve zvoleném jazyce, který je čitelný a srozumitelný pro počítač. Tyto jazyky

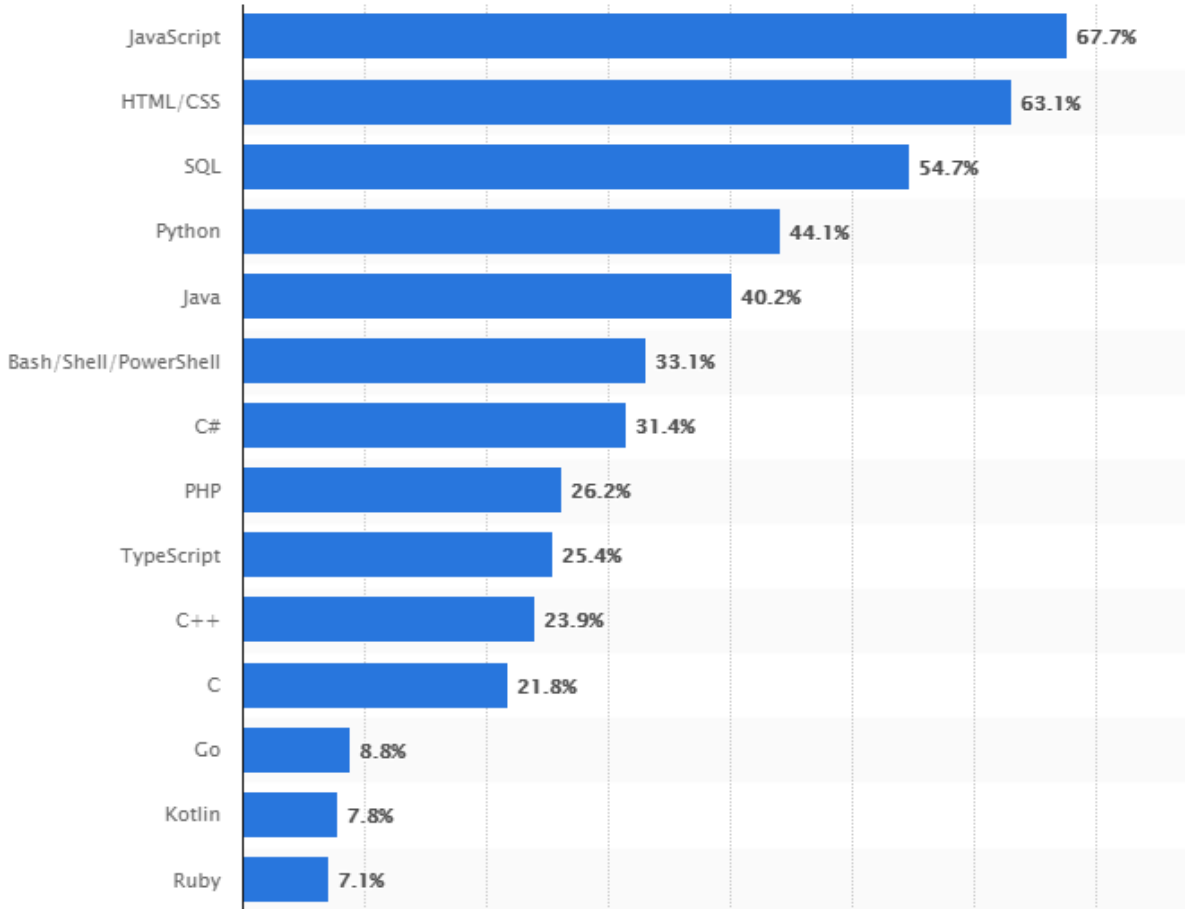
se nazývají programovací jazyky. Jejich účel je komunikace a dorozumění mezi člověkem a počítačem. Programátor tedy pomocí programovacího jazyka popisuje, jakým způsobem má počítač splnit požadovanou úlohu. (Bory, 2016)

Programovacích jazyků je opět velké množství a dělí se hlavně podle jejich míry abstrakce na nižší a vyšší programovací jazyky. Nižší programovací jazyk se svou mírou abstrakce velmi podobá strojovým instrukcím procesoru a na specifickém procesoru je také závislý. Příkladem nižšího programovacího jazyka je Assembler. Naopak zápis algoritmu ve vyšším programovací jazyku se podobá tomu, jak by daný problém zpracoval člověk. Aby mu počítač rozuměl, do strojového kódu se převádí kompilátorem. Příkladem vyšší programovacího jazyka je například C#, s kterým pracuji v rámci praktické části nebo Java či Python. (Blahuta, 2017)

Programovací jazyky se také můžou dělit podle přístupu programování, a to na strukturované programování a objektově orientované programování. Ve strukturovaném programování se algoritmus rozděluje na dílčí úlohy, které se poté spojují v jeden celek. U objektově orientovaného programování je program založený na objektech, které uchovávají informace o sobě a lépe popisují reálný svět. Mají své vlastnosti a schopnosti a jsou schopny samostatně reagovat na události. Program z pohledu objektově orientovaného přístupu je tedy množina vzájemně spolupracujících objektů. (Danel, 2012)

V kontextu mé práce je videohra tedy program a algoritmy napsané programovacím jazykem říkají počítači, co se v ní bude odehrávat. Na následujícím obrázku (Obrázek 2) jsou zobrazeny nejpoužívanější programovací jazyky mezi vývojáři na celém světě. Tato statistika pochází ze začátku roku 2020.

Obrázek 2: *Nejpoužívanější programovací jazyky*



(Zdroj: Statista, 2020)

2.4 Skript

Script je program nebo část programu zapsaná pomocí programovacího jazyka. Tento kód je po sobě jdoucí série příkazů, algoritmus, vykonávající nějaký úkol. Skript je obvykle uložen jako samostatný soubor a dá se upravovat bez potřeby rekompilování hlavní spustitelné části programu.

Skriptování je základní složkou všech videoher. Skripty zajišťují veškeré herní mechaniky, kontrolují dodržování pravidel a starají se například také o to, aby počítač vůbec reagoval na vstup od hráče a události ve hře se odehrávaly správným způsobem a ve správný čas. (Unity Technologies, 2018b)

V herním enginu Unity se ke skriptování používá objektově orientovaný programovací jazyk C# vyvinutý společností Microsoft.

2.5 Herní engine

Game engine nebo herní engine je softwarová aplikace, která poskytuje integrované vývojové prostředí a nástroje pro vizuální vývoj a je navržena speciálně pro konstrukci a vývoj videoher. Umožňuje vývojářům soustředit se výhradně na logiku a design videohry. Stará se o technickou implementaci, umožňuje nastavovat různé aspekty a spravuje assety. (Studytonight Technologies, n.d.-b)

Herní engine má několik hlavních komponent a to jsou:

- Vstup - herní engine poskytuje podporu pro řadu vstupních zařízení, jako je myš a klávesnice nebo ovladač. Existuje několik různých způsobů zpracování vstupu, nejčastěji používané jsou tzv. events nebo události. Vstupní událost je zaregistrována počítačem (například kliknutím pravým tlačítkem myši nebo stisknutím klávesy) a poté se spustí kód na základě toho, jaký vstup byl přijat. (Studytonight Technologies, n.d.-b)
- Grafika - grafika je jedním z nejdůležitějších faktorů. Navrhuje a vytváří se pomocí grafických programů jako například velmi populární Blender nebo Maya. Poté se importují do herního enginu, který poskytuje mnoho dalších funkcí, jako jsou světelné efekty, stíny nebo animace, které napomáhají tomu, aby finální produkt vypadal, co nejlépe. (Studytonight Technologies, n.d.-b)
- Fyzika – pro simulaci fyziky je v herním enginu obvykle další dílčí komponenta, která je známá jako Physics Engine nebo fyzikální engine. Je to softwarová komponenta, která umožňuje provádět relativně přesnou simulaci většiny fyzických funkcí, jako je pohyb těla, změna hmotnosti a rychlosti těles, dynamické chování tekutin, gravitaci, detekci kolizí, rotaci a další. (Studytonight Technologies, n.d.-b; Gregory, 2018)
- Umělá inteligence – implementace umělé inteligence (AI) ve hrách se obvykle provádí pomocí hotových skriptů, které jsou navrženy a napsány programátory, kteří se na umělou inteligenci specializují. AI ovládá mnoho prvků a používá se hlavně k tomu, aby nehumánní postavy (NPC) byly responzivní, adaptivní a svým chováním a inteligencí připomínaly člověka. Umělá inteligence je schopná aktivně měnit úroveň dovedností hráčova oponenta (počítače) na základě toho, jaké má hráč schopnosti. Díky tomu se hráči videohra zdá více přizpůsobená. (Mathur, 2018)
- Zvuk – tato komponenta se používá k ovládání zvukových efektů a hudby. Zvuk ve videohrách obvykle nevychází z reproduktorů tak, jak byly zaznamenán. Většina herních enginů má prostředky k umisťování zvuků do 3D světa. Díky funkcím enginu

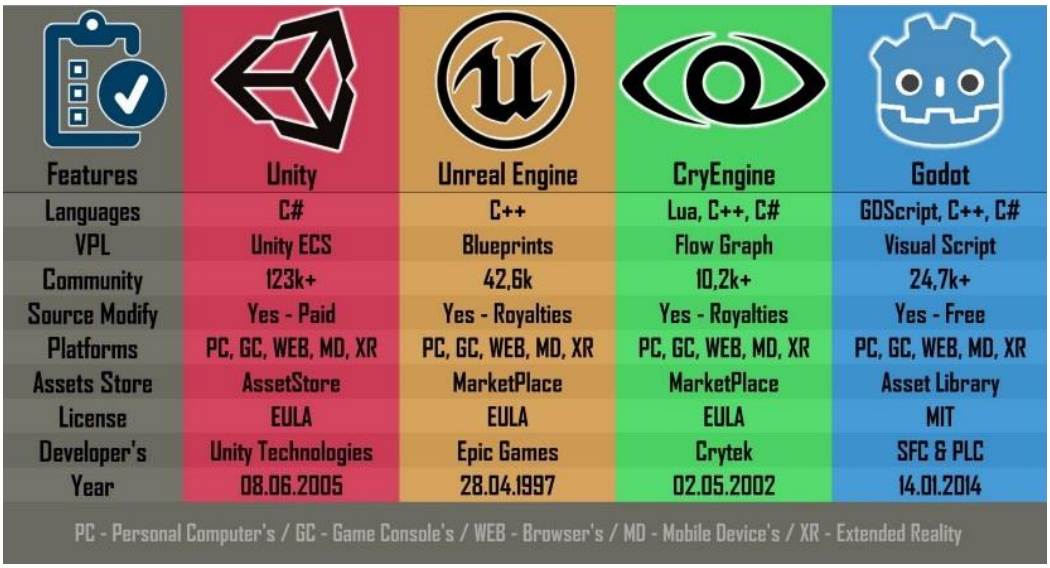
upraví hlasitost podle toho, kde se hlavní postava pohybuje vzhledem ke zdroji zvuku. Realismus zvuku se dá také ovlivnit upravením výšky tónu nebo dozvuku a dalších funkcí, které herní enginy poskytují. (Studytonight Technologies, n.d.-b)

- Síťové připojení – online aspekt hry je v dnešní době skoro samozřejmostí. Většina herních enginů poskytuje úplnou podporu pro tyto požadavky na hru více hráčů a nabízí mnoho dalších dílčích komponent a hotových skriptů k řešení síťového připojení k serverům. (Studytonight Technologies, n.d.-b; Gregory, 2018)

Herních enginů je na trhu velké množství a liší se svojí cenovou dostupností, podporovanými programovacími jazyky, podporovanými operačními systémy, jednoduchostí používání, možností tvořit jen 2D nebo i 3D videohry a dalšími aspekty. Tyto herní enginy jsou k dispozici veřejně a fungují na bázi předplatného nebo jednorázového poplatku. Spoustu herních vývojářských studií ale vyvíjí hry ve svých vlastních herních enginech, které byly vytvořeny specificky pro konkrétní herní studio a jeho zaměstnance. Tyto enginy nejsou veřejně dostupné.

Na následujícím obrázku (Obrázek 3) jsou zobrazeny nejpopulárnější herní enginy, které jsou veřejně dostupné a jejich vlastnosti.

Obrázek 3*: Nejpopulárnější veřejně dostupné herní enginy*



(Zdroj: IRONKAGE, 2020)

2.6 Asset

Asset je jakákoli položka nebo objekt, který lze při tvorbě videohry použít. Jedná se například o obrázek, texturu, zvuk nebo animaci. I skript může být asset. Tento asset může být vytvořen mimo herní engine v příslušném programu a poté nahrán do projektu. Určité assety lze vytvořit i přímo v herním enginu. Jedním ze zvláštních případů assetu v herním enginu Unity, v kterém budu v rámci praktické části pracovat, je tzv. Prefab. Prefab je asset, který lze vytvořit z jakéhokoliv objektu, který už má přidané nějaké vlastnosti a komponenty a chceme ho použít vícekrát. Pokud bychom chtěli náhle změnit vlastnosti daného objektu, stačí je jen upravit u tohoto Prefab assetu a změny se automaticky projeví na všech instancích daného objektu ve hře

3 Proces vývoje videohry

Vývoj videohry je velmi složitý a komplexní proces, a to ať se jedná o tvorbu jednoduché tzv. „indie“ videohry, kdy se vývojový tým skládá jen z několika členů nebo dokonce jen jednoho člověka až po vývoj masivních AAA titulů, které jsou financovány a vytvářeny po dobu několika let a podílí se na nich stovky lidí. Termín AAA značí trojmístný rozpočet, který se pohybuje v řádech milionů. Doba a náklady na vývoj se tedy liší v závislosti na rozsahu a ambicích konkrétního projektu. Téměř všechny videoherní produkce ale procházejí koncepční fází, fází návrhu, fází výroby, fází testování a fází postprodukce. (Jirkovský, 2011) V následujících podkapitolách tyto fáze popíšu společně s oblastmi vývoje a s nimi příslušnými týmovými rolemi.

3.1 Oblasti vývoje

U větších projektů v podstatě neexistuje možnost, že by všechny oblasti vývoje zastával jeden člověk, a tak vznikají specializace, jak tomu je i v ostatních odvětvích.

