VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VR INTERAKTIVNÍ APLIKACE

VR INTERACTIVE APPLICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MAREK VALENTA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ MILET**
SUPERVISOR

BRNO 2018

Abstrakt

Tento dokument se primárně zabývá popisem vývoje interaktivní aplikace pro virtuální realitu. Na začátku je popis historie VR, která sahá až do 19. století. Následuje popis zařízení, které se momentálně pro VR používají, a pro které je tato aplikace vytvořena. Dále je představeno vývojové prostředí Unity 3D, což je, společně se základním popisem neuronových sítí velká součást celého projektu. Druhá polovina popisuje návrh a implementaci aplikace simulujícího magický svět včetně nového konceptu kouzlení, který v žádné VR hře nebyl dosud použit, a jde o rozpoznání gest pohybu ruky, pro tvorbu kouzla, pomocí neuronové sítě. Dále práce obsahuje popis celé aplikace, místností a objektů. Závěr rozebírá problémy a slepé uličky, které byly vyzkoušeny ale nevyhovovali.

Abstract

This document is primarily concerned with describing the development of an interactive virtual reality application. At the beginning is a description of the history of the VR, dating back to the 19th century. Following it is a description of the devices that are currently being used for the VR and for which this application is created. The Unity 3D development environment is presented, which is, together with the basic description of neural networks, a major part of the whole project. The other half describes the design and implementation of simulation of a magical world, including a new magic spell system that has not yet been used in any VR game, which is the recognition of hand gestures for spell creation by using neural network. The work also contains a description of the whole application, rooms and objects. The conclusion discusses the problems and dead ends that have been tried but failed.

Klíčová slova

Virtuální realita, neuronová síť, rozpoznávání gest, Unity3D, Tensorflow, aplikace, hra

Keywords

Virtual reality, neural network, gesture recognition, Unity3D, Tensorflow, application, game

Citace

VALENTA, Marek. *VR INTERAKTIVNÍ APLIKACE*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet

VR INTERAKTIVNÍ APLIKACE

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Mileta. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Marek Valenta
22. května 2018

Poděkování

Děkuji mému vedoucímu Ing. Tomáši Miletovi za vedení a veškerou pomoc při vypracování této práce.

Obsah

Kapitola 1

Úvod

Virtuální realita (dále jen VR) je momentálně stále častěji zmiňované téma, které v budoucnu bude stále nabývat na síle. První zmínky jsou starší, než se může zdát, ale až v posledních letech nastává počátek této technologie. Samozřejmě díky tomu má daleko k dokonalosti, ale velké množství výrobců se snaží, více či méně, tuto technologii ovlivňovat, vyvíjet a zlepšovat.

Aplikace, která v této práci bude popsána, bude tzv. „Proof of concept“ použití neuronové sítě ve VR pro rozpoznávání gest. Bude to interaktivní aplikace, ovládaná pomocí ovladačů, kde hlavním cílem bylo vytvořit základy pro kouzlení neboli rozpoznání mávnutí ruky v prostoru. Po přečtení by měl mít čtenář základní povědomí o tom jak takovou aplikaci vytvořit.

Na začátku této práce jsou informace o tom, co to VR je a jaká je překvapivá historie této technologie, protože VR je mnohem starší než se může zdát a to záleží na úhlu pohledu, jak se na ni díváme. Budou zde informace od první zmínky až do současnosti spolu s vizionářskými vidinami některých lidí, kteří předběhli dobu, ale také neúspěchy v začátcích velkých firem, když se pokoušeli o tuto technologii.

Ve druhé kapitole jsou popsána zajímavá zařízení pro VR. Jednak jsou to ty, které přišly s něčím jednoduchým a chytrým oproti konkurenci, nebo život mladého nadšence do VR, který tuto technologii nenávratně změnil k lepšímu a posunul o kus dopředu. Také bude krátce vysvětleno nové VR zařízení treadmill, které podle mého v budoucnu velmi ovlivní tuto technologii. Rozebereme výhody a nevýhody jednotlivých zařízení a částečně jejich specifikace. Dále zde bude popis zařízení HTC Vive, který byl použit pro tvorbu této aplikace.

Ve třetí kapitole bude popsáno vývojové prostředí Unity, protože ve zbytku práce se tyto termíny budou využívat. Je zde uvedena základní struktura tohoto programu, a jeho hlavní stránka. Dále komponenty, které existují a většina z nich je i použita v aplikaci. Následuje vysvětlení základního objektu Unity GameObject, společně se skripty, které dávají tomuto objektu funkčnost a jak tyto objekty řadit do hierarchie a proč to vůbec dělat. Na konci bude část o podpoře VR v Unity.

Čtvrtá kapitola pojednává o základech neuronových sítí, které jsou zde v aplikaci použity. Stručně je zde popsána konvoluční neuronová síť, její učení pomocí algoritmu BackPropagation a tvorba datové sady. Je důležité si uvědomit tyto základy pro další části této práce.

Kapitola Návrh obsahuje návrh aplikace, což je část práce, která se bude snažit o to, abychom ukázali přednosti VR oproti normálním PC aplikacím. Bude zde popsán návrh hry, kde se hráč bude moct stát kouzelníkem. Jistě je to sen spousty lidí vyzkoušet si, jaký

je to pocit mít nadpřirozené schopnosti a tato aplikace přesně toto nabídne. Hra musí být navržena tak, aby hráč přesně věděl, co dělat v jakých situacích a aby ho tutoriál provedl základy hrou. Popsány budou také všechny scény hry tak, jak jdou chronologicky po sobě. Dále zde budou popsány postupy, některých kritických postupů ve VR jako je tvorba menu a pohyb v aplikacích. Je to z důvodu, že VR je nová technologie a postupy nejsou zcela zavedeny. Proto tato část může pomoci v návrhu jiných aplikací. V neposlední řadě bude uveden rozbor a architektura neuronové sítě a popis tvorby datové sady, protože tato část je hlavním bodem této aplikace. Jako poslední zde bude popsáno ovládání hry jako pohyb, kouzlení atd.

Kapitola implementace bude pojednávat o cestě, kterou jsem absolvoval při vytváření takovéto aplikace. Začne základy programování pro VR, kde budeme čerpat ze třetí kapitoly, kde jsou uvedeny nástroje pro tvorbu takové aplikace. Dále zde budou uvedeny skripty využity na ovladačích, protože skrze ně se hra bude ovládat. Poté zde bude rozbor snímání a ukládání pohybu hůlky, a proč jsem se rozhodl pro TrailRenderer, spolu se všemi cestami, kterými jsem se vydal, abych uspokojil potřeby aplikace, tedy odezvu a spolehlivost. Nejdůležitější část bude o rozpoznání gest, tedy aplikování neuronové sítě na nahraná data. Tato část bude pojednávat o všech slepých uličkách, na které jsem narazil, ale i tyto informace jsou důležité pro budoucí projekty, které takovou funkčnost budou implementovat. Jsou zde tedy uvedeny a zhodnoceny přístupy, které jsem zkoušel, a z této části by měl čtenář odejít s tím, jak a proč správně vytvořit detekci gest. Zbytek kapitoly pojednává o detailech implementace menu a tutoriálu, kde se například uvede implementace a chyby knihy kouzel.

Kapitola 2

Virtuální realita

VR je technologie, která se snaží o to, abychom se v počítačem vygenerovaném světě cítili a chovali jako v reálném světě. Hlavní ideou je, abychom odstínili vlivy vnější reálné reality a nahradili je simulací. Snažíme se o to, aby naše smysly vnímaly falešné vjemy jako reálné. Tato idea sice zní velice jednoduše, ale doposud se to moc nedaří a vždy se musíme smířit s kompromisem. Momentálně se nejvíce vyvíjí headset pro VR, což je zařízení, které si nasadíme na hlavu a vidíme počítačem vygenerovaný obraz, do sluchátek nám hraje vygenerovaný zvuk a pohyb hlavy popřípadě těla se přenáší do toho, co vidíme.

2.1 Historie

Historie je velice zajímavá, protože lidstvo se již dlouho snaží o VR. Není to tedy téma vztahující se k našemu století. Ze článku [4] vyplývá, že pokud VR vyložíme tak, že se nás snaží přesvědčit o tom, že jsme jinde, než ve skutečnosti jsme. První zmínky jsou datovány do 19. Století, kdy existují nástěnné malby, které vyplňují celé zorné pole, takže se můžeme cítit jako součástí nějaké události.

