UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Jan Mesarč

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Vývoj multiplatformní hry v Unity3D

Jan Mesarč

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Mesarč**
Osobní číslo:	**I14141**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Vývoj multiplatformní hry v Unity3D**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit multiplatformní hru v nástroji Unity3D. Hra by měla být dostupná minimálně pro platformy Windows, iOS, Android.

V teoretické části bude popsána problematika tvorby hry v nástroji Unity3D a problémy, které je nutné vyřešit při tvorbě multiplatformní hry.

V praktické části bude implementována samotná hra. Tématem hry bude hráčem ovládaná kosmická loď. Hráč se bude muset bránit proti okolním asteroidům a UFO. Hra by měla rovněž obsahovat další omezující prvky (nutnost udržovat palivo, zdraví kosmické lodi, ...) či vylepšující prvky (vylepšení zbraně hráče, aj.).

Rozsah grafických prací:

Rozsah pracovní zprávy: **30-40 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

GOLDSTONE, Will. Unity 3.x game development essentials: game development with C# and Javascript. 2nd ed. Birmingham, UK: Packt Publishing, 2011. ISBN 978-184-9691-444.

NAGEL CH. et al. C# 2008. Programujeme profesionálně. Brno, 2009. ISBN 978-80-251-2407-7.

Vedoucí bakalářské práce: **Ing. Roman Diviš**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Mgr. Josef Horálek, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 11. 05. 2017

……

Jan Mesarč

PODĚKOVÁNÍ

V první řadě bych rád poděkoval vedoucímu práce Ing. Romanu Divišovi, za odborné vedení mé bakalářské práce, za trpělivost, ochotu a v neposlední řadě za cenné rady a připomínky. Dále bych rád poděkoval mojí rodině, která mě neúnavně podporuje po celou dobu mých studií.

ANOTACE

Bakalářská práce se zabývá problematikou vývoje multiplatformní hry. Cílem teoretické části je seznámit čtenáře s problematikou vývoje hry, popis problematiky přizpůsobení hry více platformám, seznámení s nástrojem Unity3D. Praktickou částí bakalářské práce je samotná hra, která demonstruje využití nástroje Unity3D pro tvorbu multiplatformní hry, jenž je určena pro mobilní zařízení.

KLÍČOVÁ SLOVA

Vývoj her, multiplatformní vývoj, Unity3D, C#, iOS, Android, Windows 10

TITLE

Development of multiplatform game in Unity3D

ANNOTATION

Bachelor thesis deals with the development of multi-platform games. The theoretical part is to introduce the issues of game development, description of problems adapting more games platforms, familiarity with the tool Unity3D. The practical part of the thesis itself is a game that demonstrates the use of tools for creating multiplatform Unity3D game, which is designed for mobile devices.

KEYWORDS

Game development, multiplatform development, Unity3D, C#, iOS, Android, Windows 10

OBSAH

SEZNAM ZKRATEK A ZNAČEK

2D	dvoudimenzionální
3D	trojdimenzionální
MMO	Hra obrovského počtu hráčů (anglicky Massively Multiplayer Online)
IDE	Integrated development environment (česky Vývojové prostředí)
AI	Umělá inteligence (anglicky Artificial Intelligence)
AAA	Označení pro hry s nejvyšším rozpočtem na vývoj, vysoký ekonomický risk
PC	Osobní počítač (anglicky Personal Computer)

ÚVOD

Tématem mé bakalářské práce je vývoj multiplatformní hry pomocí nástroje Unity3D. Vývoj her je mým velkým koníčkem, kterému se věnuji již od střední školy. Je to dynamické odvětví, které se neustále rozrůstá a také je náročné, protože spojuje hodně profesí, například programátory, grafiky, animátory, zvukaře a další. Herní vývoj skrývá hodně úskalí a tvrdé práce, proto si kladu za cíl této práce popsat celý proces vývoje hry od počátku až k hotovému produktu.

Mým úkolem bylo navrhnout a vytvořit hru, především herní logiku. Jelikož jsem především programátor tento úkol se mně nebo mi povedl, proto zvukové materiály byly použity z volně dostupné databáze freesound.org [11]. Grafické materiály jsou výsledkem práce mého kolegy Petra Drahoše.

Kompletní vývoj herního titulu je velmi náročný proces, zejména u špičkových (AAA) titulů. Prvním krokem k úspěšné hře je vytvoření kvalitního konceptu. Herní designér poté sestaví tzv. design dokument, který popisuje celou hru (herní postavy, grafiku hry, jednotlivé úrovně, vlastnosti hráče, zbraní atd.). Tento rozsáhlí dokument (řádově až tisíce stran) se zpracovává několik měsíců, protože je kladen důraz, aby zahrnoval veškeré aspekty herního titulu. Jakmile jsou veškeré informace poskládány a koncept hry vytvořen, nastupuje na řadu vývoj.

Proces tvorby můžeme dělit na různé etapy. Prvním a velice důležitým krokem je výběr herního engine nebo vytvoření vlastního. Tvorba vlastního herního engine je velice náročná a nákladná věc, proto je pro začínající nebo malá herní studia mnohem výhodnější vybrat z dostupných herních engine. Navíc mnoho z nich je již ke stažení zdarma. Dále je potřeba vytvořit modely, textury, animace, hudbu, zvuky, skripty a mnoho dalších. Proto je herní průmysl velmi náročný, protože k úspěšným a větším titulům je potřeba mnoho lidí a zdrojů.

Multiplatformní vývoj je v dnešní době velice rozšířen z důvodu mnoho platforem na trhu. Obnáší své klady i zápory. Nativní vývoj aplikací dosahuje lepšího výkonu, nižší náročnosti na danou platformu, ale zároveň je potřeba vývoj opakovat pro každou platformu, což je nákladné (více programátorů, více zdrojů). Naproti tomu multiplatformní vývoj nám šetří čas a drtivá většina kódu bývá shodná pro všechny platformy, kdy cenou za tento komfort je nižší výkon oproti nativním aplikacím.

Teoretická část této práce vás provede postupně každou etapou vývoje herního titulu. Vysvětlíme si postup od návrhu, konceptu a výběru technologií až po samotnou implementaci, otestování a sestavení hry.

Praktickou částí bakalářské práce je herní titul *Rocket of Milky Way*. Titul jsem vytvořil jako ukázku práce s nástrojem *Unity3D*. Jedná se o 2D arkádu, kde hráč je představován vesmírnou lodí, která proplouvá vesmírem. Cílem hry je doletět co nejdále. Hráč si musí hlídat úroveň paliva, které postupně ubývá. Doplnit ji může sbíráním předmětů. Dále hráč musí sestřelovat asteroidy a nepřátelské mimozemšťany. Odměnou za sestřelování nepřátelských objektů je vylepšení hráčovi rakety. V následujících kapitolách naleznete řešení problematiky vývoje her.

1 VÝVOJ MULTIPLATFORMNÍ HRY

Úkolem této bakalářské práce je vyvinout multiplatformní hru. Proces vývoje herního titulu se skládá z mnoha dílčích kroků. Postupným skládáním jednotlivých kroků lze dosáhnout úspěšného herního titulu, který má potenciál uspět. Práce se bude zabývat problematikou multiplatformního vývoje pomocí nástroje *Unity3D*, jak navrhnout herní koncept a jak tento koncept uskutečnit.

Na začátku vývoje hry je potřeba vytvořit abstraktní koncept, poté koncept upřesnit a rozšířit. Dalším krůčkem je design herního titulu. Grafika je vizitkou videohry, proto je důležité vědět, jak s ní pracovat. Game design také obnáší návrh herního světa, postav, herních objektů nebo situací, které hráč musí řešit. Dnešní trh nám nabízí široké spektrum herních zařízení, které jsou popsány níže. Je potřeba si uvědomit jaký zážitek z hraní nabízí jednotlivé platformy, především je nutností promyslet jednotlivé aspekty hry, například ovládání a náročnost na hardware. Každá platforma nám nabízí různé způsoby ovládání (herní ovladače, klávesnice, dotykové obrazovky). Návrh obsahuje i výběr potřebných technologií, které jsou potřeba k implementaci hry.

Vývoj herního titulu je velmi komplexní problém a není možné, aby jeden člověk ovládal veškeré potřebné dovednosti na vysoké úrovni.

1.1 Koncept

Na začátku musí být myšlenka, vize herního titulu. Všechny dobré koncepty jsou postavené na silné, solidní myšlence [1, str. Ideate]. Aby výsledný herní titul měl velký potenciál uspět a získal si herní komunitu, herní návrhář musí ovládat spoustu důležitých dovedností pro tvorbu videoher.

Postupně tvoříme koncept celé hry. Abstrahujeme základní koncept, principy titulu. Mechaniky hry definují žánr a hratelnost, ale také jakým způsobem hra sděluje svůj příběh. Dobrým principem, jak vytvořit skvělý koncept je metoda brainstorming. Zapisujeme si veškeré myšlenky, které nás napadají a poté vybereme ty nejlepší nápady. Zvolíme si téma, které definuje vzhled hry. Dále je nutné do konceptu zahrnout důležité aspekty jako:

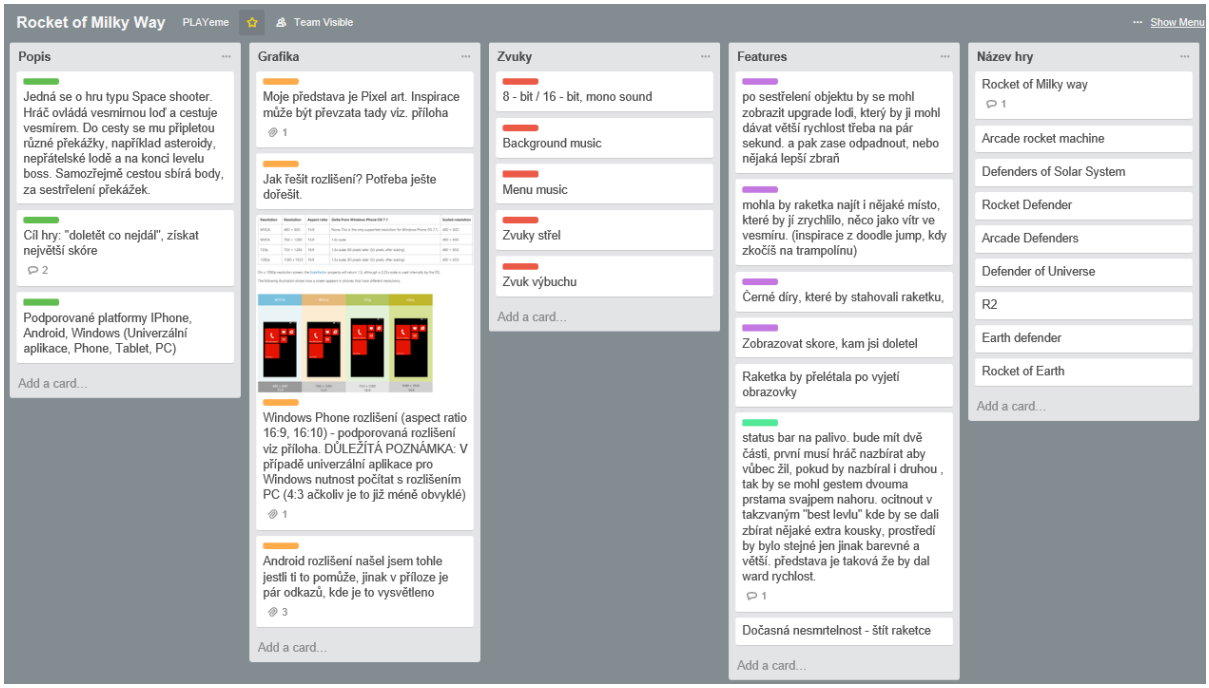
- Antropologie – zabývá se hráčem, jak se cítí a co chce zažít, cíle hry a jak vyřešit dosažení cíle.
- Vlastnosti (anglicky features) – jedná se o klíčové vlastnosti hry (moment překvapení, jedinečný zážitek z ovládání nebo speciální možnosti pro hráče atd.).
- Matematika – žádná hra se bez matematiky neobejde, a proto je důležité ovládat principy, které nám pomůžou lépe navrhnout herní mechaniky, například ovládání výsledné hry, fyziku, práce s kolizemi atd.
- Design zvuku – důležitá vlastnost pro vtažení hráče do hry, zvuky a hudba nám pomáhají navodit atmosféru a nezapomenutelný emocionální zážitek.

1.1.1 Vlastnosti herního titulu

V následujícím kroku vybereme vlastnosti hry, které zahrneme do vývoje. Potřebujeme jedinečné a klíčové vlastnosti, například jaké možnosti vylepšení nabídneme hráči. To bývá jedním z největších neúspěchů během vývoje hry, protože příliš mnoho vlastností prodlužuje dobu vývoje a představuje nejčastější důvod ukončení vývoje.

1.1.2 Koncept – Rocket of Milky Way

Můj námět představuje raketa, která letí hlubokým vesmírem. Cílem je doletět co nejdále a dosáhnout nejvyššího skóre. Překážky jsem zvolil meteority a nepřátelské mimozemšťany. Meteoritů je více druhů a liší se velikostí nebo rychlostí pohybu. Hráč je odměňován za sestřelení překážek navýšením skóre a možností získat vylepšení rakety. Grafická stránka hry je stylizována jako Pixel Art, který vytváří atmosféru retro stylu, 8bitových her z minulého století. Hra je doprovázena 8bitovými zvukovými efekty. Prostor pro pohyb rakety je dvoudimenzionální (2D), protože tento prostor je jednodušší než 3D prostor. Ovládání je primárně vytvořeno pro dotykové obrazovky.