3.1.1 Herní design

Game designer neboli herní designér vytváří hlavní myšlenku, nápad a pravidla, která videohru utváří. Herní designéři budují herní svět a herní mechaniky a dělají zásadní rozhodnutí, které tvarují výsledný produkt a udávají směr, kterým videohra míří. Nemusí vědět všechno o programování, ale musí vědět vše, čeho je programátor schopen dosáhnout. Herní designér je široký pojem, který se skládá z mnoha dalších různých specializací jako designér úrovní, designér soubojového systému, systémový designér, kreativní ředitel a další. Hlavním úkolem designéra je udělat hru zábavnou. Videohra, která není zábavná je selhání herního designu. (Grygar, 2011; Rogers, 2014)

3.1.2 Programování

Programátoři pomocí programovacích jazyků vytváří kód a implementují různé nástroje herního enginu, herní mechaniky do samotné videohry, programují fyziku nebo píšou skripty obstarávající funkcionalitu a hratelnost. Obvykle se dělí na dvě skupiny, a to na tzv. runtime programátory, kteří pracují jak na herním enginu, tak na samotné videohře a poté na programátory nástrojů, kteří pracují s různými nástroji, které jsou pak k dispozici pro vývojový tým. Programátoři celou videohru přivádějí k životu. (Rogers, 2014)

3.1.3 Výtvarný design

Než se vůbec začne na hře pracovat, musí se určit, jak bude vlastně vizuálně vypadat. Výtvarný tým uměleckých designérů v této fázi úzce spolupracuje s hlavním designérským týmem na definování vizuální identity dané videohry. Oblast výtvarného designu zahrnuje několik uměleckých pozic a to hlavně:

- Koncepční umělci - poskytují vývojovému týmu vizi toho, jak bude konečný produkt vypadat. vytvářejí skici a obrazy na základě nápadů a myšlenek hlavního herního designéra. Jsou velmi zásadní v rané fázi vývoje.
- 3D modeláři - vytvářejí trojrozměrný virtuálního herní svět. Tato pozice se obvykle dělí na dvě další, a to modeláře předmětů v popředí videohry, kteří vytvářejí různé assety jako postavy, vozidla, zbraně atd., a poté modeláře předmětů v pozadí videohry, kteří vytvářejí terén, budovy, mosty apod.
- Umělci vytvářející textury - tyto textury jsou aplikovány na povrchy různých trojrozměrných modelů, které vytvoří 3D modeláři.
- Osvětlovači - umisťují zdroje světla do celého herního světa, a to jak statické, tak i dynamické. Upravují barvu, intenzitu a směr světla.
- Animátoři - animují postavy a předměty ve hře.
- Umělci vytvářející UI (uživatelské rozhraní) - navrhují ikony a prvky HUD (heads-up display), který zobrazuje různé informace a údaje o hlavní postavě. (Rogers, 2014)

3.1.4 Zvukový design

Zatímco skladatelé vytvářejí hudební podklad pro videohry v podobě písní, zvukoví designéři vytvářejí všechny zvukové efekty, které jsou ve hře slyšet, od zvuku kliknutí v hlavním menu po zvuk střílející zbraně. Zvuk je stěžejní součástí každé videohry a může velmi zásadním způsobem obohatit celý herní zážitek. Samotné hraní videoher bez zvuku je mnohem těžší a méně přínosné. (Rogers, 2014)

3.1.5 Příběh a narativní design

Tuto oblast vývoje zastávají spisovatelé a narativní designéři. Můžeme tedy rozlišit dvě různé role psaní, a to je narativní designér, který se na návrhu podílí a druhým je spisovatel, který není do samotného návrhu zapojen není. Hranice mezi těmito pozicemi se ale liší v závislosti na herním studiu. (Gaider, 2016)

Narativní designéři od základu vytvářejí úkoly, které hráč musí splnit. Rozhodují o tom, jak bude úkol probíhat krok po kroku a jaký je jeho cíl. Také vymýšlejí jednotlivé příběhové linie pro různé herní postavy a mají na starosti to, jakým způsobem bude hráč danou situaci vnímat. Naproti tomu spisovatel je někdo, kdo do návrhu zapojen není. Spisovatelé tvoří příběh v kontextu návrhu, který jim byl poskytnut. Narativní designéři rozhodují o tom, kde úkol proběhne, o jaké postavy se bude jednat a jak úkol skončí. Zatímco spisovatelé dávají kontext danému úkolu. Vypracují příběh a charakter postav psaním cutscén a dialogů. (Gaider, 2016; Rogers, 2014)

3.1.6 Testování

Testeři jsou odpovědní za nalezení co největšího počtu chyb, které mohli vývojáři v procesu vytváření hry přehlédnout. Poskytují programátorům objektivní informace a co nejvíce podrobností o nalezeném problému a o tom, jestli může být chyba reprodukována. Vývojáři videohry musí chybám důkladně a detailně rozumět, aby je mohli opravit. (Rogers, 2014; Gregory 2018)

3.1.7 Produkce

Producent videohry je osoba dohlížející na celý vývojový tým. Mezi jeho odpovědnosti patří najímání a budování týmu, podepisování smluv, kontrolování rozpočtu videohry nebo řešení sporů mezi kreativními a programovacími vedoucími. Spravuje pracovní harmonogramu týmu a zajišťuje, aby všichni dodržovali stanovené termíny. Je zástupcem vývojového týmu v jednání s vyšším managementem a vydavateli, koordinuje vytváření externích prvků jako hudby nebo cutscén, může také zajišťovat testování nebo lokalizaci. (Rogers, 2014, Jirkovský 2011)

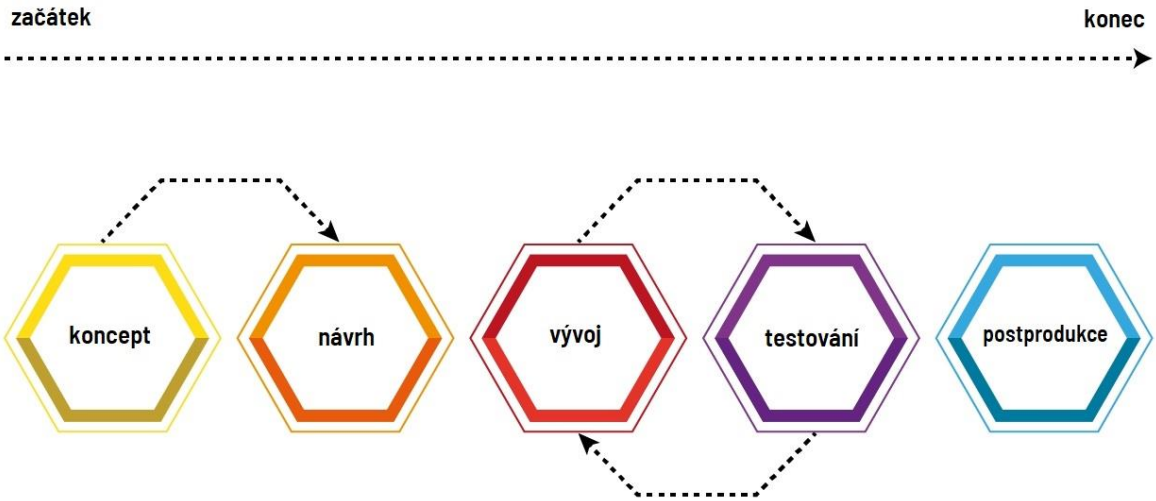
3.1.8 Ostatní

Vývojový tým, který vytváří videohru, nefunguje samostatně. Je často podporován dalšími odděleními jako je managment, marketing nebo administrativa. Pokud těmito odděleními nedisponují, využívají je velmi často externě, jako například externí marketingovou firmu, aby zvýšili povědomí o svém produktu. (Rogers, 2014)

3.2 Etapy vývoje

Etapy vývoje dobře znázorňuje následující obrázek (Obrázek 4).

Obrázek 4: *Diagram průběhu vývoje videohry*



(Zdroj: *Ferro & Sapio*, 2018)

3.2.1 Koncepční fáze

Koncepční fáze je nejranější fází vývoje hry. Obvykle je na několika stránkách popsaná hlavní myšlenka hry a některé základní herní prvky. Je také většinou stanovená cílová platforma a jsou vytvořeny koncepční obrázky. (Ferro & Sapio, 2018)

3.2.2 Návrhová fáze

Po koncepční fázi je třeba dále popsat funkční a technické specifikace videohry více do hloubky. Tyto specifikace mohou zabrat několik desítek stránek v závislosti na rozsahu hry, obzvláště když se jedná o AAA hry, které vyžadují velmi rozsáhlé dokumentace. Tyto dokumentace se skládají ze všeho, co lze považovat za užitečné jako popis obsahu, průběhu a vlastností hry. Jsou rozšiřovány a upravovány v různých vývojových fázích až do úplného konce projektu, stejně jako samotná videohra. Obvykle se vypracuje také krátká demo verze, která by měla v nejlepším případě přilákat investory. Manažeři na základě designu a záměru sestavují různé plány jako rozpočet, alokaci zdrojů, strukturu týmu atd. (Ferro & Sapio, 2018, Rogers, 2014)

3.2.3 Výrobní a testovací fáze

Výrobní fáze trvá nejdelší dobu. Produkce probíhá podle dokumentace a plánů vytvořených v návrhové fázi. Harmonogram určitých činností musí mít správnou návaznost.

Na počátku produkční fáze se vytvoří přibližný prototyp s tzv. placeholdery a měl by demonstrovat hlavní herní mechaniky. Placeholdery jsou assety, které jsou ve hře pouze za účelem testování. V rámci produkční fáze dosahuje vyvíjená videohra čtyři hlavní milníky, a to pre-alfu, alfu, betu a tzv. gold master. Ve fázi pre-alfa se vývoj soustředí především na programování a vytváření assetů. V alfa fázi je videohra tehdy, když už je možné ji hrát, ale ještě není hotová a celý koncept videohry se už nebude měnit. V tuto chvíli se k videohře dostávají také i testeři. K dosažení beta verze musí být hra dokončena, ale má stále velké množství potencionálních chyb, které je třeba eliminovat. Tato fáze slouží především k analýze získaných dat od testerů a opravování chyb, které testeři nahlásí a k dodatečnému upravování různých aspektů videohry. Fáze beta testování může být buď uzavřená a v tom případě se testeři vybírají přímo na základě přihlášky nebo otevřená. Ta je populární především u her pro více hráčů, kde je potřeba testovat i stabilitu připojení k herním serverům a toho se nejlépe dosáhne při velkém množství testerů. Ve chvíli, kdy videohra dosáhne milníku gold master, je dokončena a připravena na vydání a distribuci. (Ferro & Sapio, 2018, Jirkovský 2011)

3.2.4 Postprodukce

Poté, co se videohra dokončí a je připravena k vydání začíná fáze postprodukce. Ta má několik dalších částí. První z nich je marketing. Marketing videoher zahrnuje marketingovou strategii a marketingový plán. Marketingová strategie přímo souvisí s hráčskou základnou, na kterou hra míří a typu videohry. Marketingový plán se soustředí na to, jak danou videohru dostat k zákazníkovi. Hlavní cílem této fáze je prodej vytvořeného produktu. Další dílčí fází postprodukce je distribuce. O tu se většinou stará vydavatelská společnost, který vývoj hry financuje, poté ji propaguje, provádí marketingovou kampaň a finálně spolupracuje s distributorem na distribuci. Ta se dělí na fyzickou distribuci, tedy prodej fyzické videohry v kamenném obchodě a digitální distribuci, což znamená, že výsledná videohra se k zákazníkovi dostane pomocí internetu, konkrétně skrz tzv. digitální distribuční kanál. Nejznámějším distribučním kanálem pro koupi videoher je platforma Steam, kde je k dispozici k zakoupení dohromady přes 50 000 titulů. Hráč tedy nevlastní fyzickou kopii hry, ale má ji uloženou na svém disku v počítači. Po vydání a distribuci vývoj hry ale nekončí. Díky různým druhům hardwaru, který hráči po celém světě používají, mohou vznikat další chyby, a tak vývojářský tým musí tyto chyby dodatečně opravit. Kromě pravidelných aktualizací videohry z důvodu vyvažování nějaké herní mechaniky může být do hry po vydání

přidáván také nový herní obsah. Vydání nového obsahu je v dnešní době velmi běžné, protože to zvyšuje znovuhratelnost. (Pickell, 2019; Jirkovský, 2011)

4 Typy videoher

Existuje mnoho žánrů videoher, ale všechny spadají do jednoho ze dvou obecných grafických stylů, a to je 2D nebo 3D.

4.1 2D

Ve 2D videohrách je možné pohybovat se pouze ve dvou dimenzích (kvůli tomu 2D), a to doleva nebo doprava a nahoru nebo dolů. Protože nemají tolik možností pohybu, jsou 2D videohry často značně jednodušší než jejich 3D protějšky. Mnoho 2D videoher je lineárních, což znamená, že primárním cílem je v podstatě dostat se od začátku na konec úrovně. Ovládání je často také relativně jednoduché, protože postava má omezenou možnost pohybu a počet interakcí s jinými objekty. Kamera ve 2D hrách je také výrazně zjednodušena. Obvykle se dívá na hru přímo ze strany, takže neexistuje žádná perspektiva jako ve 3D titulech. (Stegner, 2020)

4.2 3D

3D videohry naproti tomu poskytují pohyb ve třech dimenzích. To znamená, že se hráč může pohybovat v prostředí „reálného světa“, kde se může otáčet o 360 stupňů a ve kterém mají objekty výšku, délku a hloubku. Prvky jako světlo a zvuk se většinou chovají jako v reálném životě. 3D videohry jsou mnohem komplexnější než 2D videohry. Jedním z největších rozdílů je perspektiva kamery. V mnoha 3D hrách je možné pohybovat kamerou nezávisle na postavě, což hráčovi umožňuje dívat se na svět videohry z různých úhlů. Zvýšená komplexnost ovlivňuje také samotné herní mechaniky. Místo jednoduchých cílů jako dosáhnout konce úrovně se v 3D videohrách dává za úkol plně prozkoumávat prostor, řešit různé hádanky a další. (Stegner, 2020)

4.3 Žánry

Ačkoli byly žánry videoher kdysi pevně dané, v dnešní době to už není zdaleka pravda. S postupem času rozmanitost žánrů roste a svět videoher se neustále vyvíjí a s ním i názvy herních kategorií. V této kapitole se pokusím popsat jednotlivé typy her podle žánrů tak, jak jsou dnes nejvíce relevantní. Definice jednotlivých žánrů a zařazení konkrétní hry do specifického žánru ale není úplně jednoznačné, protože některé videohry mnohdy pokrývají hned několik žánrů najednou.

4.3.1 FPS a TPS

Shooters neboli střílečky je žánr s dlouholetou historií, z něhož se vyvinuli dva hlavní subžánry, a to střílečka z pohledu první osoby (FPS) a střílečka z pohledu třetí osoby (TPS). I zde figuruje určité překrývání, protože mnoho současných titulů umožňuje přepínat mezi první a třetí osobou. Zásadním rozdílem je perspektiva. FPS simuluje v podstatě to, co vidí vaše herní postava. TPS ukazuje celou vaši postavu a okolní prostředí. (Studytonight Technologies, n.d.-a)

Předpoklad pro tyto hry je jednoduchý, postřílet nepřátelé, kteří stojí v cestě k cíli. Zatímco většina střílečeк se dělí na FPS a TPS, často se o nich mluví jako o herních prvcích v jiných hrách. Dobrým příkladem je jedna z nejúspěšnějších her vůbec, Grand Theft Auto. V základu je to sandboxová hra, ale má také prvky TPS a zároveň má hráč i volbu hrát hru z pohledu první osoby, tedy jako FPS. (Pavlovic, 2020)

4.3.2 RTS

Pojem real-time strategy neboli strategická hra, byl původně vytvořen jako marketingový výraz pro hru od Westwood Studios, Dune II. Tyto typy her existovaly roky předtím, než většina hráčů vůbec věděla, že takový žánr je. Díky jejich trvající popularitě a růstu nových subžánrů zůstávají RTS hry značnou součástí videoherního světa. (Pavlovic, 2020)

Podstatou RTS her je, aby hráč získával zdroje a díky tomu rozvíjel počet bojových jednotek. Pomocí těchto jednotek potom soutěží proti oponentovi v „reálném čase“, na rozdíl od tahové strategie. Proto termín „real-time strategy“. Na hru se obvykle kouká pohledem shora dolů. Dobrým příkladem RTS videohry je například série Age of Empires. (Pavlovic, 2020)

4.3.3 RPG

Předpoklad hry na hrdiny neboli RPG je v mnoha hrách velmi jednoduchý, a to vytvořit nebo převzít kontrolu nad postavou, která se poté může pomocí systému úrovní a zkušeností vylepšit. (Pavlovic, 2020) RPG žánr je základním kamenem videoher, ale žádná hra nemůže tuto kategorii řádně reprezentovat, protože se rozrostl do mnoha subžánrů, a to hlavně:

- MMORPG - kombinuje masivně multiplayerový online formát s RPG hratelností. Pravděpodobně nejznámějším titulem je World of Warcraft.
- ARPG - akční RPG kladou velký důraz na souboj, ale zahrnují mnoho charakteristik standardních RPG. Dobrým příkladem je série Zaklínač.