V roce 1838 výzkum Charlese Wheatstona demonstroval [4], že lidský mozek zpracovává dva rozdílné 2D obrazy, pro každé oko jedno, do jedné 3D scény. Při dívání na dva stereoskopické obrázky přes stereoskop získáme hloubku obrazu, kterou si mozek domyslí. Pozdější výzkum vedl k vytvoření populárního View-Master stereoskopu 2.1, který byl používán pro „virtuální turismus“. Tento princip je využit u jednoduchých brýlí Google Cardboard 2.2, které se stali velmi populárními.

Obrázek 2.1: View-Master stereoskop.

Obrázek 2.2: Google Cardboard.

V roce 1929 Edward Link vytvořil „Link trainer“ [4] pravděpodobně první letecký simulátor, který byl zcela elektromechanický. Tento simulátor byl velice jednoduchý, pro chod potřeboval pár motorů pro naklánění a např. pro tvorbu turbulencí. Stal se velice populárním hlavně za druhé světové války, kdy si tento přistroj vyzkoušelo na 500 000 lidí.

Ve 30. letech 19. století spisovatel Stanley G. Winbaum vytvořil první model VR ve své povídce „Pygmalion's Spectacles“ [4], jež obsahuje ideu brýlí, které nechají nositele zažít virtuální svět pomocí holografie, pachu, chuti a doteku. Tento popis byl velice vizionářský a dnešní výzkum se o takové brýle částečně snaží.

V roce 1968 Ivan Sutherland na univerzitě v Utahu společně s pomocí svých studentů vytvořil první headset pro VR a Rozšířenou realitu [4]. Tento headset byl ale velmi těžký, takže bylo nutné být zavěšen na strop a museli jste být přivázaní k zařízení. Tento systém je samozřejmě vizionářský, ale byl velmi jednoduchý a zobrazoval jen hodně špatný obraz jednoduchých místností.

V roce 1991 začínáme vidět zařízení, která byla přístupnější, ale stále nebylo možné tato zařízení vlastnit doma [4]. Firma Virtuality Group vytvořila řadu arkádových her a zařízení. Uživatelé měli možnost za pomocí VR brýlí hrát zajímavé stereoskopické 3D vizualizace a dokonce některé byly spojeny, aby si hráči mohli zahrát hry pro více hráčů.

V letech 1993 firma SONY oznámila headset pro VR, který se již velmi podobá dnešním headsetům [4]. Zařízení umožňovalo detekci pohybu hlavy, stereo zvuk a LCD displej. I přes velmi nízkou cenu tento model zůstal jen jako prototyp, protože po celou dobu byl provázen technickými chybami. Podobné zkušenosti má i Nintendo se svým zařízením Virtual Boy (originální jméno VR-32) a to i přesto, že tento model způsobil velmi velké vzrušení mezi lidmi. Byla to první konzole, která podporovala VR, ale displej byl jen dvoubarevný, byl velký nedostatek her a také používání bylo velmi nepohodlné, takže žádná revoluce se nekonala.

2.2 Zařízení

Zařízení pro VR vyvíjí momentálně, mimo jiné, tito hlavní zástupci: SONY, HTC, Oculus, Google, Apple, Samsung. Aktuálně je nejvíce populární headset a ten se dá dělit na ty, které potřebují PC nebo herní konzoli, nebo ty, jenž potřebují mobilní zařízení. První typ potřebuje v headsetu velmi kvalitní displej, protože oko takto blízko je velmi citlivé. Do druhého se vkládá mobilní zařízení, které na svůj displej zobrazí dva obrazy, pro každé oko jeden. Na oba typy potřebujeme čočky, které nám obraz dobře zobrazí.

2.2.1 SONY PlayStation VR

Levnější a stále přijatelně kvalitní headset, který je nutný připojit k zařízení PlayStation (PS). Ve sféře konzolí je to unikát a technologii VR dokázal velice rozšířit díky 2 mil. prodaným kusům [20].

Jak je uvedeno v [16], na rozdíl od konkurence nepotřebuje velmi výkonnou grafickou kartu a spokojí se, se zabudovaným grafickým čipem v PS. Toto je dosaženo za pomocí PS kamery, které sleduje 9 různých zdrojů světla z headsetu a světla z PS Move, což jsou ovladače do ruky. Nevýhoda této technologie je, že pokud zdroj světla ovladače zakryjete např. tělem, tak kamera není schopná určit polohu ovladače. Díky tomu, že je použita jen jedna kamera, která snímá pohyb, nelze příliš určovat pohyb po místnosti. Takže oproti např. HTC Vive, nedokáže tato technologie nabídnout tolik volnosti.

Obrázek 2.3: Google Daydream View.

Obrázek 2.4: Samsung Gear VR.

Toto zařízení má pak další velkou výhodu a to tým lidí, který vytváří hry pro PS VR. Hry pro VR jsou velmi náročné na implementaci, protože je málo vývojářů a spotřebitelů. Díky tomu je tato platforma velmi rozšířená s bohatou herní i hráčskou základnou.

PS VR aktuálně umožňuje zahrát si na 150 her a díky velkým prodejům se SONY rozhodlo, že bude více investovat do této sféry a do konce roku zde bude 280 her [9].

2.2.2 Google a Samsung

Tyto dvě firmy se snaží o realizaci VR společně s mobilním přístrojem. Google začal pomocí již zmiňovaného Google Cardboard a teď má Google Daydream View (2.3), Samsung má Samsung Gear VR (2.4).

Google začal podporovat virtuální realitu pro telefony vydáním papírových brýlí Google Cardboard [11], pomocí kterých šlo z telefonu vytvořit headset. Toto řešení má oblibu dodnes, hlavně díky své nízké ceně. Společně s tímto samozřejmě přišly změny do systému android, aby více podporoval VR. Google Daydream View přidává také bezdrátový ovladač, kterým lze aplikace a hry ovládat. Největší nevýhoda se jeví v tom, že podpora mobilních telefonů je velmi malá.

Nevýhodu Googlu Samsung nemá, protože všechny jeho novější modely tuto podporu mají. Model Samsung Gear VR pomáhala vytvořit společnost Oculus a dokázali stvořit skvělý headset pro Samsung mobilní telefony. Součástí je samozřejmě ovladač, který je velmi podobný tomu od Googlu.

2.2.3 Oculus

Okolo 15 let se Palmer Luckey [13] zamiloval do konceptu VR a byl jejím vášnivým sběratelem. Nebyl spokojený s tím, jak kvalitní a drahé byly headsety a proto se v 16 letech rozhodl, že začne vyvíjet vlastní. Velkou změnu v jeho životě a technologie VR byl crowfudingový projekt na kickstarteru, kde vybral za první tři dny 1 000 000$. Byl to úspěch, který vedl k prvnímu modelu Dev Kit 1 (2.5). Tento model byl přímo mířený na vývojáře, aby si osvojili technologii a zjistili její potenciál. Také bylo jasné, že tento model se stane startem nové éry, protože za cenu 350$ byl daleko levnější a technologicky lepší než konkurenční produkty. Nevýhodou bylo velmi nízké rozlišení displeje a nemožnost určovat pozici v prostoru, umělo to jen zjistit naklonění a otočení hlavy.

Tento pramen [13] uvádí, že další verze zařízení Dev kit 2 (2.6) byl vylepšená verze svého předchůdce. Nevýhoda prvního typu byl displej o rozlišení 640800 nový typ má rozlišení 960*1080 toto je obrovský nárůst pixelu, který se na první pohled projeví. Obrovský pokrok

Obrázek 2.5: Dev Kit 1.

Obrázek 2.6: Dev Kit 2.

Obrázek 2.7: Oculus Rift.

bylo představení kamery, která snímá chytře ukryté led diody v headsetu a díky tomu je tedy schopný rozpoznat, kde v prostoru se headset nachází. Tento model se vydal jen těsně před tím, než tuto firmu zakoupila společnost Facebook.

Poslední model Oculus Rift (2.7) je již plnohodnotné VR dnešní doby. Donedávna potřeboval externí ovladač, ale teď již disponuje vlastními ovladači Oculus Touch, které se od ovladače HTC Vive velmi liší. Ovladač disponuje ovládací analogovou páčku jako na gamepadech, dvěma tlačítky na horní straně, spouští a postranním tlačítkem pro prostředníček pro snazší chytání objektů. Kromě toho je na horní straně ovladače vyhrazená plocha pro palec, pokud zrovna nic nedělá a na každém ovladači je také systémové tlačítko, které v pravé ruce vyvolá Universal Menu a v levé si ho může použít hra, jak bude potřeba.