Obrázek 1 - Koncept vytvořený v programu Trello

Obrázek 1, zobrazuje základní koncept herního titulu *Rocket of Milky Way*. Koncept obsahuje popis principu hry, grafický návrh s řešením pro rozlišení jednotlivých platforem, zvukovou stránku hry a v neposlední řadě vlastnosti (Features), kde jsou uvedeny návrhy na vylepšení hry, například vylepšení hráče, přidání nepřátel atd. Tento koncept byl základem pro vytvoření praktické části bakalářské práce.

1.2 Herní engine

Herní engine je nejzákladnější jádro hry, které nabízí API pro komunikaci s ním. Jedná se o speciální nástroj, který umožnuje tvorbu her pro různé platformy (PC, mobilní zařízení, herní konzole). Nástroj pomáhá vývojáři spojit dohromady herní prvky například herní fyziku, modely, materiály atd. Herní engine běžně poskytuje funkce jako renderování grafiky (2D, 3D), fyzický engine (simulování fyziky), detekce kolizí, zvukové manažery, animace, práce se sítí, skripty a mnoho dalších funkcí. Rozsah služeb u jednotlivých enginů se liší, můžeme tak nalézt, jak se jednoduché knihovny starají o vykreslení, tak i rozsáhlé enginy s vlastním IDE. [2]

1.2.1 Výběr herního engine

Nejdůležitějším krokem je výběr herního engine. V dnešní době nemusíme začínat vytvářením vlastního engine, ale můžeme si vybrat ze široké škály dostupných herních engine. Při výběru volíme podle námi stanovenými kritérii. Musíme zvážit možnosti, které nám herní engine umožnuje, například podporované platformy, pro které můžeme hru sestavit. Také je důležitá komunita, která rozvíjí a udržuje engine aktuální. Dále musíme zvážit výkonost a hardware nároky.

Unreal Engine je herní engine vytvořený firmou *Epic Games*. První verze vyšla v roce 1998 a od té doby byla několikrát vylepšena a doplněna tak, aby mohla být použita v nejnovějších titulech. Jádro je napsáno v programovacím jazyce C++ a podporuje mnoho platforem, například *Microsoft Windows, Linux, Mac OS X* na osobních počítačích. Podporuje také herní konzole [3]. *Unreal Engine* je velmi dobrým pro tvorbu AAA titulů a nabízí velkou podporu komunity v podobě videí a návodů. Nicméně je to výkonný nástroj pro tvorbu opravdu velkých titulů.

CryEngine je herní engine vytvořený německým vývojářským studiem *Crytek*. Tento nástroj se těší velké popularitě. Podporuje velké množství platforem, například *Microsoft Windows, Linux, PlayStation, Xbox, iOS, Android* [4]. *CryEngine*, jako v případě *Unreal Engine*, je vhodný pro tvorbu velkých AAA titulů, protože poskytuje opravdu velký výkon.

Kromě těchto dvou velkých herních engine existuje mnoho dalších, například *Hero Engine*, který je vhodný pro tvorbu MMO titulů, *GameMaker*: *Studio* je vhodný pro tvorbu 2D her. *Cocos2D* je také vhodný pro tvorbu 2D her, a navíc je kompletně zdarma, avšak nevýhodou je, že podporuje malé množství platforem.

Já jsem si pro svoji práci zvolil herní engine *Unity3D*, z důvodu jeho velké všestrannosti. Podporuje celou řadu platforem a je v hodný pro tvoření 2D i 3D her. Také je stále aktuální a je velmi dobře přijímán komunitou, což přináší velkou řadu návodů, tipů a rychlou odezvu na případné dotazy. Takže je vhodný pro osvojení si základů a principů vývoje her. Následující kapitola je věnována popisu tohoto nástroje.

1.3 Game Design

Jedna z nejdůležitějších etap vývoje herního titulu je Game Design. Návrh hry je umění aplikovat design a estetiku tak, aby vytvořila hru, která usnadní interakci mezi hráči a zábavou

nebo vzděláním, cvičením, experimentálním účelem [1, str. Design]. Tento proces lze uplatnit jak na hry, tak na další interakce, zejména na ty virtuální.

Design hry vytváří cíle, pravidla a výzvy, která definují samotnou hru, ať už se jedná o sport, stolní hru, videohru, herní roli nebo simulaci, která vytváří žádoucí interakci mezi jejími účastníky, popřípadě diváky.

1.3.1 Výběr platformy

Na výběr máme širokou škálu platforem a každá nabízí specifické vlastnosti, které musíme vzít v potaz.

Jednou z nejuniverzálnějších platforem je osobní počítač. Dnešní herní PC jsou vybaveny výkonným hardware a jsou přizpůsobivé pro ovládání. Můžeme připojit periferní zařízení jako ovladač, klávesnici, počítačovou myš nebo dotykovou obrazovku. PC nabízí nespočetně periferií. Na osobní počítače jsou také k dispozici zdaleka nejvíce exkluzivních herních titulů, od rozsáhlých on-line her přes simulátory a náročné strategie. Na výkonném počítači pak multiplatformní hry, vypadají zdaleka nejlépe, za předpokladu kvalitního vývoje. Nevýhodou herních počítačů je vysoká pořizovací cena.

Herní konzole jsou zařízení, které můžeme vnímat jako protipól osobních počítačů. Jedná se o uzavřenou platformu s předem danou specifikací a vyrobené za účelem hraní her. Výběr je široký, ale dva největší zástupci jsou *Microsoft Xbox* nebo konzole *PlayStation* od firmy *Sony*.

Novinkou je virtuální realita, která hráče „přenese“ do virtuální reality. Umožnují hráči, aby se integroval se simulovaným prostředím. Technologie virtuální reality může vytvářet iluzi skutečného světa, například výcvik boje, pilotování a mnohé další. Také může vytvářet iluzi fiktivního světa počítačové hry. Proto má velké předpoklady pro využití v mnoha oborech (lékařství, sport, armáda, konstrukce).

V neposlední řadě jsou tu chytré telefony, dnes již běžně rozšířené zařízení. Jejich výhodou je přenositelnost a výborně se hodí pro příležitostné hry, například logické tituly, arkády, adventury, strategie. Poslední zmiňovaná platforma není primárně určená k hraní her a má omezený výkon ve srovnání s předešlými platformami.

1.3.2 Business Model

Jednou z finálních etap konceptu je promyšlení obchodního plánu. Ten zahrnuje, propagaci, monetizaci a jakým způsobem budeme řešit šíření titulu. Dnešní doba zjednodušuje propagaci pomocí sociálních sítí a internetu a jako médium nám nabízí mnoho možností zahrnující využití cílené reklamy [1, str. Monetize]. Většinou se jedná o placenou propagaci. Jeden z nástrojů, který je poskytován zdarma, je například *Ad Duplex*. Tento reklamní systém funguje na principu zobrazování naší reklamy v jiných aplikacích, které využívají tento systém, a naopak zobrazuje reklamy jiných aplikací v našem herním titulu. Monetizace herního titulu je důležitou součástí konceptu. *Rocket of Milky Way* je dostupná v režimu Free to Play. To znamená, že je dostupná úplně zdarma.

1.4 Sestavení prototypu

Vytvoření prototypu, je zahájením vývoje herního titulu. Prototypem myslíme sestavení malé sekce, která reprezentuje nejdůležitější herní mechaniky. V této fázi ověřujeme, zda naše představy herního titulu pracují, jak jsme zamýšleli. Doporučeným postupem je vytvořit, několik prototypů a postupně je vyzkoušet a zvolit ten nejlépe vyhovující. Postupně přidáváme funkce, vlastnosti a prototyp vyzkoušíme [1, str. Build]. Tento cyklus se neustále opakuje, sestavíme prototyp, otestujeme a popřípadě přecházíme k dalšímu bodu testování.

Vhodným prostředkem jsou ukázková řešení v podobných titulech. Právě *Unity3D* má dobré komunitní zázemí, kde můžeme najít spousty ukázek kódu nebo celých titulů. Tím si můžeme osvojit nové techniky a postupy, čímž zefektivníme naši hru.

1.5 Testování

Testování hry je důležitou součástí vývoje hry. Během testování hrají hráči hru a poskytují zpětnou vazbu vývojářům, například do jaké míry jsou použitelné komponenty herních mechanik nebo jednotlivých prvků ovládání, srozumitelnost cílů a pravidel, snadné učení a potěšení z hraní hry [1, str. Test]. Vývojář na základě zpětné vazby reviduje návrh, herní komponenty, prezentaci cílů a pravidel před tím, než se otestují znova. Cyklus se opakuje, dokud není herní titul připraven na finální vydání.

Během testování mohou být identifikovány různé problémy, které musí být odstraněny před vydáním finální verze. Testováním ověříme správný návrh hry a popřípadě upravíme náročnost, ovládání, mechaniky atd. Tvoříme dokumentaci chyb (anglicky bug) se kterými se musíme následně vypořádat. Testování videohry je vysoce technická oblast vyžadující mnoho odborných znalostí, analytickou kompetenci, kritické hodnocení a především vytrvalost.

2 UNITY3D

Unity3D je velmi kvalitní multiplatformní herní engine s integrovaným vývojovým prostředím (IDE), který je dostupný zdarma. Je snadno pochopitelný i pro začátečníky, a proto je to skvělý nástroj pro osvojení si základních principů při tvorbě herního titulu. *Unity3D* je slavné díky rychlému vytváření prototypů a velkému počtu videoher, které jsou pomocí tohoto nástroje vyvíjeny.

Herní engine vytvořila společnost *Unity Technologies*. První verze *Unity* (1.0.0) byla vytvořena vývojáři David Helgason, Joachim Ante a Nicholas Francis v Dánsku. Celý projekt vznikl 6. června 2005. Cílem projektu bylo vytvořit výkonný herní engine s profesionálními vývojovými nástroji pro amatérské herní vývojáře [5]. Herní engine je navržen pro tvorbu her pro PC, konzole, mobilní zařízení a nově i pro virtuální realitu.

Unity3D od verze 5.0 je ke stažení ve 4 verzích: Personal, Plus, Pro, Enterprise. Personal verze je dostupná zdarma a poskytuje všechny vlastnosti herního engine. Herní tituly využívající Personal edici je možné použít i pro komerční účely, pouze s jedinou podmínkou, že hranice příjmu nesmí překročit částku 100 000,- dolarů. Po překročení částky je nutné si zakoupit licenci. Edice Plus nabízí navíc výkonnostní reporty, možnost přizpůsobit grafickou podobu editoru, certifikační kurz, slevu na balíčky v online obchodě *Unity* (Asset store), balíčky assets, které obsahují 3D modely, zvuky, animace a mnoho dalšího obsahu. Verze Pro obsahuje vše předešlé, a navíc k tomu přidává prémiovou podporu a zvětšuje slevu v Asset store na 40 procent. Poslední edicí je Enterprice, která je speciálně přizpůsobena konkrétním potřebám [6, str. Unity documentation].

Nástroj *Unity3D* se využívá nejen k vývoji videoher, ale i pro vývoj 3D aplikací, protože engine je velmi obecný [7]. Prázdné scény nejsou nijak specifické a dají se snadno nastavit například k prohlížení modelů.

Práce v *Unity3D* se skládá ze dvou částí, programování a návrh scény, modelů a komponent. Unity podporuje 2D i 3D tvorbu. V současné době (12. 5. 2017) *Unity3D* podporuje 25 různých platforem [6, str. Platform-specific].

Tabulka 1 - Seznam podporovaných platforem

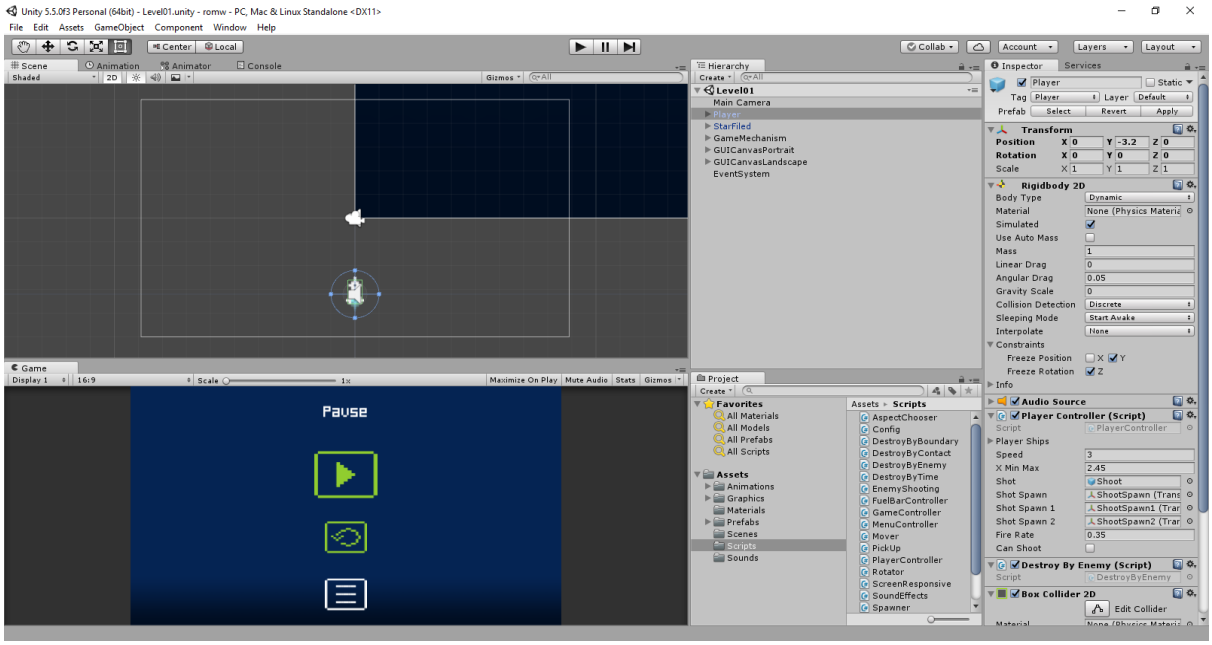
Virtuální realita	Desktopové a webové aplikace	Konzole	Mobilní platformy	Chytré televizory
Oculus Rift	Windows	PlayStation 4	iOS	Android TV
Google Cardboard	Windows Store Apps	PlayStation Vita	Android	Samsung Smart TV
Steam VR	Mac	Xbox One	Windows Phone	tvOS
Playstation VR	Linux	Wii U	Tizen	-
Gear VR	Steam OS	Nintendo 3DS	Fire OS	-

Virtuální realita	Desktopové a webové aplikace	Konzole	Mobilní platformy	Chytré televizory
Microsoft Hololens	Facebook Gameroom	Nintendo Switch	-	-
Daydream	WebGL	-	-	-

S důrazem na přenositelnost, *Unity3D* se zaměřuje na následující API: *Direct3D* a *Vulkan* pro *Windows* a *Xbox 360*, *OpenGL* pro *Mac*, *Linux*, *Windows*, *OpenGL ES* pro *Android* a *iOS* a proprietární API pro herní konzole. Skvělou vlastností *Unity3D* je, že umožnuje nastavení komprese textur a rozlišení pro každou platformu, kterou podporuje zvlášť [6, Platform-specific]. Proto je *Unity3D* velmi pozoruhodný herní engine, pro svou schopnost cílit na různé platformy. Jak je vidět z tabulky 1, *Unity3D* podporuje mobilní platformy, webové prohlížeče, stolní počítače, konzole i televizní platformy.

