Mendelova univerzita v Brně
Provozně ekonomická fakulta

Virtuální realita na mobilních zařízeních

Diplomová práce

Vedoucí práce:
Ing. David Procházka, Ph.D.

Bc. Marek Musil

Brno 2017

Děkuji panu Ing. Davidu Procházkovi, Ph.D. za odborné vedení práce, cenné rady, nápady a velkou ochotu. Dále děkuji za maximální podporu mé rodině a přítelkyni.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Virtuální realita na mobilních zařízeních** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně 3. 1. 2017 .....

Abstract

MUSIL, M. Virtual reality on mobile devices. Diploma thesis. Brno: MENDELU 2017.

This thesis is focused on development of a game based on mobile virtual reality. The models used in the game are created in *3ds Max* application. The game itself is implemented using the *Unity* engine and the scripts are written in *C#* language. The user can control the game naturally using the sight on the basis of the cell phone data. Target platform of the game are mobile devices with operating system *Android* and the game requires using of virtual reality viewer *Cardboard*.

Keywords

Virtual reality, mobile devices, Cardboard, Android, mobile game, Unity, 3ds Max.

Abstrakt

MUSIL, M. Virtuální realita na mobilních zařízeních. Diplomová práce. Brno: MENDELU 2017.

Tato práce se zabývá tvorbou hry z prostředí mobilní virtuální reality. Modely použité ve hře jsou vytvořeny v programu *3ds Max*. Samotná hra je vytvořena v prostředí herního enginu *Unity* a skripty použité ve hře jsou implementovány v programovacím jazyce *C#*. Hra uživateli umožňuje přirozené ovládání pohledem na základě dat ze vstupních zařízení chytrého telefonu. Aplikace je určena pro zařízeních s operačním systémem *Android* a vyžaduje použití brýlí pro virtuální realitu *Cardboard*.

Klíčová slova

Virtuální realita, mobilní zařízení, Cardboard, Android, mobilní hra, Unity, 3ds Max.

Obsah

1 Úvod a cíl práce

1.1 Úvod

Virtuální realita je jedním z aktuálních témat v oblasti informačních technologií, ačkoli její prvopočátek sahá již k roku 1962. Od této doby přichází s pravidelností každých 10 až 15 let vlny zvýšeného zájmu, přičemž v mezičase je virtuální realita z hlediska mainstreamových technologií téměř zapomenuta.

Období, ve kterém se nacházíme nyní, představuje dosud největší vlnu obnovení zájmu o tuto technologii a troufám si tvrdit, že se nacházíme na prahu doby, kdy virtuální realita nalezne uplatnění nejen u technologických nadšenců, ale i v každodenním životě lidí. Vystoupení technologie z okraje zájmu vždy přináší jednak její výrazné zlevnění, ale hlavně kvalitní obsah, který je pro životaschopnost kterékoliv platformy naprosto klíčový.

Rok 2016 byl z toho pohledu pro virtuální realitu přelomový a na trh dorazilo vlastní řešení od velké řady předních technologických společností. Jako hlavní zástupce ze zařízení pro zprostředkování virtuální reality je možno jmenovat například *Oculus Rift*, *Playstation VR*, *Samsung Gear VR* nebo *HTC Vive*. Každý z výrobců se vydal lehce jiným směrem, ačkoli v principu se u všech zařízení jedná o virtuální brýle.

Z hlediska přístupu lze zařízení rozdělit na dražší a více specializované (například *Oculus Rift* nebo *HTC Vive*) a levnější univerzálnější řešení (*Samsung Gear VR*). Vůbec nejlevnější variantou je použití mobilního telefonu, kterým uživatel disponuje, a dokoupení VR brýlí. Již je pryč doba levných *Cardboardů*[1] a na trhu je možno vybírat z řešení, které není nutné před obličejem držet rukou. Lepší typy brýlí disponují popruhy pro uchycení na hlavě, čočkami s dioptrickou korekcí a pohodlným polstrováním.

V rámci Mendelovy univerzity se jedná o první diplomovou práci, která se virtuální realitou na mobilních zařízeních zabývá. Vývoj hry na mobilní zařízení přirozeně vyžaduje znalosti z oblasti 3D modelování a programování, nicméně k tomuto základu se připojuje i mnoho dalších oborů. Přínosem práce pro čtenáře by mohlo být shrnutí těchto kroků do jednoho celku a ulehčení práce dalším, kteří se vydají podobnou cestou.

1.2 Cíl práce

Cílem této diplomové práce je návrh a implementace hry pro virtuální realitu na mobilních zařízeních s přirozeným ovládáním pomocí vstupních zařízení telefonu. Hra bude umožňovat průchod virtuálním prostředím a interakci s objekty, které se v prostředí nachází. V rámci hry se bude odehrávat scénář, který má uživatel za úkol dokončit.

[1] Brýle pro virtuální realitu vyrobené z kartonu.

Dílčí cíle diplomové práce

Pro dosažení cíle je nezbytné splnit následující dílčí cíle:

- Seznámit se s problematikou virtuální reality.
- Prozkoumat existující řešení v oblasti virtuální reality.
- Zvolit softwarové prostředí pro vytvoření hry.
- Vytvořit modely, které se ve hře budou nacházet.
- Navrhnout způsob ovládání chůze a způsob interakce s prostředím.
- Vytvořit scénář a výslednou hru exportovat na cílovou platformu.

2 Virtuální realita

Virtuální realita (v práci bude využívána zkratka VR) je v současné době velice často skloňované téma. V této kapitole bude definováno, co virtuální realita představuje a jaké jsou její základní komponenty. Dále bude zmíněno využití virtuální reality v praxi a její problémové oblasti zejména z pohledu vnímaní uživatele. Následně budou zhodnoceny přístupy k ovládání mobilní virtuální reality u stávajících her.

2.1 Definice VR

Kniha *Virtual Reality Technology* (Burdea – Coiffet, 2003) definuje virtuální realitu z hlediska funkcionality jako simulaci s využitím počítačové grafiky, která je použita pro vytvoření realisticky vypadajícího světa. Virtuální realita je vysokoúrovňové uživatelské rozhraní zahrnující simulaci a interakci v reálném čase a využívá smysly uživatele jako jsou zrak, sluch, hmat, čich nebo chuť (Burdea – Coiffet, 2003).

Klíčovou vlastností virtuální reality je interaktivita v reálném čase, což znamená, že systém je schopen vyhodnotit vstup od uživatele a současně s ohledem na tento vstup promítnout změnu do virtuálního prostředí. Simulace ve virtuální realitě reaguje na vstup od uživatele v podobě gest, pohybů, ústních příkazů a tak dále (Burdea – Coiffet, 2003).

V knize *Virtual Reality Technology* (Burdea – Coiffet, 2003) jsou jako hlavní vlastnosti virtuální reality definovány interaktivita, imerze a také představivost. Interaktivita zahrnuje možnost provádění úkonů, kterými uživatel mění virtuální prostředí. Imerze (ponoření) vytváří pocit, že se uživatel v umělém prostředí reálně nachází. Představivost uživatel potřebuje k vnímaní a uvěření virtuálnímu světu (Burdea – Coiffet, 2003).

Základní komponenty zařízení pro zprostředkování virtuální reality jsou:

- zobrazovací zařízení,
- vstupní zařízení pro snímání pohybu,
- výstupní zařízení pro zvuk, hmat, čich, chuť,
- výpočetní jednotka,
- program/hra.

Příbuzná VR je realita augumentovaná. Zatímco VR pracuje v umělém prostředí, které je kompletně generováno pomocí počítačové 3D grafiky, augumentovaná realita oproti tomu do reálného prostředí doplňuje informace, počítačem vytvořené modely, obrazy atd. Augumentovaná realita může být založena na projektorech, průhledných displejích překrývající výhled, popřípadě na obrazovkách, skrze které je prováděna video kompozice obrazu (Ong – Nee, 2004).

2.2 VR v praxi

Již v minulosti byla virtuální realita používána ve vojenství a zdravotnictví, kde ji bylo možné nasadit nasadit i přes vysokou cenu hardwaru (Burdea – Coiffet, 2003). Zlevnění technologie ale otevírá nové možnosti v oblasti rehabilitace, prezentace, zábavy i umění.

Vojenství

Virtuální realita je využívána k tréninku v letecké armádě, pozemní armádě i v námořnictvu. Bez rizika zranění, nebo smrti, mohou vojáci trénovat v leteckých simulátorech, bitevních simulátorech, lékařských simulátorech, v simulátorech vozidel nebo ponorek a tak dále. VR nemůže nahradit reálný výcvik, ale může vojákům výrazně pomoci zvládat stresové situace, které konflikt přináší (Virtual Reality Society, 2016).

Rovněž může být virtuální realita prostředkem pro překonání posttraumatických šoků a vojáci trpící traumatem z bojiště se mohou učit své symptomy zvládat v bezpečném prostředí (Virtual Reality Society, 2016).

Hlavní výhodou tohoto nasazení VR je úspora času i finančních prostředků (Virtual Reality Society, 2016). Velkým benefitem je ale i možnost analýzy postupu simulátorem, která umožňuje podrobně zkoumat každou interakci provedenou uživatelem (Parkin, 2015).

Často používaným nástrojem pro nábor i trénink vojáků je taktická multiplayerová hra *America's Army* z roku 2002. Dále jsou běžně nasazovány specializované VR simulace vyvíjené například společnostmi Plextek[2], DoDAAM[3] a dalšími (Parkin, 2015).

V současné době je v armádě jako zobrazovací zařízení nejčastěji používán *Oculus Rift*, který je doplněn dalšími prvky simulace (haptika, čich atd.) pro umocnění zážitku z prezentované situace (Parkin, 2015).

Zdravotnictví a rehabilitace

V minulosti byl rozvoj technologie virtuální reality v oblasti zdravotnictví zpomalován absencí standardů v datových formátech, ověřovacích opatřeních i cenou samotné VR technologie. VR přináší do zdravotnictví řadu výhod v podobě snadnějšího vzdělávání lékařů (chyby způsobené na virtuálním modelu místo na reálném) nebo reálnější postupy certifikace (přesnější měření chirurgické dovednosti) (Burdea – Coiffet, 2003).

Zlevnění technologie otevírá nové možnosti v oblasti rehabilitace. Pro pacienty s popáleninami vede použití VR ke snížení bolesti ze zranění. Studie provedená ame-

[2]Společnost spolupracující s Britskou vládou na vojenských simulacích od konce 80. let (Parkin, 2015).

[3]Vojenský dodavatel z Jižní Koreji experimentující na poli virtuální reality (Parkin, 2015).

rickou vládou v roce 2011 prokázala, že VR hra *SnowWorld* popáleným pomáhala od bolesti více než morfium (Carson, 2015).

Pacientům trpícím bolestí fantomové končetiny[4] nasazení virtuální reality přináší uvolnění. V praxi vše funguje tak, že senzory snímají nervové výstupy z mozku a promítají pohyb například chybějící paže do hry. Pacient má za úkol plnit jednoduché úkoly, čímž postupně získává kontrolu nad bolestí (Carson, 2015).

Článek *Astrojumper: Motivating Children with Autism to Exercise Using a VR Game* (Finkelstein et al., 2010) pojednává o využití CAVE[5] VR hry *Astrojumper* pro motivaci dětí s autismem k pohybu. Hráč musí provádět fyzické pohyby k vyhnutí se kolizi a za postup získává body.

Pro mladé dospívající s autismem může být virtuální realita využita pro trénování sociálních dovedností. Uživatelé jsou umísťování do situací jako jsou pracovní pohovory nebo schůzky s opačným pohlavím, přičemž mají za úkol rozvíjet své sociální dovednosti (Carson, 2015).

Aplikace *DEEP* pro *Oculus Rift* pomáhá uživatelům překonávat úzkost pomocí meditativního dýchání, které je zároveň jediným ovládacím prvkem hry. *DEEP* používá pás kolem hrudníku, který monitoruje dech. Dýchání přemísťuje hráče z místa na místo, čímž je *DEEP* hratelný i pro uživatele, kteří nejsou schopni používat jakýkoli ovladač (Carson, 2015).

Prezentace techniky

Některé společnosti začínají využívat virtuální realitu pro prezentaci svých výrobků. Pro platformu *Oculus Rift* vydal například *Canon* v průběhu listopadu 2016 aplikaci s názvem *Camera Simulator by Canon Labs* (Oculus VR, 2016). Ve hře je možné aktuálně vybírat ze tří fotoaparátů a tří objektivů. Hráč může nastavovat expozici, měnit rovinu zaostření nebo ovládat pohyb postavy s cílem získat co nejlepší snímek.

Obrázek 1: Aplikace *Camera Simulator by Canon Labs* slouží k vyzkoušení zrcadlovek výrobce *Canon* (Oculus VR, 2016)

[4]Pocit pacienta, že chybějící část těla je stále připevněna k tělu.
[5]Virtuální realita využívající promítání na 3–6 stěn.

V současné době se ve hře nachází fotoaparáty s velikostí čipu APS-C[6] i Full-frame[7], takže je možné sledovat vliv tohoto prvku na výslednou fotografii. Pro objektivy může být aplikace ještě lépe využitelná, jelikož je možné dobře sledovat vliv ohniska na změnu úhlu záběru. Přesto se domnívám, že by *Camera Simulator by Canon Labs* mohl mít větší úspěch na rozšířenější platformě typu *Google Cardboard*.

Zábava, umění a ostatní využití

V oblasti zábavy je popularizace virtuální reality nejvíce zřejmá. Pro dostupné platformy přicházejí hry, které buď virtuální realitu podporují jako jiný způsob zobrazení, nebo jsou pro virtuální realitu přímo vytvořeny a jejich herní design z této platformy v maximální možné míře těží.

Za první kategorii lze zmínit například válečnou hru *War Thunder*, která pracuje s VR pohledem z kokpitu, ale k ovládání hry slouží standardně myš a klávesnice. Pro *HTC Vive* již nyní existuje řada her, které v herním designu využívají pohyb po prostoru při řešení různých logických úkolů. Jako zástupce této kategorie lze zmínit *The Gallery: Call of the Starseed*. Mezi hry, které z VR platformy v maximální možné míře těží, patří *Tilt Brush*. Náplní tato hra mírně přesahuje do umění, jelikož se jedná o kreslení ve 3D prostoru. Dle zkušeností uživatelů jde o jeden z nejlepších VR herních zážitků roku 2016 (Prasuethsut, 2016).

V oblasti umění nachází virtuální realita uplatnění při prezentaci uměleckých děl (například obrazy a sochy), při koncertech a v mnoha dalších oblastech (Google Play, 2016).

Článek *Virtual Reality Applications in Forensic Psychiatry* pojednává o využití virtuální reality pro kriminální psychiatrii. Soudní rozhodovatelé musí být schopni porozumět násilnému chování pachatelů a VR pomáhá s obnovením jak z metodologického, tak z teoretického úhlu pohledu na vyšetřovaný čin (Benbouriche et al., 2014).

2.3 Problémové oblasti VR

Jeden z problémů virtuální reality je její snášení uživateli. Z hlediska technologie je základním klíčem k úspěchu dodržení snímkové frekvence alespoň 25 snímků/s (lépe ale 60 snímku/s), co nejvyšší možné rozlišení obrazu a dodržení latence aplikace nižší než 50 ms, která musí být ideálně neměnná v čase (Burdea – Coiffet, 2003). Ani tyto podmínky ale nezaručí bezproblémové přijetí virtuální reality ze strany uživatele.

Jak o problematice pojednává článek *VR sickness cannot be solved by hardware improvements alone*, samotné vylepšování hardwaru na lepší vstřebávání virtuální reality nestačí a je nutno se zabývat i samotným softwarem (Handrahan, 2015). Problémem paradoxně může být, když je zážitek z virtuální reality příliš intenzivní, a to

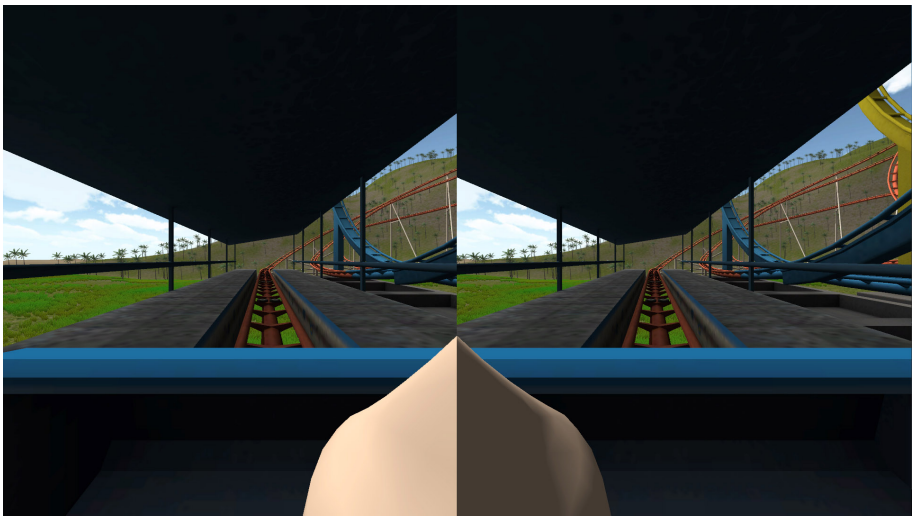
[6]Fotoaparát s čipem o velikosti přibližně 25×16 mm.