2.2.4 HTC Vive

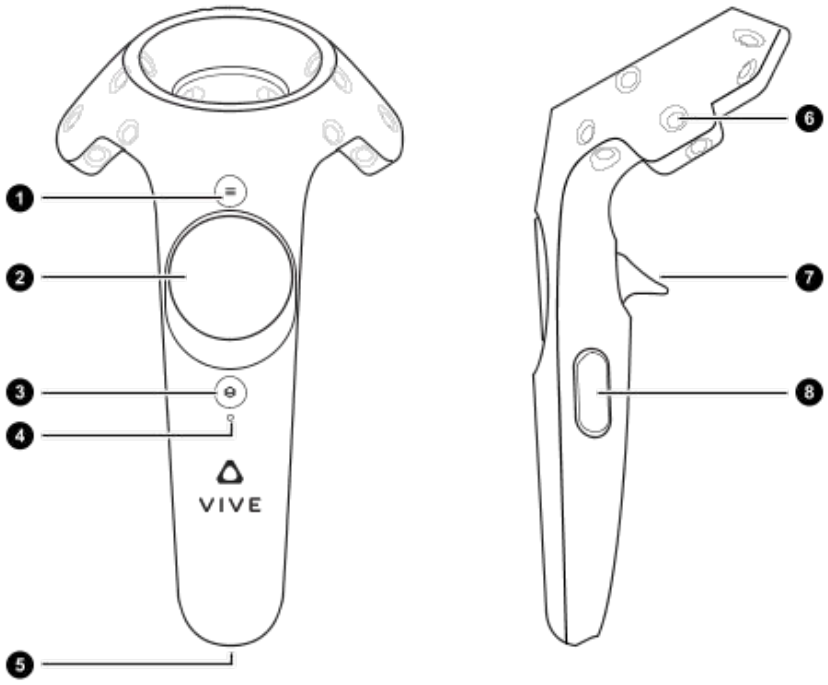
Toto zařízení bylo vyvinuto ve společenství HTC a Valve, a díky těmto firmám vznikl velký konkurent pro Oculus Rift, protože spojení VR se službou steam byl důležitý krok.

První předprodej pro Vive byl 29. února 2016 a začal se vydávat 5. květnu téhož roku, obsahoval headset, dva ovladače a dva senzory. Vive byl již od počátku dražší než Rift, ale díky spolupráci se službou Steam měl Vive velkou výhodu týkající se her a to byla velký posun. Další plus bylo v ovladačích, protože Rift je v době vydání neměl a Vive tedy představoval lepší interakci s prostředím.

Technologie pro sledování pohybu

Existují dvě nekompatibilní verze způsobu jak detekovat pozici headsetu a ovladačů, obě jsou založeny na infračerveném světle, jen se mírně liší ve způsobu.

První „SteamVR tracking v1.0“ pro fungování potřebuje dvě krabičky, které obsahují LED žárovky pro reset signál, dva motory s LED žárovkou, přesnou pozici těchto krabiček



Obrázek 2.8: Ovladač a jeho popis 1. Nabídka menu tlačítko, 2. multifunkční trackpad, 3. Systémové tlačítko, 4. stavová led kontrolka, 5. Micro-USB port, 6. Senzory pro určení polohy, 7. Trigger, 8. Tlačítka pro stisk sevření ruky. Převzato z [2].

a headset/ovladače disponující senzory na infračervené záření. První motor s LED žárovkou se točí vertikálně a druhý horizontálně. Oba motory se točí rychlostí 60 otáček za sekundu a střídají se ve svícení, takže jednu chvilku projede celou místností horizontální paprsek a hned po tom projede vertikální. Jakmile oba paprsky proběhnou, blikne reset signál. Samotná detekce potom probíhá tak, že headset čeká na reset signál, pokud ho dostane, začne počítat čas a čeká na vertikální záblesk a na základě času zjistí, v jaké vertikální rovině se nachází. Poté čeká na horizontální záblesk a zjistí, na jaké horizontální rovině se nachází, pak zase přichází reset signál a takto se to opakuje. Pomocí geometrie lze potom dopočítat přesnou polohu headsetu.

První „SteamVR tracking v2.0“ funguje velmi podobně jako předchozí verze s tím rozdílem, že v záblesku posílá data, které obsahují např. ID krabičky, naklonění motoru, čas atd. Díky tomu v krabičce nemusejí být motory dva, ale jen jeden všesměrný, a nemusí být ani reset signál. Takto řešená detekce je levnější a přidává novou možnost a to škálovatelnost, takže je po té možné mít více takových krabiček a mapovat velké a členité prostory.

2.2.5 Ovladač

Ovladač u HTC Vive [2] slouží pro interakci s okolním virtuálním světem. Jeho schéma je na obrázku 2.8 a má 24 senzorů, multifunkční trackpad, trigger a zpětnou vazbu pomocí vibrací. Je napájen za pomocí baterií a je bezdrátový. Pro pozdější reference je zde uveden obrázek a popisem tlačítek.

Obrázek 2.9: Omni konkávní disk pro chůzi na místě.

Obrázek 2.10: Omni postroj pro držení těla.

2.2.6 Treadmill pro VR

Jeden z největších problémů ve VR je reálný pohyb hráčů, a proto zde bude popsáno jedno z možných řešení. HTC Vive a Oculus Rift řeší pohyb tak, že existuje omezený prostor, kde se může hráč pohybovat, ale pro hry je takto omezený prostor nedostatečný. Je žádoucí, aby hráč měl možnost mít větší prostor, což je často řešeno tak, že hráče teleportujete na ukázané místo. Je nutné, aby to byla teleportace, protože lidem se rychle dělá špatně, pokud se scéna hýbe, ale ve skutečnosti stojí na místě. Proto vzniklo zařízení treadmill neboli „běžecký pás“. [12]

Firma Virtuix [12][1] má zařízení treadmill Omni, které pracuje dost podobně jako běžecký pás jen s tím rozdílem, že se pod vámi nepohybuje nějaký pás, ale vy kloužete, když děláte krok. Zařízení Omni vytvořilo konkávní disk 2.9, který se speciálními boty má velmi malé tření a můžete na místě chodit ve všech směrech. Díky tomu, že disk je konkávní, tedy prohnutý dolů, tak při každém kroku se za pomocí gravitace vrátíte do startovní pozice. Senzory na botě pak mapují cestu vaší nohy a přenáší tento pohyb do hry, kde děláte velmi realistický krok. Celou dobu je nutné být v přístroji v postroji 2.10, který drží hráče na stejné pozici nad diskem. Volnost pohybu je skoro neomezená a po chvilce si tělo zvykne na to, že je připevněno do stroje a kroky se začnou zdát být normální.

Zařízení od firmy Virtuix je kompatibilní s HTC Vive a Oculus Rift [1], dále je velká výhoda kompatibilita do aplikací a her, protože podobný pohyb je již do her vložen. Cena tohoto zařízení je 1250$ za kompletní vybavení.

2.2.7 Porovnání HTC Vive a Oculus Rift

Jak je uvedeno v článku [14], HTC a Oculus jsou přední výrobci headsetů. Oba tyto headsety jsou si velmi podobné v hardware specifikaci. Oba disponují OLED displejem o rozlišení 2160x1200 s obnovovací frekvencí 90Hz, 110 stupňové zorné pole, v dnešní době i oba disponují ovladači. Velký rozdíl je v pokrytí pohybové zóny kde HTC disponuje 4,5mx4,5m, kdežto Oculus jen 1,5mx1,5m při použití dvou senzorů a 2,5mx2,5m při použití tří senzorů. Cena a minimální hardwarové nároky mají velmi podobné.

Kapitola 3

Vývojové prostředí Unity 3D

Existují dva velcí hráči na tomto poli a to Unity 3d a Unreal Engine. V této práci je použit Unity 3D, a proto zde bude uvedeno lehké seznámení s tímto enginem a jeho hlavní rysy, abychom se na tyto informace poté mohli odkazovat v části návrhu a implementace.

Unity 3D je multiplatformní herní engine, který je velice populární pro své ovládání. Je zde možnost zaměřit se při programování na funkčnost hry a neřešit vykreslování, pokud ovšem defaultní vykreslování stačí. Unity obsahuje pracovní plochu, do které vkládáme objekty. S těmito objekty lze manipulovat (posun, rotace atd..). Dále je zde možnost přidávat jim vlastnosti (komponenty).