2.1 Unity3D editor

Unity editor se skládá z mnoha pod oken. Nejvíce používaná okna jsou: *Project Browser*, *Inspector*, *Game View*, *Scene View* and *Hierarchy* [6, str. Main Windows]. Každé okno slouží ke specifickému účelu. Všechny části jsou modulární, takže vzhled editoru si lehce přizpůsobíte svým potřebám.



Obrázek 2 - Unity3D editor

2.1.1 Project Browser

Každý projekt má svou strukturu, kterou je dobré při tvorbě projektu vhodně dodržovat. Projekt obsahuje složku *Assets*, ve které jsou ukládána všechna data projektu [6, str. Project Window]. V této složce si tvoříme svou strukturu, a proto je dobrým zvykem vytvářet příslušné složky.

Projekt *Rocket of Milky Way* má obsáhlejší strukturu, a proto složka *Assets* obsahuje mnoho složek a podsložek. Důležité složky zahrnují:

- Scenes – složka obsahuje sestavenou scénu, například menu nebo herní úroveň,
- Graphics – zde je ukládána veškerá potřebná grafika, například sprity hráče, nepřátel a tak podobně,
- Scripts – složka obsahuje veškeré použité skripty, například pro ovládání hráče, přizpůsobení obrazovky a veškerou herní logiku,
- Prefabs – složka obsahuje vytvořené prefaby, které slouží jako předpřipravené herní objekty (například hráč se skládá z několika různých komponent).

2.1.2 Inspector

Projekty v editoru *Unity* se skládají z mnoha herních objektů obsahujících skripta, zvuky a ostatní grafické elementy například nastavení světla. *Inspector* je jedním z nejdůležitějších částí editoru [6, str. Inspector window]. Zobrazuje detailní informace o vybraném herním objektu ve scéně, například připojené komponenty a jejich vlastnosti. Umožnuje nám jednotlivé nastavení komponent modifikovat.

2.1.3 Game View

Okno *Game View* zobrazuje výslednou scénu z pohledu hlavní kamery. Reprezentuje aktuální podobu hry. Využívá se k testování provedených změn. Při spuštění se editor přepne do takzvaného play módu, kde můžeme hru testovat. Výhodou je možnost upravovat jednotlivé objekty ve scéně a výsledek se projeví okamžitě. Pozor na ukládání změn, protože vše, co nastavíme v play módu je dočasné a bude to ztraceno ve chvíli, kdy se přepneme zpět. Editor ve výchozím nastavení ztmavne při vstupu do play módu [6, str. Game View], abychom mohli indikovat, v jakém módu se nacházíme. Ovládací panel play módu můžete vidět níže.

Obrázek 3 - Ovládání play módu

Prvním tlačítkem vyvoláme přepnutí do play módu, prostřední tlačítko slouží k pozastavení aktuální scény, což bývá užitečné, protože máme čas prohlédnout si jednotlivé komponenty, co obsahují, jakých hodnot nabývají, případně tyto hodnoty upravit. Poslední tlačítko napravo panelu slouží ke krokování scény.

2.1.4 Scene View

Okno *Scene View* je interaktivní náhled do světa, který vytváříme. Využívá se k přidávání, odebírání a pozicování herních objektů, jako například kamery, světla a všech ostatních herních objektů [6, str. Scene View]. Umožňuje výběr, manipulaci a úpravy objektů ve scéně. Můžeme na scénu nahlížet ze všech úhlů a přepínat mezi spoustou typů zobrazení, mezi základní patří 2D a 3D pohled.

Objekty můžeme vkládat do scény přímo operací „Drag and Drop“ nebo pomocí skriptů. V případě vkládání pomocí skriptů se objekty generují až za běhu hry. Podrobnosti o herních objektech můžeme najít v okně *Hierarchy*.

2.1.5 Hierarchy

Hierarchy okno obsahuje seznam každého herního objektu v aktuální scéně. Některé mohou být přímou instancí Assets souborů, například 3D modely [6, str. Hierarchy window]. Zvláštní kategorií jsou takzvané Prefabs objekty. Jedná se o definované herní komponenty, které se zpravidla skládají z mnoha specificky nastavených herních objektů. Prefabs většinou tvoří větší část hry. Názornou ukázkou je například hráč, který se skládá z komponent: skript PlayerController, transform (pozice v herní scéně), Rigidbody 2D (Unity fyzikální engine), atd.

2.2 Assets

Asset reprezentuje jakoukoliv položku, která může být použita ve hře nebo projektu. Asset může pocházet ze souboru vytvořeného mimo projekt Unity, například 3D model, zvukový soubor, obrázek nebo kterýkoliv jiný typ souboru, který podporuje Unity [6, str. Asset Workflow]. Existují také některé typy assets, které mohou být vytvořeny v Unity, například Animation Controller, Audio Mixer nebo Render Texture.

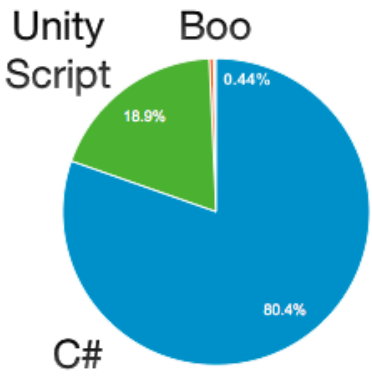
2.3 Unity3D script

Psaní skriptů je základním kamenem každé hry. Skripty potřebuje i nejjednodušší hra, například pro zachycení vstupů od hráče nebo řešení událostí, které nastanou v průběhu hry. Skripty se používají k vytvoření grafických efektů, animací, fyziky objektů nebo k implementaci AI systémů [6, str. Scripting].

Chování herních objektů je kontrolováno komponenty, které jsou k objektu přiřazené. *Unity3D* má mnoho vestavěných komponent, které mohou být velmi všestranné. Brzy však zjistíme, že mnohdy potřebujeme jít nad rámec toho, co mohou poskytnout [8]. Možným řešením je implementace vlastních herních funkcí. *Unity* nám umožnuje vytvářet vlastní komponenty pomocí skriptů. Ty umožnují spouštět události, měnit vlastnosti komponent v průběhu času nebo reagovat na vstup od uživatele jakýmkoli způsobem, který se nám líbí.

Od počátku nástroj Unity podporoval tři programovací jazyky: *C#, Unity Script*, *Boo*. [6] Podle statistiky využívá přes 80 % uživatelů *C#.* Unity Script neboli Javascript používá necelých 20 % uživatelů a *Boo script* využívá 0,44 % uživatelů, proto byl poslední zmiňovaný jazyk od verze 5.0 odstraněn [6, str. Unity documentation]. Pokud aktualizujete starší projekt na novější verzi a využíváte *Boo script*, bude stále pracovat jako v předešlé verzi. Aktuální verze *Unity* 5.6 (12. 5. 2017) nativně podporuje dva programovací jazyky:

- *C#* – programovací jazyk vyvinutý společností *Microsoft*, průmyslový standard podobný jazykům Java nebo C++ [9].
- *Unity Script* – jazyk určený speciálně pro použití s *Unity3D*, jedná se o upravený Javascript.



Obrázek 4 - Využití programovacích jazyků

Kromě těchto programovacích jazyků lze použít i další jazyky .NET, které jsou schopny kompilovat kompatibilní DLL (dynamické knihovny).

Skriptování (nebo vytváření skriptů) slouží k tvorbě vlastních doplňků k funkcím *Unity* editoru, pomocí rozhraní *Unity Scripting API*. Když vytvoříme skript a připojíme jej k objektu, skript se objeví inspektoru stejně jako vestavěná komponenta [8]. Je to proto, že skripty se stanou součástí projektu, jakmile je uložíme.

Z technického hlediska se každý skript, který vytvoříme, kompiluje jako typ komponenty, takže editor považuje náš skript za vestavěnou komponentu. Skvělou vlastností editoru je, že pokud definujeme atributy ve skriptu jako public, budou dostupné k editaci i v *Inspector*. Takže můžeme upravovat atributy, aniž bychom museli otevřít skript. To bývá výhodné například v play módu, protože můžeme v reálném čase měnit parametry, například rychlost pohybu hráče, množství generovaných objektů atd.

2.3.1 Struktura C# skriptu

Editor Unity nám vygeneruje předdefinovanou podobu skriptu, při jeho vytvoření. Jedná se o základní herní smyčku. Klíčové jsou dvě metody *Start* a *Update*. Metoda Start slouží k prvotní inicializaci proměnných, nastavení atributů atd. *Update* je nekonečný cyklus, který se volá dle počtu snímků za sekundu, ideálně šedesát krát za sekundu [9].

Skripty vytváří spojení s vnitřní implementací Unity, která vychází z vestavěné třídy *MonoBehavior*. Můžeme uvažovat o třídě jako o způsobu vytváření nového typu komponent, který lze připojit k hernímu objektu. Pokaždé, když připojíme komponentu skriptu k hernímu objektu, vytváříme novou instanci objektu definovaného modelem. Název třídy je převzat z názvu souboru, proto je důležité, aby oba názvy byly shodné, jinak nám skript nepůjde připojit k objektu. Vygenerovaný skript můžete vidět níže.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Ukazka : MonoBehaviour {
      // Metoda slouží k inicializaci
      void Start () {
```

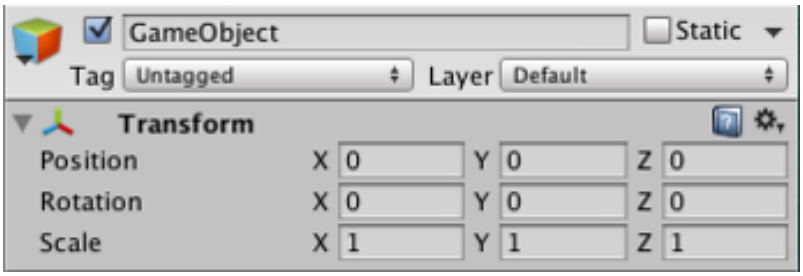
```
        }
        // Metoda je volána jednou za snímek
        void Update () {