[7]Fotoaparát s čipem o velikosti 36×24 mm.

natolik, že aktivuje ochranné mechanizmy našeho těla. Skutečnost, že oči vnímají pohyb a tělo je zároveň stacionární, působí u některých osob smyslové zmatení, které může vyústit v nevolnost. Příčinou této kinetózy je nesoulad vestibulárního systému s vjemem, který vidí oči. Běžné jsou ale i záchvaty strachu, dezorientace a závratě. Čím je zážitek ze hry intenzivnější, tím větší riziko výše zmíněného hrozí.

Obecně většina činností, které způsobí nevolnost v reálném životě, budou vést k nevolnosti i ve virtuální realitě včetně příliš rychlého pohybu nebo otáčení. Tímto problémem běžně trpí akční hry z pohledu první osoby, ve kterých se hráč pohybuje nepřirozeně rychle a provádí prudké zrychlení následované okamžitým zpomalením (Handrahan, 2015). Rovněž se ukázalo, že postupné zrychlování činí uživatelům větší problémy než okamžitá akcelerace a je nutné se vyhnout simulaci pohybů hlavy.

Na výzkumu příjímání virtuální reality a možném zlepšení se ve světě podílí mnoho týmů. Zajímavý experiment vznikl na *Purdue University*, kde se výzkumníci do scény pokusili umístit virtuální nos (viz obr. 2). Tento krok vedl k výraznému nárůstu snášenlivosti virtuální reality mezi testovanými uživateli, přičemž uživatelé nos ve scéně ani nevnímali. Za tímto zlepšením pravděpodobně stojí fakt, že mozek lépe vnímá scénu, ve které se nachází pevné vizuální prvky, což se osvědčilo například u leteckých simulátorů obsahující kokpit (Whittinghill, 2015).



Obrázek 2: Experimentální přidání virtuálního nosu do renderované scény (Whittinghill, 2015)

Dle vlastní zkušenosti je rovněž vhodné hráče na virtuální prostředí nechat přivykat postupně a nestavět ho rovnou při spuštění aplikace do stresující situace. Na základě poznatků z této kapitoly vyplývá pro implementaci požadované hry několik pravidel:

- Testované zařízení musí umožňovat běh aplikace alespoň 25 snímků za sekundu.
- Pohyb ve hře nesmí být příliš rychlý.
- Akcelerace u chůze musí být okamžitá.

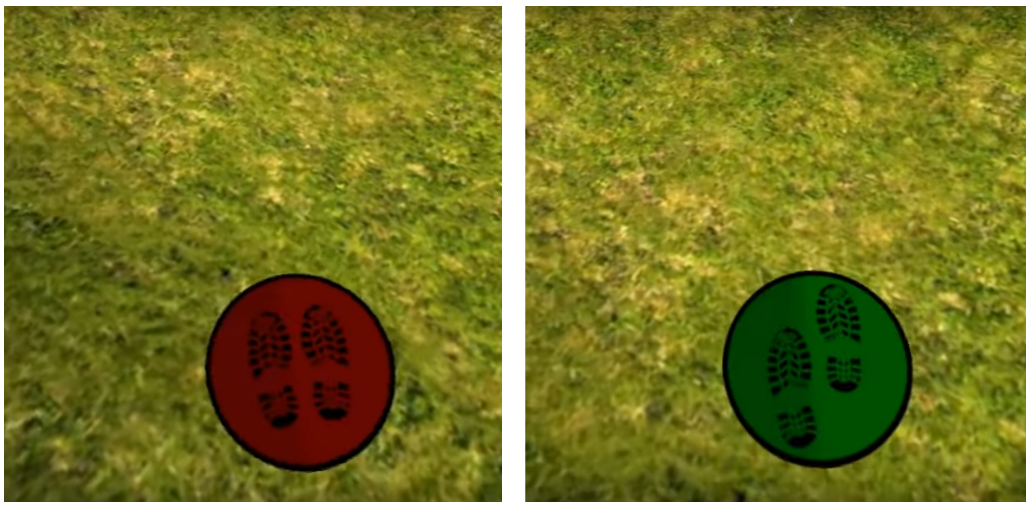
- Aplikace nesmí simulovat pohyb hlavy.
- Zejména z počátku se ve hře nesmí vyskytovat situace, která by mohla způsobit nevolnost v reálném prostředí.

2.4 Existující hry pro mobilní VR

V *Google Play* se nachází velké množství her pro mobilní VR. Pro zhodnocení stávajícího řešení na tomto poli bylo vybráno několik zástupců, u kterých byl následně rozebrán způsob interakce s prostředím a jeho vhodnost pro účely této diplomové práce.

Tuscany Dive

Více než o hru v pravém slova smyslu se jedná o prezentaci technologie virtuální reality. Aplikace původně vznikla jako prezentační demo pro *Oculus Rift* a na *Android* byla pouze portována (Durovis, 2016).



Obrázek 3: Ovládání chůze pomocí herního prvku, který je umístěný pod herní postavou (pohledem na ovládací prvek je chůze aktivována/deaktivována) (Google Play, 2016)

Aplikace nenabízí herní obsah a prostředí je možné pouze procházet bez interakce. Aktivace i deaktivace chůze probíhá pohledem na ovládací prvek umístěný pod herní postavou.

House of Terror VR

Ve hře *House of Terror VR* uživatel prochází starým domem a má za úkol řešit logické hádanky.

V této hře je veškerá interakce i chůze přizpůsobena pro použití externího ovladače a hru není možné ovládat pomocí pohledu, přestože se ve hře nenachází speciální způsob interakce, který by se s přirozeným ovládáním vylučoval.

Obrázek 4: Hra *House of Terror VR* nemá žádné UI prvky a po prvním spuštění není ani zřejmé, jak hru ovládat (Google Play, 2016)

Wizard Academy VR Cardboard

Je hra z fantasy prostředí, ve které hráč prozkoumává město a jeho okolí, přičemž má za úkol ničit různé herní objekty a řešit logické hádanky.

Obrázek 5: Hra *Wizard Academy VR Cardboard* z fantasy prostředí nabízí několik způsobů ovládání paprsku pro ničení herních objektů (Google Play, 2016)

U hry *Wizard Academy VR Cardboard* je na výběr z více způsobů ovládání interakce. K zaměřování a potvrzení interakce je možné využít ovladač, který je možné vytvořit i v domácích podmínkách za pomocí tiskárny, lepenky a lepící pásky. Pozice tohoto ovladače v prostoru i jeho použití jsou analyzovány pomocí fotoaparátu (Realiteer, 2016). Ve hře je možné vybrat ze dvou verzí, a to z *Real Control 1* a *Re-*

al Control 2. Dále je možné ovládat interakci pomocí pohledu a jeho setrváním na předmětu interakce alespoň po dobu jedné sekundy.

Chůze se ovládá simulovanou chůzí na místě v reálném prostředí. Hra reaguje na houpavý pohyb, který je promítán do virtuálního pohybu ve hře.

Zombie Shooter

Zombie Shooter je akční hra s náplní v podobě střílení nepřátel v ponurém prostředí různých tunelů. Hráč pouze zaměřuje nepřátele a střelba je automatická.

Obrázek 6: Ve hře *Zombie Shooter* může hráč pouze zaměřovat nepřátele (Google Play, 2016)

Pohyb v této hře je rovněž automatický a průchod hrou je tedy striktně lineární bez možnosti volného pohybu.

2.5 Srovnání metod interakce

Některé hry pro převedení do virtuální reality nepotřebují změnu ovládání a upraveno je pouze zobrazení rozdělené obrazovky pro stereoskopické vidění. Mezi zástupce těchto her patří například *VR X-Racer*, kde ovládání probíhá nakláněním přístroje tak, jak je běžné u závodních her na mobilní platformě.

Velké množství her pro mobilní VR využívá pohyb automatický, popřípadě hráče dle potřeby teleportuje. Výhodou tohoto přístupu je zpravidla dobré snášení pohybu uživatelem, nicméně hra v tomto případě postrádá volnost. Mezi hry s automatickým pohybem patří různá technologická dema, relaxační prohlídky prostředí, ale i simulátory jízdy na horské dráze. Z testovaných her automatický pohyb vyu-

žívá výše zmíněný *Zombie Shooter* a pro účely mé diplomové práce hodnotím tento způsob ovládání chůze jako nevhodný.

Dalším přístupem k ovládání pohybu, který je aplikován například ve hře *Tuscany Dive*, je ovládání chůze pomocí prvku umístěném pod herní postavou. Výhodou je možnost prohlížet herní prostředí, aniž by docházelo k samovolné aktivaci chůze. Přesto považuji toto řešení za uživatelsky velmi nepřívětivé, protože pohledem přímo pod postavu hráč ztrácí pojem o prostoru a pohybu v něm. Přesnost pohybu je velmi omezena, jelikož pro orientaci je nutné zvednout pohled a pro následnou deaktivaci hlavu zase co nejrychleji sklopit. Tento způsob ovládání rovněž nereflektuje požadavek na přirozenost ovládání aplikace vytvářené v rámci této diplomové práce.

Speciální přístup k ovládání chůze se nachází ve hře *Wizard Academy VR Cardboard*, kde aktivace chůze probíhá simulovanou chůzí na místě v reálném prostředí. Jedná se o poměrně přirozený způsob ovládání, nicméně z pohledu uživatele není herní zážitek příjemný. V aplikaci se nachází simulovaný pohyb hlavy, což dle části 2.3 může vést k nevolnosti.

Využití ovladače v případě hry *House of Terror VR* sice umožňuje pohodlné a přesné ovládání hry, ale nesplňuje podmínku přirozenosti ovládání bez manuální interakce ze strany uživatele.

Použití předmětů ve hrách probíhá většinou skrze zaměření objektu pohledem a manuální aktivací pomocí magnetu *Cardboard trigger*. Druhým často využívaným přístupem je automatická aktivace po určité časové prodlevě, což představuje z mého pohledu uživatelsky příjemnější přístup a navíc splňuje požadavek na přirozené ovládání aplikace.

Dle článku *Controller-less Interaction Methods for Google Cardboard*, ve kterém byla zkoumána spojitost mezi způsobem interakce a výsledným hodnocením uživatelů na *Google Play*, je uživateli nejlépe přijímáno využívání náklonu k ovládání aplikace. Nejnižší hodnocení zaznamenaly aplikace, které k ovládání potřebují externí ovladač (Yoo et al., 2015). Za tímto výsledkem pravděpodobně stojí nízké zastoupení externích ovladačů v *Cardboard* komunitě.

Nižšího hodnocení dosáhly také aplikace využívající k ovládání magnet (Yoo et al., 2015). Na základě zkoumání konkrétních recenzí lze vyvodit závěr, že řada uživatelů má zařízení pro použití magnetu špatně nakonfigurováno a magnet v tomto případě nefunguje správně. Dalším důvodem může být absence magnetu u některých brýlí, které nesplňují standard pro *Google Cardboard*, ale přesto se na trhu pod tímto označením prodávají.

3 Zařízení pro zprostředkování virtuální reality

Jak bylo zmíněno v sekci 2.1, virtuální realita se skládá z několika komponent. V současné době nejrozšířenější řešení jsou takzvaně Head-Mounted Display (dále HMD), neboli obrazovka připojená k hlavě. V této kapitole práce budou představeny některé zařízení typu HMD, které se nachází na trhu, a blíže bude představeno zařízení zprostředkovávající mobilní virtuální realitu.

3.1 Situace na trhu v roce 2016

V roce 2016 dorazilo na trh řešení od celé řady výrobců a všichni se bez výjimky vydali cestou virtuálních brýlí (Lamkin, 2016). Obecně je možné vybírat z levnějších a dražších řešení, přesto každý z výrobců přistoupil k virtuální realitě lehce odlišně.

Mezi hlavní zástupce dražších zařízení pro zprostředkování virtuální reality patří *Oculus Rift*, *HTC Vive*, *Sony Playstation VR* nebo *Fove O*. Ve všech případech se jedná o brýle, které obsahují displej (nebo dvojici displejů) a k běhu potřebují výpočetní jednotku. Všechna zařízení jsou vybavena senzory, které slouží pro sledování pohybu hlavy. Obraz pro každé oko je mírně posunutý a vytváří tak stereoskopický efekt (Burdea – Coiffet, 2003).

Obrázek 7: Virtuální brýle (zleva): *Oculus Rift*, *HTC Vive*, *Playstation VR* a *Fove O* (Lamkin, 2016)

Oculus Rift jsou virtuální brýle, které v roce 2013 odstartovaly současnou vlnu velkého zájmu o virtuální realitu. Celkové rozlišení integrovaného displeje je 2160 × 1200 pixelů a pracuje s obnovovací frekvencí 90 Hz. Cena brýlí *Oculus Rift* je přibližně 17 800 Kč[8] (Lamkin, 2016).

HTC Vive je výsledkem spolupráce společností *HTC* a *Valve*. Jedná se o set, který obsahuje dva ovladače a set pro sledování hráče v místnosti (*Lighthouse room tracking*). *HTC Vive* je v současné době považován za vůbec nejlepší zařízení pro zprostředkování virtuální reality, ale s cenou přibližně 24 400 Kč[9] se zároveň jedná o nejdražší řešení (Lamkin, 2016).

[8]Cena uváděná na stránkách `www.wareable.com` činí 549 Eur a přepočet na Kč je platný k 9. 12. 2016.

[9]Cena uváděná na stránkách `www.wareable.com` je 759 Eur a přepočet na Kč je platný k 9. 12. 2016.

Sony Playstation VR využívá jako výpočetní jednotku *Sony Playstation 4*. S cenou 11 200 Kč[10] představuje zajímavé funkční rozšíření pro stávající majitele konzolí *Sony Playstation 4*, ačkoli z hlediska technologie (sledování pohybu a rozlišení obrazu) za předchozími dvěmi zmíněnými zařízeními zaostává.

Fove O se od ostatních zařízení odlišuje ve schopnosti sledovat oči hráče pomocí infračerveného senzoru, což otevírá jak nové způsoby ovládání, tak možnost simulovat hloubku ostrosti, protože systém ví, do kterého bodu se uživatel přesně dívá. S cenou přibližně 15 300 Kč[11] se řadí mezi levnější zařízení, ale v současné době je dostupná pouze vývojářská verze zařízení (Lamkin, 2016).

Mezi levnější vstupenky do virtuální reality patří *Google Cardboard*, *Google Daydream View* a *Samsung Gear VR*, nicméně ve všech případech je vyžadováno, aby byl do brýlí vložen chytrý telefon, který poté funguje jako displej, výpočetní jednotka a sada senzorů pro sledování pohybu hlavy. Zážitek je tedy vždy výrazně ovlivněn kvalitou vloženého mobilního zařízení.

Obrázek 8: Levnější brýle pro virtuální realitu na mobilních zařízeních zleva: *Google Cardboard*, *Google Daydream View* a *Samsung Gear VR* (Lamkin, 2016)

Cardboard byl společností *Google* ohlášen v roce 2014 a představoval opravdu levnou alternativu, jak zažít virtuální realitu. Běžný chytrý mobilní telefon s operačním systémem *Android* totiž obsahuje všechny potřebné senzory pro sledování pohybu hlavy. Jak název napovídá, *Cardboard* je vytvořen z papírového kartonu, nicméně v současné době je na trhu na výběr z mnoha brýlí, které jsou vyrobeny z plastu a díky popruhům je již není nutné před obličejem držet v rukou. *Cardboard* je zpravidla vybaven posuvným magnetem, který v aplikacích slouží pro potvrzování nabídek. Průměrná cena této varianty je přibližně 480 Kč[12], ale není v ní zahrnut nákup mobilního zařízení (Lamkin, 2016).

Google Daydream View představuje evoluci *Cardboardu* a na trh dorazil teprve 10. listopadu 2016. Platforma oproti *Cardboard* nově obsahuje softwarové i hardwarové specifikace a kompatibilní telefony budou v budoucnu označovány jako

[10]Cena uváděná na stránkách `www.wareable.com` je 349 Eur a přepočet na Kč je platný k 9. 12. 2016.

[11]Cena uváděná na stránkách `www.wareable.com` je 599 $ a přepočet na Kč je platný k 9. 12. 2016.

[12]Cena uváděná na stránkách `www.wareable.com` je 15 Eur a přepočet na Kč je platný k 9. 12. 2016.

Daydream-ready. U *Google Daydream View* je posuvný magnet nahrazen odděleným ovladačem, takže není nutné pro potvrzování herních nabídek udržovat ruce na brýlích. Cena *Google Daydream View* činí 2 200 Kč[13] bez mobilního zařízení (Lamkin, 2016).

Samsung Gear VR pracuje výhradně s telefony *Samsung*. Jedná se opět o stejný koncept představený v původním *Google Cardboard*, který je doplněn o tlačítka a vyroben z kvalitních materiálů. Cena *Samsung Gear VR* je přibližně 2 500 Kč[14].

Konstrukce Google Cardboard

Konstrukčně je *Google Cardboard* velice jednoduchý a základní verze se skládá pouze z papírového obalu, magnetu a dvou čoček. Za vyšší cenu se dají pořídit brýle vyrobené z plastu, vybavené pohodlným polstrováním a popruhy pro upevnění na hlavu. V následujícím textu bude popisována tato pokročilejší varianta, protože byla používána i v průběhu vývoje aplikace.

Obrázek 9: Brýle z kartonu s pořizovací cenou 120 Kč (nalevo) a kvalitnější brýle *iGET Virtual R1* s pořizovací cenou 450 Kč, které umožňují mimo jiné i nastavením dioptrické korekce

Brýle pro mobilní VR nemají vlastní displej, ale jako zobrazovací zařízení využívají displej telefonu. Displej je v tomto případě rozdělen na dvě poloviny tak, aby bylo možné zobrazovat pro každé oko odděleně posunutý obraz. Výsledkem toho-

[13]Cena uváděná na stránkách `www.wareable.com` je 69 Eur a přepočet na Kč je platný k 9. 12. 2016.