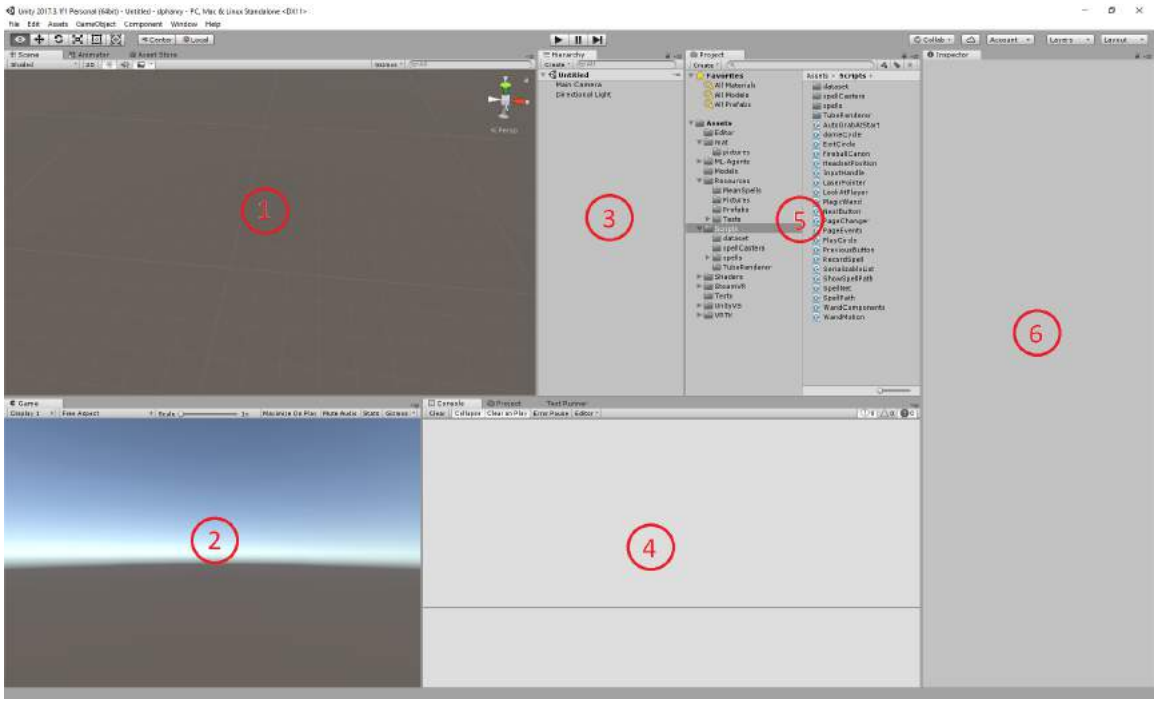
3.1 Rozvržení

Vývojové prostředí Unity umožňuje velmi jednoduchou a intuitivní práci. Obsahuje hlavní okno „Scene“ se scénou (3.1, část 1), kde lze vidět všechny objekty, které jsou přidány do scény. Lze je modifikovat pomocí posuvu, rotace a změny měřítka, také zde lze kliknout pravým tlačítkem myši a objekty vytvářet. Další část je okno „game“ (3.1, část 2), kde se zobrazí scéna z pohledu kamery hned po spuštění aplikace, zde lze různě hru ovládat např. kliknutím myší nebo pomocí klávesnice. Panel „Hierarchy“ (3.1, část 3) obsahuje všechny objekty a zobrazuje jejich hierarchický vztah, zde je možné tuto hierarchii měnit. Toto okno také slouží k rychlé orientaci mezi objekty a jejich hledání. Důležité informativní okno při spuštěné aplikaci je „Console“ (3.1, část 4), kde se vypisují veškeré logy, warningy a errory. Dále okno „Project“ (3.1, část 5), které zobrazuje všechny soubory v projektu. V projektu funguje hierarchie složek, která není jen z důvodu organizace, ale Unity přímo vyžaduje nebo umí využívat různé složky. Jako poslední je tu okno „Inspector“ (3.1, část 6), kde dochází k nastavování komponent u GameObjectu.

3.2 Komponenty

Jsou to atributy objektů a mohou nabývat různých typů např. funkce herního enginu a uživatelské skripty. Funkce herního enginu jsou skripty, které nabízí přímo Unity 3D a slouží např. ke kolizím objektů, jejich fyzikálním vlastnostem atd. Skripty psané uživatelem je možno psát v jazyce C nebo Java Script. V návrhu a implementaci si ukážeme, jak s těmito komponenty pracovat. Co vše je nutné pro to, aby aplikace fungovala.

Aby Unity docílila přesvědčivého fyzikálního chování, musí objekt ve hře správně zrychlovat a být ovlivňován kolizemi, gravitací a jinými silami. Jednotlivé zabudované fyzikální



Obrázek 3.1: Rozvržení Unity 3D, kde 1. Scene, 2. Game, 3. Hierarchy, 4. Console, 5. Project, 6. Inspector.

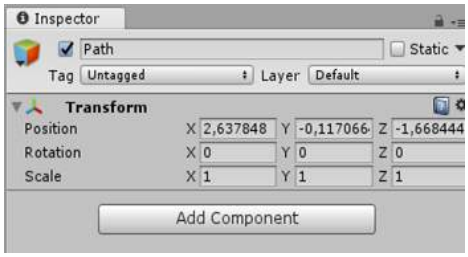
motory společnosti Unity poskytují komponenty, které pro vás zajišťují fyzikální simulaci. Pouze za pomocí několika komponentů a jejich nastavení parametrů můžete vytvořit objekty, které se chovají pasivně realistickým způsobem (tj. Budou přesunuty kolizemi a pády, ale samy se nebudou pohybovat). Řízením těchto komponent ze skriptů můžete dát objektu dynamiku vozidla, stroje nebo dokonce kusu látky. Unity obsahuje 3D a 2D fyziku a pro oba tyto typy jsou tu speciální komponenty.

Collider [18] je bounding box objektu, který se stará o kolize nebo jen dává informace o tom, jestli nějaké bounding boxy kolidují, a pokud ano, tak kde kolidují a s jakou silou. Existují různé typy jako je, Box Collider nejjednodušší bounding box tvaru kvádru, Sphere collider tvaru koule, Capsule collider tvaru kapsle a Mesh collider, který je nejsložitější, protože jeho bouding box je počítaný na jeho reálnou polygonovou podobu. Vždy pokud je možné se využívá co nejjednodušší Collider, protože výpočty kolizí jsou náročné a počítají se pro každý objekt s každým.

Rigidbody [19] je komponenta, která spolupracuje s Collider a to tak, že objektu dodává hmotnou podobu, která je schopná reagovat na gravitaci. Pokud je na objekt přiřazen Rogidbody a Collider objekt padá kvůli gravitaci, při nárazu má určitou velikost, váhu a podle toho akceleruje.

3.3 Objekty

Unity obsahuje velké množství objektů, které můžeme používat, aniž bychom je vytvářeli. Jednoduché modely, které můžeme jednoduše dát do scény, jako jsou kostka, koule, atd. můžeme jim samozřejmě dát materiál a měnit jejich velikost.



Obrázek 3.2: Prázdny GameObject.

Obrázek 3.3: GameObject s komponentou Light.

Další jsou „Effects“ objekty ty obsahují tyto typy: Particle system, Trail renderer a Line renderer. Particle system pro tvorbu kouře, ohňostroje, jisker a tak podobných. Efekt Trail renderer pro vykreslení předem dané stopy při nějaké události jako např. kolo ve smyku, štětec na papíře, atd. Line renderer pro vykreslování čar při zadání bodů. Všechny tyto efekty budou použity v aplikaci, protože velmi zjednodušují jejich vytváření.

Velké množství světelných objektů, které simulují různé světla jako např. všesměrné, bodové, plošné. Také budou použity pro některé efekty v této aplikaci, protože Unity přímo implementuje jejich fyziku a tedy je velmi jednoduché je používat.

Prefab objekty jsou meta objekty, které mají stejnou důležitost jako třídy v objektově orientovaném programování a to tak, že přidávají znovu použitelnost a jednoduchou úpravu na jednom místě. Unity tedy umožňuje vytvořit objekt a dát mu nastavené komponenty, tento objekt po té uložit jako Prefab, který si bude pamatovat všechny nastavení. Pokud tento objekt budeme chtít dvakrát ve scéně, použijeme tento Prefab pro tvorbu kopie. Poté pokud něco změníme u Prefabu, změna se projeví u všech instancí vytvořených na základě tohoto Prefabu.