        }
}
```

Nejdůležitější definované metody jsou *Start* a *Update*. Metoda *Update* je místo, kam vložíme kód, který se provádí při každém snímku. To může zahrnovat pohyb objektu, spouštění funkcí, reakci na vstup od uživatele, v podstatě vše, co je třeba vyřešit během hry v průběhu času [6, str. Scripting]. Aby metoda *Update* fungovala korektně, je potřeba před jejím spuštěním nastavit proměnné, přečíst preference nebo navázat spojení s jiným herním objektem. K těmto účelům slouží metoda *Start*, která se volá po spuštění skriptu, ještě před prvním voláním metody *Update*.

2.4 Herní Komponenty

Herní objekty (anglicky Game Objects) jsou základní objekty v *Unity*. Představují znaky, vlastnosti, rekvizity, scenérie. Sami o sobě nevykonávají mnoho činností, ale spíše se jedná o kontejnery pro herní komponenty, které implementují skutečnou funkcionalitu [6, str. Asset Workflow]. Například objekt světlo je vytvořen připojením komponenty světlo k hernímu objektu. Takže komponenty jsou matice a šrouby herních objektů a určují jejich chování ve hře.

Každý herní objekt má vždy připojenou komponentu *Transform* (reprezentuje pozici a orientaci) a není možné ji odstranit. Je to proto, že komponenta *Transform* určuje, kde se daný objekt nachází, jak je otočen a také obsahuje měřítko objektu. Bez této komponenty by herní objekt neměl místo v herním světě.



Obrázek 5 - Komponenta Transform

Flexibilita je jeden z největších aspektů komponent. Když připojíme k hernímu objektu komponentu, nacházejí se v ní různé hodnoty nebo vlastnosti, které mohou být upraveny v editoru při vytváření hry nebo skriptem při spuštění hry. Existují dva hlavní typy vlastností – hodnota a odkaz neboli reference.

Ostatní komponenty, které dávají objektu jeho funkce, lze přidat z menu podokna Inspector nebo ze skriptu. Existuje také mnoho užitečných předem vytvořených objektů (primitivní tvary, kamera atd.)

2.4.1 Main Camera

Stejně jako kamery se ve filmech používají k zobrazení příběhu publikum kamery, v Unity se používají k zobrazení herního světa. Ve scéně budete vždy mít alespoň jednu kameru, ale můžete jich mít více [6, str. Cameras].

Každá scéna má v základním nastavení jeden objekt Main Camera. Bod místa na obrazovce je definován v pixelech. Levý dolní okraj obrazovky je (0,0). Pravý vrchol je (šířka v pixelech, výška v pixelech). Pozice na ose Z je uváděna jako vzdálenost v pixelech od komponenty Main Camera.

2.4.2 Lights

Světla jsou nezbytnou součástí každé scény. Zatím co meshes a textury definují tvar a vzhled scény, světla definují barvu a náladu prostředí [6, str. Lights]. Práce se světly chce hodně procvičení, než bude výsledný efekt vypadat hezky. Světla se dají kombinovat a vytvářet tak složitější motivy. Světla mohou být přidána do scény z nabídky editoru GameObject a z následné podnabídky Light.

2.5 Důležité funkce Unity

Unity disponuje spoustou důležitých částí a funkcí, které nám pomáhají a usnadňují život při vývoji videoher. Obsahuje jak fyzikální engine, tak i jádro pro renderování grafiky.

2.5.1 Fyzický herní engine

Unity má vestavěnou podporu *PhysX* od společnosti NVIDIA. Jedná se o fyzikální herní engine, který využívá grafické karty k výpočtu fyziky [5]. V případě užití *PhysX* je procesor výpočtů fyziky ušetřen, protože grafická karta to udělá za něj. To zajišťuje plynulejší chod hry a také lepší grafiku. Velkou výhodou je podpora simulací, animací, materiálů a kolizí objektů v reálném čase.

2.5.2 Renderování grafiky

Unity3D podporuje sadu knihoven *DirectX* vyvíjenou společností *Microsoft*, která jsou používaná na platformách *Windows* a *Xbox*. Dále podporuje *OpenGL*, který je taktéž použitelný v systému *Windows*, ale dále také na operačních systémech *Mac OS* a *Linux* [6, str. Graphics Overview]. Pro mobilní platformy je podporován *OpenGL*, který se využívá v mobilních operačních systémech *Android* a *iOS*. Renderování grafiky v nástroji Unity3D je velmi pokročilé a podporuje například funkce bump mapping (vytváří iluzi textury, která pak vypadá jako 3D objekt), parallax mapping (vylepšeny bump), reflection mapping (výpočet odrazu světla), shadow mapping (výpočet stínu objektu) [5] a mnohé další.

Unity podporuje velikou škálu modelů a formátů z nejrůznějších grafických softwarů například *Autodesk Maya, Cinema4D, Adobe Photoshop, 3ds Max, Blender* a mnoho dalších. Přes uživatelské rozhraní nástroje Unity, mohou být importovány do enginu a dále se s nimi může pracovat [6, str. Graphics HOWTOs].

2.6 Asset store

Unity *Asset store* je obchod, kde můžeme nalézt spousty užitečných věcí k vývoji. Jsou zde assets vytvořené členy komunitou ale i společností Unity Technologies. K dispozici je široká škála assets, zahrnující vše od textur, modelů a animací až po příklady projektů, výukové programy a rozšíření editoru [6, str. Asset Store]. Obchod je dostupný z jednoduchého rozraní zabudovaného do editoru Unity. Z tohoto rozhraní jsou assets staženy a importovány do projektu.

Okno *Asset Store* můžeme otevřít výběrem položky Window a z podnabídky vybereme *Asset store*. K přístupu do obchodu je nutné si bezplatně založit účet.

Uživatelé Unity mohou publikovat a prodávat svůj obsah, který vytvořili.

2.7 Služby spojené s Unity3D

Nástroj Unity3D nabízí řadu integrovaných služeb pro vytváření her, zvýšení produktivity a nástroje pro analýzu hráčů, kteří výslednou hru hrají. Pokročilejší funkce jsou dostupné v placených verzích, ale i pro verzi Unity3D Personal, která je dostupná zdarma, je množství nabízených služeb zdarma široké. Nejpoužívanější služby jsou uvedeny v následujících kapitolách.

2.7.1 Unity Ads

Služba *Ads* nabízí možnost monetizace hry přidáním reklam. Jednotlivé reklamy jsou navrženy tak, aby se stali přirozenou součástí hry. Na vývojáři je, aby se rozhodl, jakým způsobem budou podávány reklamy hráčům. V základě jsou dostupné dva modely. Prvním modelem je, že vývojář určí, kdy se má reklama zobrazit, jak dlouho nebo zda ji může hráč přeskočit. Druhým a lépe přijímaným modelem je, že hráč si rozhodne sám, zda reklamu zhlédne nebo ne [6, str. Unity Ads]. Vývojář může rozhodnutí podpořit například odměnou za zhlédnutí videa. Došel hráči život? Zhlédni reklamu za jeden život. Hráč si tedy sám vybere, zda si chce prohlédnout reklamu nebo ne. Dobrou vlastností na této službě je, že jako vývojář máte kontrolu nad reklamou ve vaší hře. Navíc velké plus přináší multiplatformní podpora, kdy tato služba funguje na všech podporovaných platformách *Unity3D*.

2.7.2 Unity Analytics

Služba *Analytics* poskytuje snadný a rychlí přístup k důležitým informacím, které pomáhají zlepšit příjem ze hry a zážitek pro hráče [6, str. Unity Analytics]. Poskytuje následující nástroje:

- Metric Monitor – poskytuje přehled na vysoké úrovni o tom, jak je hra používaná.
- Data Explorer – shromažďuje informace konkrétních případů hraní hry, pomáhá sledovat důležité události, jako úprava obtížnosti a jak se projevila na hraní hry.
- Funnel Analyzer – Porozumění pokroku hráče při hraní hry, kde uvízne nebo lehce projde úroveň.
- Segment Builder – Vytváří z uživatelů hry skupiny založené na jedinečných herních scénářích a vzorců chování.

2.7.3 Unity Cloud Build

Unity Cloud Build usnadňuje, snáze vytváří a sdílí aktuální stav hry. Automaticky kompiluje, nasazuje a testuje hru. Snadno a rychle můžeme hru šířit týmu nebo testerům a proces opakovat [6]. Nastavení trvá sekundy a pracuje se stávajícím úložištěm zdrojů, které si nastavíme. Můžeme použít například *Git* nebo jakýkoliv jiný verzovací systém. Stačí *Cloud Build* připojit k úložišti a ve chvíli kdy na úložišti provedeme změnu, služba sestaví hru a rozešle lidem, které nastavíme. Sestavení hry provede automaticky, nebo si můžeme nastavit manuální spuštění, v tom případě musíme sami vyvolat akci služby. Výbornou vlastností je, že určíme platformy, pro které se má hra sestavit a *Cloud Build* se již postará o zbytek. [6, str. Unity Cloud Build]

2.7.4 Unity Collaborate

Aktuálně (12. 5. 2017) je *Collaborate* ve fázi beta verze. Služba poskytuje jednoduchý způsob, jak sdílet, ukládat a synchronizovat projekt mezi členy týmu [6, str. Unity Collaborate]. Projekt je sdílen technologií cloud hosting, takže celý tým může přispívat k projektu a každá změna se eviduje. Proto je snadné vrátit se zpět v případě chybného kroku.

2.8 Problémy spojené s multiplatformním vývojem

Problémy, které musíme řešit pro jednotlivé platformy se liší. Mobilní platformy jako chytré telefony nebo tablety neposkytují výkon srovnatelný například s herní konzolí nebo osobním počítačem. Proto musíme s rozvahou používat hardwarové zdroje a snažit se o co nejmenší náročnost, abychom zajistili hladký chod hry na slabším hardware. V herním průmyslu se spíše nepoužívá zapouzdřený kód, protože přístup k atributu třídy skrze metodu je řádově pomalejší než v případě přímého přístupu, kdy je atribut veřejný. Jeden z největších problému je rozdíl výkonu platforem.