[14]Cena uváděná na stránkách `www.wareable.com` je 99 $ a přepočet na Kč je platný k 9. 12. 2016.

to posunu je stereoskopický efekt, neboť v mozku se oba obrazy spojí do jednoho a vytvoří tak iluzi 3D obrazu (Burdea – Coiffet, 2003).

Obrázek 10: Lepší typy brýlí pro mobilní VR často obsahují různé zásuvné šuplíky pro umístění telefonu, aby byla dodržena ideální vzdálenosti od čoček brýlí

Posunem magnetu dochází ke změně magnetického pole v okolí telefonu a při správné kalibraci v aplikaci *Cardboard* je tento posun vyhodnocen jako potvrzení nabídky. Špatná kalibrace v zařízení často vyústí v nefunkčnost magnetu, a proto bude samotná kalibrace zmíněna v další části této práce.

Obrázek 11: Posuvný magnet je zpravidla umístěn na některé straně brýlí a slouží pro potvrzení herních nabídek

Kvalita čoček ovlivňuje dojem z vnímaného obrazu a spolu s rozlišením displeje se výrazně podílí na věrohodnosti zobrazované scény. Levnější brýle mají čočky vždy

plastové, v dražších brýlích se mohou objevit čočky skleněné. Čočky jsou v brýlích potřebné z důvodu neschopnosti oka ostřit na velmi krátké vzdálenosti (vzdálenost displeje od oka je v brýlích od 3 do 7 centimetrů) (VR Lens Lab, 2016).

Mezi důležité parametry čoček patří nízké sférické zkreslení a co nejnižší chromatická aberace (barevná vada). Pro zachování kompaktních rozměrů a nízké váhy čoček jsou v brýlích často využívány fresnelovy čočky (VR Lens Lab, 2016).

Obrázek 12: Čočky jsou z hlediska kvality VR nejdůležitějším prvkem brýlí a zpravidla jsou rovněž nejdražší položkou celého setu

3.2 Senzory v mobilních zařízeních

Většina mobilních zařízení s operačním systémem *Android* disponuje senzory pro měření pohybu, rotace, orientace a dalších vstupních údajů[15]. Tyto senzory jsou schopné poskytovat data s vysokou přesností a je možné je použít pro určení pohybu v 3D prostoru, popřípadě pro monitorování blízkého okolí zařízení (Google Developers, 2016). Tím se otevírají široké možnosti pro herní využití těchto senzorů, které budou zmíněny v následujících podkapitolách.

Platforma operačního systému *Android* podporuje tyto tři hlavní kategorie senzorů:

- **Senzory pohybu** měří akceleraci a rotaci ve třech osách x, y, z. Do této kategorie spadá akcelerometr (senzor zrychlení), gyroskop a senzor rotace.
- **Senzory prostředí** měří různé parametry prostředí jako je teplota okolního vzduchu, hodnota tlaku, osvětlení nebo vlhkosti. Do této kategorie se řadí barometr, senzor osvětlení a teploměr.
- **Senzory pozice** dodávají hodnoty, které reprezentují pozici přístroje. Do této kategorie spadá senzor orientace, magnetometr a GPS senzor (Google Developers, 2016).

[15]Senzory používají i zařízení s jiným operačním systémem, nicméně v této práci bude aplikace vyvíjena pro operační systém *Android*.

K údajům z těchto senzorů je možné přímo přistupovat prostřednictvím *Android sensor framework*. Skrze *framework* je možné zjistit, zda je senzor v zařízení přítomen a jaký je rozsah měření a rozlišení senzoru, číst data ze senzoru a přidávat nebo odebírat události, které jsou na údajích ze senzoru závislé (Google Developers, 2016).

Akcelerometr a gyroskop

Existuje celá řada senzorů, které umožňují určit pozici a orientaci v prostoru. Nejběžnějšími z nich jsou akcelerometr a gyroskop. Ačkoli oba senzory mají podobný účel, jejich způsob měření je však odlišný (Goodrich, 2013).

Akcelerometr slouží k měření zrychlení v jednotlivých osách x, y, z a je navržen tak, aby při změně z klidového stavu zaznamenával změnu zrychlení. Pomocí akcelerometru je možné změřit fyzickou akceleraci i gravitační zrychlení v ose a tím určit natočení zařízení. (Google, 2016). Mezi zástupce her využívajících akcelerometr patří typicky závodní hry, u kterých se zatáčení auta ovládá náklonem telefonu.

Akcelerometr je kompaktní zařízení obsahující mikroskopické krystaly, jež reagují na vibrace spojené se zrychlením. Jakmile je zařízení uvedeno do pohybu, krystaly generují napětí, z kterého je možné zjistit velikost akcelerace (Goodrich, 2013).

Příklady hodnot čtených z akcelerometru jsou:

- Při volném pádu je vektor x, y, z nulový.
- Pohyb zařízením zleva doprava generuje kladnou hodnotu zrychlení na ose x.
- V klidovém stavu (například zařízení položeno na zemi) je hodnota zrychlení v ose z rovna hodnotě 9,81, což odpovídá akceleraci zařízení v ose z ($0\ m \times s^{-2}$) mínus síla gravitace ($9{,}81\ m \times s^{-2}$).
- Pokud je zařízení ze země zvednuto kladným směrem v ose z, je hodnota zrychlení v této ose vyšší než $9{,}81\ m \times s^{-2}$ (Google, 2016).

Gyroskop v telefonu dodává informace orientaci zařízení s využitím zemské přitažlivosti. Gyroskop měří úhlovou rychlost, a to opět ve třech osách x, y, z (Google, 2016).

Konstrukčně se gyroskop skládá z volně rotujícího setrvačníku, který je pomocí otočných os spojen se svým rámem. Setrvačník má tendenci zachovávat svoji osu rotace nezávisle na poloze rámu, což umožňuje určit směr natočení zařízení (Goodrich, 2013). Hodnoty získané z gyroskopu jsou u operačního systému *Android* dostupné prostřednictvím funkce *sensors_event_t.gyro* a všechny hodnoty jsou udávány v radiánech za sekundu (*rad/s*) (Google, 2016).

U her je gyroskop často využíván ve spojení s akcelerometrem, díky čemuž je zajištěna vyšší přesnost ovládání a lepší odezva zařízení na pohyb uživatele.

Ve vztahu k virtuální realitě představují akcelerometr a gyroskop nejdůležitější senzory, jelikož jsou používány k určení rotace hlavy uživatele (hlava v prostoru má ale šest stupňů volnosti: pozici x, y, z a rotaci x, y, z).

Bohužel vzhledem k úsporám nejsou některé současné telefony vybavovány gyroskopem a jeho funkce je zastoupena akcelerometrem a magnetickým senzorem[16]. Hodnoty z gyroskopu ale nemohou být pomocí těchto dvou zařízení emulovány z důvodu horší konzistence dat a horší odezvy na pohyb zařízením. Telefony bez gyroskopu proto nejsou standardně považovány jako *Cardboard* kompatibilní, ačkoli některé hry lze i přesto s nimi hrát.

Magnetický senzor a geolokace

Magnetický senzor dodává informace o okolním magnetickém poli, které je měřeno ve třech osách x, y, z. Měření ze senzoru jsou dostupné skrze *sensors_event_t.magnetic* a hodnoty jsou udávány v jednotce *mikrotesla* (µT) (Google, 2016).

Magnetický senzor je ve hrách používán pro potvrzování nabídek. Senzor reaguje na změnu magnetického pole při pohybu magnetu v brýlích *Google Cardboard* (označováno jako *Cardboard trigger*).

Pro geolokaci zařízení je v současné době nejčastěji používáno satelitů *GPS*, nicméně některé mobilní telefony jsou schopny přijímat i signál ze satelitů *GLONASS* (Rusko) a *COMPASS* (Čína) (Griffin, 2011).

Hry využívající lokalizační služby v herním designu byly v minulosti z důvodu své náročnosti rozšířeny spíše mezi nadšenci, než mezi běžnými hráči. V roce 2016 ale zaznamenala enormní úspěch hra *Pokémon GO*, u které se ukázalo, že chytrý herní design v kombinaci s motivací hráče může i běžné uživatele přimět k navštěvování určitých lokací pro další postup hrou. Pro virtuální realitu ale není využití lokalizačních služeb vhodné, protože osoba ponořená do virtuálního prostředí ztrácí pojem o prostředí reálném, což může vést k nehodě.

Ostatní senzory

V telefonech je přítomna řada dalších senzorů. Například kamera představuje senzor, který nachází široké uplatnění u augumentované reality. Některé senzory slouží pro lepší kalibraci ostatních senzorů (například teploměr), jiné rozšiřují a usnadňují použití přístroje (například barometr pro měření atmosferického tlaku nebo senzor otisku prstu pro rychlejší přístup do zařízení).

Vliv přesnosti senzorů na kvalitu virtuální reality

Zážitek z virtuální reality je výrazně ovlivněn přesností výstupů z akcelerometru a gyroskopu. Pro co nejlepší zážitek z virtuální reality musí aplikace na uživatele reagovat s co nejkratší odezvou a zároveň množství chyb a špatně vyhodnocených pohybů je nutno udržet na minimální úrovni. Chyby senzorů s největším vlivem na kvalitu virtuální reality jsou:

[16]U některých přístrojů je gyroskop v zařízení fyzicky přítomen, ale je zakázán softwarově a ponechán pouze pro dražší verzi zařízení – například telefon *Honor 7 Lite* a *Honor 7*.

- **Posun** (*drift*) je systematická chyba, která se projevuje změnou výstupního signálu nezávisle na měřené veličině. K eliminaci posunu se používá kalibrace, popřípadě data z ostatních senzorů.
- **Šum** (*noise*)je náhodná chyba signálu v čase. Šum může být odstraněn filtrací, popřípadě co nejlepším odstupem signálu od šumu.
- **Odezva** (*latency*) představuje jeden z klíčových aspektů pro virtuální realitu. Odezvu je možné vylepšovat jak hardwarovou, tak softwarovou optimalizací (Google, 2016).

4 Vývoj her pro mobilní virtuální realitu

Existuje celá řada nástrojů, které umožňují vyvíjet profesionálně vypadající hry pro mobilní virtuální realitu. V následující kapitole bude představeno samotné *Google VR* a dále bude provedena analýza nástrojů použitelných na vývoj her pro tuto platformu.

4.1 Google VR

Společnost *Google* označuje své technologie pro virtuální realitu souhrnně jako *Google VR*. Mezi tyto technologie patří *Daydream*, *Cardboard* a *Jump*, přičemž dvě zmíněné byly popsány již dříve. Pod pojmem *Jump* se skrývá vlastní standard pro natáčení 360° videí. Kamery jsou uspořádané do kruhu a software *Jump Assembler* se stará o bezešvé spojení videí do jednoho 360° videa. Výsledné video je možné umístit na *YouTube* a prohlížet například v *Google Cardboard* nebo *Google Daydream* (Google VR, 2016).

Osvědčené postupy Google VR

Návrh her pro virtuální realitu je výrazně odlišný od návrhu her standardních. Na stránkách *Designing for Google Cardboard* se nachází postupy pro *Google VR*, které jsou založeny na základě evaluace stávajících VR her. Jedná se ale i o tipy, které zohledňují základy lidského vnímání a jsou použitelné napříč všemi VR platformami (Google Design, 2016). Některé z těchto zásad byly popsány již v části 2.3 a následně tedy budou zmíněny pouze poznatky, které jsou uplatnitelné při návrhu požadované herní aplikace.

Sledování pohybu hlavy

- Sledování pohybu hlavy je nutné zachovat v průběhu celé aplikace. I krátká přestávka ve sledování pohybu hlavy může způsobit nevolnost.
- Překryvný 2D obraz je lepší nahradit 2D plochou umístěnou v 3D prostoru. V menu je nutné zachovat alespoň jeden stupeň volnosti, lépe ale tři stupně volnosti (rotace v osách x, y a z).
- Nečekané zaseknutí aplikace například při načítání objektů může opět způsobit problémy s nevolností (Google Design, 2016).

Ovládání aplikace

- Jestliže se v aplikaci nachází ovládací *UI* prvky, je nutné tyto prvky umístit do zorného úhlu uživatele ihned po spuštění aplikace.
- Jestliže je v aplikaci možný pohyb, *UI* prvky se musí pohybovat spolu s herní postavou.

- Pokud aplikace využívá automatické potvrzení interakce, uživatel musí být informován o průběhu interakce.
- Aplikace využívající automatické potvrzení interakce by měla umožňovat i okamžitou manuální interakci pomocí posuvného magnetu.
- Ovládací prvky v prostoru musí být umístěny dostatečně daleko od sebe, aby bylo možné předcházet chybné interakci.
- Uživatelé jsou schopni poměrně přesně zaměřovat malé objekty pohledem. Pro ještě jednodušší zaměření je možno využít vizuální prvek fungující jako zaměřovač (Google Design, 2016).

Zpětná vazba

- Textové instrukce jsou ve virtuální realitě špatně čitelné a pro komplexnější informace je vhodnější využívat hlasové instrukce.
- Pro maximální ponoření uživatele do hry je vhodné v aplikaci využívat zvukové efekty.
- Zvuk je možné využít k připoutání pozornosti uživatele na jakýkoli objekt.
- Na mobilní platformě je možné v omezené míře využívat i haptickou zpětnou vazbu jako další prostředek pro komunikaci s uživatelem (Google Design, 2016).

Konfigurace a kalibrace zařízení

Ke konfiguraci mobilního zařízení pro virtuální realitu se využívá aplikace *Cardboard* dostupná v obchodě *Google Play*. Tato aplikace kontroluje, zda telefon disponuje všemi důležitými senzory, a uvádí uživatele do světa *Cardboard* virtuální reality.

Po prvním spuštění je uživatel vyzván k naskenování *QR* kódu brýlí pomocí kamery. Tento kód se nachází buď na krabici od VR setu, nebo přímo na samotných brýlích. V případě nepřítomnosti *QR* kódu je možné vyzkoušet některý z kódů dostupných na internetu a vybrat vyhovující pro konkrétní brýle, přičemž všechny aplikace využívající *Google VR SDK* dle toho přizpůsobí své zobrazení. Součástí *Google VR SDK* je rovněž automatická konfigurace stereoskopického zobrazení v aplikaci pro přiřazený model virtuálních brýlí (Google Design, 2016).

Dále je v aplikaci uživatel vyzván k použití magnetu umístěném na brýlích, čímž je provedena kalibrace pro použití *Cardboard* tlačítka. Při výměně brýlí pro VR je vhodné smazat uživatelská data aplikace *Cardboard* skrze *Android* nastavení, aby bylo možné provést opětovnou kalibraci magnetu (ke změně *QR* kódu není třeba tyto data mazat).

4.2 Herní engine

Herní engine vzniká oddělením klíčových funkčních komponent hry jako jsou 3D grafický vykreslovací systém, fyzikální systém, audio systém a další komponenty od obsahových částí hry. Základní komponenty jsou v tomto případě univerzální a je možno je použít jako základ pro jinou hru bez nutnosti zásadních změn v jejich funkční logice. Herní engine vzniká zpravidla pro účely konkrétního žánru hry a při použití na stejný žánr hry dosahuje nejlepších výsledků. Existují i univerzální herní enginy, nicméně vždy existuje určitý kompromis mezi univerzálností na úkor optimality (Gregory, 2009).

Většina současných herních enginů je multiplatformní, což znamená, že výslednou hru je možné portovat na více hardwarových platforem (Crawlist Team, 2016).

4.3 Vývojové prostředí her

Na trhu je dostupná celá řada univerzálních multiplatformních enginů, z jejichž zástupců je možno jmenovat například *Unity*, *Unreal Engine*, *CryEngine* a *ShiVa3D*.

Unity je herní engine a vývojové prostředí vyvíjené společností *Unity Technologies* a představeno bylo původně pouze pro *Mac OS* v roce 2005. *Unity* v současné době umožňuje export na více jak 20 herních platforem, mezi kterými mimo běžné platformy nechybí ani *BlackBerry 10*, *Wii U*, *Linux* nebo *Windows Phone 8*. Tento herní engine nabízí jedny z nejlepších licenčních podmínek na trhu a velké množství vývojářů si vystačí s verzí zcela zdarma, která nabízí více než dostatečnou funkcionalitu (Crawlist Team, 2016).

Unreal Engine 4 je herní engine vyvíjený společností *Epic Games* a poprvé byl představen v roce 1998 při uvedení FPS hry[17] *Unreal*. *Unreal Engine* je jeden z nejlépe hodnocených herních enginů pro vývoj vysokorozpočtových her a byl použit například pro vývoj her typu *Batman: Arkham Asylum*, *Mass Effect* a dalších (Crawlist Team, 2016). Použití enginu je zdarma při obratu hry do 76 800 Kč[18] za čtvrtletí (Epic Games, 2016).

CryEngine 3 je dílem společnosti *Crytek* a na trh dorazil v roce 2009. Hlavní předností tohoto nástroje je vynikající grafický výstup a v rámci trhu má integrovány nejpokročilejší nástroje pro práci se zvukem. Rovněž engine obsahuje pokročilé nástroje pro práci s fyzikou, částicovým systém i s umělou inteligencí protivníků. *CryEngine 3* je k dispozici k použití zcela zdarma nezávisle na zisku ze hry, ve které je použit (Crawlist Team, 2016).