3.4 Funkce objektů

Každý objekt, který je ve scéně musí být typu GameObject. Tato vlastnost dovoluje chovat se ke každému objektu stejně a předávat si je v parametrech. GameObject funguje jako jakýsi kontejner pro komponenty, které dodávají objektu funkčnost, ale sám o sobě toho moc neumí, např. světlo se vytvoří tak, že prázdnému GameObjet (3.2) přidáme komponentu Light (3.3). Skrz tuto třídu je ale možné přistupovat ke všem komponentám. Takže ze skriptu pomocí GameObject.GetComponent lze přistupovat ke komponentám jako je např. Light. Z obrázků 3.2 a 3.3 lze také vidět, že každý GameObject má komponentu Transform, protože každý objekt ve scéně musí mít svou polohu, rotaci a měřítko. Dále lze takto přidávat vlastní skript, který ovlivňuje chování objektu.

3.5 Hierarchie v objektech

Každý objekt ve scéně může mít hierarchický vztah k jinému objektu. Jsou zde dva vztahy child (potomek) a parent (rodič). Každý potomek může mít právě jednoho rodiče a každý rodič může mít neomezené množství potomků. Tyto vztahy se využívají z důvodu např. transformační závislosti, kdy změna velikosti nebo rotace rodiče ovlivní i potomka, dále posílání zpráv mezi všechny jeho potomky nebo pro přístup ke komponentům potomků. Tyto vlastnosti hierarchie urychlují proces vyhledávání a komunikace.

Důležitě je si uvědomit, že díky těmto závislostem je po té rotace potomka relativní a je spjata společně s rodičem. Tato nepříjemná vlastnost způsobuje problémy při implementaci a je nutné si na to dávat pozor.

3.6 Tagy, pojmenování a vrstvy

Tagy a pojmenování jsou velmi využívané vlastnosti v Unity aplikacích, ačkoli používání jmen není nejlepší volba, někdy může být ta správná. Jméno musí mít každý objekt, ale nemusí být jedinečné, tag musí mít také každý objekt a jedná se o identifikátor z pravidla separující více objektů stejných vlastností. Na základě těchto dvou identifikací jsme schopni tyto objekty dohledat, nebo zjistit o jaké se jedná. Toho lze využít např. při kolizi objektu, kdy lze zjistit jméno objektu, se kterým započala kolize, a na základě tohoto jména reagovat.

Vrstvy jsou více než identifikátory a z pravidla se nepoužívají běžně ve skriptu. Jejich nejdůležitější vlastnost je, že se podle nich řídí fyzika Unity. V nastavení lze nastavit, jestli na sebe budou dvě vrstvy reagovat, jen jednosměrně reagovat nebo nereagovat vůbec.

3.7 Podpora VR

Pro Unity 3D existuje více FrameWorků, které velmi zpříjemní práci s VR, protože obsahují základní prvky, jako jsou čtení všech senzoru zařízení VR a promítnutí je do Unity. Takže se nám pohyb hlavy v Unity jeví jako pohyb kamerou, pohyby ovladačů jako pohyb ovladačů v aplikaci společně s nasloucháním na stisk tlačítek a pohyb osoby v prostoru. Tyto prvky jsou stěžejní pro tvorbu jakékoli aplikace pro VR.

3.7.1 SteamVR Plugin

Jak vyplývá ze článku [6], toto SDK dovoluje vývojáři zaměřit se na vývoj programu, který bude fungovat se všemi majoritními druhy virtuální reality od „seated headset“, neboli headsetu, u kterého se jen sedí, do „room scale“ kde se s headsetem lze volně pohybovat po místnosti. Dále obsahuje programový přístup k součástím HTC Vive jako jsou ovladače, jejich model, poloha a interface pro všechny ovládací prvky na ovladači, jako jsou tlačítka a touchpad. Další součástí je zobrazování toho, co vidí uživatel v headsetu přímo v Unity Play mode, aby normální vývojářské okno zůstalo netknuto. Tento plugin do Unity je velmi důležitý, protože bez něj nelze vyvíjet pro HTC Vive.

Dále tento plugin obsahuje prefaby pro snadnou implementaci do aplikace. Tyto prefaby zahrnují již zmíněné modely ovladačů a jejich nastavení a reagování na pohyb a stisknutí. Dále obsahuje kameru, která je napojená na headset. Takže inicializace aplikace je velmi jednoduchá po použití těchto prefabů.

3.7.2 VRTK - Virtual Reality Toolkit

Tento toolkit dovoluje abstrahovat zařízení VR [7], takže lze jeden program spustit na HTC Vive, Oculus Rift a další. Samozřejmě pokud ale do aplikace přidáme vlastní skripty, které budou závislé na nějakou platformu, tento toolkit nám stačit nebude a budeme muset skript psát pro každou platformu zvlášť. Dále obsahuje skripty, které zahrnují základní práci s VR, jako jsou:

- Pohyb ve virtuálním světě
- Interakce jako jsou dotýkání, uchopování a používání předmět
- Interakce s Unity3D UI elemety jako jsou např. tlačítka
- 2D a 3D kontrolní prvky jako jsou páky a tlačítka

Funguje tak, že umí pracovat s různými pluginy, jako je SteamVR a při spuštění lze nastavit, jaký se bude aktuálně spouštět. Jeho další výhoda je že umí simulovat VR přímo v unity, takže existuje velmi omezená možnost, jak si vyzkoušet aplikaci bez VR zařízení. Bohužel neobsahuje pokročilou simulaci ovladačů, což je pro mou práci hlavní.

3.7.3 Překlad na různá zařízení

Díky open source Software Mono, který je založený na platformě .NET Framework dovoluje, aby byl program spustitelný na jiných platformách díky tomu, že přeložení C# do NET bytecode. Když pak Unity sestavuje aplikaci, tak vkládá NET bytecode interpretovaný v nativním kódu, založeném na Mono. Při spuštění aplikace se pak spouští tento bytecode. Díky této vlastnosti společně s VRTK je možné opravdu psát aplikace, které lze spustit prakticky všude.

3.7.4 Formát modelů

Každý program pro tvorbu modelů má svůj vlastní formát, do kterého zvládne uložit všechny data, jako jsou body, normály, uvw souřadnice, textury a spoustu dalšího. Unity 3D ale tyto formáty nezvládne, takže je nutno exportovat z těchto programů do jiných formátů jako jsou např. fbx a obj. Oba tyto formáty podporují export bodů, uvw map a normál, ale neobsahují textury a materiály. Takže je nutné exportovat model a po té na něj aplikovat materiály přímo v unity. Tento postup je obecně používaný ale existuje i alternativa.

Novější verze Unity 3D podporují i nativní formáty některých programů jako jsou Photoshop, 3Ds Max, Blender, atd., ale je nutné je mít nainstalovány všude, kde se s těmito objekty pracuje. Takže pokud se programátor rozhodne využívat tyto formáty, tak při jakékoli manipulaci s těmito objekty na jakémkoli počítači musí mít nainstalovány tyto programy.

Kapitola 4

Umělé Neuronové sítě

Tento výpočetní model volně a zjednodušeně napodobuje chování biologického mozku. Je to tedy systém, který se postupně učí daný úkol a snaží se v něm zlepšovat. Historie těchto sítí sahá až do poloviny 20. století, kdy docházelo k prvním pokusům. Vždy byly omezovány kvůli výpočetní síle, která je velmi důležitý pro tento typ algoritmů. Proto je teď v posledních letech tak obrovský zájem o tyto technologie díky vzniku dostupného výkonného hardware, který zvládne obrovské množství operací. V této kapitole si lehce nastíníme základní prvky konvolučních neuronových sítí.