- CRPG - termín „Počítačové RPG“ se používá k popisu západních RPG vytvořených primárně pro platformu osobních počítačů. Reprezentující hrou tohoto subžánru je například série Fallout.
- TRPG – tato zkratka znamená taktickou hru na hrdiny, která se podobá strategickým hrám, ale klade větší důraz na kreativní myšlení a krátkodobé rozhodování. Příkladem je série XCOM.
- Roguelike – jedná se o tahovou hru, kde musí hráč po smrti začít hrát úplně znovu. Hra má tzv. „tiles“ grafiku. Jeden z novějších titulů je například The Binding of Isaac. (Studytonight Technologies, n.d.-a; Pavlovic, 2020)

4.3.4 MOBA

Multiplayer online battle arena nebo MOBA je stále populárnější žánr. Propojením s řadou dalších stylů, MOBA hry sdílejí mnoho prvků se strategickými hrami (RTS). Na hru se díváme z perspektivy shora dolů, což zviditelňuje mapu a zdroje, které je třeba sbírat. (Pavlovic, 2020)

Hlavním rozdílem mezi MOBA a RTS hrami je postava a role hráče. V MOBA hře se obvykle ovládá pouze jedna postava. To je významný kontrast s většinou RTS videoher, kde se většinou ovládá více jednotek najednou. (Pavlovic, 2020)

MOBA hry také upřednostňují hru pro více hráčů a týmovou hru. Hrajete s a proti ostatním lidskými hráčům a týmovou spoluprací se snažíte dosáhnout vítězství. V tomto žánru dominují hry Dota 2 a League of Legends. (Studytonight Technologies, n.d.-a; Pavlovic, 2020)

4.3.5 Sandbox

Termín „sandbox“ je často spojován s prvky jako otevřené volby hráčů, které mají následky na další průběh hry, otevřeným herní světem a nelineární hratelností. Žánr sandboxu se za posledních pár let velmi rozrostl. (Pavlovic, 2020)

V těchto hrách mají hráči často ne příliš konkrétní cíle a hra disponuje ne narativním způsobem vyprávění. Hráč plní řadu úkolů, které můžete splnit mnoha různými způsoby a pohybuje se na rozsáhlé herní mapě. Tato mechanika hráče většinou pohltí a umocní jeho herní zážitek. Jednou z nejznámějších sandboxových her je například Minecraft, který je současně také nejprodávanější hrou na světě. (Pavlovic, 2020)

4.3.6 Akční adventury

Akční adventury patří mezi jeden z prvních hybridních žánrů a zaměřují se hlavně na děj a souboje skrz pohlcující příběh a zábavné herní mechaniky. Do těchto kritérií se vejde opravdu mnoho her. Jedním z klasických zástupců akčních adventur je série Legend of Zelda, která vyšlapala cestu pro další řadu videoherních franšíz. Adventury zahrnují také svůj vlastní žánr. Mnoho z nich spadá do subžánru tzv. „point-and-click“, ve kterém hráči obvykle řeší záhady nebo hádanky z pohledu první osoby. Pomyslná hranice mezi akcí a adventurou se dá nastavit podle toho, jak hra vyvažuje příběh a funkce, jako je například simulovaný souboj. Do tohoto žánru patří také velké množství komerčních AAA videoher s velkou produkcí jako Assassin's Creed od studia Ubisoft nebo poměrně nová herní série s tématikou Star Wars od studia EA. (Studytonight Technologies, n.d.-a; Pavlovic, 2020)

4.3.7 Simulace a sportovní

Tyto žánry se postupem času velmi hodně vyvinuly a ve skutečnosti se stále jedná o ten samý princip. Avšak až s pokrokem grafické technologie začaly tyto videohry nabízet jedinečné a pohlcující zážitky. Nejnovější iterace poskytují působivou úroveň detailu a ukazují, co všechno je vlastně u her možné. (Pavlovic, 2020)

Rozmanitost sportovních her se rozšířila a nabízí plnohodnotné partnerství s významnými sportovními organizacemi a ligami, od závodních tratí po hřiště nebo tenisové kurty. NBA 2K a série FIFA jsou dva populární příklady, které simulují profesionální basketbal a fotbal, zatímco například série Forza je simulací automobilových závodů. (Pavlovic, 2020)

Žánr simulace zahrnuje mnoho sandboxových titulů, her o budování světa a her pro virtuální realitu. Typickým zaměřením je vytvoření pohlcujícího a realistického herního světa, ať už jde o svět farmaření, řízení zoologické zahrady nebo pilotování letadel. (Studytonight Technologies, n.d.-a; Pavlovic, 2020)

4.3.8 Logické a společenské

Puzzlers neboli logické hry a společenské hry kladou velký důraz na herní mechaniky. V logických hrách se hráč snaží vyřešit hádanku či hlavolam nebo je před něj postavena nějaká výzva jako například bludiště. Do žánru logických her můžeme zařadit videohry od Hledání min či Tetrisu až po více pohlcující hry, s plně vyvinutým prostředím a hybridní hratelností. Dobrým příkladem je například série Tomb Raider, kde jsou hádanky a hlavolamy zabudovány do konvenčního dobrodružného příběhu a prostředí, což z nich dělá klíčovou herní mechaniku. (Studytonight Technologies, n.d.-a; Pavlovic, 2020)

Společenské hry často a zcela pochopitelně obsahují prvky pro více hráčů a používají je jako základní stavební kameny. Hlavní účelem těchto her je zábava pro více lidí najednou. Série Mario Party je obzvláště populární společenská hra a zahrnuje ji již více než 10 titulů. (Pavlovic, 2020)

4.3.9 Hororové

Hororové hry a survival hry neboli hry zaměřené na přežití se navzájem také hodně překrývají. Zejména hororové hry často sdílejí některé základní funkce se survival hrami, i když opak je méně častý. Jelikož jsou v těchto hrách z velké části vždy i FPS prvky, probíhá často debata o tom, jak tyto videohry definovat. (Studytonight Technologies, n.d.-a; Pavlovic, 2020)

Základní mechanika hry zaměřené na přežití se soustředí na sbírání různých zdrojů a nástrojů k tomu, aby mohl hráč čelit hrozbám a udržel svou postavu naživu. Do této kategorie by mohl patřit populární Minecraft nebo povedená nezávislá hra Don't Starve, ve které je hlavním cílem, jak už název napovídá, nevyhladovět v nepříznivém prostředí. (Pavlovic, 2020)

Hororové hry je velmi široká kategorie. Téměř cokoli se zombie, postapokalyptickým příběhem nebo spoustou děsivých situací, je považováno za horor. Tyto tituly jsou velmi často také psychologické a využívají napětí k většímu vnoření hráče do hry. (Pavlovic, 2020)

4.3.10 Platformové

Platformové hry nebo také plošinovky získaly své pojmenování díky tomu, že hlavní postava se pohybuje po plošinách, a to buď v podobě běhu, lezení nebo skákání. Plošiny slouží jako nástroj k překonání úrovně nebo naopak jako překážka. Platformové hry mají velmi často boční pohled a jednoduché ovládání. Hráč v nich velmi často ovládá pouze hlavní postavu a prochází náročnými úrovněmi. Legendární hra Donkey Kong je často považována za první skutečný příklad. Tato hra předala pochodeň další legendární hře Super Mario Bros. Plošinovky jsou extrémně oblíbené u nezávislých studií a hráčů. V praktické části vytvářím videohru spadající právě do tohoto žánru. (Pavlovic, 2020)

5 Videohry jako nástroj pro učení

Když je člověk plně ponořen do nějaké problematiky, učí se. Videohry se od ostatních médií liší tím, že se člověk přímo účastní. Umožňují hráčům vstoupit do různých rolí, konfrontovat různé problémy, činit patřičná rozhodnutí a zkoumat důsledky jejich rozhodnutí. Dobře navržené videohry poskytují rovnováhu mezi výzvami a odměnami a ty mohou vést k hluboké angažovanosti. Hráči mohou postupovat vlastním tempem a jejich neúspěch se netrestá. Vždy existuje další pokus, jak daný problém vyřešit. Videohry také podporují kritické myšlení, dovednosti, spolupráci, systémové myšlení a kreativitu, což jsou vlastnosti nezbytné pro fungování v rychle se měnícím světě současnosti. (Brand, 2017)

Podstatou učení se prostřednictvím videoher je získávání konkrétních zkušeností. Před hráče jsou postaveny překážky, které musí překonat. K překonání těchto překážek je třeba myslet neotřelým způsobem, což vede k novým poznatkům a většinou existuje i více způsobů, jak vyřešit konkrétní problém a dosáhnout cíle. Každá videohra s kvalitním designem vyvolává v hráči pocit frustrace z opakujícího se neúspěchu. Poté ale pokrok a zvládnutí zmiňovaných překážek poskytuje obohacující zkušenosti a motivuje hráče k dalšímu hraní a dokončení úkolu. (Brand, 2017; Janíčková, 2009)

5.1 Vzdělávací hry a jejich vlastnosti

Výše zmíněný efekt najdeme ve velkém množství videoher. Tyto hry často integrují učení, aniž by byly výslovně vzdělávací. To je ale jen jeden ze dvou pohledů na učení ve videohrách. Existují také takové videohry, které se přímo zaměřují na vzdělávání. Jsou to tzv. edukační videohry. Tyto tituly mají silnou vzdělávací složku, ale často nemají takový dopad, jako většina komerčních videoher. Pro správnou motivaci hráče by měl být součástí hry i kvalitní příběh, nejenom dobré herní mechaniky a překážky, kterým hráč vzdoruje. Správná kombinace obsahu, příběhu a principu hraní v jednom celku je doslova umění. Hra bez nápadu a bez myšlenky nemá smysl, dílčí úrovně a úkoly bez přítomnosti dobré příběhové linie neupoutají hráče takovým způsobem, aby ve hře pokračoval. Hlavní účelem videohry je především zábava. Velké množství vzdělávacích her ale této skutečnosti nedokáže dosáhnout. Bez dobrých herních mechanismů dostatečně nevtáhnou hráče do děje. Příliš se koncentrují pouze na vzdělávací prvek, zatímco ostatní aspekty hry pokulhávají. Všechny komponenty by měly mít stejnou prioritu, jako ta vzdělávací. Výzvou těchto edukačních her je tedy hlavně rovnováha mezi zábavným obsahem a koncentrací na to, aby hráče naučily konkrétním dovednostem. (Brand, 2017; Janíčková, 2009; Widitiarsa, 2019)

Vzdělávací hry jsou vyvíjeny ku prospěchu třetí osoby a měly by být založeny na vědeckých teoriích o vzdělávání a vývojáři by se měli soustředit na požadovaný vzdělávací výsledek. Vzdělávací videohra je typ produktu, ve kterém je zakomponována pedagogika a vlastnosti her, které motivují a jsou zábavné. Na vývoji se už nepodílí jen herní designéři, programátoři a další, ale také různí odborníci a pedagogové v příslušném oboru. Společně by se měli soustředit na vytvoření kvalitního produktu se vzdělávacím cílem. Nejde o to přeměnit zábavnou videohru na vzdělávací, ale o prezentování tradičního školního materiálu v podobě videohry. (Brand, 2017; Janíčková, 2009)

5.2 Příklady vzdělávacích her

5.2.1 The Oregon Trail

Videohra The Oregon Trail (Obrázek 5) je považována za první vzdělávací hru, kterou vytvořili tři studenti Don Rawitsch, Bill Heinemann a Paul Dillenberger v roce 1971. Jejím účelem bylo naučit žáky osmých tříd ve školách v Minnesotě americké dějiny a podporovat tvůrčí myšlení. Náplní hry a cílem studentů bylo, aby se ujali role západních osadníků překračujících severoamerický kontinent na cestě k pobřeží Tichého oceánu. Hráči si museli vybrat, které předměty si ponesou, jak rychle budou cestovat a museli učinit rozhodnutí, co dělat, když dojde jídlo nebo výpravu postihne nemoc. (Pepple, 2018; Janíčková, 2009)

V roce 1974 se vyvinula verze pro širokou distribuci do škol. Distribuce začala nejprve v Minnesotě a poté se šířila po celých Spojených státech. Od té doby je hra stále dostupná v novějších verzích a objevuje se na všech hlavních platformách, od počítačů až po chytré mobilní telefony. V sedmdesátých a osmdesátých letech minulého století, kdy byl přístup k počítačům vzácný, The Oregon Trail pomáhal nejen učit hráče o americké historii, ale také je seznámil s užíváním počítače. Bylo prodáno více než 65 milionů kopií a jedná se o možná nejstarší nepřetržitě dostupnou videohru, která byla kdy vyrobena. Je průkopníkem kombinace učení a hraní videoher. (Pepple, 2018)

Obrázek 5: *Ukázka ze hry The Oregon Trail*

(Zdroj: Jenkins, 2017)

5.2.2 Attentat 1942

Attentat 1942 (Obrázek 6) je výuková videohra od českého studia Charles Games, které je složené hlavně ze zástupců Filozofické fakulty Univerzity Karlovy a Matematicko-fyzikální fakulty Univerzity Karlovy. Je to oceňovaná „point-and-click“ adventura z roku 2017. (Attentat 1942, n.d.-b)

Vypráví příběh z doby nacistické okupace a z pohledu těch, kteří ji zažili na vlastní kůži. Je postavena na dialozích s přeživšími, interaktivních komiksech a autentických historických záběrech. Hráč mluví se svědky těchto událostí a jsou mu předloženy jejich vzpomínky. V rámci příběhu hráč zjistí, že dědečka hlavní postavy zatklo gestapo bezprostředně po atentátu na Reinharda Heydricha. Cílem hry je zjistit, proč byl zatčen a jakou roli při atentátu hrál. (Attentat 1942, n.d.-b)

Hlavní herní funkcí jsou především dialogy. Hra obsahuje interaktivní komiksy, vzácné digitalizované filmové záběry, náročné minihry a rozhovory ve filmovém stylu, které byly napsány týmem profesionálních historiků. Hra slouží jako dějepisná učební pomůcka a interaktivní způsob toho, jak studenty poučit o tomto období naší historie. (Attentat 1942, n.d.-b)

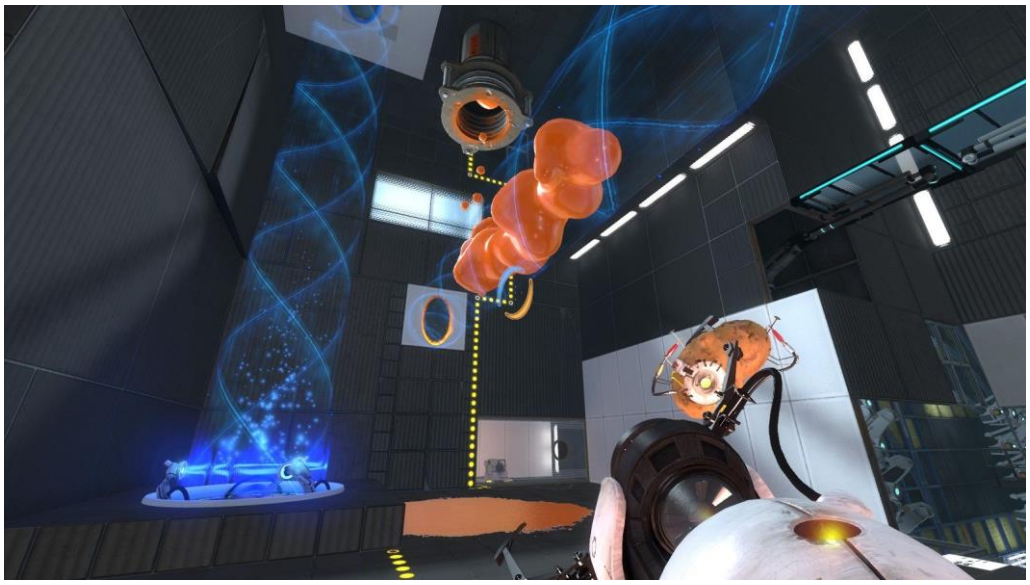
Obrázek 6: *Ukázka ze hry Attentat 1942*

(Zdroj: Attentat 1942, n.d.-a)