[17] First-person shooter, neboli střílečka z pohledu první osoby

[18] Obrat uváděný na oficiálních stránkách `www.unrealengine.com` činí 3 000 $ a přepočet na Kč je platný k 9. 12. 2016.

ShiVa3D je herní engine, který umožňuje export aplikací na více jak dvacet platforem včetně široké řady mobilních zařízení. Základní verze vývojového prostředí *ShiVa* stojí 5 120 Kč[19], pokročilá verze stojí 25 600 Kč[20] (Crawlist Team, 2016).

4.4 Srovnání Unity 5 vs. Unreal Engine 4

V současné době je balíček *Google VR SDK* dostupný pro herní enginy *Unity* a *Unreal*, a proto v následující části bude provedeno srovnání těchto dvou vývojových prostředí.

Skriptování vlastního obsahu

Primární programovací jazyk pro psaní skriptů v *Unity* je *C#* a dále je také podporován *Javascript*. Dříve podporovaný jazyk *Boo* byl ve verzi *Unity 5* odstraněn (Unity Manual, 2016). *Unity* má integrovaný editor zdrojového kódu *MonoDevelop*, popřípadě lze zvolit jiné vývojové prostředí pomocí nastavení projektu.

Unreal Engine umožňuje skriptovat reakce na uživatele a chování hry pomocí programovacího jazyku *C++*, nebo prostřednictvím vizuálního programování *Blueprint Editor*.

Vizuální schopnosti

Poslední verze *Unity* (*Unity 5*) přinesla výrazné zlepšení grafických schopností enginu. Podporována je většina současných technologií, nicméně i přesto *Unity* zaostává za schopnostmi *Unreal Engine*, který patří v tomto oboru ke špičce.

Editor materiálů je v případě *Unreal Engine* pokročilejší a umožňuje širší nastavení fyzikálních vlastností. *Unity* ale při podobných scénách dosahuje vyšší snímkové frekvence.

Podpora Google VR

Unity i *Unreal Engine* nativně podporují virtuální realitu na mobilních zařízeních, což znamená, že ani v jednom případě již není nutné do aplikace přidávat rozšíření v podobě *Google VR SDK*. V případě *Unity* stačí při exportu aplikace zvolit cílovou platformu a vybrat příslušné *VR SDK* (*Cardboard*), v případě *Unreal Engine* je nutné aktivovat doplněk *Google VR* a poté projekt nakonfigurovat pro *Cardboard*.

[19]Cena uváděná na oficiálních stránkách `www.shiva-engine.com` je 200 $ a přepočet na Kč je platný k 9. 12. 2016.

[20]Cena uváděná na oficiálních stránkách `www.shiva-engine.com` je 1 000 $ a přepočet na Kč je platný k 9. 12. 2016.

Import 3D modelů

Unity podporuje souborové formáty řady 3D nástrojů včetně *3ds Max*, *Maya*, *Blender*, *Cinema4D*, *Modo*, *Lightwave and Cheetah3D* a tak dále. Z obecných formátů jsou podporovány *.FBX*, *.OBJ* a *.DAE* (*Collada*) (Unity Manual, 2016).

Unreal Engine podporuje 3D souborové formáty *.FBX* a *.OBJ*. Ostatní formáty je před importem do *Unreal Engine* nutné konvertovat.

Cena

Dle oficiálních webových stránek `www.store.unity.com` je Unity zdarma při ročním obratu společnosti do 100 000 $. Při ročním obratu do 200 000 $ je poplatek za použití vývojového prostředí *Unity* 35 $ za jednoho vývojáře a při obratu vyšším jak 200 000 $ je poplatek za jednoho vývojáře 125 $ (Unity Technologies, 2016). V minulosti byla u verze zdarma funkcionalita omezena (export na platformy, profilování, atd.), nicméně v současné době ve vývojovém prostředí žádná zásadní funkce nechybí.

Unreal Engine 4 je zdarma, pokud obrat produktu za čtvrtletí nepřesáhne 3 000 $. Pokud je obrat z produktu vyšší, činí poplatek za použití vývojového prostředí 5 % z obratu produktu.

Vyhodnocení srovnání Unity 5 vs. Unreal Engine 4

Z hlediska vytváření vlastního chování hry vítězí jednoduchost vizuálního skriptování v *Unreal Engine 4*. *Unity* standardně neumožňuje podobný přístup a vytváření skriptů vyžaduje znalost programování.

Od nové verze *Unity 5.4.1.f1* je příkladná podpora virtuální reality pro mobilní platformu. Standardně stačí zvolit jako cílovou platformu *Android* a při exportu povolit podporu virtuální reality *Cardboard*.

Vizuální schopnosti nejsou z pohledu vývoje hry pro mobilní platformu tolik důležité, jelikož aplikace nemůže využít plného vizuálního potenciálu a mnohem důležitější je celková optimalizace enginu.

Cena nebyla do porovnání zahrnuta, protože oba enginy jsou pro menší projekty k dispozici zdarma. Dále je zcela rozdílný přístup k licencování při větších projektech v případě prodejního úspěchu.

Tabulka 1: Srovnání enginů Unity 5 a Unreal Engine 4 (plný počet bodů je 5 = nejlepší)

	Unity 5	**Unreal Engine 4**
Skriptování	3	5
Vizuální schopnosti	4	5
Podpora Google VR	5	4
Import 3D modelů	5	3

5 Metodika

Aby bylo možné požadovanou aplikaci implementovat, bude nezbytné zvolit cílovou platformu pro implementaci, vytvořit modely jednotlivých objektů ve hře, zvolit vývojové prostředí pro hru a skriptovací jazyk, navrhnout způsob pohybu v aplikaci, navrhnout způsob interakce uživatele se hrou a vytvořit scénář.

Jako cílovou platformu jsem zvolil mobilní zařízení s operačním systémem *Android*. Jedná se o dostupné řešení, u kterého je možné při pohybu ve virtuálním prostoru těžit z absence jakýchkoli kabelů. Prostředí pro vývojáře je díky společnosti *Google* velmi dobře připraveno a celá platforma je podrobně popsána v rozsáhlé dokumentaci dostupné online.

Z množství programů sloužících k modelování 3D objektů jsem vybral *3ds Max*, který se řadí mezi špičku v mnou preferovaném polygonálním modelování. S programem mám bohaté zkušenosti a k tvorbě modelů *3ds Max* využívám od roku 2006.

Na základě analýzy nástrojů pro vývoj her provedené v části 4.4 jsem jako herní engine a vývojové prostředí vybral program *Unity*. *Unity* je nástroj dostupný pro účely této práce zdarma, disponuje příjemným uživatelským rozhraním, exceluje kompatibilitou s modelovacími nástroji a v neposlední řadě mám s tímto vývojovým prostředím bohaté zkušenosti z dřívějších projektů.

Skripty pro interakci budou implementovány v programovacím jazyce *C#*. Jazyk *C#* je využíván pro všechny inicializační skripty v aplikaci a využívá ho rovněž *Google VR SDK*. Ve verzi *Unity 5* byla většina skriptů (například pro obrazové efekty atd.) přepsána z jazyka *JavaScript* do jazyku *C#* (Unity Manual, 2016). Programovací jazyk *C#* je navíc preferovaný mezi uživateli a většina témat na diskuzních fórech je řešena právě pro tento jazyk (Unity Technologies, 2016).

Způsoby ovládání pohybu analyzované v části 2.5 byly pro požadovanou aplikaci vyhodnoceny jako nevyhovující a pro vlastní aplikaci bude nutné navrhnout vlastní řešení, které bude splňovat požadavek zadání na přirozenost ovládání.

Pro interakci uživatele se hrou bude použit způsob ovládání pohledem s odpočtem času, který dobře funguje například ve hře *Wizard Academy VR Cardboard*. Toto řešení nevyžaduje od uživatele manuální interakci a splňuje tak zadání.

Pro návrh scénáře je nejprve nutné určit žánr hry. Pro požadovanou aplikaci jsem jako vhodný žánr vyhodnotil logickou adventuru a pro motivaci hráče bude nutné v další fázi vytvořit odpovídající scénář.

6 3D modelování prostředí

Hru vyvíjenou v rámci této diplomové práce jsem se rozhodl pojmout jako logickou adventuru z pohledu první osoby. Od tohoto rozhodnutí se následně odvíjí i návrh herního prostředí a tvorba modelů. Následující kapitola představí program 3ds Max, jeho nástroje a na demonstračním objektu bude předvedeno použití těchto nástrojů.

6.1 3ds Max

3ds Max je jedním z nejpopulárnějších 3D nástrojů a je používán v řadě oborů. Jedná se o univerzální nástroj pro modelování objektů, vytváření animací a osvětlení, texturování objektů a následný rendering včetně post-process efektů.

Program je vyvíjen společností *Autodesk* a představuje špičku na poli nástrojů pro polygonální modelování. Mezi další způsoby vytváření objektů v programu *3ds Max* patří modelování pomocí primitivních těles ve spojení s *boolean* operacemi a modelování pomocí *NURBS* křivek. Mimo samotné modelování je vzhled objektů možné upravovat množstvím modifikátorů (Autodesk, 2016).

6.2 3D modely objektů

Pro vytvoření opravdového dojmu z prostředí musí být vytvořeno množství modelů. Tyto modely musí dosahovat přiměřené složitosti vzhledem k cílové platformě aplikace. V této části budou na objektu kytary vysvětleny techniky použité při vytváření všech modelů v aplikaci. Většina modelů ve hře je vytvořená na základě reálných předmětů a jsou vymodelovány dle skutečných rozměrů odměřených přímo na objektech.

Objekty ve hře jsou většinou vytvořeny ze standardních primitiv[21] a modelování probíhá zpravidla převedením objektu na editovatelný (v *3ds Max* označováno *Editable Poly*[22]). Při tvorbě objektů ve hře byly využívány následující nástroje:

- **Cut** přidává hrany do objektu.
- **Cap** uzavírá označenou hranu.
- **Extrude** vytláčí označené polygony o zadanou vzdálenost.
- **Bridge** vytváří propojení mezi párem polygonů/hran.
- **Inset** vloží polygon do označeného polygonu.
- **Bevel** kombinuje *Extrude* a zmenšení koncového polygonu objektu.
- **Select and Uniform Scale** umožňuje měnit velikost jak celých objektů, tak pracovat s jednotlivými polygony.

[21] Například kvádr, koule, válec a jiné.
[22] Umožňuje editovat objekt pomocí bodů, hran nebo polygonů.

- **Select and Rotate** slouží k rotaci celých objektů, ale i samotných polygonů.
- **Select and Move** standardně pohybuje objektem, při editaci objektů je možné pohybovat jedním, nebo skupinou bodů.

Vzhled a vlastnosti modelovaných objektů je dále možno ovlivnit pomocí modifikátorů, které se přidávají skrze panel *Modify*. V této práci byly nejčastěji používány:

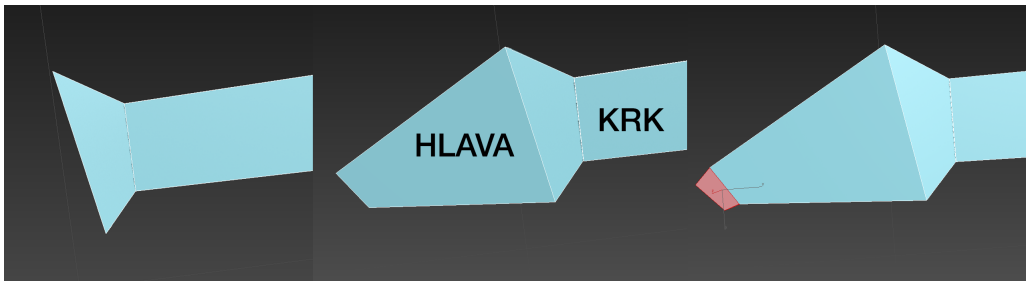
- **UVW Map** slouží pro mapování textury na objekt.
- **MeshSmooth** se používá k zaoblení hran objektu, přičemž míra zaoblení je odvislá od počtu iterací modifikátoru.
- **Bevel** oproti nástroji dostupném v *Editable Poly* slouží pro vytvoření 3D objektu z tvaru a často je využíván pro tyto účely při modelování textu.

Model elektrické kytary

Kytara je jeden z důležitých herních objektů aplikace a její použití je nutné pro další postup hrou. Z tohoto důvodu byl objekt vytvořen složitější s větším množstvím polygonů i přes vyšší náročnost na vykreslování takového objektu. Zároveň je na tomto objektu možné ukázat většinu technik, které se následně používají při modelování dalších objektů.

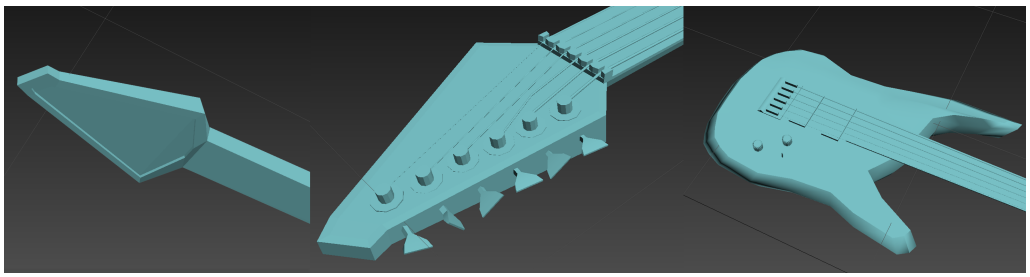
Jako základní primitivum pro krk kytary byl zvolen kvádr (*Box*) a všechny délky mají počet segmentů roven jedné, neboť všechny tvary a detaily budou objektu přidávány v dalších krocích. Aby bylo možné objekt upravovat na úrovni polygonů, je nutné jeho převedení na *Editable Poly*.

Samotné modelování je realizováno pomocí nástrojů, které jsou zmíněny v kapitole 6.2. Kombinováním nástroje *Extrude* s nástroji *Select and Uniform Scale* a *Select and Rotate* je možné vytvářet složitě vyhlížející tvary velmi rychle a intuitivně viz obr. 13.



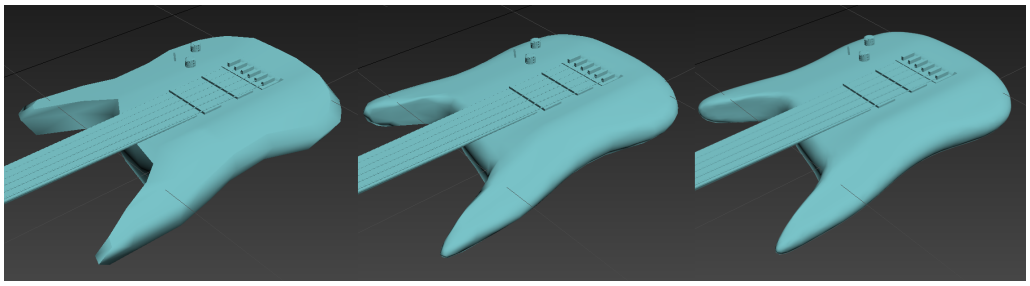
Obrázek 13: Postupné modelování hlavy kytary kombinací nástrojů *Extrude*, *Select and Uniform Scale* a *Select and Rotate*

Pro doplnění detailů do objektu je možné využít *Inset*, který do plochy vloží plochu stejného tvaru. Opět pomocí nástroje *Extrude* je možné tuto plochu vytáhnout z původní kladným i záporným směrem. Některé detaily je vhodnější vytvářet jako samostatné objekty viz obrázek 14.



Obrázek 14: Přidávání detailů pomocí nástrojů dostupných v *Editable Poly* a dalších objektů

Pro objekty, které se mají jevit jako zaoblené, je možné využít modifikátor *MeshSmooth*. Efekt tohoto modifikátoru je možné vidět na obr. 15.



Obrázek 15: Tělo kytary na levém obrázku bez modifikátoru *MeshSmooth*, uprostřed s použitím modifikátoru *MeshSmooth* a *Iterations* nastaveným na 1, obrázek napravo s *Iterations* nastaveným na 2

6.3 Texturování

Samotný tvar není pro objekt dostačující a pro realistický vzhled je nutné k objektu přiřadit materiál. Základem materiálu se u většiny objektů stává 2D textura, která je umístěna na 3D model. Při tvorbě modelů byly zpravidla použity bezešvé textury (označované seamless), které byly staženy ze specializovaných webových stránek[23].

Vzhledem k cílové platformě byly všechny textury zmenšeny na přijatelný kompromis mezi rozlišením a výslednou velikostí. Některé specifické textury byly vytvořeny v programu *Adobe Photoshop* (snímače kytary a jiné).

Vytvoření materiálu pro objekt probíhá skrze *Material Editor* umístěný v hlavní nabídce programu. Po vybrání volného slotu pro materiál je možno nastavit základní barvu. Tlačítkem umístěným vedle materiálu *Diffuse* lze otevřít *Material Browser* a vybráním *Bitmap* se otevře průzkumník, ve kterém je možno přidat materiálu texturu. V režimu *Editable Poly* lze přidělovat textury jednotlivým polygonům a kombinovat tak více materiálů na jednom objektu.

[23]Webové stránky `www.textures.com`, ze kterých je možno stáhnout za každých 24 hodin 15 textur v rozlišení 1024 × 1024 pixelů.

Po nanesení textur je nutné objektu přidat modifikátor *UVW Map*, který zajistí správné namapování textury. V nastavení modifikátoru lze zvolit mapovací primitivum, které ale nutně nemusí být ve shodě s modelovaným objektem. V modifikátoru se dále nastavují rozměry, na které se má textura mapovat i počet jejího opakování.