Dalším problémem je také responzivita pro různé velikosti a typy obrazovek. Musíme výslednou aplikaci přizpůsobit pro dané rozlišení a zajistit, aby se nám například modely a textury v různých rozlišeních nerozpadli. V případě praktické části *Rocket of Milky Way*, je dostupná pro dvě zobrazení, prvním je zobrazení na výšku například pro poměr stran 9:16 a 10:16. Druhým zobrazením je na šířku, takže 16:9, 16:9 nebo 4:3. Poměr stran je kontrolován průběžně, a v případě změny rozlišení nebo velikosti okna je na tuto událost reagováno, například změnou rozložení obrazovky, aby došlo k zachování hratelnosti správné funkčnosti aplikace.

Unity3D nám velmi pomáhá a velká část kódu je pro většinu podporovaných platforem stejná. Herní engine Unity3D sám na pozadí převádí události a kód pro danou platformu. Vývojář se tedy může soustředit na vývoj.

3 ZVOLENÉ PLATFORMY

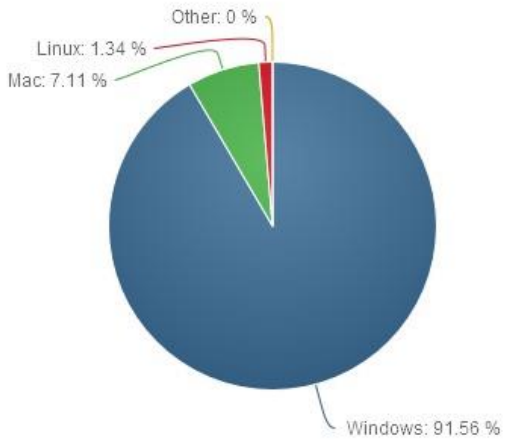
Tato kapitola se věnuje popisu platforem, které jsem zvolil jako cílové pro praktickou část této práce. Jak je již představeno v předchozí kapitole, Unity3D podporuje velké množství platforem. První koncept herního titulu *Rocket of Milky Way* byl určen pouze pro mobilní platformy (iOS, Android, Windows Phone), ale postupně jsem návrh rozšířil o další platformu Standalone. Takto jsem učinil, protože jsem aplikaci přizpůsobil pro Windows 10 Universal Application, která zahrnuje podporu zařízení od firmy *Microsoft*. Verze pro osobní počítače je téměř identická s verzí pro *Windows 10*.

Vytvoření her pro mobilní zařízení, vyžaduje jiný přístup, než jaký bychom použili pro stolní PC hry. Na rozdíl od trhu s PC je váš cílový hardware standardizován a není tak rychlý nebo výkonný jako počítač s vyhrazenou grafickou kartou [6, str. Platform-specific].

3.1 Standalone

Platforma, která představuje osobní počítače s operačními systémy *Windows*, *Mac OS* a *Linux*. Historie vývoje her sahá až do roku 1952, kdy byla napsána jedna z prvních počítačových her na světě [12]. Jednalo se o hru jménem *OXO* a byla to zjednodušená verze piškvorek na ploše 3 x 3. Autorem byl student Alexandr S. Douglas, který hru napsal jako svoji dizertační práci na univerzitě v Cambridge. Dalším důležitým milníkem vývoje her byl rok 1958, kdy William Higinbotham se svými spolupracovníky vytvořil hru *Tenis for Two*. Nutno podotknout, že tenkrát se ještě nejednalo o osobní počítače ve smyslu, v jakém je známe dnes. Jako vykreslovací jednotky byly použity osciloskopy a pro hru *Tenis for Two* byl sestrojen speciální počítač. Technologický skok, který umožnil velký posun v před, je za posledních 50 let enormní. Proto od té doby vývoj her urazil obrovskou cestu až k nynějším velkým AAA herním titulům. Z těchto počátků se nám postupně rýsovala nejstarší herní platforma osobních počítačů.

Standalone je v současné době (12. 4. 2017) nejpoužívanější platformou pro hraní her. Platforma má velmi rozmanitý hardware, takže výkon jednotlivých počítačových sestav se značně liší, proto je potřebné dát si pozor a správně určit nároky na vybavení počítačové sestavy. Také ovládací prvky se mohou značně lišit (dotyková obrazovka, klávesnice, joystick atd.). V případě titulu *Rocket of Milky Way* jsou nároky na vybavení zanedbatelné, jedná se o nenáročný herní titul. Na následujícím obrázku vidíme zastoupení jednotlivých operačních systému na platformě Standalone. [1], [6, str. Standalone]



Obrázek 6 - Operační systémy a jejich zastoupení rok 2015 [10]

3.2 Android

Systém Android je mobilní operační systém založen na Linuxovém jádře a je volně dostupný jako otevřený systém (anglicky open source). Světlo světa spatřil v říjnu roku 2008. Je vyvíjen konsorciem OHA (Open Handset Alliance). Konsorcium se skládá z mnoha firem, například Google, HTC, Intel, LG, nVidia, Qualcomm, Samsung a další. Využívá se na chytrých telefonech, tabletech, chytrých televizích a dalších zařízení. V současné době (12. 5. 2017) patří mezi nejrozšířenější operační systém na světě [13].

Tvorba her pro platformu Android je podobná jako vývoji pro platformu *iOS* nebo *Windows 10 Universal Application*. Hardware není standardizován pro všechny zařízení, takže nastávají větší komplikace s odladěním výsledné hry, než je tomu například u iOS. [6, str. Android]

3.3 Windows 10 Universal Application

Představuje architekturu aplikací založenou na platformě, která byla vytvořena společností *Microsoft* a byla poprvé představena v systému *Windows 10*. Účelem této softwarové platformy je pomoci při vývoji univerzálních aplikací, které běží na *Windows 10* a *Windows 10 Mobile*, aniž by bylo nutné, přepisovat ji pro každou platformu zvlášť [14]. Tento typ aplikace bude moci být spuštěn na osobním počítači s operačním systémem *Windows 10* nebo mobilním zařízením s *Windows 10 Mobile* (chytré telefony, tablety).

3.4 iOS

iOS je mobilní operační systém vytvořený firmou *Apple Inc*. Původně byl operační systém určen pouze pro mobilní zařízení *IPhone*, ale začal se používat i na dalších zařízeních firmy *Apple Inc*. jako *iPod*, *iPad* a *Apple TV*.

Vytváření her pro zařízení, jako je iPhone a iPad, vyžaduje jiný přístup, než jaký bychom použili pro stolní PC hry. Na rozdíl od trhu s PC je cílový hardware standardizován a není tak rychlý nebo výkonný, jako počítač s vyhrazenou grafickou kartou. Z tohoto důvodu musíme přistupovat k vývoji her pro tyto platformy trochu jinak. Také funkce dostupné v aplikaci Unity pro iOS se mírně liší od funkcí pro stolní počítače [6, str. iOS].

4 IMPLEMENTACE PRAKTICKÉ ČÁSTI

Praktická část bakalářské práce je videohra *Rocket of Milky Way*. Jedná se o herní titul, který je inspirován hrami z konce minulého století. Protože v oboru vývoje her jsem nový, rozhodl jsem se postupovat pozvolně a začít vývojem od nejjednodušších titulů, postupně k větším a složitějším. Mým cílem bylo vytvořit první jednodušší 2D hru a naučit se pracovat s nástrojem *Unity3D*.

Před samotnou implementací je třeba si vše patřičně rozvrhnout. Bez řádné přípravy je velká pravděpodobnost, že projekt skončí dřív, než dosáhneme patřičných výsledů. Implementace bez náležité průpravy se může využít k vytvoření prototypu na kterém vyzkoušíme herní mechaniky. Ovšem tento kód by neměl být použit před důkladným návrhem.

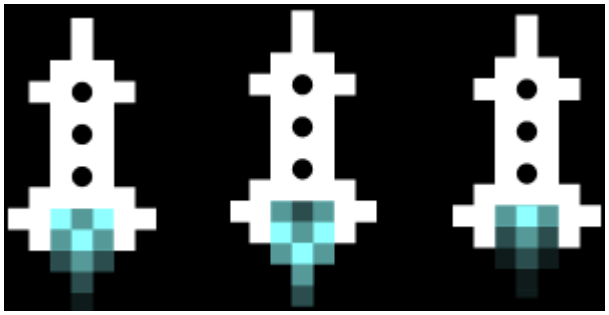
Samotná implementace je náročná, protože musíme dobře navazovat jednotlivé části projektu, jinak se můžeme lehce zaplést do nekonečné spirály oprav a předělávání.

4.1 Návrh hry

Při návrhu herního titulu *Rocket of Milky Way* jsem postupoval od nejjednodušších elementů k těžším úkonům. Prvně jsem navrhl prostředí, hra se odehrává v hlubokém vesmíru, kde hráč je představován vesmírnou raketou, která letí vstříc nebezpečí.

4.1.1 Grafická stránka hry

Chtěl jsem vytvořit tzv. retro styl, proto jsem zvolil techniku zvanou Pixel Art. Jedná se o druh počítačové grafiky, která je vytvářena postupně po jednotlivých pixelech. Výhodou je, že na tvorbu Pixel Art můžeme použít téměř každý 2D grafický editor. Nevýhodou je, že práce vyžaduje více času, protože musíte každý pixel vybarvit zvlášť, aby došlo k co nejrealističtější grafické podobě. Na obrázku 6, můžeme vidět raketu, která představuje hráče ve hře.

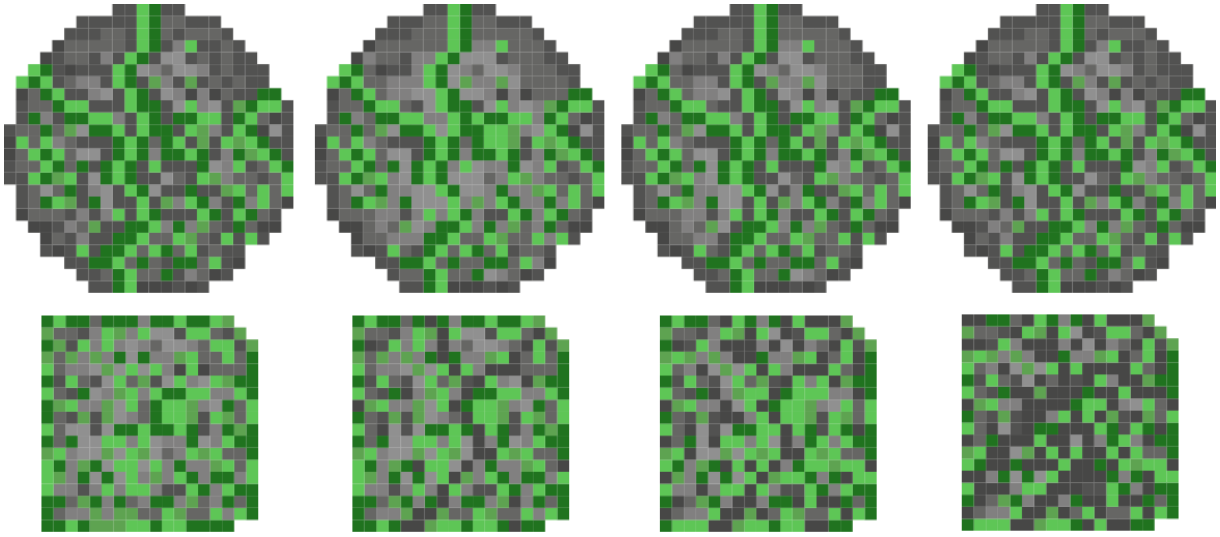


Obrázek 7 - Raketa hráče

Animace motoru je dosažena střídáním těchto tří obrazů rakety. Rychlým střídáním dosáhneme iluze hořícího motoru. Obdobným způsobem jsou řešeny animace nepřátelských lodí.

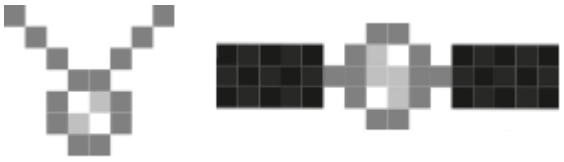
Obrázek 8 - Nepřítel hráče

Časově náročnější bylo vytvořit asteroidy. Ve hře se vyskytuje celkem osm objektů typu asteroid a dva objekty v podobě družic. Asteroidy se liší velikostí, rychlostí, jakou padají nebo počtem střel, které jsou potřeba na jejich likvidaci. Největší asteroidy jsou složeny ze čtyř obrazců.



Obrázek 9 - Ukázka největších asteroidů

Dalším herním objektem jsou družice. Tyto objekty lze sestřelit jednou střelou, vystřelenou hráčem a do hry byly přidány za účelem balancování obtížnosti, protože po sestřelení družice se často vyskytne palivo, které hráč potřebuje doplňovat. Mají tedy mnohem větší pravděpodobnost na výskyt paliva, po jejich sestřelení než jakýkoliv jiný objekt.

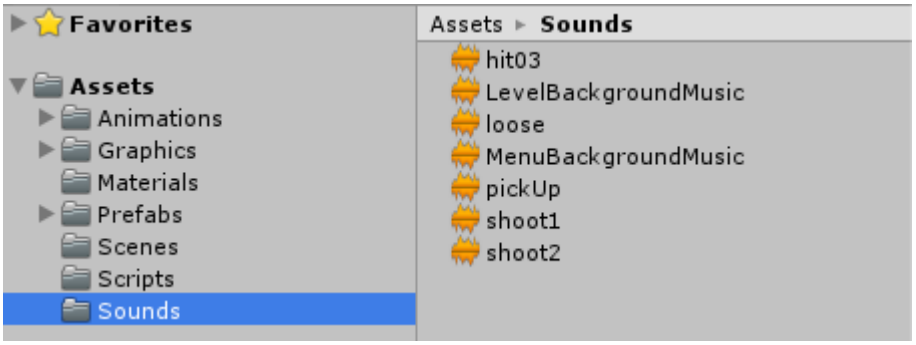


Obrázek 10 - družice

4.1.2 Zvuky

Jak jsem již psal, hra je situovaná jako retro styl, takže zvuky byly zvoleny 8bitové, aby bylo celé téma zachováno a zapadalo do sebe. Pro zvuky byla využita databáze volně dostupných zvuků freesound.org [11]. Jsem především programátor a tvorba zvuků, hudby a efektů je mě poměrně vzdálena. Zvuky a hudba používaná ve hře je pod svobodnou licencí, je k dispozici

zdarma, a to i pro komerční účely. Herní titul obsahuje celkem sedm zvukových efektů, včetně doprovázející hudby. Na následujícím obrázku můžeme vidět seznam.



Obrázek 11 - Seznam zvuků a hudby

- Hit03 – zvukový efekt po zasažení hráče
- LevelBackgroundMusic – Hudební podklad hry
- Loose – zvukový efekt, který se přehraje při konci hry
- MenuBackgroundMusic – hudba na pozadí hlavního menu
- pickUp – zvukový efekt, který signalizuje kolizi s objektem určeným k sebrání (palivo, vylepšení hráče atd.).
- shoot1 – zvukový efekt střely hráče
- shoot2 – zvukový efekt střely nepřítele

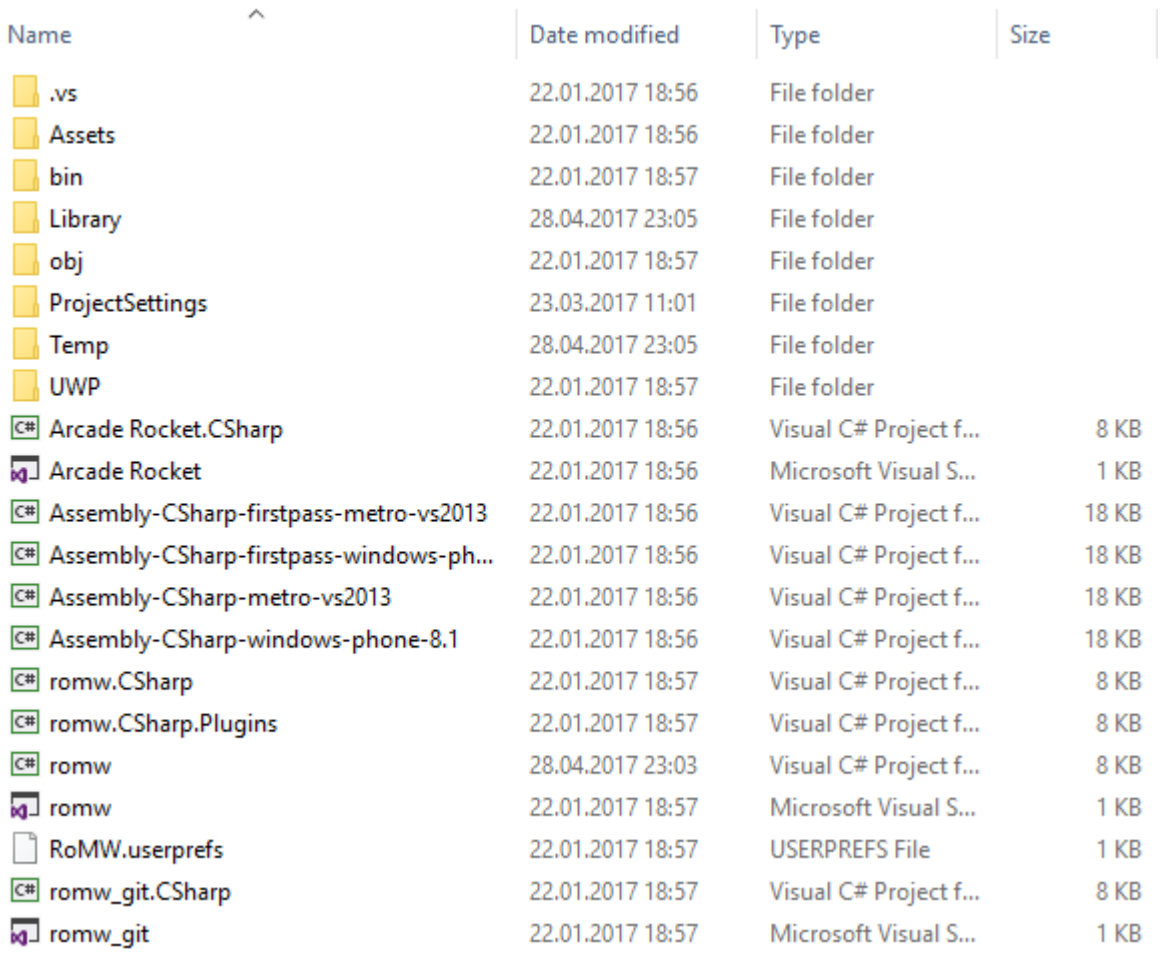
Zvukové efekty a hudba je velmi důležitou součástí hry, protože může navodit atmosféru a lépe vtáhnout hráče do děje. Také je vhodné ji používat k signalizaci událostí, například sebrání předmětu, oznámení konce hry nebo zásahu hráče.

4.1.3 Cíl hry