4.1 Struktura

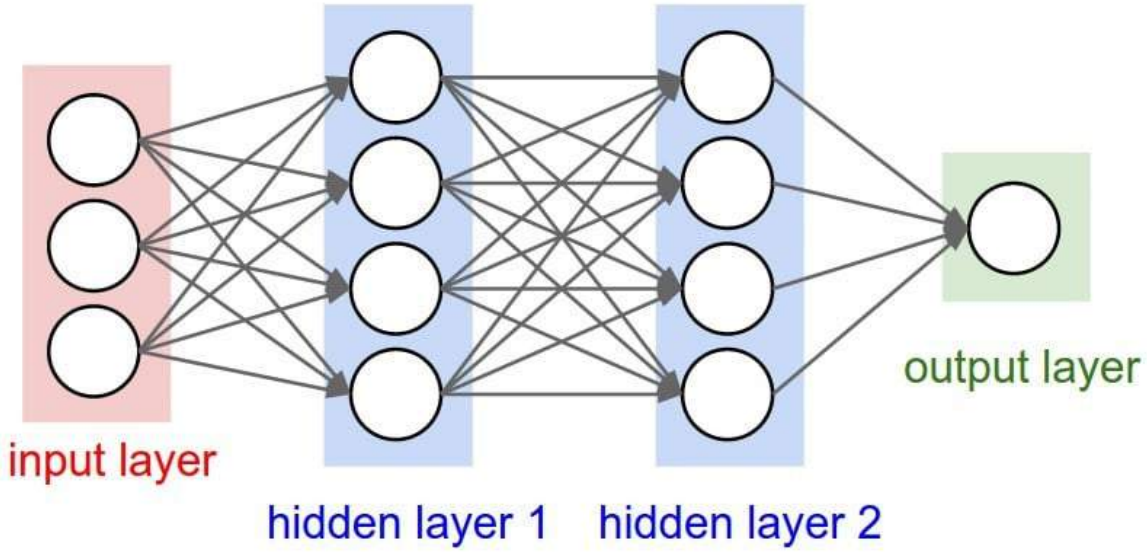
Jak je řečeno v [17], základem těchto sítí je stejně jako v biologickém mozku neuron, ale má jen jeden vstup a libovolný počet výstupů. Dále jsou tu synapse, což jsou spoje mezi jednotlivými neurony a každá synapse má danou váhu. Tyto tři věci lze napsat jako funkci popsanou rovnicí 4.1, kde x_i je vstup neuronu, ω_i je váha synapse, Θ je práh a označuje prahovou hodnotu aktivace neuronu, S(x) je přenosová funkce neuronu a Y je vstup neuronu. Nejdůležitější jsou tedy váhy synapsí, které určují zkušenost neuronu a čím je hodnota vyšší, tak je jeho vstup důležitější.

$$Y(x) = S\left(\sum_{i=1}^{N}(x_i\omega_i)\right) \tag{4.1}$$

Tyto neurony se po té seskupují do vrstev sítě, kde každý výstup z vrstvy je vstup do další [17]. Takovým způsobem se vytvoří celá neuronová síť, jako je na obrázku 4.1. Existují různé druhy vrstev a v případě konvolučních sítí jsou to konvoluční vrstvy, kde se za pomocí konvoluce projíždí vstupní struktura a výstupy se dávají jako vstupy do další sítě. Dále vrstva plně propojená, která se dává jako konečná a určuje výslednou hodnotu sítě.

4.2 Učení

Jak je popsáno v [10], učením se myslí nastavení vah celé sítě, tato operace je časově velmi náročná, vyžaduje co možná největší počet dat. Existují dva druhy učení, první učení s učitelem, které bude použita v této práci a druhé bez učitele. V prvním případě jde o to, že známe výsledek, takže síti jsme schopni říct, jestli rozhodla dobře, nebo ne. Je tedy nutné mít datovou sadu a k ní výsledky. Učení probíhá typicky pomocí algoritmu



Obrázek 4.1: Jednoduchá struktura sítě, která se skládá ze vstupní vrstvy, dvou plně propojených vrstev a výstupní vrstvy. Převzato z [3].

Backpropagation, který pracuje tak, že vyhodnocené řešení se porovná s očekávaným a tím se zjistí, o kolik se neuronová síť „spletla“. Zpětně se na základě toho vypočítává, o kolik se mají změnit váhy neuronů, aby se tato odchylka od správného řešení co nejvíce snížila.

Pro učení se vytváří dvě datové sady jedna trénovací, která je co možná největší a druhá testovací, kde jsou prvky vybrány tak, aby co možná nejlépe pokryli veškerou variabilitu problému.

4.3 Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou velmi podobné běžným neuronovým sítím, jsou také tvořeny neurony, které mají naučitelné váhy. Každý neuron dostane nějaké vstupy, provede operaci a použije nelinearitu. Stále platí všechno, co bylo vyvinuto pro učení normálních neuronových sítí. Tyto sítě jsou specifické tím, že mají konvoluční vrstvy, kdežto normální neuronové sítě mají jen plně propojené. Tímto způsobem je možné ušetřit zdroje při výpočtu, protože nejsou všechny neurony propojeny se všemi, ale jsou propojeny jen s malou částí předchozí vrstvy. Další specifikací je, že jako vstup je typicky obraz tedy datová struktura, kde existují závislosti na okolí bodu. Není tedy nutné, aby tato struktura byla obrázek, jen musí mít podobné vlastnosti. Lze tedy použít zvukovou stopu, mračna bodu atd. Pro tento projekt je toto velmi výhodné, protože obrázek se skládá z bodů a každý bod má typicky tři barvy. V našem případě máme taky body a ty jsou složeny z x, y, z hodnot souřadného systému.

V této práci nevyužijeme všechny typy vrstev, které jsou dostupné jako pooling (zmenšení obrazu), dekonvoluce, atd., ale využijeme jen konvoluční vrstvu a reshape.

4.4 TensorFlow

TensorFlow [15] je systém strojového učení, který je robustní a dokáže fungovat v různorodých odvětvích. Používá dataflow graf (což je standartní programovací model pro paralelní

výpočty), který reprezentuje výpočet, sdílený stav a operace, které tento stav mění. Mapuje uzly grafu datového toku na mnoha strojích v clusteru a uvnitř stroje napříč různými výpočetními zařízeními, včetně vícejádrových procesorů, univerzálních grafických jednotek (GPU) a vlastních ASIC, známých jako jednotky zpracování tenzoru (TPU). Tato architektura poskytuje vývojáři aplikace flexibilitu. Zatímco v předchozích návrzích "parametrického serveru"je do systému zabudováno řízení sdíleného stavu, TensorFlow umožňuje vývojářům experimentovat s novými optimalizacemi a trénovacími algoritmy. Dále podporuje celou řadu aplikací se zaměřením na trénink a inference na hlubokých neuronových sítích. Několik služeb od společnosti Google používá TensorFlow ve výrobě. Tento software byl vydán jako open-source projekt a stal se široce používán pro výzkum strojního učení.

Pro tento projekt je vybrán tento nástroj z důvodu jeho referencí, a díky tomu, že se velmi dobře dokáže přizpůsobit zadání. Pracuje s programovacím jazykem Python, který bude použit i v tomto projektu.

Kapitola 5

Návrh

V této kapitole popíšeme návrh aplikace, která by měla ukázat, čeho je schopná VR. Cíl aplikace je rychlá ukázka pro všechny, kteří nepřišli do styku s touto technologii a aby si ji zamilovali během prvních pár minut. Téma aplikace je kouzlení jako např. v Harry Potterovi za použití hůlky. Takže aplikace bude splňovat příjemné, rychlé, intuitivní ovládání a menu, aby si uživatel myslel, že to není demo, ale aby měl pocit, že se právě nachází ve světě čar a kouzel.

Hra při spuštění bude v místnosti menu, ze které budeme moci hru spustit anebo vypnout. Po spuštění hry bude základní provedení hrou, aby uživatel hned věděl, co a jak dělat a jaké má možnosti. Hned po tomto základním výcviku bude následovat menší mapa s arénou, kde hráč bude muset použít kouzla k tomu, aby dosáhl co největšího skóre. Nepřátelé zde budou vyobrazeni jako děla střílející kouzlené ohnivé koule. Hráč bude mít za úkol se jim co nejdéle vyhýbat a používat obranné kouzla.

5.1 Menu

Díky tomu, že VR je nová technologie ještě nejsou úplně definované a otestované způsoby, jak věci dělat. Takže se hodně experimentuje a vývojáři se snaží o to, jak novou dimenzi využít. Zde bude seznámení s přístupy, které se využívají a také návrh menu v naší aplikaci.

5.1.1 Způsoby menu ve VR

Existují tři přístupy, jak udělat menu ve VR. Jedná se o skeuomorfní menu, flat menu s prvky geometrie a opravdové 3D rozhraní.

Ideou skeuomorfního menu [8] je modelovat to, co mají představovat v reálném světě, např. pokud vytváříme prohlížení fotek, můžeme to modelovat jako procházení reálného alba fotek. Pro uživatele to potom znamená, že vybere album a v něm si bude listovat. Toto grafické zpracování uživateli co možná nejvíce umožní vžít se do role.