Tímto práce s texturami v programu *3ds Max* končí, jelikož testováním jsem došel k závěru, že nejlepších výsledků při dalších úpravách textur je možné dosáhnout přímo v *Unity*.

6.4 Export modelů a import do Unity

3ds Max dokáže modely exportovat do řady formátů, z jejichž zástupců lze jmenovat například *Filmbox* (.FBX), *3D Studio* (.3DS), *COLLADA* (.DAE) nebo *Shockwave 3D* (.W3D).

Z vlastní zkušenosti se mi jako nejméně problémový formát pro spolupráci s *Unity* osvědčil prvně jmenovaný *Filmbox*, který jsem následně používal pro export všech modelů použitých v aplikaci.

Z programu 3ds Max je možné exportovat buď jednotlivé modely, nebo celé scény. Samotný export se provádí pomocí hlavního menu (logo *3ds Max*) a výběrem možnosti *Export*. Z rolovacího menu *Save as type* je nutné vybrat *Autodesk (.FBX)* a stisknutím *Save* se otevře nabídka pro nastavení exportu. Položku *Current Preset* je vhodné nastavit na *Autodesk Media and Entertainment* a provést změny u několika položek na kartě *Include*:

- Zrušit položku *Lights*, protože tvorba světel bude probíhat v programu *Unity*.
- Zrušit položku *Animation*, jelikož animace budou vytvářeny skrze skripty v *Unity*.
- Povolit položku *Embed Media* pro zahrnutí textur k exportovanému modelu.

Na kartě *Advanced Options* doporučuji pro bezproblémovou spolupráci s poslední verzí *Unity 5* zvolit verzi *FBX 2014*, nebo novější.

Po potvrzení nastavení je proveden export, přičemž v případě chyby v některém z exportovaných objektů je zobrazeno chybové hlášení, popřípadě upozornění. Chyby se zpravidla týkají neuzavřených objektů nebo objektů se špatně použitými modifikátory.

7 Interakce s prostředím a export aplikace

V této kapitole bude popsáno herní vývojové prostředí *Unity*, které bude využito k vytvoření výsledné aplikace. Dále bude představena funkcionalita *Google VR SDK* i způsob, jakým bude v aplikaci řešen pohyb a interakce hráče. Následně bude popsána další gamifikace prostředí, přizpůsobení aplikace pro běh na mobilních zařízení a export aplikace s uveřejněním na *Google Play*.

7.1 Unity

Unity je vývojové prostředí pro vytváření 3D i 2D her a podporuje celou řadu platforem od běžných operačních systémů pro osobní počítače (*Windows*, *OS X*, *Linux*), přes mobilní operační systémy (*Android*, *iOS*, *Windows Phone*) až po konzolové operační systémy (*PlayStation*, *Xbox One*, *Wii U*, *Nintendo 3DS*, *PlayStation Vita* nebo *Steam OS*). Z hlediska virtuální reality *Unity* podporuje *Oculus Rift*, *PlayStation VR*, *Google Cardboard*, *Google Daydream*, *Gear VR*, *Steam VR* nebo *Microsoft Hololens* (Unity Technologies, 2016). *Unity* je nepřetržitě vyvíjeno a podporované platformy jsou s každou další verzí rozšiřovány.

Funkcionalita vývojového prostředí i samotného enginu je neustále vylepšována. Například aktuálně nejnovější verze *Unity 5.5* přidává široké možnosti práce s úvodní obrazovkou, rozšiřuje práci s integrovaným nástrojem pro vytváření animací, přidává nové moduly pro částicový systém a obsahuje řadu optimalizačních vylepšení. Všechny změny jsou vždy podrobně popsány na oficiálních stránkách[24] včetně změn, které nejsou zpětně kompatibilní.

Unity se vyznačuje vynikající kompatibilitou s 3D modelovacími nástroji a podporuje velkou řadu různých formátů. Mimo to jsou podporovány i různé obrazové formáty, audio, video a další.

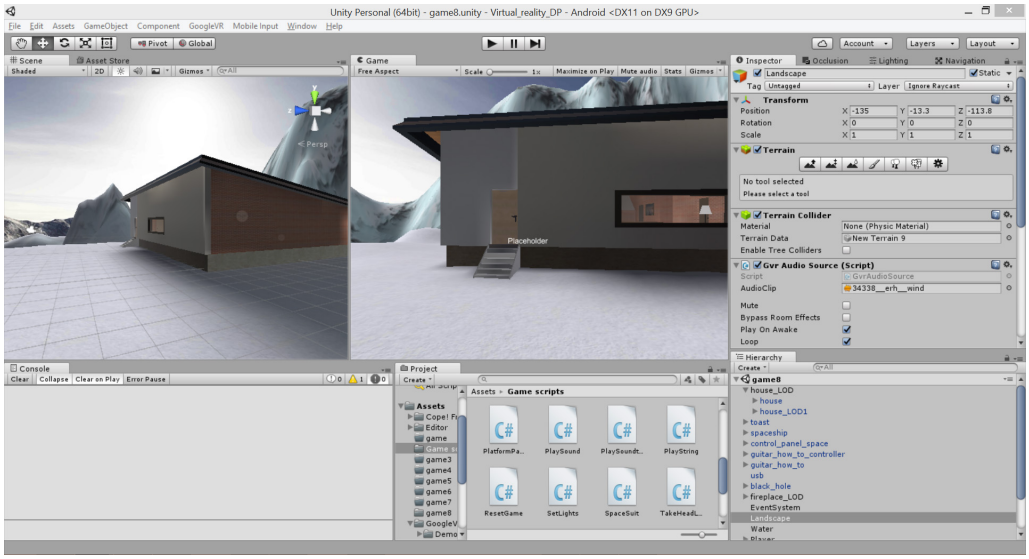
Více jak 15 000 volně dostupných modelů, textur, skriptů, zvukových efektů a mnoho dalšího lze stáhnout přímo ve vývojovém prostředí skrze integrované rozhraní *Asset Store* (Unity Technologies, 2016).

Rozhraní programu je plně konfigurovatelné a zobrazení je možné přizpůsobit dle osobních preferencí. Uživatelské rozhraní se skládá z oken (viz obr. 16), které lze přidávat i odebírat, popřípadě je možné je v případě potřeby slučovat. Základní menu pro práci s projektem se nachází v horní části, nicméně funkcionalitu tohoto menu lze až na nabídky *File*, *Edit* a *Assets* nahradit funkcemi v jednotlivých oknech.

Inspector

Karta *Inspector* slouží pro kompletní nastavení označeného objektu. V této kartě je možné nastavit pozici i rotaci objektu, přidat objektu komponenty a ovlivnit jejich vazby, nastavit objektu statický identifikátor a řadu dalších atributů.

[24] `https://unity3d.com/unity/whats-new`



Obrázek 16: Personalizované rozhraní programu *Unity* s množstvím oken

V tomto okně se zároveň otevírají nastavení projektu a editoru (například nabídky v *Edit* → *Project Settings*).

Prefab

Komponenty jsou standardně jednotlivě přidávány do scény a nastavení každé z nich je nezávislé na ostatních. V některých případech tato nezávislost může vést k problémům, protože objekty se vykytují na mnoha místech, ale jejich chování nelze hromadně ovlivnit. Pro tyto případy je vhodné využít *Prefab*, které představují výchozí objekt a umístěním do scény vznikají jeho instance.

Použití *Prefabů* navíc snižuje množství renderovacích dávek, čímž se snižuje výpočetní složitost scény. Pro co nejvyšší výkon aplikace je tedy nutné v maximální možné míře využívat *Prefaby*, protože výrazně zjednodušují správu velkých projektů a zvyšují množství vykreslovaných snímků za sekundu.

Collider

Collider je vestavěná komponenta *Unity* využívaná fyzikálním enginem pro detekci kolizí. Výhodou je, že tvar *Collideru* může být často výrazně jednodušší než zobrazovaný model, jelikož není viditelný.

Pokud je v nastavení importovaného objektu na kartě *Inspector* povolena položka *Generate Colliders*, generuje *Unity* pro objekt fyzikální model stejného tvaru (*Mesh Collider*). Tato reprezentace je ale často zbytečně komplikovaná a je vhodné tento *Mesh Collider* nahradit *Collidery* primitivními – *Box*, *Sphere* a *Capsule*. V tomto případě se nepovoluje položka *Generate Colliders* v nastavení importova-

ného objektu, ale konkrétnímu objektu ve scéně se pomocí karty *Inspector* manuálně přidělí jedna z výše zmíněných fyzikálních reprezentací.

Pro většinu objektů ve vlastní aplikaci je využíváno primitivních *Colliderů*, neboť výpočet jejich kolizí je méně náročný na výkon CPU. Pro optimalizaci výpočtu fyzikálního modelu je rovněž nutné všem pohyblivým objektům s *Colliderem* přidat komponentu *Rigidbody*. Reakce objektu na gravitaci se povoluje pomocí tlačítka *Use Gravity*. V případě, že objektem má být možné pohybovat pouze pomocí skriptu, je nutné zakázat položku *Use Gravity* a povolit položku *Is Kinematic*.

7.2 Příprava projektu a import 3D modelů

Unity rozlišuje projekty a scény. Obecně platí, že v rámci projektu jsou všechny zdrojové soubory sdíleny a scéna může využívat pouze některé z nich. V praxi se například úprava skriptu v projektu projeví ve všech scénách, které se v projektu nachází.

Konfigurace aplikace pro operační systém *Android* se provádí skrze hlavní nabídku *File* → *Build Settings*. Zde je nutné vybrat položku *Android* a volbu potvrdit stisknutím *Switch Platform*.

7.3 Google VR SDK

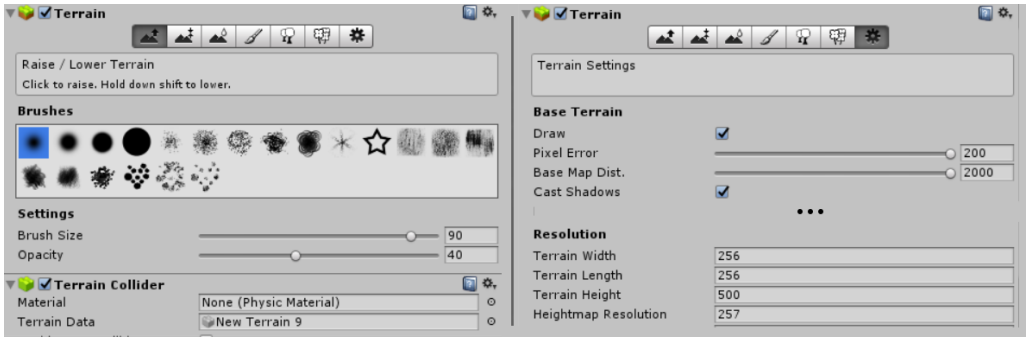
Google VR SDK pro *Unity* umožňuje jednoduše adaptovat existující aplikaci vyvíjenou v rámci vývojového prostředí *Unity*, nebo vytvořit od základu zcela novou aplikaci (Google VR Dev, 2016). *SDK* se stará o sledování pohybu hlavy, vykreslování a konfiguraci stereo obrazů, detekci použití magnetu a prostorové audio efekty. Dále *SDK* obsahuje základní *Prefab* pro interakci pohledem, ukázkové demo a VR emulaci pro *Unity Editor*.

Od září 2016 je podpora *Google VR* přímo integrována v rámci *Unity GVR* verze. V nastavení exportu aplikace pro cílovou platformu nyní nově stačí povolit nabídku *Virtual Reality Supported* a dostupných *Virtual Reality SDKs* vybrat *Cardboard*.

7.4 Tvorba terénu

Součástí *Unity* je pokročilý nástroj pro vytváření terénů. K modelování i texturování terénu jsou využívány štětce, jejichž chování se mění na kartě *Inspector*. Štětce je možné využít pro snížení nebo zvýšení terénu, pro nastavení konkrétní výšky terénu, pro vyhlazení plochy, texturování, přidávání stromů a detailů (rostliny, keře apod.). Hlavní síla tohoto nástroje spočívá v možnosti nastavit hustotu štětce, čímž je umožněno například prolínání několika textur.

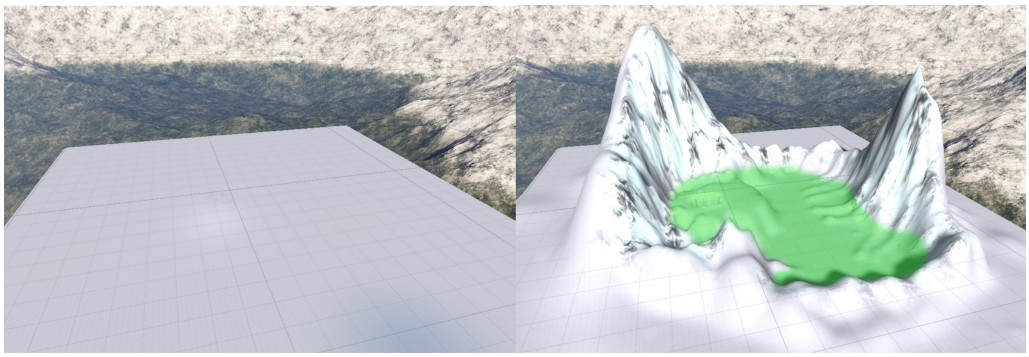
Přidání nového terénu do scény probíhá pomocí nabídky *GameObject* → *3D Object* → *Terrain*. Vytváření krajiny probíhá kombinací různých tvarů a druhů štětců,



Obrázek 17: Nastavení chování, tvaru, tloušťky a hustoty štětce v levé části obrázku, nastavení vlastností terénu v pravé části obrázku

zvyšováním a snižováním jednotlivých částí terénu a prolínáním textur. Do krajiny je možné pomocí štětců rovněž přidávat stromy i libovolné detaily odvozené z *Prefabu*.

Vzhledem k nižšímu výkonu cílové platformy je vhodné výchozí hodnoty v nastavení terénu změnit. Šířka i délka terénu byla snížena na 256 metrů a hodnota *Heightmap Resolution* byla v aplikaci nastavena na *257* pixelů.

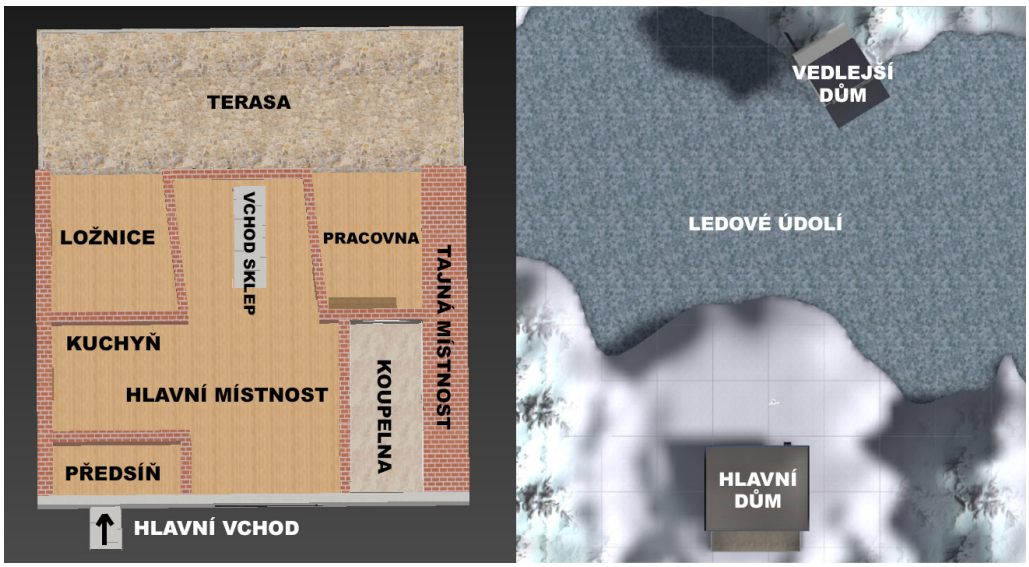


Obrázek 18: Výchozí terén se základní texturou v levé části obrázku a výsledný terén doplněný dalšími texturami v pravé části obrázku (zelené vybarvení znázorňuje oblast dostupnou pro volný pohyb hráče)

Pomocí modelování terénu je ve hře vytvořena přirozená hranice pro omezení volnosti pohybu. Pohyb je omezen na prostor údolí a hranici tvoří strmý svah, takže se ve hře nenachází žádné neviditelné stěny.

7.5 Scénář a obsah hry

Hra obsahuje scénář a obsahově se nejvíce přibližuje k logické adventuře z pohledu první osoby. Ihned po spuštění hry se hráč objeví před opuštěným domem, ve kterém mu je skrze hlasového průvodce sděleno, že má za úkol vyřešit záhadu o ztraceném vědci. V domě se nachází interaktivní předměty, jejichž použití hráče posouvá v rámci hry dále.



Obrázek 19: Základní rozvržení místností v domě (levý obrázek) a rozvržení herní mapy (pravý obrázek)

Příběh je hráči vyprávěn skrze otevřené knihy, ze kterých je při interakci pohledem přehrána krátká hlasová zpráva vytvořená převodníkem textu na řeč (*TTS*) zdarma[25].