5.2.3 Portal

Portal a Portal 2 (Obrázek 7) jsou logické videohry vytvořené americkým studiem Valve. Jsou to pravděpodobně nejvíce komerčně úspěšné a kriticky uznávané videohry, které by se daly zařadit do kategorie vzdělávání. Nejsou to sice vyloženě vzdělávací hry, ale vyžadují a prohlubují kritické myšlení, kreativitu a schopnost řešit problémy. Hlavní postava řeší hádanky a problémy v trojrozměrném světě. Hráči disponují ručním portálovým zařízením, tzv. „portal gun“, které slouží k umisťování propojených portálů na stěny, podlahy nebo stropy. Jakmile je umístěn pár portálů, jakýkoli objekt vstupující do prvního portálu opustí ten druhý. V druhé iteraci této videohry existuje také editor, kde hráči mohou samotné úrovně, tzv. testovací místnosti“, vytvářet a sdílet je pro ostatní členy komunity. Na některých amerických školách byla videohra Portal 2 použita při výuce fyziky. Z vlastní zkušenosti vím, že koncepty fyziky hra demonstruje velmi dobře. Od zkoumání gravitace a tření až po uvažování o rychlosti pohybu a zachování hmoty objektů. Prohlubuje také i prostorové uvažování a opravdu nutí hráče kriticky myslet nad konkrétní překážkou. (KQED, 2012)

Obrázek 7*: Ukázka ze hry Portal 2*



(Zdroj: New Game Network, 2011)

6 Vývojové prostředí Unity

Unity je herní engine vyvíjen společností Unity Technologies. Je to vývojové prostředí používané pro vytváření videoher na velké množství platforem jako Windows, Mac, Linux, WebGL, současnou a předchozí generaci herních konzolí, Nintendo Switch, mobilní platformy Android a iOS, virtuální realitu Oculus Rift a další. Unity je jedním z nejpoužívanějších herních enginů vůbec. Mezi hlavní faktory toho, proč je Unity tak oblíbený herní engine je hlavně jeho dostupnost. Je nabízen v několika verzích, přičemž základní kategorie s názvem Personal je zcela zdarma. Poté jsou k dispozici ještě verze Plus, Pro a Enterprise, které nabízí další funkcionality nad rámec základní verze, které ale nemají dopad na výslednou kvalitu hry. Dalším důvodem popularity Unity je to, že je možné vytvořenou videohru exportovat na více platforem najednou bez nutnosti nějak výrazně upravovat kód. Hlavní oblastí úprav je především vstup uživatele. (Saarelainen & Pakarinen, 2013)

Z počátku byl engine Unity uzpůsobený pouze pro vývoj 3D videoher, ale postupem času byly přidány i nástroje pro podporu 2D vývoje. Vývoj hry jako takový se odehrává jednak v Unity Editoru, kde se pracuje především s organizací objektů ve scénách, a poté v jakémkoliv vývojovém prostředí pro psaní skriptů, které podporuje programovací jazyk C#. Unity je nativně spojeno s prostředím programu Visual Studio od společnosti Microsoft. (Šveigr, 2018)

6.1 Unity Asset Store

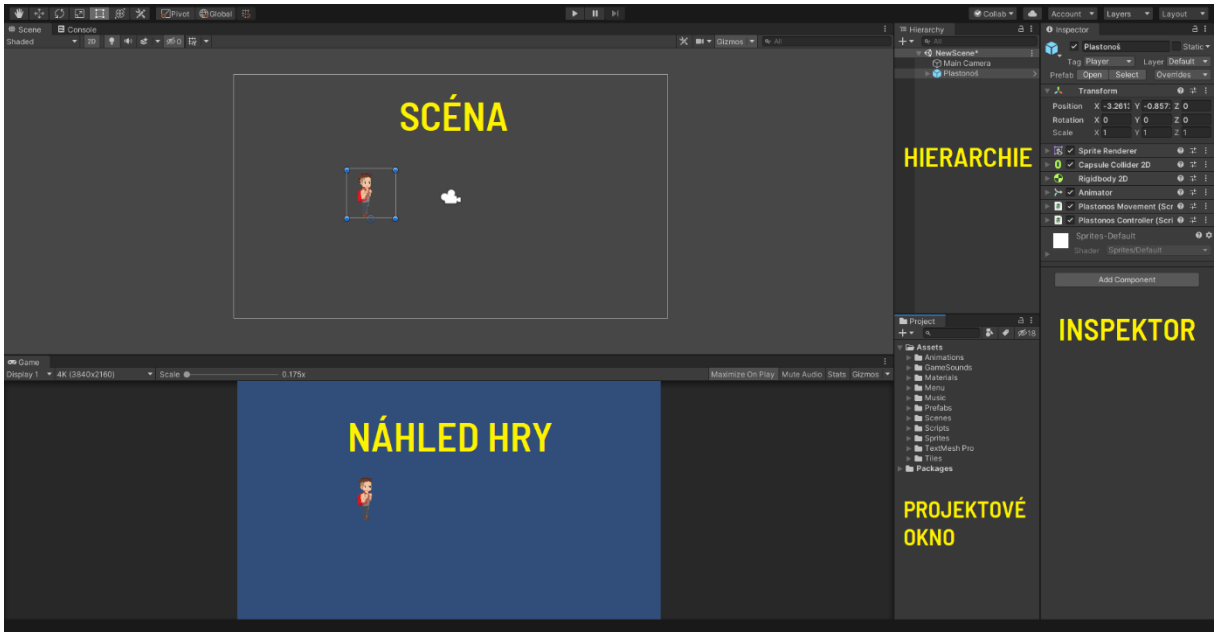
Další faktor přispívající k popularitě Unity je Unity Asset Store. Tento obchod obsahuje knihovnu bezplatných i zpoplatněných assetů, které společnost Unity Technologies i členové Unity komunity vytvářejí. Uživatelé Unity se tak mohou stát vydavateli a prodávat obsah, který vytvořili. V obchodě jsou dispozici všechny možné druhy assetů jako textury, 3D a 2D modely, soubory zvukových efektů, sety animací, ukázkové projekty, výukové programy a různá další rozšíření pro Unity Editor v podobě pluginů. K zakoupeným a staženým assetům lze poté přistupovat rovnou z integrovaného rozhraní v Unity Editoru. Odtud je možné stahovat a importovat assety přímo do konkrétního projektu.

6.2 Editor

Unity Editor je rozdělen do několika hlavních oken (Obrázek 8). S těmito jednotlivými okny se dá libovolně pohybovat a uživatel si může nastavit jejich velikost a umístění dle vlastních preferencí. Má také možnost vytvořit úplně nové okno a přiřadit mu vlastní

funkcionalitu. Toto rozpoložení oken si může uživatel uložit na měnit je na základě konkrétního projektu, v kterém zrovna pracuje. V následujících podkapitolách popíšu ty nejdůležitější okna. (Ferro & Sapio, 2018; Pereira, 2014)

Obrázek 8*: Unity Editor*



(Zdroj: autor, 2021)

6.2.1 Hierarchie

Okno Hierarchie obsahuje veškeré objekty neboli GameObjects v aktuální scéně. GameObject může být cokoli od hlavní postavy přes rekvizitu až po scenérii v pozadí. GameObject je v podstatě jakýsi kontejner, do kterého se dají umístit tzv. komponenty. Sám o sobě je GameObejct neviditelný a obsahuje pouze komponentu Transform, která udává informace o jeho pozici ve scéně, velikosti a natočení. V Unity existují i další komponenty pro osvětlení, animaci, vykreslování, chování objektu (skript), fyziku, detekci kolizí atd. Tyto komponenty dávají konkrétním objektům funkčnost. GameObject obsahuje komponentu, a ta mu tedy ve scéně poskytuje určité funkce. Pokud se objekt přidá nebo odebere ze scény, objeví se a zmizí také z okna Hierarchie. (Ferro & Sapio, 2018; Pereira, 2014)

6.2.2 Scéna

Scéna je pohled do světa, který se při vývoji videohry vytváří. Je to interaktivní prostor, kde je možné vybrat, přeskupit a umístit všechny objekty v rámci projektu jako postavy,

kamery, světla a všechny ostatní typy objektů. Je možné změnit jejich velikost, natočení apod. (Ferro & Sapio, 2018)

6.2.3 Náhled hry

Náhled hry neboli Game view je velmi podobný pohledu na scénu. Zobrazuje ale aktuální výsledek scény tak, jak ho uvidí hráč při samotném hraní. Je možné měnit poměr stran a rozlišení tohoto okna, aby se vývojář mohl ujistit, že hra bude vypadat dobře na různých obrazovkách. Zároveň je možné si v tomto okně danou scénu zahrát a otestovat, zda vše funguje tak, jak má. (Ferro & Sapio, 2018; Godbold, & Jackson, 2016)

6.2.4 Inspektor

V okně Inspektor jsou zobrazeny všechny informace týkající se vybraného objektu. Je to jedno z nejpoužívanějších oken v Unity Editoru. Po zvolení daného objektu se v Inspektoru zobrazí informace jak o konkrétním objektu, tak i o jeho komponentách. Tyto informace se týkají vlastností a nastavení nejen objektů ve scéně, ale také ostatních položek v rámci Unity, jako jsou assety, fyzické materiály, zvuky atd. (Godbold, & Jackson, 2016)

6.2.5 Projektové okno

V Projektovém okně je zobrazen každý soubor neboli asset, který je součástí projektu. Assety se mohou importovat nebo vytvořit přímo v Unity. Jsou to například různé modely, skripty, zvukové efekty atd. Assety mohou být z projektového okna přetaženy rovnou do scény. V tomto okně se také nachází adresářová struktura. Všechny assety jsou logicky umístěny ve složce Assets. (Ferro & Sapio, 2018)

6.2.6 Konzole

Konzolové ono zobrazuje chyby, varování a další zprávy, které Unity vygeneruje nebo na základě napsaného kódu. Tyto zprávy či varování mohou být jednoduché chyby syntaxe v kódu nebo i složitější problémy týkající se výkonu. (Ferro & Sapio, 2018)

6.3 Klíčové komponenty

6.3.1 Transform

Transform se používá k určení pozice, rotace a měřítka herních objektů ve scéně. Funkce objektu je definována komponentami k němu připojenými. GameObject má vždy připojenou komponentu Transform a není možné ji odebrat nebo ho vytvořit bez ní. S Transform komponentou se manipuluje ve 3D prostoru na osách X, Y a Z ve 2D prostoru pouze na X a

Y. Transform lze upravit přímo ve scéně nebo změnou jeho vlastností v Inspektoru. Ve scéně se můžete Transfrom upravit pomocí nástrojů přesunout, otočit a měřítko. (Unity Technologies, n.d.-c)

6.3.2 Sprite Renderer

Sprite je v podstatě 2D grafický objekt, který na sobě má vykreslený nějaký obrázek nebo texturu. Sprity tvoří grafické prostředí videohry. Komponenta Sprite Renderer vykresluje sprite a řídí, jak se zobrazuje ve scéně pro 2D i 3D projekty. Pokud se vytvoří sprite v Hierarchii, Unity automaticky vytvoří GameObject s připojenou komponentou Sprite Renderer. V rámci této komponenty je možné měnit RGB nastavení barvy daného obrázku nebo například jeho průhlednost. (Unity Technologies, 2017a)

6.3.3 Camera

Kamera je komponenta, která zachycuje a poté zobrazuje svět videohry. Ve scéně může být neomezený počet kamer. Lze je nastavit tak, aby se vykreslovaly v libovolném pořadí, na jakémkoli místě na obrazovce nebo pouze v určitých částech obrazovky. Kamera má možnost ortografického zobrazení, což znamená, že odstraní veškerou perspektivu z pohledu kamery. To je většinou užitečné při tvorbě 2D videoher. (Unity Technologies, 2018a)

6.3.4 Collider

Collider je komponenta, která definuje tvar objektu za účelem fyzických kolizí. Collider, který je neviditelný, nemusí mít přesně stejný tvar jako GameObject. Hrubá aproximace je často efektivnější a ani není ve hře rozeznatelná. Nejjednoduššími typy collideru jsou tzv. primitivní typy. Ve 2D se jedná o Box Collider 2D, Circle Collider 2D a Capsule Collider 2D. Jeden objekt může mít libovolný počet colliderů a při neobvyklém tvaru objektu se k sobě mohou skládat. Může ale nastat případ, kdy ani složené collidery nejsou dostatečně přesné. Ve 2D je tedy možné použít také Polygon Collider 2D, v rámci kterého je možné tvar přesně určit pomocí jednotlivých bodů polygonu. (Unity Technologies, n.d.-h)

6.3.5 Rigidbody

Rigidbody zajistí, že na objekt budou působit vlivy fyziky. Rigidbody může přijímat sílu a točivý moment, což zapříčiní to, že se objekty budou pohybovat realistickým způsobem. Jakýkoli GameObject musí obsahovat Rigidbody, aby byl ovlivňován gravitací a působily něj síly vyprodukované pomocí skriptů nebo interakcí s jinými objekty prostřednictvím fyzického enginu. Rozdíl mezi Rigidbody a Rigibody 2D spočívá v tom, že ve 2D se objekty mohou

pohybovat pouze v rovině X a Y a mohou se otáčet pouze kolem os těchto rovin. (Unity Technologies, n.d.-a)

6.3.6 Script

Chování objektů je řízeno komponentami, které jsou k nim připojeny. Přestože vestavěné komponenty v Unity mohou být velmi univerzální, při implementaci některých herních funkcí se musí jít nad rámec toho, co mohou poskytnout. Unity tedy umožňuje vytvářet vlastní komponenty pomocí skriptů. Ty umožňují spouštět herní události, upravovat vlastnosti komponent v reálném čase a reagovat na vstupy od uživatele. Skripty se mohou vytvořit přímo v Unity. (Unity Technologies, 2018b)

6.3.7 Animator

Komponenta Animator se používá, pokud chceme objektu ve scéně přiřadit animaci. Komponenta Animator vyžaduje referenci na tzv. Animator Controller (ovladač animací), který definuje, které animační klipy použít a jak je uspořádat a přecházet mezi nimi. Je běžné mít více animací jednoho objektu a přepínat mezi nimi, když nastanou určité podmínky. Například přepnout z animace chůze na animaci skoku pokaždé, když hráč stiskne mezerník. Animator spravuje různé animační klipy a přechody mezi nimi pomocí tzv. state machine (stavový stroj), což je v podstatě takový diagram animací. (Unity Technologies, 2017b)

6.3.8 Audio Source

Audio Source neboli zdroj zvuku je komponenta, která přehrává zvukový klip ve scéně. Bez přiřazeného zvukového klipu nedělá nic. Klip je zvukový soubor, který bude přehrán. Komponenta zdroj zvuku je ovladač pro spuštění a zastavení přehrávání tohoto klipu a úpravu jeho dalších vlastností. (Unity Technologies, n.d.-b)

6.3.9 Particle System

Particle System neboli částicový systém simuluje a vykresluje mnoho malých obrázků najednou nazývaných částice, aby vytvořil vizuální efekt. Každá částice představuje samostatný grafický prvek. Částicové systémy jsou užitečné, když je třeba vytvořit dynamické objekty, jako je oheň, kouř nebo nějaký druh kapaliny. (Unity Technologies, n.d.-g)

6.3.10 Canvas

Komponenta Canvas neboli plátno představuje prostor, ve kterém je zobrazeno uživatelské rozhraní. Všechny prvky uživatelského rozhraní musí být potomky objektu, ke kterému je připojena komponenta Canvas. Když se vytvoří objekt libovolného prvku

uživatelského rozhraní jako například obrázek, text nebo tlačítko, automaticky se vytvoří i objekt Canvas, pokud ve scéně ještě žádný není. (Unity Technologies, n.d.-d)

7 Tvorba vlastní videohry – praktická část

V této kapitole popisuji praktickou část bakalářské práce, a to tedy vytváření vlastní videohry. Pomocí ukázek kódu a grafických assetů vysvětlím můj postup při vývoji. Hru jsem vytvářel v herním enginu Unity, konkrétně verzi 2019.4.10f1, a jehož funkce a výhody jsem popisoval v předchozí kapitole. Skripty jsem psal v programu Microsoft Visual Studio 2019. Tento program jsme používali při studiu, takže jsem s ním již obeznámen a dobře se v něm orientuji. Další výhodou používaní toho programu je tzv. IntelliSense, což je obecný termín pro různé funkce na úpravu kódu, jako doplnění rozepsaného kódu nebo informace o parametrech. IntelliSense podporuje Unity funkce a třídy.