Flat menu [8] jako je na obrázku 5.1 staví na rozhraních, které známe z počítačové a mobilní grafiky. Staví na základních blocích, jako jsou kachle nebo karty a hierarchické strukturování. Aplikace na mobilních zařízeních využívají často uspořádané kachle, které je možné seskupovat do skupin, nebo složek pro vytvoření hierarchie. Takto jednoduché rozhraní se stalo velmi populárním, a proto se dostává i do VR, kde lze tyto prvky jednoduše graficky znázornit jako grafické tělesa, které jsou zobrazeny na kulové, nebo válcové těleso kolem uživatele.

Obrázek 5.1: Takto vypadá Flat menu, které se velmi často používá jako jednoduchá alternativa menu s reálným 3D rozhraním.

Reálné 3D rozhraní [8] je zatím velmi vzácné, i přestože jsou asi nejvíce vhodné pro VR. Poblémem však je, že pro flat menu máme roky testování a frameworků, kdežto pro tento druh menu toho zatím moc není. Hlavní výhodou oproti flat designu je ten, že další přidaná dimenze umožňuje zobrazit více informací a jako lidé jsme zvyklí žít ve 3D světě a interagovat s 3D objekty, takže tato varianta je nám nejbližší. Tento typ je tedy asi nejlepší a také nejsložitější jak ze strany implementace, tak ze strany grafické a návrhové.

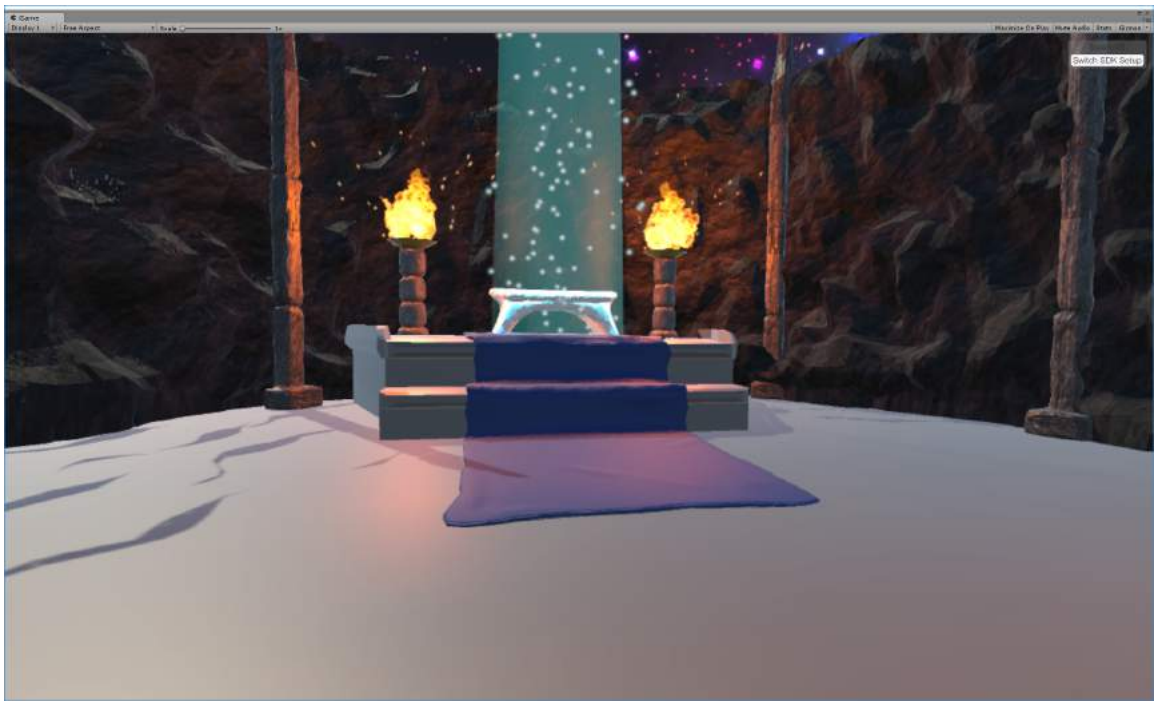
5.1.2 Menu v aplikaci

Menu bude reálné 3D rozhraní obrázek 5.2, kdy při pohledu hned hráč musí vědět, co dělat. Hráč se po zapnutí hry objeví v historické místnosti, kde budou na zemi dva magické znaky podobné jako na obrázku 5.3, které při vstoupení (teleportací pomocí pohybu) hráče teleportují do dané lokace. Tyto kruhy budou vyzařovat lehké světlo, aby působili magickým dojem, a bude v nich a nad nimi zapsáno slovo, které charakterizuje, co daný kruh dělá. První kruh bude mít nápis „Play“ a hráč se přes něj dostane přímo do hry, druhý „Exit“, který ukončí hru.

Díky tomuto způsobu implementace by si hráč neměl připadat že je v menu, ale že je již ve hře pro navození lepší atmosféry, která je ve VR velmi důležitá.

5.2 Hra

Hra začíná při vstupu na kruh „Play“, který hráče teleportuje do místnosti, kde se odehrává tutoriál. Tato scéna hráče naučí vše potřebné o kouzlení a mechanikách hry pro boj v další části hry. Po dokončení tutoriálu hráč bude teleportován do světa, kde již budou protivníci.



Obrázek 5.2: Ukázka menu ve hře a na schodech je po té portál do části tutoriálu.

Obrázek 5.3: **Vlevo** obrázek znázorňující spuštění hry, nebo přechod do jiné místnosti. **Vpravo** pro odchod nebo ukončení aplikace.

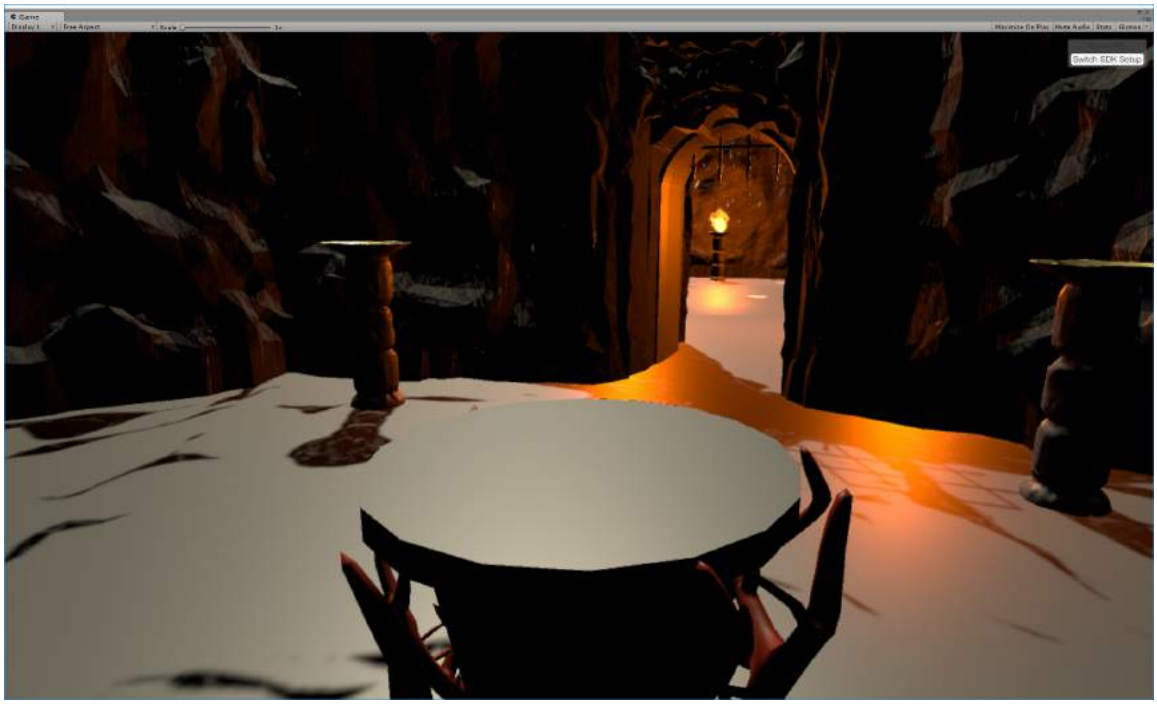
Obrázek 5.4: Ukázka turoriálu, kde lze vidět stůl s hůlkou, jak jsou nasvíceny, aby hráč věděl, že tuto hůlku má vzít.