Herní titul *Rocket of Milky Way* je koncipován jako „endless game“. Cílem je udržet se na živu co nejdéle, a tím získat největší skóre. Hráč potřebuje k přežití alespoň jeden život a ukazatel paliva nesmí klesnout na nulovou hodnotu. Palivo je generováno v průběhu hry a také je určitá pravděpodobnost jeho výskytu po sestřelení nepřátelských lodí nebo asteroidů.

4.2 Struktura projektu

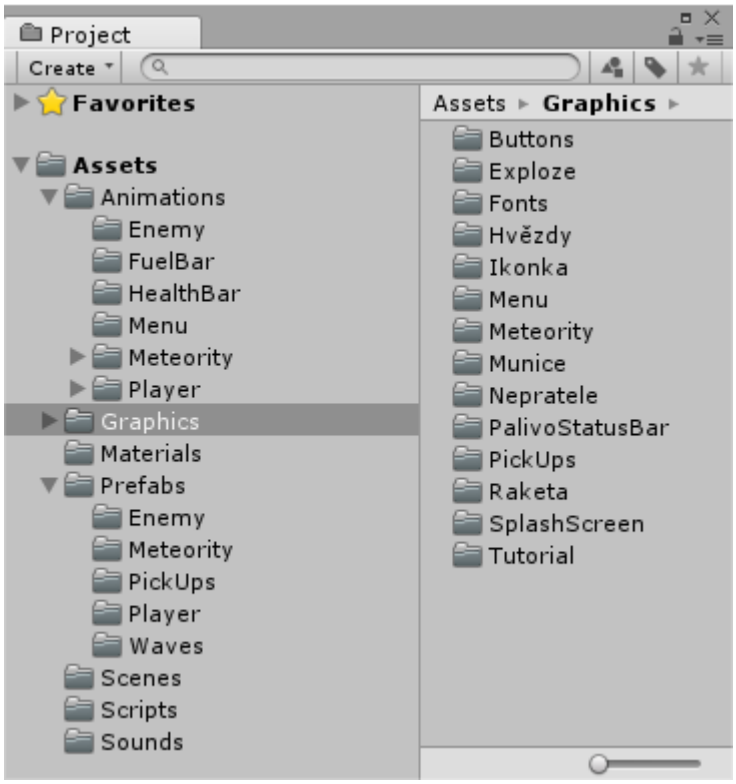
Unity3D nám vygeneruje projekt, který obsahuje různé složky a nastavení. Struktura se rozšiřuje postupně s projektem a s platformami pro které hru sestavíme. Náhled na strukturu celého projektu *Rocket of Milky Way* vidíme níže.



Name	Date modified	Type	Size
.vs	22.01.2017 18:56	File folder	
Assets	22.01.2017 18:56	File folder	
bin	22.01.2017 18:57	File folder	
Library	28.04.2017 23:05	File folder	
obj	22.01.2017 18:57	File folder	
ProjectSettings	23.03.2017 11:01	File folder	
Temp	28.04.2017 23:05	File folder	
UWP	22.01.2017 18:57	File folder	
Arcade Rocket.CSharp	22.01.2017 18:56	Visual C# Project f...	8 KB
Arcade Rocket	22.01.2017 18:56	Microsoft Visual S...	1 KB
Assembly-CSharp-firstpass-metro-vs2013	22.01.2017 18:56	Visual C# Project f...	18 KB
Assembly-CSharp-firstpass-windows-ph...	22.01.2017 18:56	Visual C# Project f...	18 KB
Assembly-CSharp-metro-vs2013	22.01.2017 18:56	Visual C# Project f...	18 KB
Assembly-CSharp-windows-phone-8.1	22.01.2017 18:56	Visual C# Project f...	18 KB
romw.CSharp	22.01.2017 18:57	Visual C# Project f...	8 KB
romw.CSharp.Plugins	22.01.2017 18:57	Visual C# Project f...	8 KB
romw	28.04.2017 23:03	Visual C# Project f...	8 KB
romw	22.01.2017 18:57	Microsoft Visual S...	1 KB
RoMW.userprefs	22.01.2017 18:57	USERPREFS File	1 KB
romw_git.CSharp	22.01.2017 18:57	Visual C# Project f...	8 KB
romw_git	22.01.2017 18:57	Microsoft Visual S...	1 KB

Obrázek 12 - Struktura celého projektu

Pro nás je důležitá složka Assets, kde si ukládáme a spravujeme naše data. V této složce jsem si vytvořil hierarchii, která mně(mi) pomáhá udržet v projektu pořádek. Jak je vidět na obrázku 13, struktura složky Assets je obsáhlejší, i když se jedná o menší projekt. V této složce jsem si vytvořil strukturu, kde jsou soubory a data rozděleny podle oborů do kterých patří, aby byla zachována dobrá orientace v projektu.

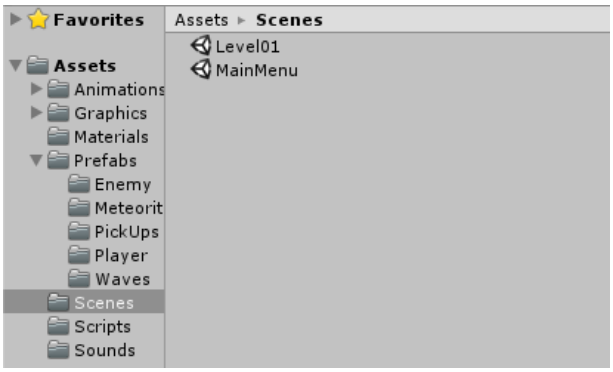


Obrázek 13 - Struktura Asset složky

- Animations – v této složce uchovávám animační klipy a jejich ovladače (anglicky controllers). Jsou rozčlenění dle objektů, ke kterým patří, například animace motoru hráče, nepřátelských lodí, rozpad asteroidů atd.
- Graphics – zde je uložena veškerá potřebná grafika. Je to jedna z nejobsáhlejších složek. Najdeme v ní grafiku pro hráče, asteroidy, grafické rozhraní, ikony pro publikaci atd.
- Materials – obsahuje materiál pro částicový systém, který je využit pro hvězdy v pozadí
- Prefabs – velmi důležitá složka, protože jsou v ní uloženy mnou vytvořené herní objekty, které se skládají z většího množství herních komponent, specificky nastavených pro výsledný účel hry.
- Scenes – zde najdeme dvě vytvořené scény, jedna pro samotnou hru a druhá pro menu.
- Scripts – nejdůležitější složka projektu, obsahuje veškerou herní logiku a mechanismy v podobě skriptů psaných v programovacím jazyku C#.
- Sounds – složka obsahuje všechny doprovodné zvuky a hudbu. Například střelba hráče nebo nepřátelských lodí, hudba na pozadí hry, menu atd.

4.3 Herní scény

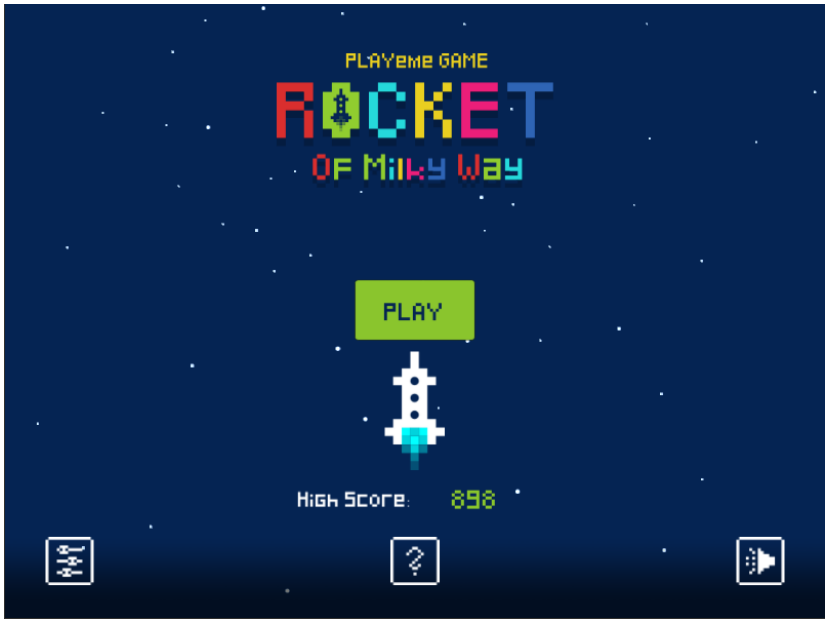
Praktická část hry se skládá pouze ze dvou herních scén. Scéna MainMenu, slouží pro úvodní hlavní menu a druhá scéna je Level01. V druhé scéně probíhá celá hra. Na obrázku 14 vidíme obsah složky Scenes, kde jsou obě scény uloženy.



Obrázek 14 – Složka Scenes, která obsahuje herní scény

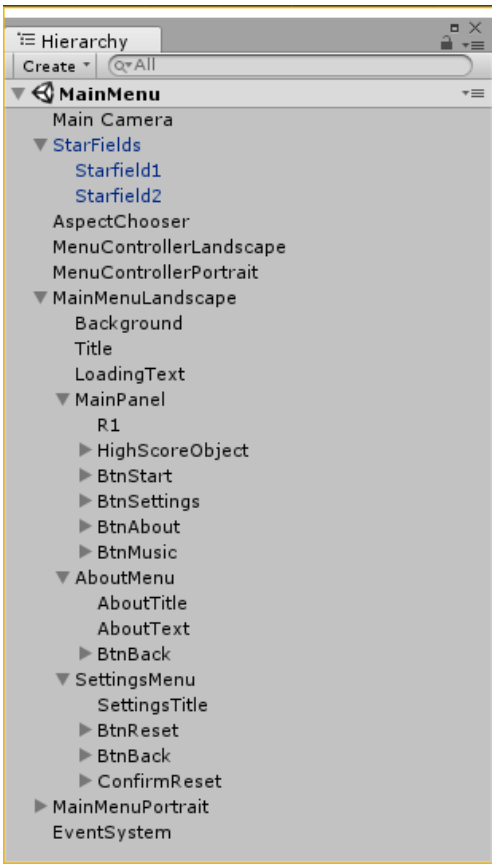
4.3.1 MainMenu scéna

Scéna MainMenu představuje hlavní menu, které se zobrazí po zapnutí hry a uvádí hráče do herního světa. Na této scéně hráč může vidět své dosavadní skóre a spustit první úroveň hry. Dále má k dispozici tři tlačítka. První tlačítko nastavení (vlevo dole), původně sloužilo k testovacím účelům, kde si hráč mohl nastavit způsob ovládání a citlivost. Na výběr byly dva způsoby ovládání. Akcelerometr nebo dotykem, ovšem první zmiňovaný neměl příliš valný úspěch u testerů. Proto byl v následujících verzích hry odstraněn. Tlačítko nastavení zde zůstalo, umožnuje nastavit hru do základního nastavení a smazat herní data. Druhé tlačítko (umístěné na střed dole), zobrazuje informace o autorovi, aktuální nainstalované verzi a číslo sestavení verze. Poslední tlačítko (vpravo dole), slouží k vypnutí nebo zapnutí zvukových efektů a hudby. Výslednou scénu si můžete prohlédnout na obrázku 15.



Obrázek 15 - Výsledná scéna MainMenu

MainMenu scéna obsahuje spoustu herních objektů a o jejich funkčnost se stará několik herních skriptů. Na obrázku 16 vidíme složení scény.



Obrázek 16 - Složení scény MainMenu

- Main Camera – Hlavní prvek, u každé scény je hlavní kamera, která zobrazuje celý výsledek
- StarFields – Tento prvek obsahuje dva částicové systémy, které vytvářejí iluzi hvězd na pozadí. Jeden systém generuje menší částice, které se pohybují pomaleji a druhý systém generuje větší, rychlejší částice s kratší životností. Výsledným efektem je, že menší a pomalejší částice se zdají být vzdálenější od hráče a naopak.
- AspectChooser – Jedná se o herní objekt, který má připojen jeden skript stejného jména. Tento skript určuje, které rozložení obrazovky bude aktivní a které ne.
- MenuControllerLandscape – Výchozí nastavení menu v případě rozložení obrazovky na šířku.
- MenuControllerPortrait – Výchozí nastavení menu v případě rozložení obrazovky na výšku.
- MainMenuLandscape – Herní objekt obsahuje potomky, kteří jsou nastaveny pro zobrazení na šířku.
- MaineMenuPortrait – Herní objekt obsahuje potomky, kteří jsou nastaveny pro zobrazení na výšku.

4.3.2 Level01 scéna

Scéna *Level01* představuje herní úroveň, která se načte po zmáčknutí tlačítka PLAY v hlavním menu. V této scéně probíhá vlastní hra. Následující obrázek ukazuje složení scény *Level01*. Vidíme zde důležité herní objekty, které slouží jako složky pro větší přehlednost. Například herní objekt *Player* zapouzdřuje hráče a jeho nastavení nebo objekt *GameMechanism*, kde

můžeme najít komponenty, které se starají o průběh hry. Hlídají stav hry, zda neproběhla událost, pro ukončení hry například hráči došlo palivo, nebo ztratil všechny životy atd.

Obrázek 17 - Složení scény Level01

- Player – Objekt obsahuje veškeré části týkající se hráče. Herní objekty A-1, B-1, C-1, D-1 jsou grafický vyobrazením vylepšení hráče. Představují čtyři modely rakety, aktivují a deaktivují se podle vyvolaných událostí například, že hráč sebere v průběhu hry předmět, který poskytuje vylepšení nebo naopak když je zasažen ztrácí vylepšení rakety. Tyto mechaniky jsou detailně popsány v kapitole herní mechaniky.
- StarField – Dva částicové systémy, které vytvářejí iluzi pohybujících se hvězd v pozadí hry (stejná funkčnost jako v případě hlavního menu).
- GameMechanism – Herní objekt, který zastřešuje herní mechanizmy hry. Najdeme zde objekty zodpovědné za generování překážek nebo naopak jejich zničení v případě, že opustí pohled komponenty *Main Camera*. Také je zde ovladač pro zvukové efekty a mnoho dalších herních komponent.
- GUICanvasPortrait a GUICanvasLandscape – Zde nalezneme obsluhu grafického rozhraní, jako zobrazování zbývajícího paliva, skóre, tlačítko pauza atd. Herní objekty jsou dva, protože jeden je nastaven na zobrazení na výšku pro mobilní zařízení a druhý obsahuje nastavené komponenty pro zobrazení na šířku, v případě spuštění hry na osobním počítači.

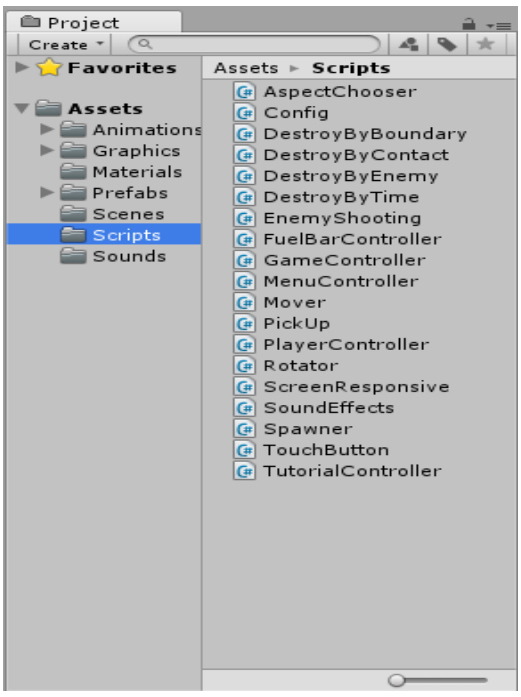
- EventSystem – Komponenta dohlíží nad prvky grafického rozhraní, jako jsou tlačítka, kdy při jejich stisknutí je vyvolána příslušná událost, na kterou jsme poté schopni reagovat.

Obrázek 18 - Ukázka scény Level01

Na obrázku 18 je zobrazena výsledná scéna Level01, hráč je představován raketou. Vlevo nahoře jsou zobrazeny zbývající životy, vprostřed je umístěno tlačítko, které slouží k pozastavení hry a vpravo nahoře je zobrazeno aktuální skóre. Horní uživatelské rozhraní podtrhuje zobrazení zbývajícího paliva.

4.4 Skripty

Projekt obsahuje celkem devatenáct skriptů. Tato podkapitola se věnuje popisu důležitých skriptů, které tvoří základní jádro hry. Následující obrázek ukazuje obsah složky *Scripts*, kde můžeme nalézt všechny skripty, které jsem pro potřebu této hry vytvořil.



Obrázek 19 - Obsah složky Scripts

4.4.1 PlayController

Tento skript se stará o řízení chodu celého hráče, ovládá pohyb hráč, vylepšení rakety a také střelbu hráče. Skript obsahuje téměř 500 řádků kódu, proto zde uvedu důležité metody a události. Kompletní skript je připojen v příloze.

Skript obsahuje tři stěžejní metody *Start, Update, FixedUpdate*. První metoda slouží k inicializaci proměnných. Metoda *Start* se provede při prvním zavolání skriptu, které probíhá při načtení scény. Nejdříve dojde k nastavení rakety hráče, aktivuje se základní raketa a ostatní se nastaví na neaktivní. Poté se zavolá metoda *InitScript*, která slouží k načtení závislých skriptů jako je *GameController* skript. Obě metody jsou zobrazeny níže.