Co se týče vizuálního vzhledu, zvolil jsem grafický styl pixel art, který je mnohými považován za nejlepší volbu pro začátečníky, a to z toho důvodu, že k vytváření obrázků je možné si vystačit jen s počítačovou myší. Na plátně libovolné velikosti se vybarvují jednotlivé pixely, což jsou obrazové body, které dohromady tvoří obrázek. Proces tvorby jednotlivých obrázků je časově velmi náročný, protože je třeba vybarvit každý pixel zvlášť. K vytvoření grafiky jsem použil editor Krita, což je grafický editor pro tvorbu nejen digitálních obrázků, ale i animací. Podporuje tvorbu pixelových obrázků a je velmi intuitivní k používání. Jednou z hlavních výhod je to, že je dostupný zcela zdarma.

Pro zvukové efekty a hudbu v pozadí jsem využil možnosti Unity Asset Store. Při hraní hry se opakuje jedna skladba. Různé zvukové efekty se spouští při sebrání předmětu, dokončení úrovně, pohybu postavy, skoku a aktivování nášlapného tlačítka.

7.1 Myšlenka a popis videohry

Mým cílem bylo vytvořit videohru, která má nějaké poslání. Vzhledem k mému zájmu o ekologii a ochranu životního prostředí jsem se rozhodl vytvořit hru, která míří především na děti a integruje vzdělávací prvky. Zábavným a interaktivním způsobem informuje hráče a problematice plastového znečištění v přírodě a v našem okolí.

Žánrově hra spadá do kategorie 2D platformer. Ve své podstatě plošinové hry vyžadují, aby se hráč pomocí funkce skoku dostal na plošiny umístěné v herním světě, procházel prostředí a dostal se na konec jednotlivých úrovní. Kategorie platformer má ovšem ještě několik subžánrů. Ten, kterým se zabývám při vytváření mé videohry je tzv. sidescroller. Sidescroller je jedním z nejběžnějších typů plošinových her. Jak název napovídá, hra se hraje v perspektivním úhlu pohledu z boku a postava se na obrazovce obvykle pohybuje z levé strany obrazovky doprava.

Co se týče další kategorizace, jedná se o tzv. collecting game neboli sběratelskou hru. Klíčovým rysem toho typu her je logicky sbírat nějaké předměty. Základní mechanika tedy spočívá v tom, že hráč musí něco sebrat, aby splnil určitou podmínku a postoupil do další úrovně. Do sběratelské hry se samozřejmě mohou implementovat i různé další komponenty a parametry jako jsou například nepřátelé, předměty s různou vzácností, frekvence objevování vzácných předmětů atd. Není důležité, co hráč sbírá, ale kdy a jakým způsobem. Při vytváření sběratelské hry je důležité apelovat na to, aby byla správně vyvážená. Hra musí být zábavná, takže by to měla být pro hráče výzva. Nesmí ale zase být natolik obtížná, aby vyvolávala velkou frustraci.

V rámci mé videohry hráč sbírá plastové odpadky v prostředí lesa. Hra disponuje deseti úrovněmi a v každé z nich musí hráč sebrat určitý počet odpadků, aby postoupil do další úrovně. K jednotlivým odpadkům se dostane po překonání určitého úseku dané úrovně. Na cestě k cíli musí čelit mimo jiné samovolně se pohybujícím platformám, dostat se na jinak nedostupná místa skrz manipulaci s kameny (Příloha 4) a rozpohybovat platformy pomocí nášlapného tlačítka (Příloha 6). Herní mechaniky jsou hráčovi postupně představovány při postupu hrou. Mezi jednotlivým úrovněmi jsou mu v podobě tzv. „cutscén“ v textové podobě podávány informace o zmiňované problematice plastového znečištění (Příloha 2).

7.2 Hlavní postava

Pomocí hlavní postavy, kterou jsem pojmenoval Plastonoš, se hráč pohybuje v herním světě. Může tedy chodit doleva nebo doprava, vyskočit a když se přiblíží k objektu kamene, může ho posunout. Sprite hlavní postavy o velikosti 64 na 64 pixelů je zobrazen na následujícím obrázku (Obrázek 9).

Obrázek 9*: Sprite hlavní postavy*

(Zdroj: autor, 2021)

Objekt hlavní postavy má několik komponent, a to Transform, který udává jeho pozici ve scéně a Sprite Renderer, který sprite vykresluje. Dále Rigidbody 2D, a to zajišťuje, že na objekt působí fyzické vlivy a poté Capsule Collider 2D, sloužící k detekci kolizí. Finálně jsou to dva skripty, které umožňují, aby hráč mohl postavou pohybovat, a to *PlastonosController* a *PlastonosMovement*. *PlastonosController* kontroluje podmínky, za kterých se hráč může pohybovat do stran, vyskočit nebo tlačit kamen, zatímco *PlastonosMovement* provádí funkci pohybu, spouští animace a zvukové efekty. V následujících odstavcích popíšu některé části kódu z obou zmíněných skriptů.

V následující ukázce (Zdrojový kód 1) je úryvek kódu, kde je možné vidět funkci *Move* ze skriptu *PlastonosController*, která je využita i ve skriptu *PlastonosMovement*. Na začátku skriptu jsem vytvořil proměnnou typu *Rigidbody2D* s názvem *rdBody2D*, která ve funkci *Start* přistupuje k Rigibody 2D komponentě objektu. Funkce *Start* se provádí ihned po spuštění hry. Pokud se na Rigibody 2D aplikuje rychlost, objekt postavy se pohne. K vyvinutí rychlosti na objekt postavy jsem použil funkci *SmoothDamp*, která postupně mění vektor směrem k požadované cílové rychlosti za určitý čas. Pohyb je tímto způsobem velmi plynulý. Pro tuto funkci je potřeba určit současnou rychlost, čas vyhlazování pohybu, který je uložen v proměnné *movementSmoothing* a poté cílovou rychlost, kterou udržuje proměnná *targetVelocity*. Této proměnné se přiřadí nový *Vector2*. Za y se přiřadí výchozí ypsilon Rigidbody hodnota a za x dosazujeme vstup od uživatele, který nabývá hodnoty od -1 po 1, podle toho, jestli hráč stisknul šipku do leva či doprava a ukládá se do číselné proměnné *move*, což je jeden z parametrů funkce *Move*. Tento parametr určuje směr, kterým se postava

bude pohybovat. Druhým parametrem funkce *Move* je proměnná *jump*, která je typu *bool*. *Bool* může nabývat dvou hodnot, a to je pravda nebo nepravda.

Zdrojový kód 1: *Funkce Move pro pohyb postavy*

```
Počet odkazů: 1
public void Move(float move, bool jump) // pohyb hráče
{
        // pohyb hráče nalezením cílové rychlosti
        Vector3 targetVelocity = new Vector2(move * 10f, rdBody2D.velocity.y);
        // postupné vyhlazení pohybu do cílové rychlosti
        rdBody2D.velocity = Vector3.SmoothDamp(rdBody2D.velocity, targetVelocity, ref velocity, movementSmoothing);
        // když vstup pohybuje doprava a hráč se dívá doleva
        if (move > 0 && !facingRight)
        {
            // otočí hráče
            Flip();
            transform.Translate(0.4f, 0, 0);
        }
        // když input pohybuje doleva a hráč se dívá doprava
        else if (move < 0 && facingRight)
        {
            // otočí hráče
            Flip();
            transform.Translate(-0.4f, 0, 0);
        }

    // když hráč chce vyskočit
    if (grounded && jump)
    {
        // přidá vertikální sílu
        grounded = false;
        rdBody2D.AddForce(new Vector2(0f, jumpForce));
        timeBeforeGroundCheck = delayGroundCheck;
    }
}
```

(Zdroj: autor, 2021)

Dále se ve funkci *Move* kontroluje podmínka změny směru pohybu. Pokud je hodnota proměnné *move* větší než 0, tedy pokud hráč stiskl šipku doprava a proměnná *facingRight* je nepravda a hráč se tedy dívá doleva, postava se otočí pomocí funkce *Flip*. Příkaz *transform.Translate* ošetřuje drobný posun postavy, který nastane po jejím otočení. Zrcadlově platí to samé i pro případ, kdy hodnota *move* je menší než nula a hráč se dívá doprava.

Poslední částí funkce *Move* je skok postavy. Pokud je hráč na zemi, což určuje proměnná *grounded* a zároveň stisknul mezerník, o čemž nám dává informaci proměnná *jump*, hráč vyskočí (Příloha 5). Skok je realizován tak, že na Rigibody 2D se aplikuje vertikální síla. Tato síla je uložena v proměnné *jumpForce*.

Na další ukázce (Zdrojový kód 2) je úryvek kódu ze skriptu *PlastonosMovement*. V rámci funkce *Update*, která se spouští jednou během každého snímku se do proměnné *horizontalMove* ukládá hráčův vstup a násobí se rychlostí pohybu *runSpeed*. Hodnota této

proměnné se přiřadí parametru *Speed*, která je definována v Animatoru jako podmínka pro spuštění animace chůze a pokud dosáhne určité hodnoty, animace se spustí.

Zdrojový kód 2: *Vybrané funkce skriptu PlastonosMovement*

```
void Update()
{

    horizontalMove = Input.GetAxisRaw("Horizontal") * runSpeed; // pohyb = vstup se vynásobí rychlostí běhu

    animator.SetFloat("Speed", Mathf.Abs(horizontalMove)); // nastaví hodnotu parametru Speed v animátoru

    if (Input.GetButtonDown("Jump")) // pokud hráč stiskne mezerník, vyskočí
    {
        isJumping = true; // hráč skáče
        jump = true; // skok probíhá
        animator.SetBool("isJumping", true); // spustí se animace skoku
    }

    if (controller.IsPushing && !isJumping) // pokud je hráč v blízkosti kamene a neskáče
    {
        animator.SetBool("isPushing", true); // parametr isPushing je pravda - spustí se animace tlačení kamene
    }
    else
    {
        animator.SetBool("isPushing", false); // jinak se animace tlačení nespustí
    }
}
Počet odkazů: 0
public void OnLanding() // při přistání hráče na zemi
{
    animator.SetBool("isJumping", false); // parametr isJumping je nepravda, vypne se animace skoku
    isJumping = false; // hráč neskáče
}

Zpráva Unity | Počet odkazů: 0
void FixedUpdate()
{
    // pohyb hráče
    controller.Move(horizontalMove * Time.fixedDeltaTime, jump);
    jump = false; // při pohybu neprobíhá skok
}
```

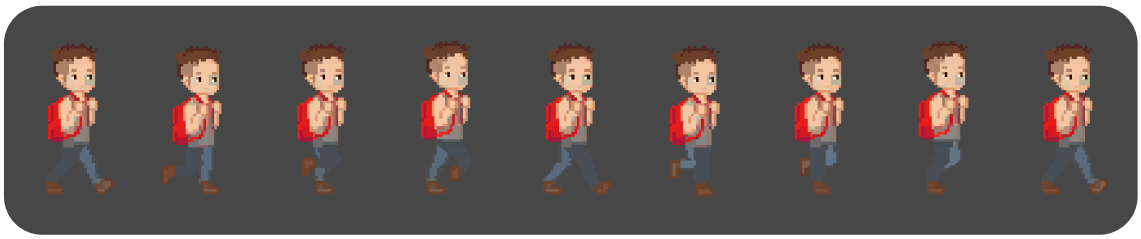
(Zdroj: autor, 2021)

Pokud hráč stiskne mezerník, hodnota proměnné *jump* a *isJupming* se změní na pravdu. Tato skutečnost se využívá dále ve skriptu. Poté se parametru *isJumping*, sloužícímu jako podmínka pro aktivování animace skoku, připíše pravdivá hodnota a spustí se animace skoku. Dále je ve funkci *Update* podmínka pro spuštění animace tlačení kamene. To se stane poté co vlastnost *IsPushing* (vrací informaci o tom, zda je hráč v blízkosti nějakého objektu kamene) je pravdivá a zároveň platí, že proměnná *isJumping* je nepravda. Hráč nemůže tlačit kamen a zároveň skákat.

Další funkcí v rámci skriptu *PlastonosMovement* je *OnLanding*. Ta je spojená s událostí *OnLandEvent* ze skriptu *PlastonosController*, která v podstatě zaznamenává moment, kdy hráč po provedení skoku dopadne zpátky na zem. Pokud se tak stane, animace skoku skončí.

Poslední funkcí, kterou zmíním je *FixedUpdate*. *Update* se spouští jednou za každý snímek a čas mezi jednotlivým snímky se může měnit například na základě výkonu počítače, zatímco *FixedUpdate* se spouští v závislosti na fyzickém enginu. Doba mezi každým provedením funkce je pevně stanovená. V rámci *FixedUpdate* by se měli provádět veškeré fyzické příkazy jako aplikování rychlosti atd. V mém případě se zde provádí funkce *Move* ze skriptu *PlastonosController*, kterou jsem popisoval dříve. Na obrázcích níže (Obrázky 10-12) jsou vidět snímky jednotlivých animací hlavní postavy.

Obrázek 10*: Animace chůze*



(Zdroj: autor, 2021)

Obrázek 11*: Animace skoku*

(Zdroj: autor, 2021)

Obrázek 12*: Animace tlačení kamene*

(Zdroj: autor, 2021)

7.3 Sbírané předměty

Pro vyobrazení plastových odpadků jsem vytvořil 13 různých obrázků, které představují předměty, jaké můžeme v lese najít. Od plastových lahví na vodu až po plastový sáček. Některé ze spritů jsou zobrazeny na následujícím obrázku (Obrázek 13).

Obrázek 13: *Sprity plastových odpadků*

(Zdroj: autor, 2021)

Každý z předmětů funguje jako Prefab. Objekt Prefabu samozřejmě obsahuje základní komponenty Transform a Sprite Renderer. Collider je v tomto případě čtvercový a předměty mají tzv. Trigger funkci. Pokud je nějaký objekt Trigger, znamená to, že v případě kolize s jiným objektem se spustí nějaká reakce. Tato reakce se implementuje pomocí funkce *OnTriggerEnter2D*. Ta definuje, co se stane, pokud collider daného objektu koliduje s colliderem objektu jiného. Důležité také je, že alespoň jeden z objektů musí obsahovat komponentu Rigibody. Naproti tomu funkce *OnTriggerExit2D* říká, co se stane, když objekty opustí prostor collideru toho druhého.

Ve skriptu *PlastonosController* jsem vytvořil funkci s názvem *AddCollectible*, která kontroluje, jestli hráč již sebral všechny plastové odpadky v úrovni (Zdrojový kód 3). První podmínka porovnává současný počet sebraných předmětů *currentNumOfCollectibles* a celkový počet předmětů potřebných k dokončení úrovně *totalNumOfCollectibles*. Pokud je momentální počet předmětů menší, proměnná *currentNumOfCollectibles* se inkrementuje o jedna.

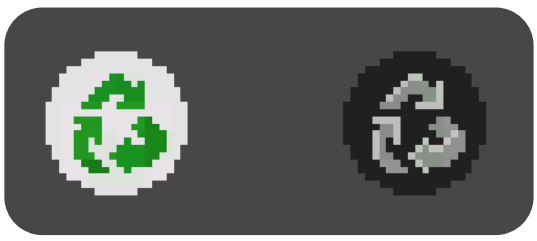
Funkce také obsahuje *switch* příkaz. Dle počtu aktuálně držených předmětů se mění obrázky ukazatelů v uživatelském rozhraní symbolizující zbývající počet předmětů k sebraní (Obrázek 14). Při kolizi hráče s plastovým odpadkem se obrázek nesebraného předmětu změní na sprite již sebraného předmětu. Proměnná *checks* je pole obrázků. S každým dalším předmětem, který hráč získal se díky indexu přistupuje k prvkům pole a mění se sprite daného prvku.