Průchod hrou je částečně nelineární a hráč začíná úkoly plnit dle toho, které části domu začne nejdříve prozkoumávat. Úkoly vedoucí ke splnění scénáře jsou:

- V předsíni se nachází skříň, ve které je nádoba na palivo.
- V hlavní místnosti je stůl, v jehož šuplíku je umístěna svítilna. Ta umožňuje hráči rozsvítit světlo pohledem pod sebe. Hru lze ale dokončit i bez akvizice tohoto předmětu.
- Přístup do sklepních prostor se nachází pod kobercem v hlavní místnosti, který se odsouvá pohledem. Ve sklepě hráč musí načerpat palivo do nádoby.
- Ve sklepě je výtah, který vede do tajné místnosti pod domem. Zde se nachází tabule, která obsahuje sekvenci tónu a obrazový návod s kytarou.
- Pro otevření tajné místnosti v pracovně je nutné přehrát na kytaru správnou sekvenci tónů.
- V tajné místnosti se nachází speciální oblek a terminál pro vysunutí plošiny k vedlejšímu domu.
- V kuchyni se nachází USB klíč.
- Televize v pracovně při opakované interakci zobrazuje barevnou sekvenci.

[25] http://www.fromtexttospeech.com/

- Počítač v ložnici zobrazuje číselnou kombinaci v případě, že hráč vzal USB klíč z kuchyně. V opačném případě je hráč vyzván k jeho hledání.
- Přístup do vedlejšího domu je chráněn zámkem, k jehož odemčení je nutné zadat správnou číselnou kombinaci.
- Ve vedlejším domě se nachází světelný zámek, k jehož otevření musí hráč zadat správnou barevnou sekvenci.
- Splnění všech úkolů a nastoupení do vesmírného modulu hráči sdělí konec příběhu.

7.6 Ovládání pohybu

V části 2.5 bylo zhodnoceno několik možných přístupů ke způsobu pohybu ve hře. Vzhledem k náplni vlastní hry byly způsoby ovládání chůze u analyzovaných her vyhodnoceny jako nevyhovující. Pro ovládání chůze jsem definoval několik parametrů:

- Chůze nebude využívat žádná tlačítka ani herní objekty.
- Spuštění chůze by měl dokázat i člověk, který neví, jak se hra ovládá.
- Aktivace i deaktivace chůze musí být možná v krátkém sledu.

Aby bylo vyhověno těmto parametrům, byl pohyb v aplikaci vyřešen pomocí pohledu, přičemž chůze dopředu je ovládána jeho sklopením dolů o 30° a více (viz obrázek 20). Jedná se o relativně přirozený způsob ovládání chůze, neboť člověk se při chůzi v reálném prostředí mírně nakloní dopředu a pohled směřuje dolů na případné překážky v cestě.

Obrázek 20: Ovládání pohybu – v levé části obrázku se hráč v aplikaci nepohybuje, v pravé části má hráč pohled sklopen dolů o více jak 30° a v aplikaci se pohybuje dopředu

Hernímu objektu hráče byl pro tyto účely nadřazen další herní objekt, který obsahuje komponenty *Collider*, *Rigidbody* a samotný skript pro ovládání chůze `Autowalk.cs`. Tento herní objekt je provázán s kamerou, díky které je možné vyhodnocovat úhel pohledu hráče. Provázání je řešeno na kartě *Inspector*, kde je do proměnné `public Camera Head` přiřazen objekt *Main Camera*. Aby byl kód pro

ovládání chůze vyhodnocován pro každý renderovaný snímek, nachází se vše uvnitř metody *Update* (Zdrojový kód 1).

Zdrojový kód 1: Kontrola úhlu pohledu pro spuštění i zastavení chůze

```
1 if (!m_isWalking && !s_isInteracting && Head.transform.eulerAngles.
      x >= m_walkingAngle && Head.transform.eulerAngles.x <=
      m_maximumAngle) {
2             m_isWalking = true;
3             m_footstep.Play();
4 } else if (m_isWalking && (Head.transform.eulerAngles.x <=
      m_walkingAngle || Head.transform.eulerAngles.x >=
      m_maximumAngle || s_isInteracting)) {
5             m_isWalking = false;
6             m_footstep.Stop();
7 }
```

V další části metody *Update* je k pohybu postavy využívána translace objektu, který je posouván dle zvolené rychlosti `m_speed` (zdrojový kód 2).

Zdrojový kód 2: Pohyb pomocí translace, který je aplikován na objekt s přiřazeným skriptem

```
1 if (m_isWalking) {
2     Vector3 Direction = new Vector3(Head.transform.forward.x, 0,
          Head.transform.forward.z).normalized * m_speed * Time.
          deltaTime;
3     transform.Translate(-Direction);
4 }
```

Trojrozměrný vektor `Direction` (řádek 2 zdrojový kód 2) se skládá z normalizovaného směrového vektoru, který je násoben rychlostí `m_speed` a proměnnou `Time.deltaTime`[26], jež zajišťuje, že velikost posunu je nezávislá na snímkové frekvenci vykreslování. Vzhledem k reprezentaci v prostoru musí být translace provedena se zápornou hodnotou `Direction`.

7.7 Interakce s předměty

V sekci 3.2 bylo zmíněno, že pro interakci s předměty ve hře se často používá magnetický senzor. Pro maximální přirozenost interakce jsem se ale rozhodl pro ovládání hry využít pouze pohled hráče. Základní funkcionalita je zajištěna pomocí *Unity* komponent. V aplikaci je nutné vytvořit reakce objektů na interakci a signalizaci průběhu pohledu uživatele.

V osvědčených postupech *Google* se nachází doporučení o nutnosti informovat uživatele o průběhu interakce (část práce 4.1). To je v aplikaci realizováno pomocí vlastního kruhového ukazatele, který nahrazuje standardní zvětšující se kruh v *Google VR SDK*. Herní postava má přiřazen prvek *Canvas* s obrázkem kruhu (obrázek

[26] Čas potřebný k vykreslení posledního snímku v sekundách.

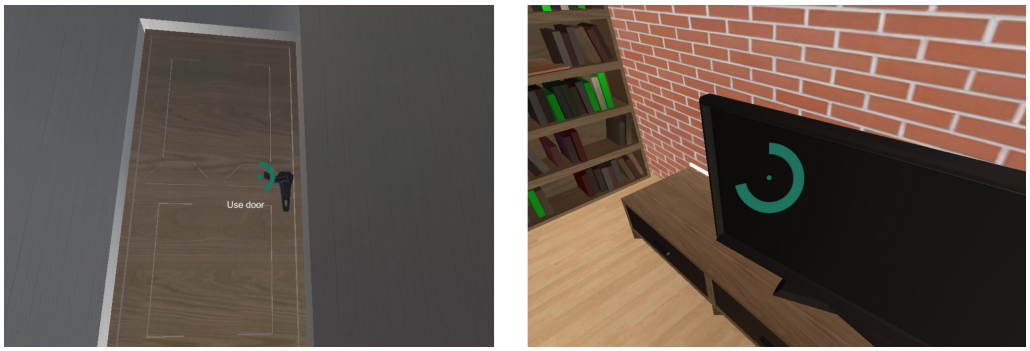
musí být importován jako *Sprite*). Na kartě *Inspector* je u obrázku nastaven *Image Type* → *Filled*, u *Fill Method* je zvolena metoda *Radial 360* (obrázek se bude odkrývat kruhově) a *Fill Origin* je nastaven na *Top*. O samotné odkrývání se stará skript `ProgressBar.cs` viz zdrojový kód 3.

Zdrojový kód 3: Obsah metody metody *Update* skriptu `ProgressBar.cs`

```
1 if (Autowalk.s_isInteracting) {
2     CirclePosition.localPosition = new Vector3 (0, 0, s_distance);
3     if (m_currentTime <= m_fillTime)
4         m_currentTime += Time.deltaTime;
5     ProgressBarImage.fillAmount = m_currentTime/m_fillTime;
6 } else {
7     ProgressBarImage.fillAmount = 0;
8     m_currentTime = 0;
9 }
```

Pokud je prováděna interakce s předmětem (v aplikaci je využíváno sdílené proměnné `s_isInteracting`) je do proměnné `m_currentTime` ukládán čas od počátku interakce. Proměnná *fillAmount* určuje úhel odkrytí obrázku, přičemž hodnota `1` znamená odkrytí 360°. Dělením času od počátku interakce celkovým časem interakce je zajištěn postupný růst hodnoty od `0` do `1`.

Řádek 2 ve zdrojovém kódu 3 se stará o posunutí ukazatele postupu do stejné vzdálenosti od uživatele jako má předmět interakce (viz obr. 21). Umístění ukazatele do konstantní vzdálenosti blízko uživatele totiž vede k neustálému přeostřování očí z ukazatele interakce do prostředí a zpět. O nastavení vzdálenosti se stará metoda `public static void SetPosition(float distance)` a její volání je doplněno do *Google VR* skriptu `GvrReticle.cs`.



Obrázek 21: Zobrazení ukazatele postupu interakce v různých velikostech dle vzdálenosti od hráče

Všechny předměty ve hře, které mají definovanou interakci, jsou aktivovány pohledem setrvávajícím na objektu po dobu 1,5 sekundy. K tomu je využíván *Physics Raycaster*, který komunikuje s *Unity* vestavěným modulem *Event System* provázaným s *Google VR* skriptem `GazeInputModule.cs`. Každý interaktivní objekt má v sobě komponentu *Event Trigger*, která komunikuje s *Colliderem* objektu a v přípa-

dě zasažení objektu paprskem generovaným pomocí komponenty *Physics Raycaster* spouští libovolnou metodu objektu, které je předávána hodnota v podobě parametru.

Ovládání pohledu pracuje s událostmi `Pointer Enter` a `Pointer Exit`. V prvním případě je metodě `SetGazedAt` v parametru předávána hodnota `True`, v druhém případě hodnota `False`. Obsah metody `SetGazedAt` v případě skriptu pro otevírání dveří se nachází ve zdrojovém kódu 4.

Zdrojový kód 4: Část skriptu pro otevírání dveří, která přebírá parametr, zda byl objekt ve hře zaměřen pohledem a předává tento parametr dále

```
1 public void SetGazedAt(bool gazedAt) {
2     m_lookedAt = gazedAt;
3     Autowalk.s_isInteracting = gazedAt;
4     Text.text = ("Use door");
5     Text.enabled = gazedAt;
6 }
```

Metoda se stará o nastavování proměnné `m_lookedAt` dle toho, jestli je objekt hráčem sledován. Stejně tak je nastavována i proměnná `s_isInteracting`, jejíž funkce spočívá v zastavení chůze při hodnotě `True`. Tato metoda je u ostatních objektů různě modifikována a jsou k ní doplňovány další hodnoty dle potřeby.

Další část kódu pro interakci s objektem je umístěna v metodě `Update`, která je volána při každém překreslení snímku. Vzhledem k 1,5 s prodlevě pro použití objektu je nutné v tomto bloku měřit počátek pohledu (zdrojový kód 5, řádek 2).

Zdrojový kód 5: Podmínka délky pohledu na objekt, která zafixuje hodnotu `m_delay` v případě pohledu na objekt

```
1 if (!m_lookedAt && m_playerClose) {
2     m_delay = Time.time + 1.5f;
3 }
```

Pokud se hráč nedívá na objekt a zároveň je v dosahu objektu, nabývá proměnná `m_delay` hodnoty součtu aktuálního času spuštění aplikace a časové prodlevy 1,5 sekundy. Zda uplynula doba nutná k použití objektu je následně měřeno v další části kódu, která zároveň mění i samotné chování objektu při splnění podmínky (zdrojový kód 6, řádek 3 a 4).

Zdrojový kód 6: Testování podmínky, zda byl objekt hráčem sledován po dostatečnou dobu

```
1 if (m_lookedAt && Time.time>m_delay && m_playerClose) {
2     m_lookedAt = false;
3     m_opened = !m_opened;
4     m_playSound = true;
5 }
```

Podmínka je kladně vyhodnocena v případě, že hráč sleduje objekt, aktuální čas spuštění aplikace je větší než `m_delay` (součet počátečního času pohledu a prodle-

vy) a hráč je v dostatečné blízkosti hernímu objektu. Protože podmínka je při běhu aplikace neustále vyhodnocována, proměnná `m_lookedAt` je nejdříve nastavena na `False`, aby při dalším překreslení snímku nedošlo k opakovanému vyhodnocení. Následně může být buď přímo spuštěna požadovaná funkcionalita objektu, nebo mohou být nastaveny proměnné pro další podmínky pro případ, že změna objektu probíhá v čase (například v tomto případě otevření dveří).

7.8 Gamifikace prostředí

Pro ozvláštnění zážitku z hraní obsahuje aplikace velké množství interaktivních prvků, které v některých případech mají vliv na hráčův postup hrou a v některých případech vliv nemají. Pro gamifikaci prostředí byly vytvořeny obecné skripty `ChangePosition.cs` a `PlaySound.cs` a řada specializovaných skriptů. První z těchto dvou skriptů změní pozici objektu při použití hráčem dle hodnot zadaných na kartě *Inspector*. Při opětovném spuštění skriptu je objekt vrácen na výchozí pozici, přičemž při obou změnách je přehrán zvuk přiřazený k hernímu objektu.

Při inicializaci je nejdříve uložena výchozí pozice do proměnné `m_defaultPos` typu `Vector3` a nová pozice do proměnné `m_newPosition` stejného typu (viz zdrojový kód 7).

Zdrojový kód 7: Inicializace skriptu s nastavením stávající i nové pozice doplněné o posun

```
1 void Start() {
2     SetGazedAt(false);
3     m_defaultPos = new Vector3 (transform.position.x, transform.
          position.y, transform.position.z);
4     m_newPosition = new Vector3 (transform.position.x + m_changeX,
          transform.position.y + m_changeY, transform.position.z +
          m_changeZ);
5 }
```

V bloku *Update* je před vykreslením každého snímku spouštěn kód, který se dotazuje, zda se hráč na objekt podíval po dostatečně dlouhou dobu (řádek 1 zdrojový kód 8) a zda je hráč dostatečně blízko (`m_playerClose`). V případě kladného vyhodnocení všech podmínek je provedena změna stavu z původního stavu do nového, nebo naopak (řádek 3 zdrojový kód 8).

Další podmínka na první pohled zbytečně opakuje `if (m_playerClose)`, nicméně ta zajišťuje spouštění další části kódu skriptu pouze v případě, že je hráč v dostatečné blízkosti hernímu objektu. Příkaz `transform.eulerAngles = Vector3.Lerp` nastavuje rotaci dle zadaného trojrozměrného vektoru s parametry stávající rotace (`transform.eulerAngles`), cílové rotace (`m_newPosition`) a poměru mezi těmito dvěma vektory (`Time.deltatime * m_smooth`). Tato část kódu by se bez podmínky `if (m_playerClose)` spouštěla při každém překreslení snímku pro každý objekt a zbytečně by tak zvyšovala režii při vyhodnocování skriptů.

Zdrojový kód 8: Vyhodnocení pohledu a nastavení pozice dveří

```
if (m_lookedAt && Time.time>m_delay && m_playerClose) {
    m_lookedAt = false;
    m_change = !m_change;
    m_playSound = true;
}

if (m_playerClose) {
    if (m_change) {
        transform.eulerAngles = Vector3.Lerp (transform.eulerAngles
            , m_newPosition, Time.deltaTime * m_smooth);
        if (m_playSound == true) {
            m_openingSound.Play ();
            m_playSound = false;
        }
    } else {
        transform.eulerAngles = Vector3.Lerp (transform.eulerAngles
            , m_defaultPos, Time.deltaTime * m_smooth);
        if (m_playSound == true) {
            m_closingSound.Play ();
            m_playSound = false;
        }
    }
}
```

Skript `PlaySound.cs` je odlehčenou verzí `ChangePosition.cs` a výkonná část skriptu se skládá pouze z řádku, který přehrává zvuk přiřazený ke skriptu (`m_sound.Play()`). Většina zvukových podkladů pro interakci byla stažena ze serveru `www.freesound.org`.

Zdrojový kód 9: Přehrání přiřazeného zvuku

```
if (m_lookedAt && Time.time>m_delay) {
    m_lookedAt = false;
    m_sound.Play();
}
```

Jedním z dalších prvků interakce s prostředím je reakce na pád herní postavy z výšky. V určité části scénáře se hráč pohybuje po úzké lávce a případný pád dolů způsobí konec hry. Herní postava má pro pohyb přiděleno *Rigidbody*, které má pro čtení přístupnou proměnnou zrychlení ve všech osách. Detekce, zda došlo k pádu, je uvedena ve zdrojovém kódu 10.

Zdrojový kód 10: Detekce pádu hráče a jeho případné přesunutí do místnosti s nabídkou restartu

```
if (m_body.velocity.y < -10) {
    m_died = true;
    m_isWalking = false;
```

```
4  }
5
6  if (m_died) {
7      if (m_body.velocity.y > -2) {
8          Dying.Play();
9          m_died = false;
10         transform.position = new Vector3 (-122f,-5.37f, -13.26f);
11         m_audioListener = GetComponentInChildren<GvrAudioListener
               >();
12         m_audioListener.enabled = false;
13     }
14 }
```

Jakmile je detekována rychlost v ose y alespoň 10 $m \times s^{-2}$, `m_died` je změněno na `True`. Dopad postavy na zem je detekován až v další části skriptu, aby nedocházelo k ukončení hry uprostřed pádu. Hráč je po dopadu přemístěn do 3D prostoru, ve kterém mu je nabídnuta možnost opětovného spuštění hry.

7.9 Přizpůsobení pro běh na mobilním zařízení

Platforma mobilních zařízení s sebou nese nevýhodu v podobě nižšího výkonu, k čemuž je od počátku nutné přizpůsobit samotnou hru.