```
public void Start(){
      playerShips[0].SetActive(true);
      for (int i = 1; i < playerShips.Length; i++){
            playerShips[i].SetActive(false);
      }

      InitScripts();
      audioSource = GetComponent<AudioSource>();
      rb = GetComponent<Rigidbody2D>();   // nactu RB
      playerWeapon = 0;
      canShoot = true;
}

private void InitScripts(){
      GameObject controllerGameObject =
      GameObject.FindWithTag("GameController");

      if (controllerGameObject != null){
            gameController =
            controllerGameObject.GetComponent<GameController>();
      }
}
```

Metoda *InitScripts* načte *GameController* skript podle stejnojmenné značky, která mu byla přidělena.

Metoda *Update* se volá jednou za snímek a ovládá střílení rakety hráče. Střelba se liší podle druhu rakety. V metodě vidíme podmínku, která je splněna v okamžik, kdy dojde ke stanovenému časovému posunu, který je pro základní raketu 0,35 sekund. Tato hodnota určuje kadenci střel. Podmínka také kontroluje, zda nenastal konec hry, v tom případě hráč přestane střílet. Ukázka podmínky.

```
// ovladani strileni
if (Time.time > nextFire && !gameController.GameOver && canShoot)
```

V těle podmínky je potom rozvětvení pomocí podmínky switch, která rozhoduje, o jaký druh střel půjde v závislosti na aktuálně aktivní typ rakety.

Metoda *FixedUpdate* se stará o obsluhu ovládání hráče. Rozdíl mezi *Update* a *FixedUpdate* je v čase provedení. Metoda *Update* se provádí jednou za snímek, ale není určeno, v jakém momentě, zatím co *FixedUpdate* se provede vždy na konci aktuálního snímku. V těle metody *FixedUpdate* je doporučeno provádět operace související s fyzikálním enginem *Unity3D*.

Ovládání hráče je řešeno dvěma způsoby. První je implementace pro klávesnice a herní ovladače. Druhý způsob je implementace pro dotykové obrazovky. Klasické ovládání, na které jsme zvyklí z osobních počítačů, je řešeno následující ukázkou kódu.

```
float horizontal = Input.GetAxis("Horizontal");
Vector2 movement = new Vector2(horizontal, transform.position.y);
rb.velocity = movement * speed;
```

Metoda *Input.GetAxis* s parametrem *Horizontal* vrací celočíselnou hodnotu v intervalu (-1;1). Je-li stisknuto tlačítko definované pro pohyb doleva návratová hodnota je -1, pro žádnou změnu pohybu je hodnota 0 a pohyb doprava vrátí pohyb 1. Poté vytvoříme proměnou typu *Vector2*, která uchovává dvě hodnoty typu *float* (jedna pro osu X a druhá hodnota pro osu Y), které předáme současnou pozici hráče na ose Y (protože hráč se pohybuje pouze po ose X, je tedy hodnota osy Y neměnná) a hodnotu, která určuje nový směr. Proměnná *rb* uchovává instanci komponenty *RigidBody2D*, která zastupuje fyzikální engine *Unity3D*. Pohyb hráče je řízen fyzikálními zákony.

Ovládání dotykem je již složitější. Způsob ovládání, které jsem navrhl, funguje tak, že si obrazovku rozdělíme pomyslně na dvě poloviny. Pokud se hráč dotkne levé poloviny obrazovky, raketa se bude pohybovat směrem doleva a naopak. Také pokud hráč se dotkne například pravé obrazovky a bez přerušení dotyku, přejede prstem na levou stranu obrazovky, raketa změní směr. Kód je okomentovaný a ukázka je níže.

```
Vector2 direction = Vector2.zero;   // inicializace nového směru
// pokud se uživatel dotkne obrazovky
if (Input.touchCount > 0){
      // prohledám všechny aktivní doteky a najdu poslední (aktuální)
      for (int i = 0; i < Input.touches.Length; i++)
      {
            // pokud je dotyk ve fázi započato
            if (Input.touches[i].phase == TouchPhase.Began)
            {
                  lastFingerIndex = Input.touches[i].fingerId;
            }
            // uložím si tento dotyk jako aktuální
            if (Input.touches[i].fingerId == lastFingerIndex)
            {
                  lastIndex = i;
            }
      }
      if (lastIndex != -1){
            Touch lastTouch = Input.GetTouch(lastIndex);

            if (lastTouch.phase == TouchPhase.Began ||
            lastTouch.phase == TouchPhase.Moved ||
            lastTouch.phase == TouchPhase.Stationary)
            {

                 if (lastTouch.position.x < (Screen.width / 2)){
                      direction = Vector2.left;
                 }
                 else if (lastTouch.position.x > (Screen.width / 2)){
                        direction = Vector2.right;
                 }
```

```
        }
    }
    rb.velocity = direction * speed;
```

Logika ovladače hráče je složitější a celý skript je v příloze.

4.4.2 GameController

Skript *GameController* zajišťuje fungování základních principů hry, proto je napojený na většinu skriptů a hlídá mnoho událostí, podle kterých se řídí tok hry. Stará se o aktualizaci grafického rozhraní, jako zobrazování aktuálního počtu životů, aktualizaci skóre, spouštění animací hráče nebo ubývání životů, spouští zvukové efekty a hlídá stav paliva atd.

Metoda *Start* inicializuje nastavení na počátku hry (po načtení scény). Metoda *Awake* načte závislosti jako ostatní skripty, které je potřeba kontrolovat a reagovat na jejich události. Následující přehled ukazuje důležité metody, které tento skript obsahuje.

- Awake – načte závislé skripty, ještě před inicializací metodou Start.
- Start – nastaví proměnné na výchozí stav.
- Update – nastavuje animátoru životů hráče aktuální počet životů.
- StartGame – spustí odpočet paliva, spustí generování překážek a počítání skóre
- AddAU – aktualizuje grafické rozhraní pro zobrazení skóre hráče
- SetGameOver – metoda ukončí aktuální hru, spustí zvukový efekt, nastaví proměnné na příslušné hodnoty, zastaví generování objektu, smaže objekty na scéně a mnoho dalšího.
- RestartGame – znovu načte celou scénu, tato metoda se zavolá po stisknutí tlačítka Restart v menu pauza.
- PlayerDead – spustí animaci výbuchu hráče a zavolá metodu *SetGameOver*.
- PlayerHealthDown a PlayerHealtUp – přidají nebo uberou hráči život.

Tento výčet metod skriptu *GameController* nepopisuje všechny metody, ale pouze ty nejdůležitější.

4.4.3 Spawner

Generování překážek, nepřátelských objektů je řízeno skriptem *Spawner*, který obsahuje dvě pole herních objektů. První pole obsahuje veškeré překážky pro náhodné generování a druhé pole nazvané *predefinedWave* uchovává předdefinované vlny nepřátelských objektů. Aktuální verze používá pouze náhodně generované objekty, protože předdefinované nejsou vzhledem k hratelnosti vyladěné.

Hlavní dvě metody pro tento skript jsou *SpawnWave* a *SpawnFuel*. První se stará o generování nepřátelských objektů (nepřátelské lodě, asteroidy) a druhá metoda náhodně v intervalu 5–15 sekund vygeneruje palivo pro hráče. Tělo metody *SpawnWave* na začátku počká pár sekund, aby se hráč mohl připravit, poté se spustí cyklus *while*, který generuje objekty, dokud není vyhlášen konec hry. V těle *while* cyklu proběhne *for* cyklus, který vygeneruje počet překážek

pro danou vlnu. Každou vlnou se provede výpočet počtu překážek na další vlnu a počet nepřátelských raket se generuje podle počtu vln. Čím vyšší je vlna, tím je větší šance na vygenerování nepřítele.

```
IEnumerator SpawnWaves()
{
      // cekani pred prvni vlnou, aby se hrac "rozkoukal"
      yield return new WaitForSeconds(spawnStart);

      while (!gameOver)
      {

            for (int i = 0; i < count; i++) // cyklus pro spawnovani
            {
                  if (gameOver)
                  {
                        break;
                  }
                  // vyber nahodneho rozsahu meteoritu dle poctu vln
                  GameObject meteor = SelectMeteorit();
                  // nastaveni nahodne pozice
                  Vector2 spawnPos = GetRandomPos();
                  // Spawn meteoritu
                  Instantiate(meteor, spawnPos, Quaternion.identity);

                  if (Random.Range(chanceToSpawn, 15.0f) >=11.5f &&
                  (count > count / 2))
                  {
                        SpawnEnemy();
                  }
                  yield return new WaitForSeconds(spawnWait); // cekani
            }
            // cekani mezi vlnama
            yield return new WaitForSeconds(waitWave);
            SetDifficulty();
            waveCount++;
      }
}
```

Druhá metoda *SpawnFuel* slouží k vyvážení hratelnosti, protože u prvních vln vzniká malé čekání mezi jednotlivými vlnami. Takže v tu chvíli nemá hráč, jak získat palivo, proto se v intervalu pár sekund generuje i bez nepřátelských objektů.

```
IEnumerator SpawnFuel()
{
      // cekani pred prvni vlnou, aby se hrac "rozkoukal"
      yield return new WaitForSeconds(Random.Range(spawnStart * 2, 12.0f));
      while (!gameOver)
      {
            if (gameOver)
            {
                  break;
            }
            // nastaveni nahodne pozice
            Vector2 spawnPos = new Vector2(Random.Range(-spawnValues.x,
            spawnValues.x), spawnValues.y);

            // Spawn meteoritu
```

```
            Instantiate(fuel, spawnPos, Quaternion.identity);

            // cekani
            yield return new WaitForSeconds(Random.Range(5.0f, 15.0f));
        }
}
```

Metoda *SetDifficulty* pokud je počet vln vyšší než 4, tak generuje náhodný počet sekund z intervalu (0,5;1.0) rozestup mezi vytvořením jednotlivých objektů. Také pokud je čekací doba mezi vlnami větší než 2,5 sekundy sníží se čekání o 0,5 sekundy.

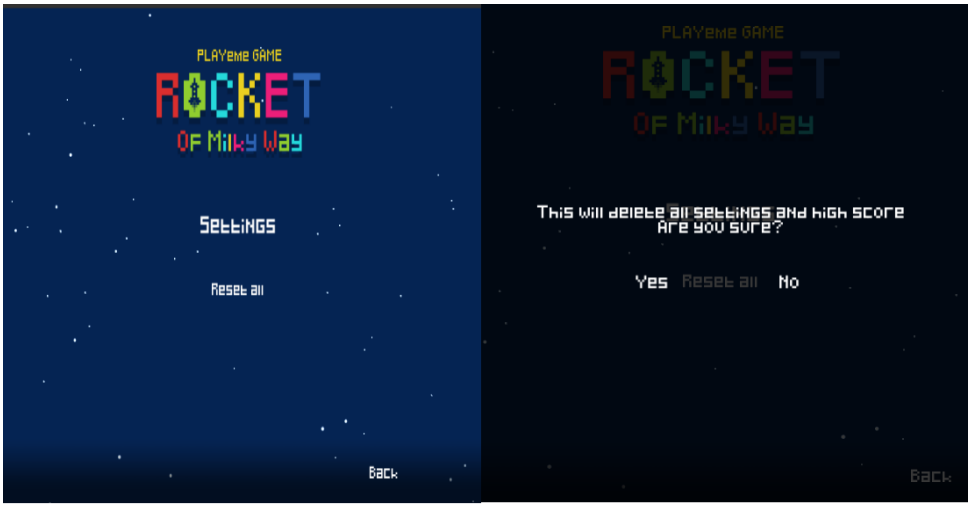
4.4.4 MenuController

Skript MenuController ovládá hlavní menu na scéně MainMenu. Podle akce uživatele zobrazuje nebo skrývá jednotlivé podmenu. V hlavním menu jsou celkem tři formuláře (aboutMenu. settingsMenu, confirmReset). Kontrolér obsahuje jednoduché metody, které pouze deaktivují nebo aktivují požadované zobrazení. Výchozí nastavení scény MainMenu je následující.