Zdrojový kód 3: *Funkce AddCollectible ze skriptu PlastonosController*

```
Počet odkazů: 1
public void AddCollectible() // přidá collectible
{
    if (currentNumOfCollectibles < totalNumOfColletctibles) // porovnává počet celkových collectibles a současný počet
    {
        currentNumOfCollectibles++; // při spuštění Triggeru přidá collectible
        switch (currentNumOfCollectibles) // dle počtu sebraných předmětů mění UI sprity ze symbolu 'nesebraného' předmětu na 'sebraný'
        {
            case 1:
                checks[0].gameObject.GetComponent<Image>().sprite = newImage;
                break;
            case 2:
                checks[1].gameObject.GetComponent<Image>().sprite = newImage;
                break;
            case 3:
                checks[2].gameObject.GetComponent<Image>().sprite = newImage;
                break;
            case 4:
                checks[3].gameObject.GetComponent<Image>().sprite = newImage;
                break;
            case 5:
                checks[4].gameObject.GetComponent<Image>().sprite = newImage;
                break;
        }
    }
}
```

(Zdroj: autor, 2021)

Obrázek 14: *Ukazatel sebraného předmětu (vlevo) a nesebraného předmětu (vpravo)*



(Zdroj: autor, 2021)

Na ukázce (Zdrojový kód 4) níže je vidět funkce *OnTriggerEnter2D* ze skriptu *Collectible*. Každý Prefab sběratelského předmětu tento skript obsahuje. Funkce má jeden parametr collision typu *Collider2D*, skrz který lze přistupovat k objektu, s kterým objekt se skriptem *Collectible* koliduje. V tělu funkce se jako první vytvoří objekt třídy *PlastonosController* s názvem *controller* a díky parmetru *collision* získáme komponentu *PlastonosController* kolidujícího objektu. V následující podmínce se ošetřuje, zda objekt *controller* není prázdný, jinými slovy, jestli kolidující předmět obsahoval komponentu

PlastonosController. Pokud objekt prázdný není, hráčovi se pomocí funkce *AddCollectible* zvýší počet získaných předmětů o jedna. Následně se přehraje zvukový efekt signalizující sebrání předmětu a objekt plastového odpadku se zničí.

Zdrojový kód 4*: Funkce OnTriggerEnter2D ze skriptu Collectible*

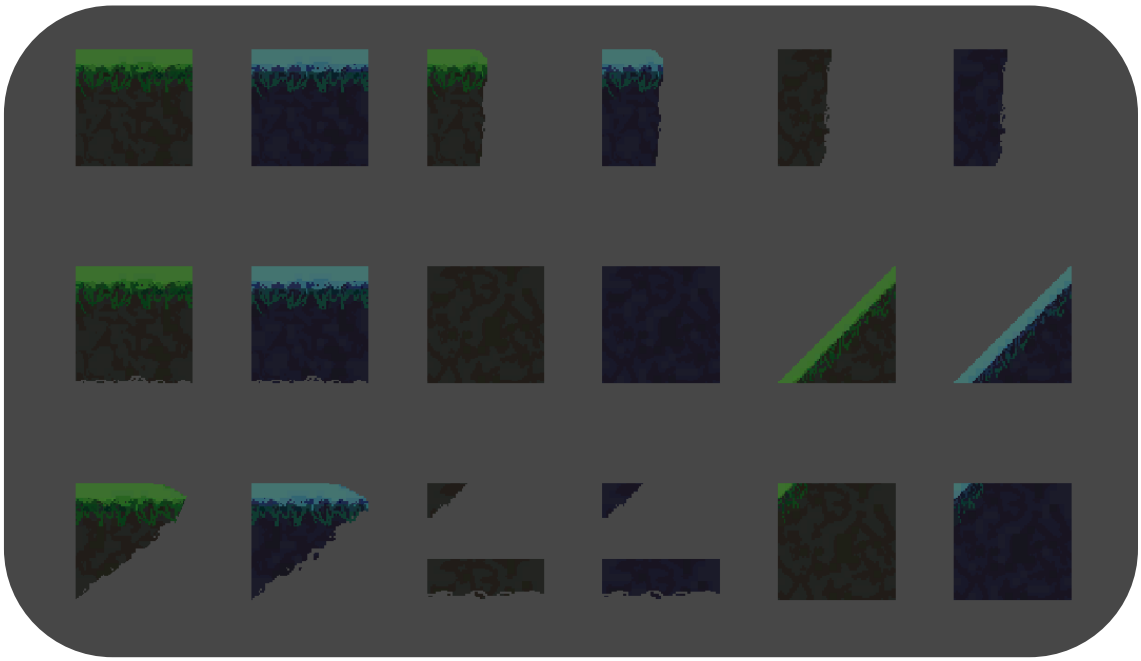
```
Zpráva Unity | Počet odkazů: 0
private void OnTriggerEnter2D(Collider2D collision) // spouští Trigger při kolizi s objektem
{   // vytvoří instanci PlayerController a získá komponentu
    PlastonosController controller = collision.GetComponent<PlastonosController>();
    if (controller != null) // kontroluje, zda není instatnce controller prázdná;
    {
        // pokud ne, přidá collectible, zahraje zvukový efekt a poté ji zničí
        controller.AddCollectible();
        FindObjectOfType<AudioManager>().Play("PickUp");
        Destroy(gameObject);
    }
}
```

(Zdroj: autor, 2021)

7.4 Prostředí

K vytvoření herního světa jsem použil tzv. tilemap techniku. Spočívá v tom, že s pomocí tzv. tiles neboli dlaždic se vytváří prostředí jednotlivých úrovní. Jednotlivé dlaždice, které na sebe navazují, se poskládají vedle sebe a v mém případě se z nich stanou platformy odlišných tvarů a různých velikostí. Celá mapa dlaždic (tilemap) má komponentu Tilemap Collider 2D, která zajišťuje, aby objekty nemohly dlaždicemi propadnout. Tvar jednotlivých colliderů je možné editovat v nastavení dlaždic podle jejich specifického tvaru. Dlaždice se importují do palety a z té se poté na základě potřeby vybírají a skládají do čtvercové mřížky, do tzv. Grid. Jedna mřížka má velikost 64 na 64 pixelů. Jelikož v se v průběhu videohry změní vizuál lesa ze zimního období na jarní, musel jsem vytvořit dva sety dlaždic. Zimní set má chladné barevné tóny, které mají vyvolávat ponurý pocit, naopak jarní set reprezentuje stav, kde se do lesa vrátil život, z toho důvodu je povrch dlaždic zelený. V rámci jednoho setu lze vytvořit dva hlavní typy platforem, přičemž délka a hloubka jednotlivých ostrůvků lze samozřejmě měnit díky návaznosti dílčích dlaždic. Sprity dlaždic jsou vidět na následujícím obrázku (Obrázek 15).

Obrázek 15: *Sprity pro vytváření platforem*



(Zdroj: autor, 2021)

Pro pohyblivé platformy jsem vytvořil GameObject s názvem *Platform*. Tento objekt se skládá ze tří spritů dlaždic. Uprostřed je dlaždice, kterou lze na obrázku výše (Obrázek 19) vidět v levém horním rohu a na každé straně objektu je dlaždice, která je vidět v levém dolním rohu obrázku. Dlaždice na levé straně je samozřejmě zrcadlově otočená. Tímto složením vznikne jeden velká platforma. Objektu se přidá komponenta Polygon Collider 2D, aby se objekt postavy mohl po plošině pohybovat.

Pohyb platformy zajišťuje skript *MovingPlatform*. Hlavní funkce tohoto skriptu lze vidět na následující ukázce (Zdrojový kód 5). Ve funkci *Start* se číselné proměnné *timer* přiřadí hodnota proměnné *changeTime*, která se dá upravit přímo v Unity Editoru a určuje, jak dlouho se bude platforma pohybovat, než změní směr. Uvnitř funkce *Update* se po spuštění hry začne hodnota *timer* snižovat v závislosti na čase. Pokud bude hodnota proměnné *timer* číslo 3, časovač poběží 3 vteřiny. Příkaz *if* říká, že pokud bude hodnota *timer* menší než nula, platforma změní směr skrz proměnnou *direction* a časovač se obnoví na původní hodnotu *changeTime*. Ve fukci *FixedUpdate* je implementován pohyb platformy. V závislosti na tom, jestli je proměnná *vertical* typu *bool* pravda či nepravda, se platforma pohybuje horizontálně nebo vertikálně. Jelikož je proměnná *vertical* přístupná přímo z Editoru, lze v

Unity nastavit i osa, po které se platforma pohybuje a díky již zmíněné proměnné *direction* také její počáteční směr.

Zdrojový kód 5*: Vybrané funkce skriptu MovingPlatform*

```
Zpráva Unity | Počet odkazů: 0
void Start()
{
    timer = changeTime; // proměnné časovač se přiřadí hodnota changeTime
}

Zpráva Unity | Počet odkazů: 0
void Update()
{
    timer -= Time.deltaTime; // odpočet časovače

    if (timer < 0) // po vypršení doby časovače se změní směr a časovači se obnoví doba,
                   // po kterou se bude platforma pohybovat daným směrem
    {
        direction = -direction;
        timer = changeTime;
    }
}

Zpráva Unity | Počet odkazů: 0
void FixedUpdate()
{
    // pohyb platformy
    Vector2 position = transform.position;

    if (vertical) // pokud je vertikální pohyb pravda, pohybuje se platforma po ose "y"
    {
        position.y = position.y + Time.deltaTime * speed * direction;
    }
    else // jinak se pohybuje platforma po ose "x"
    {
        position.x = position.x + Time.deltaTime * speed * direction;
    }

    transform.position = position;
}
```

(Zdroj: autor, 2021)

Dalším interaktivním prvkem v prostředí je nášlapné tlačítko. Má dva stavy, a to aktivní a neaktivní (Obrázek 16). Aktivování tlačítka je zobrazeno pomocí jednoduché animace a zvukového efektu. Tlačítko je spojené s dalším typem pohyblivé platformy. Tento objekt jsem nazval *PlatformOnStep* a funguje tak, že hráč musí zatížit tlačítko spojené s konkrétní platformou, aby se začala pohybovat. Hráč může tlačítko zatížit dvěma způsoby, a to, že si něj sám stoupne nebo na něj posune objekt kamene.

Obrázek 16: *Aktivní a neaktivní stav tlačítka*

(Zdroj: autor, 2021)

Funkcionalitu tlačítka zajišťuje skript s názvem *Step*. Aktivování tlačítka je řešeno opět skrz funkci *OnTriggerEnter2D* (Zdrojový kód 6) a deaktivování pomocí *OnTriggerExit2D*. Uvnitř podmínky *if* se kontroluje, jaký objekt s colliderem tlačítka koliduje. Toho se dosáhne pomocí funkce *CompareTag*. Uvnitř Unity Editoru lze jednotlivým objektům přiřadit tag neboli něco jako jmenovku. Při kolizi s objektem se porovná jeho tag a tag, který je definován jako parametr typu *string* funkce *CompareTag*. Pokud se tagy shodují, kód v těle se může provést. Hodnota proměnné *isActive* se změní na pravdu, stejně jako parametr Animatoru a animace se spustí. Proměnná *playerActive* a podmínka na dalším řádku ošetřují to, aby zvukový efekt aktivace nebyl spuštěn dvakrát v případě, kdy s tlačítkem koliduje postava i objekt kamene najednou. Obdobně jako v první podmínce pro kolizi s hráčem je řešena i podmínka pro kolizi s objektem kamene.

Zdrojový kód 6: *Funkce OnTriggerEnter2D ze skriptu Step*

```
Zpráva Unity | Počet odkazů: 0
private void OnTriggerEnter2D(Collider2D collision) // při kolizi spouští Trigger
{
    if (collision.gameObject.CompareTag("Player")) // pokud kolidující objekt má tag "Player"
    {
        isActive = true; // tlačítko je aktivní
        animator.SetBool("isActive", true); // animace pro aktivní tlačítko se spustí
        playerActive = true; // hráč stojí na tlačítku
        if (!stoneActive) // pokud hráč stojí na tlačítku, ale kámen ne
        {
            FindObjectOfType<AudioManager>().Play("Step"); // spustí se zvukový efekt tlačítka
        }
    }
    if (collision.gameObject.CompareTag("Rock")) // pokud kolidující objekt má tag "Rock"
    {
        isActive = true; // tlačítko je aktivní
        animator.SetBool("isActive", true); // animace pro aktivní tlačítko se spustí
        stoneActive = true; // kámen stojí na tlačítku
        if (!playerActive) // pokud kámen stojí na tlačítku, ale hráč ne
        {
            FindObjectOfType<AudioManager>().Play("Step"); // spustí se zvukový efekt tlačítka
        }
    }
}
```

(Zdroj: autor, 2021)

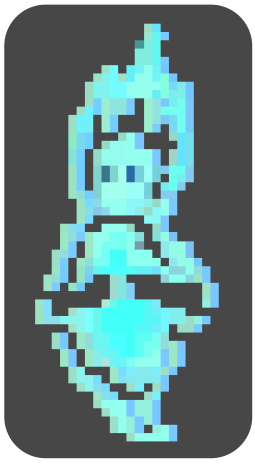
Jako cílový objekt, kterého je třeba dosáhnout pro dokončení úrovně jsem vytvořil sprite dřevěné lesní brány se symbolem recyklace (Obrázek 17). Posun do další úrovně funguje také na principu kolize, díky funkci *OnTriggerEnter2D*. V těle funkce je podmínka *if*, která kontroluje, zda má hráč v držení potřebný počet předmětů k dosažení další úrovně. Pokud ano, třída *SceneManager*, která spravuje scény v Unity, pomocí funkce *LoadScene* nahraje další scénu v pořadí. Pokud ne, hráčovi se na obrazovce pomocí komponenty Canvas zobrazí text, který ho o této skutečnosti informuje.

Dále jsem také vytvořil dva další sprity postav, které se ve hře vyskytují. Ochránce lesa a přírody, který hráče videohrou provází a sděluje mu různé informace (Obrázek 20) a pak také lesního ducha, který slouží jako jakási nápověda (Obrázek 18). V jednotlivých úrovních s hráčem komunikují pomocí dialogových oken (Příloha 3).

Obrázek 17: *Sprite brány do další úrovně*

(Zdroj: autor, 2021)

Obrázek 18: *Sprite lesního ducha*



(Zdroj: autor, 2021)

Obrázek 19: *Sprite kamene*

(Zdroj: autor, 2021)

Obrázek 20: *Sprite ochránce lesa*

(Zdroj: autor, 2021)

Kamen (Obrázek 19) je jediný objekt ve hře mimo hlavní postavu, který obsahuje komponentu Rigibody 2D. Je to z toho důvodu, aby s ním hráč mohl díky gravitaci manipulovat a vyřešit tak hádanky v různých výškách úrovně. Na následujícím obrázku je úryvek kódu z funkce *FixedUpdate* ve skriptu *PlastonosController*. Do proměnné *isPushing* (Zdrojový kód 7) se ukládá informace o tom, zda je hráč v blízkosti objektu kamene. Je k tomu využitá funkce třídy *Physics2D* s názvem *OverlapCircle*, která v podstatě říká, zda se postava na specifickém místě jejího objektu dotýká objektu kamene, který má v Unity Editoru nastavenou určitou masku. Pozice, kde se kontakt kontroluje je uložená v proměnné *handPos.position*. Průměr okruhu, v jakém se kontakt kontroluje je uložen v proměnné *checkHandRadius* a má střed právě v místě *handPos.position*. Maska vrstvy, která specifikuje povrch kamene je uložena v proměnné *whatIsRock*. Proměnnou *isPushing* vrací vlastnost *IsPushing*, kterou jsem zmiňoval při spouštění animace tlačení kamene a slouží právě k tomuto účelu.