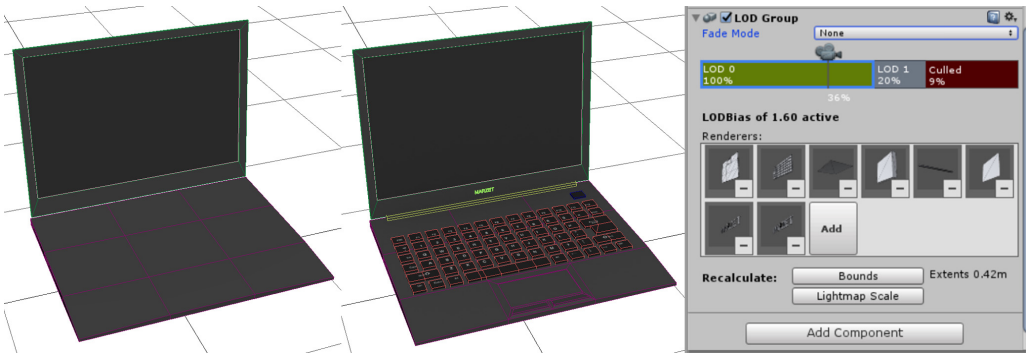
Herní objekty musí být modelovány s ohledem na zachování přiměřeného počtu polygonů. Toto pravidlo se znásobuje u objektů, které se ve scéně opakují a k jejich vykreslování dochází současně (například židle).

Level of Detail

S rostoucí vzdáleností od herních objektů klesá rozlišovací schopnost oka, ale standardně zůstávají všechny detaily při renderingu zachovány. Metoda *Level Of Detail* (dále *LOD*) v závislosti na vzdálenosti hráče mění vykreslovaný model. Pro stejný objekt tedy v rámci aplikace existuje více modelů, které se liší množstvím polygonů a koncovkou. Nejdetailnější model má koncovku *_LOD0* a s každou další redukcí detailů se číslo koncovky zvyšuje.

Vytváření *LOD* skupiny probíhá pomocí přidání prázdného herního objektu do scény skrze *GameObject* → *Create Empty*. Prázdnému objektu je nutné přidat komponentu *LOD Group*, která se stará o změnu vykreslovaného objektu z *LOD* skupiny v závislosti na ploše, kterou objekt ve vykreslovaném snímku zabírá (s rostoucí vzdálenosti plocha klesá). Plocha objektu je určena poměrem výšky vykreslovaného objektu v záběru k výšce celého záběru.

Vytvořené modely s koncovkou *_LOD0*, *_LOD1* atd. je nutné přidat v podokně scény *Hierarchy* jako podřízené dříve vytvořenému prázdnému hernímu objektu. V prázdném herním objektu na kartě *Inspector* je poté pomocí přetažení (*Drag and Drop*) přiřazen objekt pro konkrétní *LOD* úroveň. V tomto okně zároveň probíhá přidávání a odebírání *LOD* úrovní, změna jejich hranic a zároveň je možné nastavit vykreslovanou plochu objektu, od které se *Renderer* objektu zcela zakáže.



Obrázek 22: Zjednodušený notebook *LOD1* v levé části (79 polygonů), detailní model *LOD0* uprostřed (870 polygonů) a nastavení *LOD Group* v pravé části

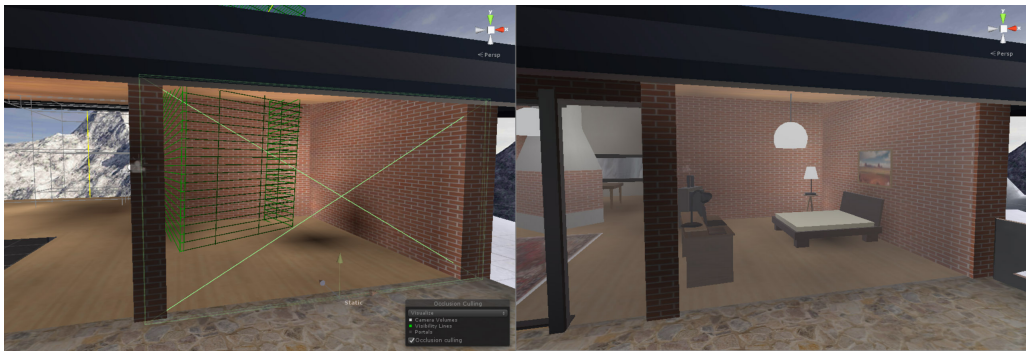
Rozptyl ovlivňující vykreslování všech *LOD* skupin ve scéně je možno nastavovat v *Quality Settings* položkou *LOD Bias*. Nižší hodnota způsobí agresivnější zásah *LOD* a aplikace vykresluje detailní model až při krátké pozorovací vzdálenosti (hodnota 1,6 viz obr. 22 znamená, že hranice 20 % poměru bude posunuta na 32 %).

Zvláště ve VR klesá schopnost uživatele vnímat vzdálené detaily a využití *LOD* u všech složitých objektů přináší velkou úsporu hardwarových prostředků při malém vlivu na výslednou kvalitu zážitku.

Frustrum Culling a Occlusion Culling

Frustrum Culling zakazuje vykreslování objektů, které se nachází mimo záběr kamery, a v *Unity* je vždy automaticky podporováno i bez přičinění uživatele.

Occlusion Culling zakazuje oproti tomu vykreslování objektů, které nejsou z pohledu kamery viditelné z důvodu zakrytí ostatními objekty. V 3D grafice jsou objekty většinou vykreslovány od nejvzdálenějších po nejbližší, čímž dochází k překreslování vzdálenějších objektů bližšími (Unity Manual, 2016).



Obrázek 23: V levé části obrázku jsou pomocí funkce *Occlusion Culling* vykreslovány pouze viditelné objekty, v pravé části jsou vykreslovány všechny objekty

Funkce *Occlusion Culling* může pracovat pouze s objekty, které jsou v *Inspectoru* označeny jako *Static* (v průběhu hry se nijak nemění jejich pozice). V podokně *Occlusion* je dále možné vybrat z možnosti *Occluder Static* a *Occludee Static*. *Occluder Static* se používá pro objekty, které nejsou průhledné a jsou dostatečně velké pro zakrytí ostatních objektů. Tato volba zůstává zakázána u průhledných objektů (nemohou sloužit jako překážka ve viditelnosti) a příliš malých objektů (zbytečné navýšení náročnosti výpočtu viditelnosti). Malé objekty je také možné z výpočtu vyřadit vhodným nastavením hodnoty *Smallest Occluder*. *Occludee Static* slouží pro označení objektů, které mohou být překryty ostatními (zpravidla všechny objekty ve scéně).

V případě velkého množství malých objektů vystupujících ve scéně jednotlivě může docházet k velkému zatěžování procesoru, neboť je nutné pro každý objekt zjistit viditelnost. Na základě dat z *Profileru* je ale možné zjistit, zda hra více zatěžuje *CPU*, nebo *GPU*. Pokud *CPU* čeká na vykreslení snímku prostřednictvím *GPU*, objeví se mezi procesy v *Profileru WaitForJob*. V tomto případě je možné zapnout agresivnější *Occlusion Culling*, protože část zátěže se tím přenese na *CPU*.

Větší zatížení *GPU* je u VR dáno faktem, že je nutno vykreslovat dva různé obrazy, na které je navíc aplikována korekce distorze obrazu.

Lightmapping

Lightmapping slouží k zjednodušení osvětlovacího modelu v aplikaci. Všechny zdroje světla ve scéně (včetně například odraženého světla od lesklého povrchu) jsou pomocí této techniky namapovány přímo do textur na objektech. Výsledkem metody *Lightmapping* je zvýšení počtu vykreslovaných snímků za sekundu při současném vylepšení vizuální stránky hry.

Lightmapping v *Unity* je prováděn skrze zabudovaný modul *Beast*, který se stará o výpočet osvětlení i samotné zapečení textur s přiřazením k objektům (Unity Manual, 2016). Nevýhodou této techniky je její použitelnost pouze na statické objekty, což je ve hrách ale často řešeno způsobem, kdy vzdálené objekty pracují s *Lightmappingem* a osvětlení objektů blíže probíhá dynamicky.

Obrázek 24: Levá část obrázku před aplikováním *Lightmappingu*, pravá část obrázku po provedení *Lightmappingu*

Metoda pracuje stejně jako v případě *Occlusion Culling* pouze se statickými objekty. Označení objektu jako *Static* v *Inspectoru* provede i jeho přidání mezi objekty pro počítání *Lightmappingu*. Ve vlastní hře je nastavení modulu ponecháno na výchozích hodnotách pouze s povolením *Ambient Occlusion*[27] a *Final Gather*[28].

Hardwarové nároky aplikace

V aplikaci je možné nastavit tři úrovně detailů. Požadavky pro běh aplikace jsou následující:

- Procesor 4x 1.5 GHz nebo lepší s odpovídajícím grafickým čipem,
- 1 GB RAM,
- Android 4.4, nebo novější verze,
- brýle *Google Cardboard*.

S výjimkou verze operačního systému se jedná o doporučenou hardwarovou specifikaci a aplikaci je možné spustit i na slabším zařízení. Bude tím ale poznamenán zážitek z virtuální reality, neboť zařízení pravděpodobně nebude schopno udržet stabilní snímkovou frekvenci alespoň 25 snímků za sekundu. Pro maximální herní zážitek je rovněž doporučeno používat při hraní sluchátka.

V aplikaci jsou dostupné tři úrovně grafických detailů, které pracují zejména s rozlišením textur a s *LOD Bias*.

7.10 Export výsledné aplikace a uveřejnění na Google Play

K distribuci a instalaci aplikací pro operační systém *Android* se využívá *Android application package*. Jedná se o archiv s koncovkou *.APK*, který obsahuje všechny zdrojové soubory i informace o aplikaci.

Po vývoji hry nastává fáze její propagace a předání publiku. Z mého pohledu nejlepší způsob distribuce herní aplikace představuje oficiální obchod *Google Play*.

Příprava pro export aplikace

Nastavení exportu pro platformu *Android* vyžaduje úpravu některých parametrů v nabídce přístupné skrze *File* → *Build settings* → *Player Settings*.

Nastavení je rozděleno do několika kategorií, přičemž nejdůležitější položky se nachází v sekci *Other Settings*. Pro export aplikace s *Google VR SDK* je nutné povolit položku *Virtual Reality Supported* a mezi *Virtual Reality SDKs* přidat *Cardboard*. V případě problému s pořadím vykreslovaných objektů ve větší vzdálenosti je nutné v nastavení *Cardboard* vynutit použití 24bitového bufferu hloubky. *Google VR SDK*

[27] Osvětlení, které počítá se zastíněním zdroje světla (Unity Manual, 2016).

[28] Využívá *Ray Tracing* pro sledování nepřímého (odražené) světlo, kterému se tak mění intenzita i barevnost (Unity Manual, 2016).

přepíše hodnoty v sekcích *Resolution and Presentation* a *Splash Image* vlastními, které jsou přizpůsobeny pro zobrazení virtuální reality.

Multithreaded Rendering povolí využití více výpočetních jader. *Static Batching* provede spojení statických objektů se stejným materiálem do jednoho, což vede k redukci počtu dávek určených k renderování, a *Dynamic batching* provede totéž s malými dynamickými objekty.

Minimum API Level specifikuje minimální verzi operačního systému *Android*, na které aplikaci bude umožněna instalace. Jelikož minimální verze *API* pro použití *Google Cardboard* je *4.4*, byla požadovaná verze nastavena právě na tuto hodnotu.

Povolení položky *Prebake Collision Meshes* sice zvětší instalační soubor, ale díky výpočtu fyzikální reprezentace modelu při kompilaci zkrátí dobu načítání aplikace. Ostatní položky zpravidla mohou zůstat na výchozích hodnotách, neboť jejich vliv výkon či velikost aplikace jsou zanedbatelné.

Sekce *Publishing Settings* je důležitá pro podepsání aplikace při exportu do *Google Play*. Je možné použít buď vlastní klíč, nebo vytvořit nový pomocí *Create a new key*.

Uveřejnění v Google Play

Pro uveřejnění aplikace v *Google Play* je nutné využít *Google* účet (nový, nebo stávající) a na stránkách `https://play.google.com/apps/publish/` se zaregistrovat jako vývojář. Registrační poplatek činí 25 $, ale požadován je pouze jednou (nikoli při přidávání každé další aplikace).

Tlačítkem *Přidat novou aplikaci* je uživatel vyzván k zadání jazyka a názvu. Pro aplikaci byl zvolen název *Forsaken Valley VR* (*Zapomenuté Údolí VR*) a jako výchozí jazyk byla zvolena angličtina. *VR* je k názvu přidáno z důvodu, že uživatelé často hledají hry a aplikace pro *Cardboard* pomocí této zkratky.

8 Diskuze

8.1 Zhodnocení řešení

Systém interakce navržený v této práci byl testován u řady uživatelů, přičemž osoby nebyly o způsobu ovládání informovány. Osoby byly rozděleny do čtyř skupin následujícím způsobem:

- **Skupina 1** složená z pěti uživatelů ve věku 20–35 let se zkušeností s hraním počítačových her.
- **Skupina 2** složená ze čtyř uživatelů ve věku 40–55 let s málo zkušenostmi z hraní počítačových her.
- **Skupina 3** složená ze dvou uživatelů ve věku 40–55 let bez zkušenosti s hraním počítačových her.
- **Skupina 4** složená ze dvou uživatelů ve věku 75–83 let bez zkušenosti s hraním počítačových her.

V této fázi nebyl využit finální scénář hry, ale scénář testovací, ve kterém uživatel musel využít opakovaně chůzi a dále interagovat s prostředím pomocí pohledu. Výsledek testování se nachází v tabulce 2.

Tabulka 2: Vyhodnocení skupin uživatelů ve schopnosti ovládat aplikaci

	Skupina 1	**Skupina 2**	**Skupina 3**	**Skupina 4**
Schopnost pohybu	100 %	100 %	100 %	100 %
Schopnost interakce	100 %	100 %	100 %	50 %
Dokončení testu	100 %	75 %	50 %	0 %

Všichni uživatele byli v aplikaci schopni chůze. Jedna osoba ze *Skupiny 4* nebyla schopna interakce a nemohla tedy dokončit scénář. Problémem v tomto případě byl příliš dlouhý čas pro interakci s předmětem, kdy osoba měla pocit, že se při pohledu na předmět nic neprovádí. Na základě tohoto sledování byl čas nutný pro použití předmětu snížen z 2 sekund na 1,5 sekundy. Druhý uživatel ze *Skupiny 4* úkolu ve scénáři nerozuměl a test rovněž nedokončil.

Jeden uživatel ze *Skupiny 2* a jeden uživatel ze *Skupiny 3* nedokončili test pro problémy se zmatením smyslů a nechutenství. V tomto případě mohl být problém jednak v nižší toleranci uživatelů, ale rovněž ve snímkové frekvenci obrazu, která na testovacím zařízení dosahovala přibližně limitních 30 snímků za sekundu.

Z tohoto pohledu hodnotím navržené ovládání jako uživatelsky velmi přívětivé, jelikož osoby pro test byly záměrně vybrány z řad méně znalých hráčů, popřípadě z řad hráčů zcela bez zkušenosti s hrami (mimo *Skupinu 4*). Pouze dva uživatelé z celé skupiny měli předchozí zkušenosti s hraním her pro virtuální realitu.

Problém nastává pouze v případě, že předmět interakce je umístěn tak, že je vyžadováno sklopení pohledu o více než 30°. V případě detekce interaktivního předmětu je chůze zastavena, nicméně u některých menších předmětů je udržení předmětu v zorném poli interakce složité a herní postava se začne při pohledu mimo interaktivní předmět opět pohybovat. Tento případ byl ale částečně eliminován herním designem a většina předmětů k použití se nachází v úrovni očí.

8.2 Alternativní metody pohybu a interakce

Jak bylo popisováno v části 2.3, jeden z velkých problémů VR je kinetóza. Vlastní řešení, které vzniklo v rámci této diplomové práce, může i přes veškerou snahu kinetózu způsobovat. Důvodem je fakt, že v aplikaci je možné se volně pohybovat.

U současných VR her je rozšířen způsob pohybu pomocí teleportace hráče v závislosti na pohledu, popřípadě na základě postupu hrou. V tomto případě je hráč vždy teleportován na stanoviště, na kterém má vyřešit úkol a poté pokračuje na další stanoviště. Výhodou využití teleportace je eliminace nemoci z pohybu.

V počáteční fázi této práce byla rovněž prozkoumávána možnost projekce reálného pohybu v prostoru do virtuální místnosti ve hře. Data z gyroskopu a akcelerometru jsou dostačující pro promítnutí základního pohybu v prostoru, ovšem již po krátké době pozice v reálné místnosti neodpovídá pozici v místnosti virtuální. Problém by se dal pravděpodobně částečně řešit pomocí filtrů, které by odstranily nevhodně vyhodnocené údaje při otáčení hlavy, nicméně pro opravdu precizní sledování pohybu uživatele v reálném prostoru je nutné využít optický či laserový systém sledování hráče v prostoru (využívá například *HTC Vive*).

Dle návrhových vzorů *Google VR* by aplikace využívající automatickou interakci s odpočítáváním měla obsahovat i možnost okamžité aktivace pomocí posuvného magnetu *Cardboard Trigger*, aby bylo možné předcházet zbytečnému zdržování uživatele při využívání interaktivních prvků. Tato funkcionalita není v aplikaci umožněna vzhledem k požadavku zadání na přirozené ovládání aplikace bez manuální interakce ze strany uživatele. Přesto se domnívám, že by bylo vhodné aplikaci o tuto možnost doplnit.