```
public void Awake()
{
      mainMenu.SetActive(true);
      aboutMenu.SetActive(false);
      settingsMenu.SetActive(false);
      confirmReset.SetActive(false);
      Time.timeScale = 1;
      loadingText.enabled = false;
}
```

V metodě Awake vidíme, že zůstane aktivní pouze hlavní menu a ostatní podmenu jsou deaktivována.

Náhled nastavení a dialogu na potvrzení vymazání herních dat zobrazuje obrázek 20.



Obrázek 20 - Náhled obrazovky nastavení a dialogu po kliknutí na tlačítko Reset all

4.4.5 ScreenResponsive

Stěžejní skript pro multiplatformní přizpůsobení obrazovky. Tento skript nastaví grafické zobrazení při spuštění nebo při změně rozlišení na odpovídající zobrazení. Obsahuje rozsáhlou metodu *ScreenSize*, která obsahuje definici zobrazení pro jednotlivé platformy. Pomocí direktiv preprocesoru, které udávají, pro kterou platformu je daný blok kódu přizpůsoben, obsahuje výsledné sestavení aplikace pouze blok kódu, který je určen pro danou platformu. Ukázka direktivy pro platformu *Standalone* a *WebGL*.

```
#if UNITY_STANDALONE || UNITY_WEBGL
#endif
```

Určení rozložení zobrazení proběhne na základě poměru stran obrazovky. Komponenta Camera nám vrací aktuální poměr stran ve scéně. Pokud je poměr větší než jedna, jedná se o rozložení obrazovky na šířku (16:9 = 1,777 atd.). Pokud je poměr menší než jedna, jde naopak o rozložení obrazovky na výšku. Podle toho jsem schopen určit orientaci obrazovky a rozhodnout se v jakém rozložení vše bude aktivováno. Ukázka nastavení podle poměru stran obrazovky.

```
if (Camera.main.aspect >= 2.2)
{
      PlayerSet(12.0f, 6.0f);
      SpawnerSet(new Vector2(11.6f, 6.0f));
}
else if (Camera.main.aspect >= 2)
{
      PlayerSet(9.9f, 6.0f);
      SpawnerSet(new Vector2(11.6f, 6.0f));
      Debug.Log("Main aspect >= 2");
}
```

4.5 Herní mechaniky

Herní titul Rocket of Milky Way obsahuje několik herních mechanik, které představují cíl hry, určují hratelnost nebo obtížnost hry.

4.5.1 Palivo rakety

Hráč si musí hlídat ukazatele paliva a průběžně palivo doplňovat, v případě, že palivo klesne na nulu hra končí. Palivo je odečítáno v průběhu času. Výchozí nastavení je, že každých 0,1 sekundy se odečte 0,005 ze zbývajícího paliva. Stav paliva je desetinné číslo v intervalu 0–1, kdy 0 znamená prázdno a 1 znamená plno. Pokud dojde hráči palivo, zbývají mu 2 sekundy a pokud do této doby nesebere palivo, hra končí. Palivo, které hráč sbírá je zobrazeno na obrázku 21.

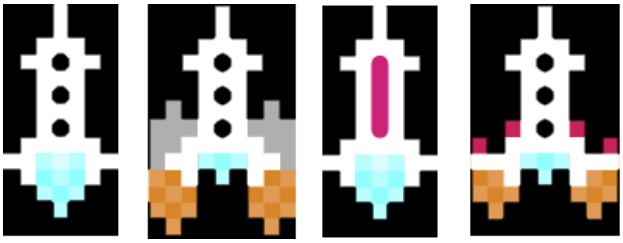
Obrázek 21 – Palivo

4.5.2 Střelba nepřátel

Nepřátelské rakety po hráči střílejí v náhodný čas. Náhodný čas střelby se generuje z intervalu 0,35 až 0,85 sekund. Střely nepřátel jsou zvětšeny o dalších 30 procent oproti střelám hráče.

4.5.3 Vylepšování rakety

Hra obsahuje celkem čtyři modely rakety. Základní raketa střílí po jedné střele jedno za 0,45 sekundy. V případě, že je hráč zasažen a má základní raketu, odečte se mu jeden život.



Obrázek 22 – Raketa typu A, B, C, D (zleva)

Druhý typ rakety B střílí po dvou střelách jednou za 0,40 sekund. Pokud je hráč zasažen, raketa je deaktivována a hráč má opět výchozí raketu. Raketa typu B je vidět na obrázku 22, druhá raketa z leva.

Raketa typu C, je nejsilnější ve hře. Hráč střílí 5 střel po dávkách jednou za 1,25 sekundy. Jednou dávkou sestřelí i největší asteroid.

Posledním typem rakety je typ D. Střely jsou uskupeny do tvaru šipky a kadence střelby je 0,4 sekundy.

5 UŽIVATELSKÁ DOKUMENTACE

Tato kapitola popisuje, jak aplikaci nainstalovat a ovládat.

5.1 Instalace aplikace

Aplikaci je možné nainstalovat prostřednictvím oficiálních obchodů pro danou platformu.

5.1.1 Android

Rocket of Milky Way můžete snadno nainstalovat prostřednictvím *Google Play* obchodu, kde můžete najít aktuální verzi hry.

5.1.2 iOS

V současné době není hra umístěna na *Apple Store*, protože pro umístění aplikace do obchodu je potřeba zakoupit vývojářský účet. Licence se platí ročně a poplatek není nízký. Proto jsem si licenci nezakoupil. Hru mohu na zařízení společnosti *Apple* nainstalovat skrze službu *Unity Cloud*, která hru sestaví a zašle emailem odkaz ke stažení.

5.1.3 Windows 10 Universal app

Pro platformu *Windows 10 Universal*, můžete nainstalovat herní titul prostřednictvím oficiálního obchodu zvaný *Windows Store*. Tento obchod můžete nalézt v operačním systému Windows 10 nebo Windows 10 mobile pro mobilní zařízení jako jsou chytré telefony a tablety. V obchodě je aktuální verze hry.

5.1.4 Standalone

Pro platformu *Windows*, *Linux* a *Mac OS* je hra dostupná v odpovídajícím balíčku. Pro platformu *Windows desktop* je možné spustit hru skrze spustitelný soubor s příponou exe.

5.2 Instrukce

Po spuštění hry se uživateli zobrazí hlavní menu, kde může kliknutím na tlačítko PLAY hru spustit. Ovládání pro osobní počítače je nastaveno ve výchozím nastavení na klávesy šipka doleva a doprava., kterými můžete ovládat pohyb rakety. Pro dotykové obrazovky je ovládání přizpůsobeno rozdělením obrazovky na dvě poloviny, dotykem na levé polovině uvedete pohyb rakety doleva a naopak.

6 ZÁVĚR

Cílem této bakalářské práce bylo vytvořit multiplatformní hru pomocí herního engine Unity3D. Teoretická část měla za úkol popsat proces vývoje multiplatformní hry.

První část bakalářské práce se věnuje popisu tvorby hry s důrazem na návrh a koncept hry. Kapitola pojednává o tom, jak by měl vypadat herní design, jak správně zvolit technologie nebo si třeba vytvořit klíčové vlastnosti herního titulu. Dále jsou probrány důležité etapy vývoje hry, jako je sestavení prototypu nebo testovací fáze.

Druhá kapitola této práce se zaměřuje na popis nástroje Unity3D. V této části je uveden popis editoru Unity3D a jeho částí. Bylo zde popsáno, co je to Asset, jak je používat a tvořit. Dále je zde rozebráno, jaké máme možnosti v případě skriptování, jaké programovací jazyky jsou používané a který je nejpoužívanější. Poukázal jsem na strukturu C# skriptu a rozebral důležité herní komponenty jako je hlavní kamera nebo světla ve scéně. Následně jsem popsal, jaké důležité funkce nám Unity nabízí a uvedl, jak funguje fyzický engine a renderování grafiky.

Třetí část se zaměřuje na zvolené platformy a jejich popis. Jsou zde představeny platformy *Standalone*, *Android*, *iOS* a *Windows 10 Universal Application*.

Čtvrtá kapitola popisuje celou implementaci praktické části bakalářské práce. Kde jsem popsal návrh herního titulu *Rocket of Milky Way*, například grafická stránka hry, zvuky a cíl hry. Je zde vyobrazena struktura celého projektu a rozebrány jednotlivé herní mechaniky. Nejdůležitější herní skripty jsou v této kapitole ukázány a vysvětleny.

Poslední kapitolou je uživatelská dokumentace, která poskytuje informace, jak herní titul nainstalovat a ovládat pro jednotlivé platformy.

Výsledná praktická část je plně fungující herní titul zvaný *Rocket of Milky Way*. Je to 2D arkáda, kde hráč v podobě vesmírné rakety letí vesmírem a sestřeluje nepřátelské objekty, například asteroidy nebo nepřátelské mimozemšťany. Hra je přizpůsobena pro mobilní platformy *iOS*, *Android* a *Windows 10 Mobile*. Dále je hra dostupná pro osobní počítače s operačním systémem *Windows*, *Linux* nebo *Mac OS*. Hra umožnuje hráči celkem tři vylepšení rakety, přičemž se každé vylepšení liší typem střílení nebo kadencí střel.

Do budoucna by se hra dala vylepšit, například přidáním znevýhodnění hráče, a to zhoršenou viditelností, snížením dočasné manévrovatelnosti rakety atd.

Seznámil jsem se s problematikou vývoje her, a tím získal zkušenost s nástrojem a herním enginem *Unity3D*. Navrhl jsem hru a následně ji realizoval, proto hodnotím výsledek práce jako kladný. Vzhledem k úspěšnému dokončení projektu budu v herním vývoji pokračovat a hru nadále rozvíjet.

Tato práce může sloužit jako podklad pro seznámení a osvojení si základů vývoje s nástrojem *Unity3D*.

POUŽITÁ LITERATURA

[1] Intel Software, *Ideate | Game Dev | Intel® Software* [online]. 2016 [citováno 3. 3. 2017]. Dostupné z: https://software.intel.com/en-us/gamedev/journey/design

[2] Miko Charbonneau, UBM, *Choosing the Perfect Game Engine* 3 [online]. 2013 [citováno 5. 3. 2017]. Dostupné z: http://www.gamasutra.com/blogs/MikoCharbonneau/20130222/187185/Choosing_the_Perfect_Game_Engine.php

[3] Unreal Engine. *Wikipedia contributors* [online]. Wikipedia, The Free Encyclopedia., 2017 [citováno 10. 3. 2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Unreal_Engine&oldid=779246355

[4] CryEngine. *Wikipedia contributors* [online]. Wikipedia, The Free Encyclopedia., 2017 [citováno 10. 3. 2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=CryEngine&oldid=777146246

[5] *Wikipedia, The Free Encyclopedia*, Unity (game engine) [online]. 2017 [citováno 10. 04. 2017]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Unity_(game_engine)&oldid=778204830

[6] Unity Technologies, *Unity - Manual: Unity User Manual (5.6)* [online]. 2017 [citováno 18. 4. 2017]. Dostupné z: https://docs.unity3d.com/Manual/index.html

[7] Ian Zamojc, Code Envato Tuts+, *Introduction to Unity3D*, 2012 [citováno 20. 4. 2017]. Dostupné z: https://code.tutsplus.com/tutorials/introduction-to-unity3d--mobile-10752

[8] GOLDSTONE, Will. *Unity 3.x game development essentials: game development with C# and Javascript*. 2nd ed. Birmingham, UK: Packt Publishing, 2011. ISBN 9781849691444.

[9] OKITA, Alex. *Learning C# programming with Unity 3D*. 2014. ISBN 1466586524.

[10] Michal Hala, *Zastoupení operačních systémů na desktopu: leden 2015* [online]. 2016 [citováno 28. 4. 2017]. Dostupné z: http://www.svethardware.cz/zastoupeni-operacnich-systemu-na-desktopu-leden-2015/39930

[11] *Freesound* [online]. Barcelona, Spain: Music Technology Group of Universitat Pompeu Fabra, 2005 [citováno 10. 5. 2017]. Dostupné z: http://freesound.org

[12] The Priesthood at Play: Computer Games in the 1950s. Videogame Historian [online]. 2014 [citováno 28. 4. 2017]. Dostupné z: https://videogamehistorian.wordpress.com/2014/01/22/the-priesthood-at-play-computer-games-in-the-1950s/

[13] Google's Android OS: Past, Present, and Future. PhoneArena.com [online]. 2011 [citováno 29. 4. 2017]. Dostupné z: http://www.phonearena.com/news/Googles-Android-OS-Past-Present-and-Future_id21273

[14] What's a Universal Windows Platform (UWP) app? *Microsoft Docs* [online]. 2017 [citováno 29. 4. 2017]. Dostupné z: https://docs.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp

PŘÍLOHY

Příloha A – *Koncept praktické části*