Zdrojový kód 7*: Kontrola blízkosti objektu kamene*

```
isPushing = Physics2D.OverlapCircle(handPos.position, checkHandRadius, whatIsRock);
```

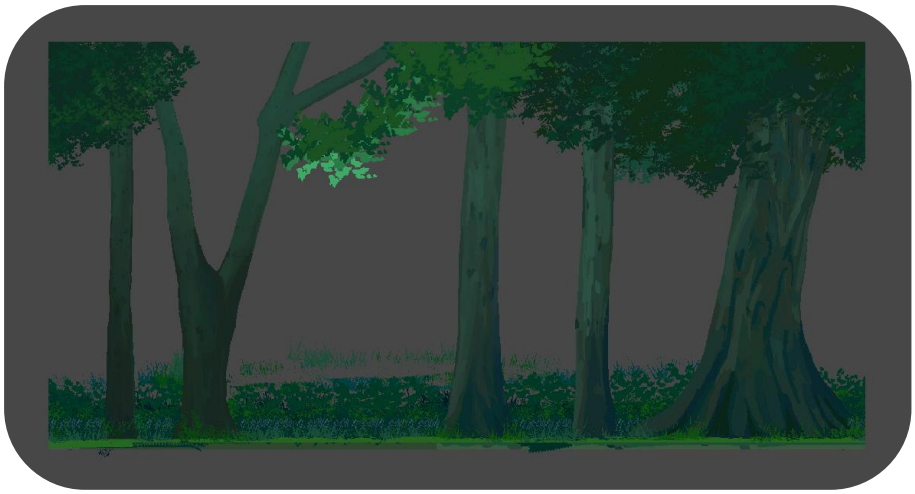
(Zdroj: autor, 2021)

Co se týče grafického pozadí jednotlivých úrovní, je to zcela jistě část práce, která zabrala nejvíce času a byla nejvíce náročná. Jedná se o scenérii lesa. Je rozdělena do pěti vrstev, které jsou složené do sebe a společně tak vytváří celkový obraz (Obrázky 21-26). Ten je proveden ve dvou variantách, stejně jako sety dlaždic, a to zimní a jarní. První vrstva obsahuje čtvercový collider, který zajišťuje, aby skrz něj postava a kameny nepropadly.

Obrázek 21*: První vrstva pozadí*

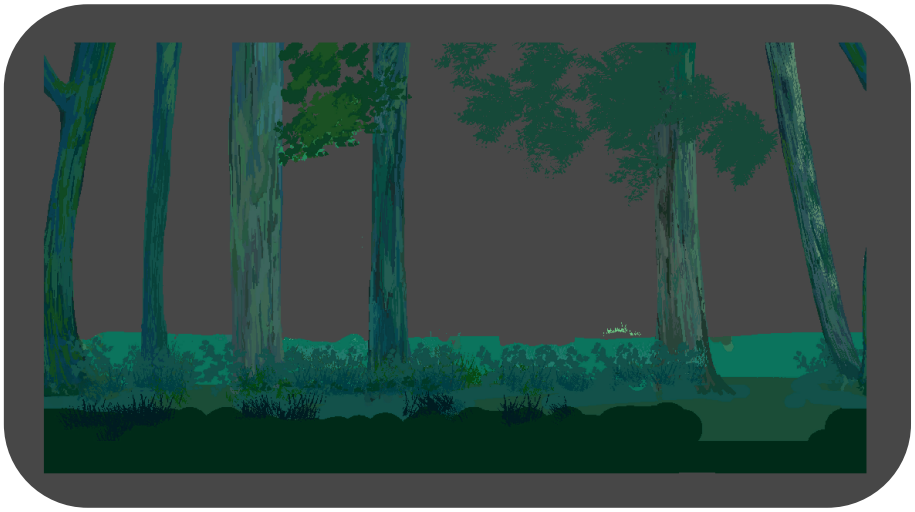
(Zdroj: autor, 2021)

Obrázek 22: *Druhá vrstva pozadí*



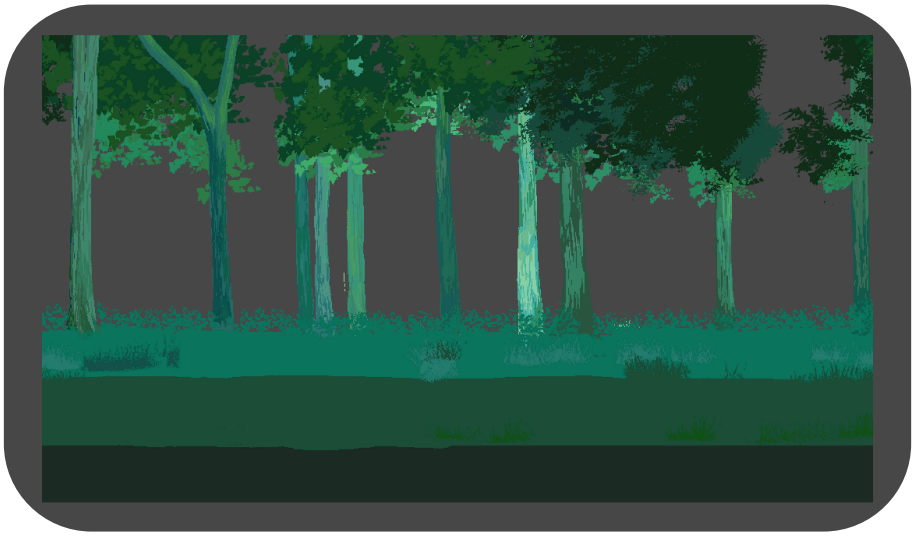
(Zdroj: autor, 2021)

Obrázek 23: *Třetí vrstva pozadí*



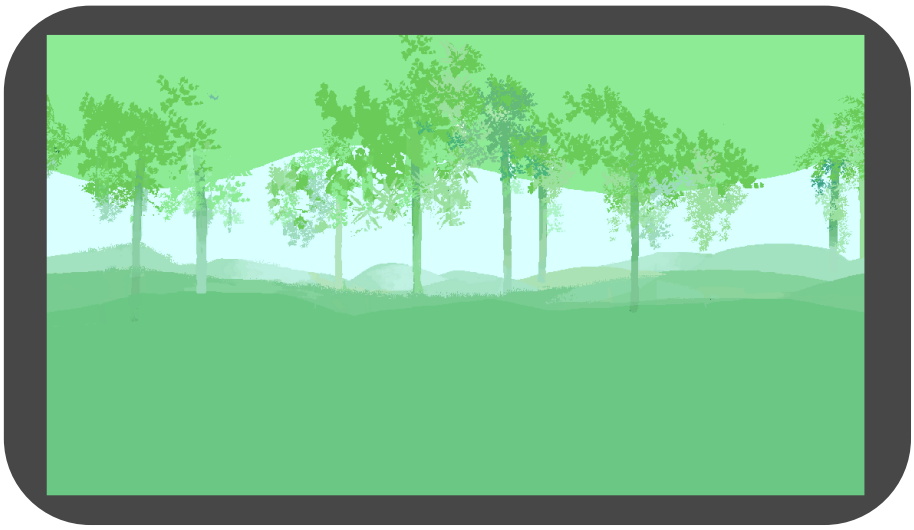
(Zdroj: autor, 2021)

Obrázek 24: *Čtvrtá vrstva pozadí*



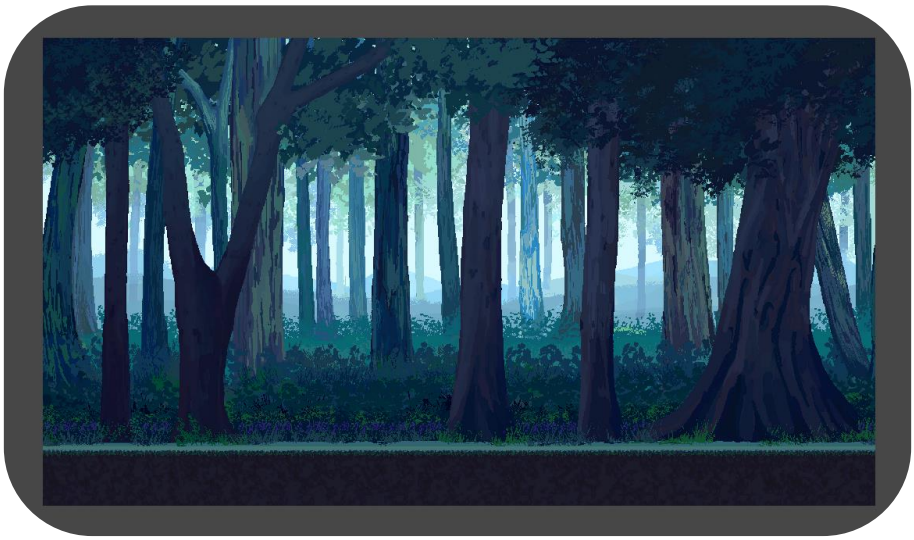
(Zdroj: autor, 2021)

Obrázek 25: *Pátá vrstva pozadí*



(Zdroj: autor, 2021)

Obrázek 26: *Pozadí úrovně - zimní varianta*



(Zdroj: autor, 2021)

Po spuštění videohry se hráč dostane do hlavního menu (Příloha 1). Zde si může zvolit novou hru, nastavit hlasitost hudby nebo zvukových efektů či vybrat konkrétní úroveň v závislosti na jeho herním postupu a finálně také vypnout aplikaci. Ve výběru úrovní se postupně zpřístupňuje možnost znovu spustit úrovně, které už byly pokořeny. Veškeré nastavení a postup se ukládají díky funkci *PlayerPrefs*, která umožňuje ukládat číselné hodnoty typu *float*. Po opětovném spuštění se tedy hráč nemusí bát, že by začínal kompletně znovu. Při pozastavení hry během hraní pomocí tlačítka Escape se hráčovi zobrazí tzv. „pause menu“. Zde má opět možnost provést zvukové nastavení, odejít do hlavního menu, vypnout aplikaci, znovu spustit aktuální úroveň nebo ve hře pokračovat.

8 Závěr

V rámci této práce jsem vytvořil 2D sidescrolling plošinovou hru pro platformu Windows s využitím nástrojů, které herní enigne Unity poskytuje. Myslím si, že po přečtení této práce by mohl mít začínající programátor, vývojář či nadšenec dobrý základní přehled o tom, co vývoj jednoduché 2D videohry obnáší.

S vývojem videohry jsem před začátkem práce neměl téměř žádné zkušenosti, a tak jsem se ponořil do studia odborné literatury a zhlédnul jsem nespočet návodů dostupných na internetu. I přes to, že mě tento čas strávený studiem velmi obohatil o cenné informace, jsem se během práce potýkal s problémy. Ty byly způsobené především minimálními zkušenostmi s herním enginem Unity a poté také nulovou znalostí a zkušenostmi s kreslením a animací, což mi zabralo pravděpodobně nejvíce času. Několikrát jsem začínal znovu, protože jsem nebyl spokojen s tím, jakým směrem se videohra udává.

Díky znalostem, které jsem získal při studiu na vysoké škole, jsem ale neměl problém se skriptováním v jazyku C#. S třídami a funkcemi, které Unity poskytuje, jsem se seznámil poměrně rychle. I přesto bylo ale třeba podrobně projít obsáhlou dokumentaci, kterou Unity poskytuje, abych si byl jistý, že konkrétní nástroje užívám správně.

S výsledkem jsem spokojen a určitě plánuji na hře dále pracovat, vylepšovat ji a rozšiřovat o nové funkce a další úrovně. Práce mě ale také motivovala začít pracovat na nových projektech s odlišnou tématikou, funckionalitou nebo grafickým stylem. Myslím si, že zkušenosti získané při tvorbě této hry budou stěžejní a velmi nápomocné při návrhu a realizaci mnohem komplexnější a zábavné videohry, kterou budu vytvářet příště.

Summary

This work is about the development of a 2D video game in a Unity game engine. Unity allows developers to create and export games for many platforms. The thesis theoretically describes the process of creating a video game and different types of video games and their use in education and finally the Unity game engine itself. These principles are then applied in the practical part in the development of a specific 2D video game for Windows using the C# programming language. Genre of the game is sidescroller which is a subgenre of platformer games. The game focuses on educating players and especially children about the issue of plastic pollution in a fun and interactive way.

Key words

Unity, video game, 2D, programming, development, C#

9 Seznam zdrojů

Attentat 1942. (n.d.-a). *Attentat 1942* [obrázek]. Dostupné z: http://attentat1942.com/press/#images

Attentat 1942. (n.d.-b). Attentat 1942 - Press Kit. [vid. 2020-03-23]. Dostupné z: http://attentat1942.com/press/

Blahuta, J. (2017). *Algoritmizace* [PDF]. Dostupné z: https://www.mvso.cz/files/algoritmizace-studijni-text.pdf

Bory, P. (2016). *C# bez přechdozích znalostí*. Brno, Česko: Computer Press.

Brand, P. (2017). Mohou nás počítačové hry učit? [vid. 2020-03-23]. Dostupné z: https://spomocnik.rvp.cz/clanek/21553/MOHOU-NAS-POCITACOVE-HRY-UCIT.html

Danel, R. (2012). *Objektový přístup k analýze a návrhu IS*. Dostupné z: http://homel.vsb.cz/~dan11/aps/Objektovy%20pristup%20k%20navrhu%20systemu.pdf

Ferro, L. S., & Sapio, F. (2018). *Unity 2017 2D Game Development Projects: Create three interactive and engaging 2D games with Unity 2017* [PDF] (2nd Revised edition). Dostupné z: https://www.packtpub.com/product/unity-2017-2d-game-development-projects/9781786460271

Gaider, D. (2016, 15. sprpen). Do you want to write video games? [vid. 2020-03-22]. Dostupné z: https://www.polygon.com/2016/8/15/12455728/how-to-get-a-job-writing-games-maybe

Godbold, A., & Jackson, S. (2016). *Mastering Unity 2D Game Development - Second Edition* [PDF] (2nd Revised edition). Dostupné z: https://www.packtpub.com/product/mastering-unity-2d-game-development-second-edition/9781786463456

Gregory, J. (2018). *Game Engine Architecture, Third Edition (3rd ed.)*. Oxford, Velká Británie: Taylor & Francis Ltd.

Grygar, A. (2011). *Design počítačové hry, design v kontextu tvorby počítačových her* (Bakalářská práce). Univerzita Tomáše Bati ve Zlíně. Dostupné z: https://digilib.k.utb.cz/bitstream/handle/10563/16384/grygar_2011_dp.pdf?sequence=1&isAllowed=y

IRONKAGE. (2020, August 14). *Game Engines Comparison* [obrázek]. Dostupné z: https://medium.com/@ironkage/game-engines-review-comparison-b0a3017ca128

Janíčková, N. (2009). *Vzdělávací potenciál 3D počítačových her*. (Bakalářská práce). Univerzita Karlova. Dostupné z: https://dspace.cuni.cz/bitstream/handle/20.500.11956/30952/BPTX_2008_1_11240_0_247604_0_63541.pdf?sequence=1&isAllowed=y

Jenkins, T. (2017, 19. května). *The Oregon Trail* [obrázek]. Dostupné z: https://medium.com/the-coastline-is-quiet/why-the-oregon-trail-is-one-of-the-most-realistic-video-games-ever-361b7ef47f20

Jirkovský J. (2011). *Game Industry: Vývoj počítačových her a kapitoly z herního průmyslu*. Praha: D.A.M.O.

KQED. (2012, 25. července). Video Game Portal Enters the Classroom. [vid. 2020-03-23]. Dostupné z: https://www.kqed.org/mindshift/22923/video-game-portal-enters-the-classroom

Lavrinčík, J. (2018). *Operační systémy* [PDF]. Dostupné z: https://www.mvso.cz/files/operacni-systemy.pdf

Mathur, N. (2018, 20. září). Best game engines for Artificial Intelligence game development. [vid. 2020-03-20]. Dostupné z: https://hub.packtpub.com/best-game-engines-for-ai-game-development/

New Game Network. (2011). *Portal 2* [obrázek]. Dostupné z: https://www.newgamenetwork.com/media/5002/portal-2/

Pavlovic, D. (2020). Video Game Genres: Everything You Need to Know. [vid. 2020-03-22]. Dostupné z: https://www.hp.com/us-en/shop/tech-takes/video-game-genres

Pepple, S. (2018, 1. září). Why Did All Children of a Certain Age Play Oregon Trail? [vid. 2020-03-23]. Dostupné z: from https://medium.com/s/story/why-did-all-children-of-a-certain-age-play-oregon-trail-6a53e27e83d8

Pereira, V. (2014). *Learning Unity 2D Game Development by Example* [PDF]. Dostupné z: https://www.packtpub.com/product/learning-unity-2d-game-development-by-example/9781783559046

Pickell, D. (2019). The 7 Stages of Game Development. [vid. 2020-03-22]. Dostupné z: https://learn.g2.com/stages-of-game-development

Retro Gamer. (2014). *Magnavox Odyssey* [obrázek]. Dostupné z: https://www.retrogamer.net/profiles/developer/the-legacy-of-ralph-baer/

Rogers, S. (2014). *Level Up!: The Guide to Great Video Game Design*. New Jersey, USA: John Wiley & Sons.