8.3 Vhodné rozšíření projektu

Dokončení hry trvá 15 až 30 minut dle schopností hráče, což je pro většinu uživatelů mobilní virtuální reality akceptovatelné. Pro lepší motivaci hráče by bylo vhodné zdokonalit vyprávění příběhu a herní zážitek prodloužit, což ale při současných schopnostech mobilních zařízení nemusí být vždy vnímáno pozitivně. Hra navíc neumožňuje postup uložit a jako vhodné rozšíření se tedy nabízí implementace alespoň jednoduchého ukládání herní pozice.

Prostředí hry by bylo vhodné doplnit dalšími interaktivními prvky, protože většina použitelných objektů se nachází uvnitř domu a doplnění dalších interaktivních prvků do krajiny by pomohlo předcházet zklamání hráče při prozkoumávání okolí.

8.4 Ekonomické zhodnocení projektu

Osobně vnímám velký potenciál na poli her pro mobilní virtuální realitu. Zatímco klasických her je pro tuto platformu dostupné nepřeberné množství, komplexních herních titulů využívajících mobilní VR je v řádech desítek. Ačkoli popularita virtuální reality v současné době výrazně roste, uživatelská základna VR mobilních her je logicky výrazně menší, než v případě klasických mobilních her. Z tohoto důvodu byla jako výchozí jazyk aplikace zvolena angličtina.

Finanční náročnost na pořízení mobilní virtuální reality se liší dle toho, zda se do ceny započítává i cena mobilního zařízení, nebo pouze cena brýlí. S přihlédnutím k faktu, že mobilní telefon je zařízení univerzální, cena za samotné rozšíření funkcionality a možnost využití telefonu pro virtuální realitu stojí v závislosti na kvalitě brýlí od 120 Kč do přibližně 2 500 Kč. Poměrně kvalitní brýle lze ale pořídit i za cenu do 500 Kč.

V současné době neexistuje pro mobilní VR hra, která by sama o sobě motivovala ke koupi brýlí. Velké množství dostupných her slouží spíše jako demonstrátor VR technologie a nenabízí žádný herní obsah.

9 Závěr

Práce se zabývá virtuální realitou na mobilních zařízeních a skládá se jak z analýzy současného stavu na poli mobilní VR, tak z popisu praktických úkonů při vývoji požadované herní aplikace. Lze říci, že zadání práce se podařilo splnit a výsledné řešení bude nyní sbírat první ohlasy prostřednictvím umístění v *Google Play*.

První část práce definuje virtuální realitu, představuje její současné využití v praxi, rozebírá problémové oblasti jejího nasazení a seznamuje s existujícími hrami, přičemž je zde diskutována i použitelnost zkoumaných řešení pro požadovanou aplikaci.

Následně jsou v kapitole 3 popsány zařízení na trhu pro zprostředkování virtuální reality a text se více zaměřuje na mobilní platformu. Popsány jsou zde senzory v mobilních zařízeních, jejich využití pro konkrétní hry ve virtuální realitě i vliv jejich přesnosti na kvalitu VR.

Na tuto kapitolu navazuje část zabývající se vývojem her pro mobilní VR. V kapitole 4 je popisována platforma *Google VR* a osvědčené postupy pro její použití, kalibrace mobilního zařízení pro virtuální realitu a dále je provedeno srovnání aktuálně dostupných vývojových prostředí.

Prostředí hry je vytvořeno pomocí programu *3ds Max* a zvláštní důraz je kladen na jednoduchost a efektivitu modelů, aby bylo možné zajistit plynulý běh na mobilním zařízení. V programu *3ds Max* byly objekty rovněž texturovány a výsledné modely byly exportovány ve formátu *.FBX* pro další použití ve vývojovém prostředí.

S ohledem na analýzu herních enginů a jejich vývojových prostředí v části 4.2 byl pro implementaci hry zvolen program *Unity*, do kterého byly importovány modely vytvořené v předchozí části. V prostředí se nachází velké množství interaktivních herních prvků včetně těch, které nemají zvláštní význam pro dokončení herního scénáře.

Na základě znalostí z teoretické části je navržena hra, která využívá pohled hráče pro přirozené ovládání aplikace. Chůze je ovládána sklopením úhlu pohledu o 30° od vodorovné úrovně. Interakce s předměty probíhá pomocí zaměření objektu pohledem a setrváním po dobu 1,5 sekundy.

Výsledná hra umožňuje průchod i interakci s virtuálním prostředím a je spustitelná na mobilních zařízeních s operačním systémem *Android* ve verzi 4.4 a novější. Hra obsahuje scénář, jehož dokončení hráči trvá přibližně 15 až 30 minut dle jeho schopností. Vizuální stránku hry považuji za zdařilou, ale hra i přes to běží na přijatelných 30 snímků za sekundu na slabších zařízeních.

Navržená hra splňuje požadavky zadání na přirozené ovládání ze strany uživatele, interakci s herními objekty a obsahuje scénář, který je možné dokončit.

Hra je v současné době umístěna na *Google Play* pod názvem *Forsaken Valley VR* (*Zapomenuté Údolí VR*) a ke stažení je dostupná zdarma. Ke hře bylo rovněž vytvořeno propagační video, které je umístěno na stránce aplikace v *Google Play*.

10 Reference

AUTODESK. *3D modeling, animation, and rendering software.* Autodesk [online]. 2016 [cit. 2016-11-17]. Dostupné z: http://www.autodesk.com/products/autodesk-3ds-max/overview.

BENBOURICHE, M. ET AL. *Virtual reality applications in forensic psychiatry.* New York: VRIC '14 Proceedings of the 2014 Virtual Reality International Conference, 2014. ISBN 978-1-4503-2626-1.

BURDEA, G. – COIFFET, P. *Virtual reality technology, 2nd ed.* Hoboken, New Jersey: A Wiley-Interscience, 2003. ISBN 0-471-36089-9.

CARSON, E. *10 ways virtual reality is revolutionizing medicine and healthcare* [online]. In: Tech Republic, 2015 [cit. 2016-11-12]. Dostupné z: http://www.techrepublic.com/article/10-ways-virtual-reality-is-revolutionizing-medicine-and-healthcare/.

CRAWLIST TEAM. *10 Best Game Engines to use* [online]. In: Crawlist, 2016 [cit. 2016-11-12]. Dostupné z: http://www.crawlist.net/2016/05/10-best-game-engines-to-use.html.

DUROVIS. *Tuscany Dive* [online]. Durovis, 2016 [cit. 2016-11-17]. Dostupné z: https://www.durovis.com/en/board_topic_2661.html.

EPIC GAMES *Unreal Engine* [online]. Epic Games, 2016 [cit. 2016-12-2]. Dostupné z: https://www.unrealengine.com.

FINKELSTEIN, S. ET AL. *Astrojumper: motivating children with autism to exercise using a VR game.* Atlanta, Georgia: CHI EA '10 CHI '10 Extended Abstracts on Human Factors in Computing Systems, 2010. ISBN 978-1-60558-930-5.

GOODRICH, R. *Accelerometer vs. Gyroscope: What's the Difference?.* In: LiveScience [online]. 2013 [cit. 2016-12-7]. Dostupné z: http://www.livescience.com/40103-accelerometer-vs-gyroscope.html.

GOOGLE. *Android Developers Guide* [online]. Google, 2016 [cit. 2016-11-15]. Dostupné z: http://developer.android.com/index.html.

GOOGLE. *Designing for Google Cardboard* [online]. Google, 2016 [cit. 2016-11-17]. Dostupné z: https://www.google.com/design/spec-vr/designing-for-google-cardboard/a-new-dimension.html.

GOOGLE. *Google Play* [online]. Google, 2016 [cit. 2016-11-15]. Dostupné z: https://play.google.com/store.

GOOGLE. *Google VR Developer* [online]. Google, 2016 [cit. 2016-11-17]. Dostupné z: https://developers.google.com/vr/.

GOOGLE. *Sensor types* [online]. Google, 2016 [cit. 2016-11-20]. Dostupné z: https://source.android.com/devices/sensors/sensor-types.html.

GOOGLE. *Virtuální realita pro každého* [online]. Google, 2016 [cit. 2016-11-18]. Dostupné z: https://vr.google.com.

GREGORY, J. *Game Engine Architecture* Boca Raton, Florida: CRC Press, 2009. ISBN 978-1-4398-6526-2.

GRIFFIN, D. *How does the Global Positioning System work?*. In: PocketGPSWorld.com [online]. 2011 [cit. 2016-11-17]. Dostupné z: https://developers.google.com/vr/.

HANDRAHAN, M. *VR sickness cannot be solved by hardware improvements alone*. In: GamesIndustry.biz [online]. 2015 [cit. 2016-11-24]. Dostupné z: http://www.gamesindustry.biz/articles/2015-03-05-vr-sickness-cannot-be-solved-by-hardware-improvements-alone.

LAMKIN, P. *The best VR headsets: The top virtual reality devices to go and buy now*. In: Wereable [online]. 2016 [cit. 2016-11-24]. Dostupné z: http://www.wareable.com/headgear/the-best-ar-and-vr-headsets.

OCULUS VR. *Camera Simulator by Canon Labs* [online]. Oculus VR, 2016 [cit. 2016-12-5]. Dostupné z: https://www.oculus.com/experiences/rift/1032993353450807/.

ONG, S. K. – NEE, A. Y. C. *Virtual and Augmented Reality Applications in Manufacturing*. New York: Springer, 2004. ISBN 1-85233-796-6.

PARKIN, S. *How VR is training the perfect soldier*. In: Wereable [online]. 2015 [cit. 2016-11-24]. Dostupné z: https://www.wareable.com/vr/how-vr-is-training-the-perfect-soldier-1757.

PRASUETHSUT, L. *The best HTC Vive games you need to play*. In: Wereable [online]. 2016 [cit. 2016-11-24]. Dostupné z: http://www.wareable.com/headgear/the-best-ar-and-vr-headsets.

REALITEER. *Build RealControl2* [online]. Realiteer, 2016 [cit. 2016-11-27]. Dostupné z: http://www.realiteer.com/diy.

SHIVATECH. *ShiVa engine* [online]. ShiVaTech, 2016 [cit. 2016-11-17]. Dostupné z: http://www.shiva-engine.com.

UNITY TECHNOLOGIES. *Unity Manual* [online]. Unity Technologies, 2016 [cit. 2016-11-17]. Dostupné z: http://docs.unity3d.com/.

UNITY TECHNOLOGIES. *Game Engine* [online]. Unity Technologies, 2016 [cit. 2016-11-27]. Dostupné z: http://unity3d.com/.

Virtual Reality Society. *Virtual Reality in the Military* [online]. Virtual Reality Society, 2016 [cit. 2016-12-3]. Dostupné z: http://www.vrs.org.uk/virtual-reality-military/.

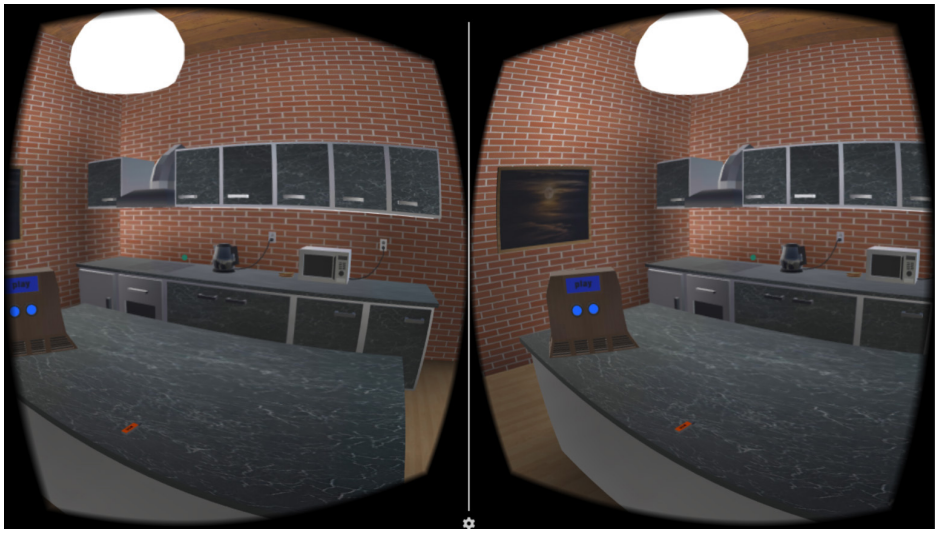
VR Lens Lab. *How Lenses for Virtual Reality Headsets Work* [online]. VR Lens Lab, 2016 [cit. 2016-12-3]. Dostupné z: https://vr-lens-lab.com/lenses-for-virtual-reality-headsets/.

Whittinghill, D. M. *Nasum Virtualis: A Simple Technique for Reducing Simulator Sickness*. In: Purdue University [online]. 2015 [cit. 2016-12-3]. Dostupné z: http://www.purdue.edu/newsroom/releases/2015/Q1/virtual-nose-may-reduce-simulator-sickness-in-video-games.html.

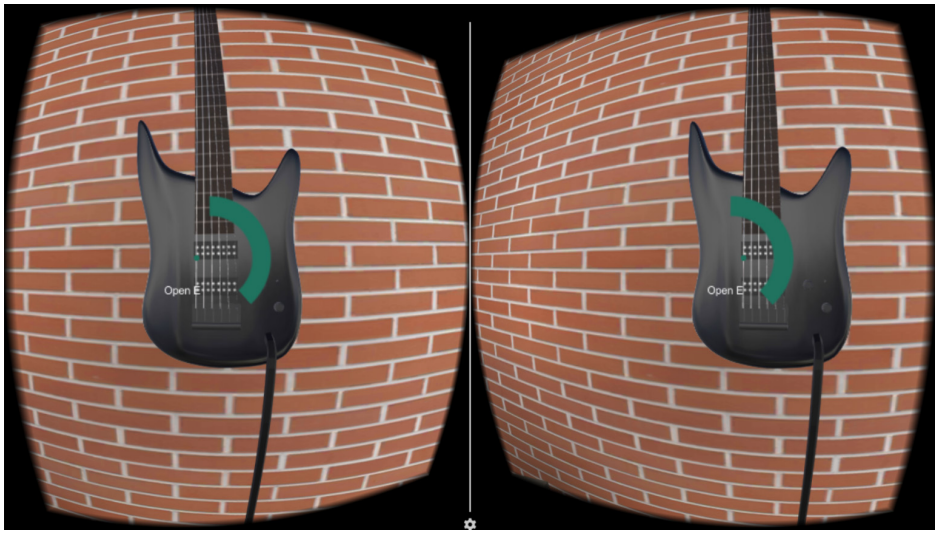
Yoo, S. et al. *Controller-less Interaction Methods for Google Cardboard*. California, Los Angeles: SUI '15 Proceedings of the 3rd ACM Symposium on Spatial User Interaction, 2015. ISBN 978-1-4503-3703-8.

Přílohy

A Snímky z výsledné hry

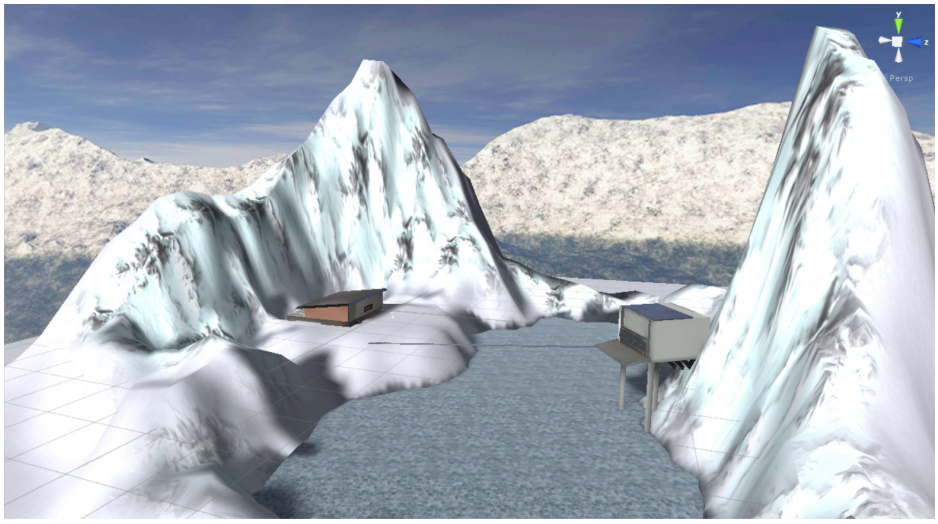


Obrázek 25: Interiér domu obsahuje množství interaktivních prvků (z objektů na tomto obrázku je možné použít digestoř, mikrovlnou troubu, konvici, rádio i USB disk)



Obrázek 26: Kytara modelována v části 6.2, na kterou je nutné přehrát pro další postup hrou sekvenci tónů

Obrázek 27: Doba potřebná pro interakci s objektem je signalizována vlastním kruhovým ukazatelem a použití většiny objektů ve hře je potvrzeno zvukovým efektem



Obrázek 28: Herní prostředí s přirozenou hranicí v podobě hor

B DVD

Přiložené DVD na zadním obalu práce obsahuje:

- kompletní projekt z vývojového prostředí *Unity* včetně všech skriptů, modelů a materiálů,
- prezentační video,
- výsledný instalační soubor hry s koncovkou *.apk*,
- práci v elektronické podobě.