Saarelainen, T., & Pakarinen, M. (2013). *2D Game Development With Unity 3D* (Disertační práce). Karelia University of Applied Sciences. Dostupné z: https://core.ac.uk/download/pdf/38099012.pdf

Statista. (2020). *Most used programming languages among developers worldwide, as of early 2020* [graf]. Dostupné z: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

Stegner, B. (2020, 7. října). 2D Games vs. 3D Games: What Are the Differences? [vid. 2020-03-21]. Dostupné z: https://www.makeuseof.com/2d-games-vs-3d-games-differences/

Studytonight Technologies. (n.d.-a). Different Genres of Game. [vid. 2020-03-21]. Dostupné z: https://www.studytonight.com/3d-game-engineering-with-unity/genres-of-game

Studytonight Technologies. (n.d.-b). What is a Game Engine?. [vid. 2020-03-20]. Dostupné z: https://www.studytonight.com/3d-game-engineering-with-unity/game-engine#

Šveigr, M. (2018). *Platformová 2D hra pro OS Android v Unity3D* (Bakalářská práce). České vysoké učení technické. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/76830/F8-BP-2018-Sveigr-Michal-thesis.pdf?sequence=-1&isAllowed=y

Tavinor, G. (2008). Definition of Videogames. [vid. 2020-03-20]. Dostupné z: https://quod.lib.umich.edu/c/ca/7523862.0006.016/--definition-of-videogames?rgn=main;view=fulltext

Unity Technologies. (n.d.-a). Unity - Manual: Rigidbody 2D. [vid. 2020-03-23]. Dostupné z https://docs.unity3d.com/Manual/class-Rigidbody2D.html

Unity Technologies. (n.d.-b). Unity - Manual: Audio Source. [vid. 2020-03-23]. Dostupné z https://docs.unity3d.com/Manual/class-AudioSource.html

Unity Technologies. (n.d.-c). Unity - Manual: Transforms. [vid. 2020-03-23]. Dostupné z: https://docs.unity3d.com/Manual/class-Transform.html

Unity Technologies. (n.d.-d). Canvas - Unity UI. [vid. 2020-03-23]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/class-Canvas.html

Unity Technologies. (n.d.-e). Unity - Scripting API: Vector3.SmoothDamp. [vid. 2020-03-23]. Dostupné z: https://docs.unity3d.com/ScriptReference/Vector3.SmoothDamp.html

Unity Technologies. (n.d.-f). Unity - Scripting API: Tilemap. [vid. 2020-03-23]. Dostupné z: https://docs.unity3d.com/ScriptReference/Tilemaps.Tilemap.html

Unity Technologies. (n.d.-g). Unity - Manual: Particle systems. [vid. 2020-03-23]. Dostupné z https://docs.unity3d.com/Manual/ParticleSystems.html

Unity Technologies. (n.d.-h). Unity - Manual: Colliders. [vid. 2020-03-23]. Dostupné z https://docs.unity3d.com/Manual/CollidersOverview.html

Unity Technologies. (2017a, 10. květen). Unity - Manual: Sprite Renderer. [vid. 2020-03-23]. Dostupné z https://docs.unity3d.com/Manual/class-SpriteRenderer.html

Unity Technologies. (2017b, 21. listopad). Unity - Manual: Animator Controller. [vid. 2020-03-23]. Dostupné z https://docs.unity3d.com/Manual/class-AnimatorController.html

Unity Technologies. (2018a, 10. květen). Unity - Manual: Camera. [vid. 2020-03-23]. Dostupné z https://docs.unity3d.com/Manual/class-Camera.html

Unity Technologies. (2018b, 19. březen). Unity - Manual: Creating and Using Scripts. [vid. 2020-03-23]. Dostupné z: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

Widitiarsa, A. (2019). *Video Games as Tools for Education*. Příspěvek byl prezentován na konferenci Journal of Game, Game Art, and Gamification, Jakarta. Dostupné také z: https://doi.org/10.5860/choice.189403

Wolf, M. J. P. (2007). *The Video Game Explosion: A History from PONG to PlayStation and Beyond.* Dostupné z: https://books.google.cz/books?id=XiM0ntMybNwC&printsec=frontcover&hl=cs#v=onepage&q&f=false

10 Seznam obrázků

11 Seznam ukázek zdrojového kódu

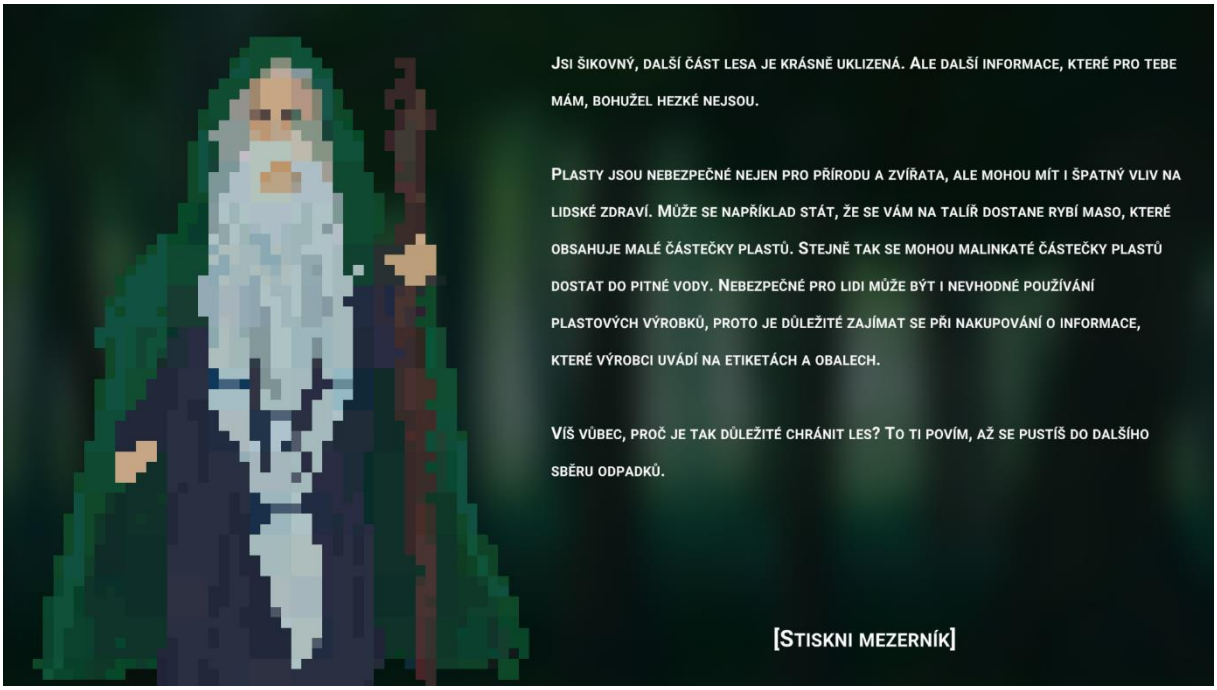
12 Seznam příloh

13 Přílohy

Příloha 1*: Hlavní menu*

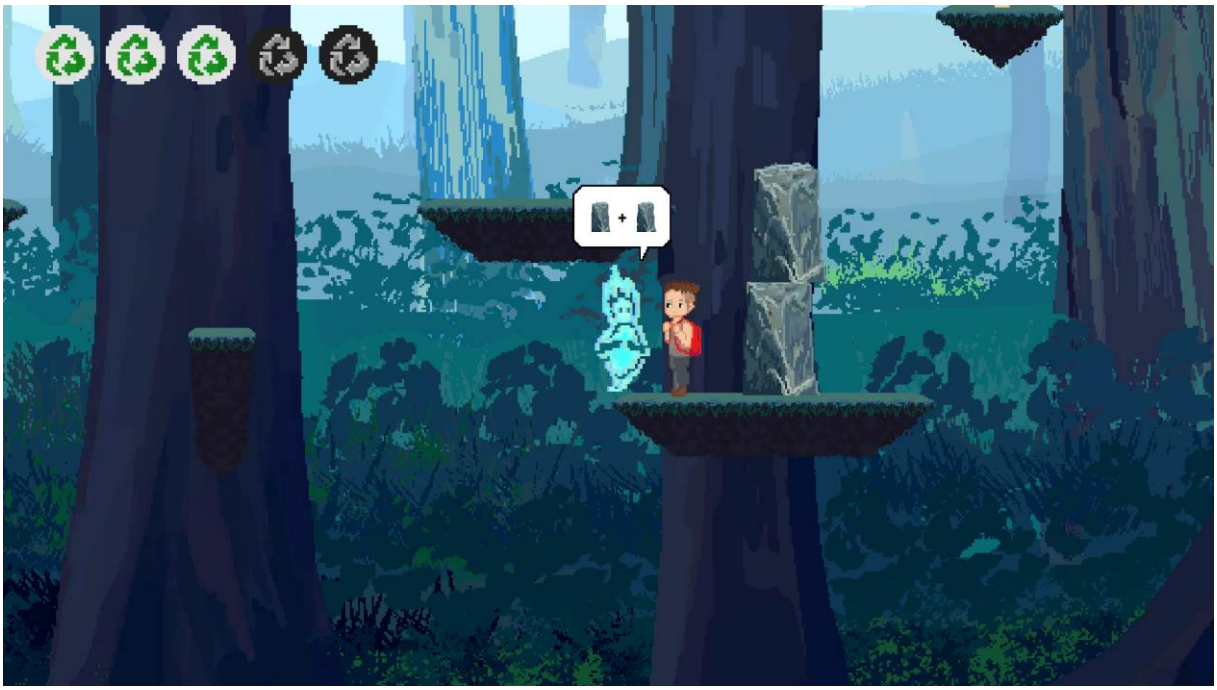
(Zdroj: autor, 2021)

Příloha 2*: Scéna mezi úrovněmi*



(Zdroj: autor, 2021)

Příloha 3: *Ukázka dialogového okna*

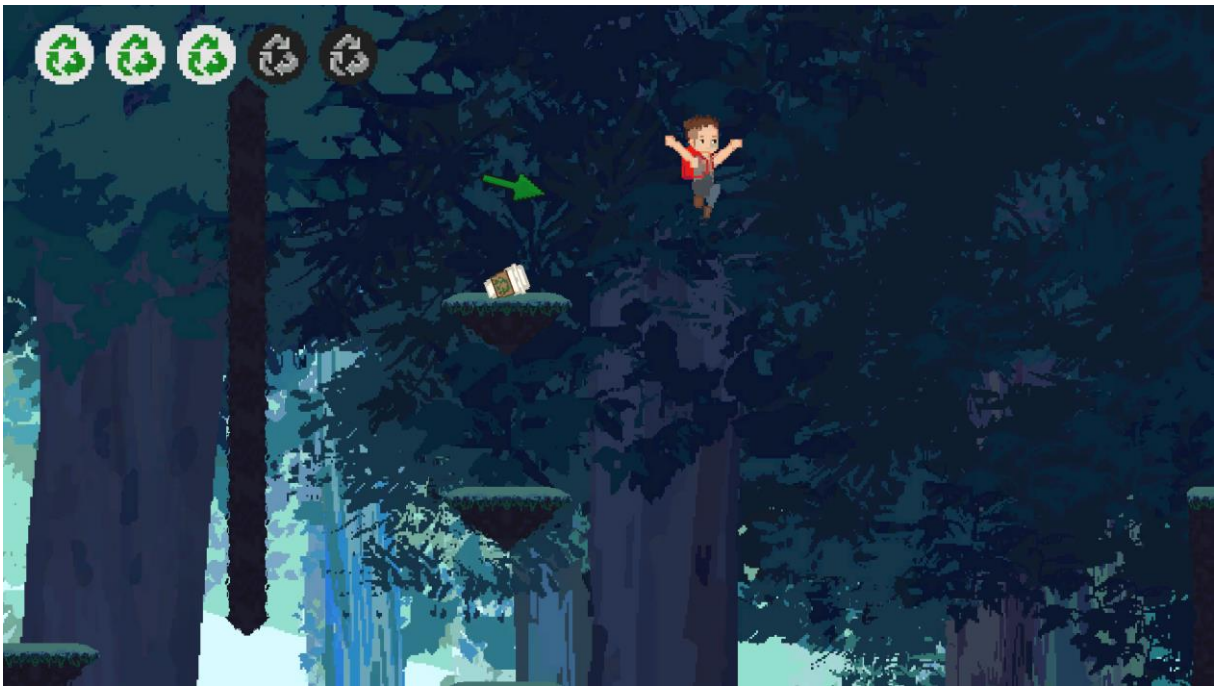


(Zdroj: autor, 2021)

Příloha 4: *Ukázka animace tlačení kamene*

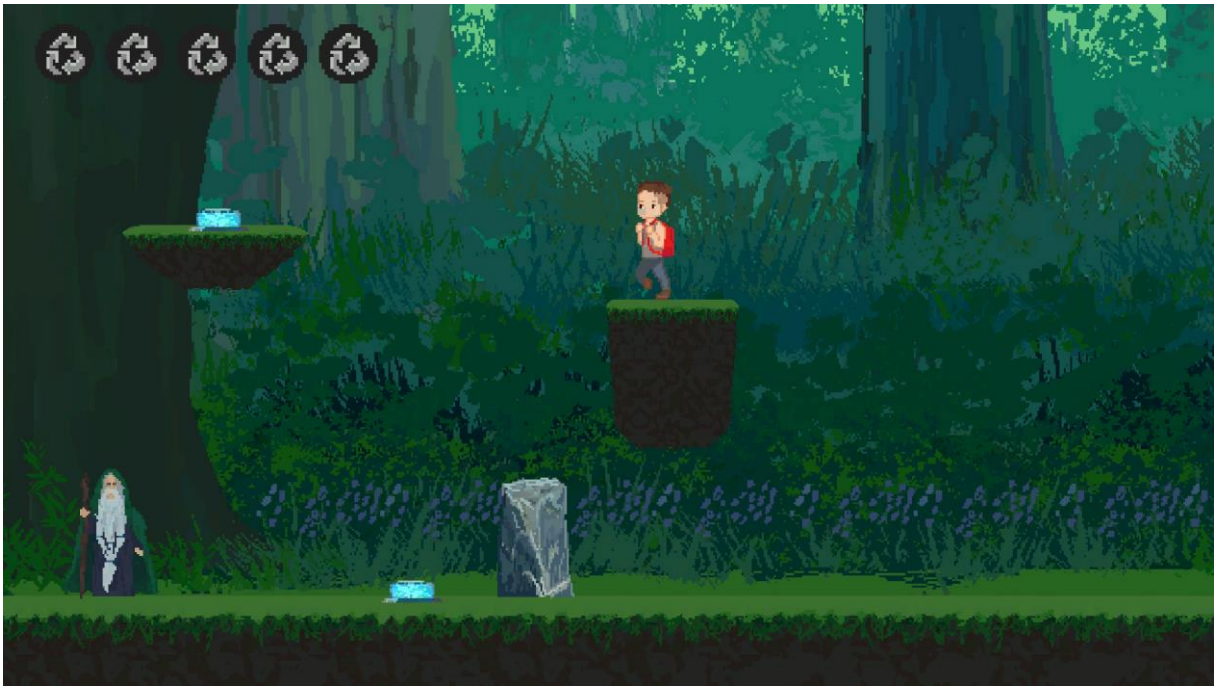
(Zdroj: autor, 2021)

Příloha 5: *Ukázka animace skoku*



(Zdroj: autor, 2021)

Příloha 6: *Ukázka nášlapného tlačítka a pohyblivých platforem*



(Zdroj: autor, 2021)