5.2.1 Tutoriál

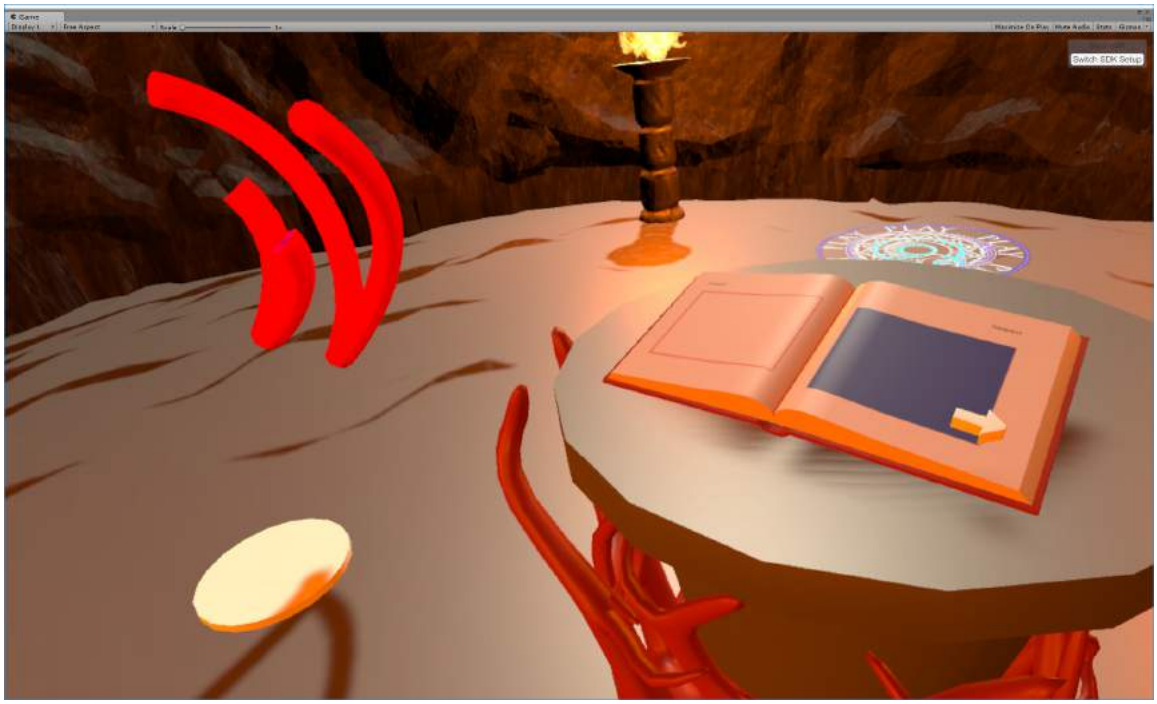
Tato scéna hráče připraví na to, co ho čeká ve hře rychlou a intuitivní cestou. Hráč se ocitne v kulaté místnosti s kruhem „Exit“ pro navrácení do menu, jedním východem do druhé místnosti a stolem umístěným uprostřed, na kterém bude ležet hůlka, jak lze vidět na obrázku 5.4. Stůl bude osvětlen, aby hráč věděl, že je důležité, aby ke stolu přišel a vzal si tuto hůlku. Po uchopení hůlky se světlo v místnosti ztlumí a rozsvítí se druhá místnost, která lze vidět průchodem 5.5. Ve druhé místnosti bude zase kulatý stůl a na něm bude kniha. Tato kniha bude mít na listech znázorněna kouzla společně s krátkým popisem a bude se v ní dát listovat 5.6. Na každý list půjde hůlkou reagovat (kliknout) a cesta tohoto kouzla se objeví hned vedle stolu společně s místem, kam si má uživatel stoupnout. Cesta bude vytvořena na základě průměru všech nahraných dat v datové sadě, díky tomu získáme nejlepší tvar gesta. Model gesta bude tvořen trubicí vyplňující se rychlostí, jak hráč bude pohybovat hůlkou, aby kouzlo provedl úspěšně. Takto se hráč naučí základní kouzla. Bude zde i kruh „Play“, který hráče přemístí do Arény.

5.2.2 Aréna

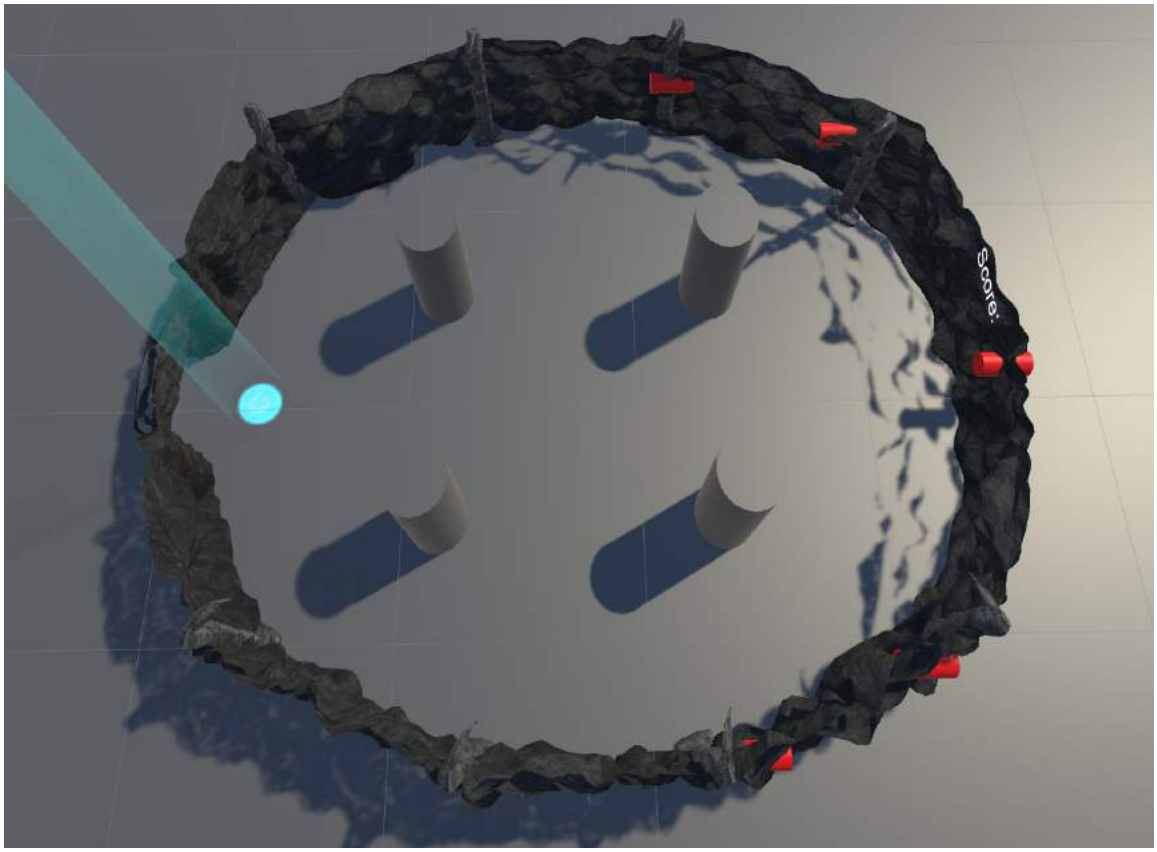
Tato scéna bude obsahovat jednu velkou kruhovou místnost se čtyřmi sloupy jako na obrázku 5.7 a půjde v ni o to, aby zde hráč vydržel co nejdéle. Bude zde také vidět text se skóre, které se bude každou vteřinu aktualizovat. Na okrajích místnosti budou kanóny, které budou střílet na hráče ohnivé koule, kterým se bude hráč muset vyhnout nebo se ubránit pomocí kouzel. Tyto kanóny budou vždy otočeny na hráče a ohnivé koule budou létat přímo na něj. Bude zde možnost pomocí ohnivé koule tyto kanóny na chvíli vyřadit z provozu. Pokud budou neaktivní, budou mít černou barvu a aktivní budou červené.



Obrázek 5.5: Ukázka turoriálu, po uchopení hůlky místnost ztmavla a druhá místnost se rozsvítila.



Obrázek 5.6: Ukázka knihy kouzel, ve které lze listovat pomocí tlačítek a po najetí hůlky do stránky se stránka zazelená a po kliknutí se objeví kouzlo (to jsou ty dva červené válce, nalevo obrázku).



Obrázek 5.7: Půdorys arény, kde je 5 kanónů, 4 sloupy, za které se může hráč schovat a napravo lze vidět počitadlo skóre.

Hráč bude mít na začátku hry určitý počet životů a pokaždé když bude zasažen ohnivou koulí, ztratí jeden. Pokud hráči dojdou životy, tak se objeví nápis "Final Score"a kanóny se stanou neaktivními. Hráč se bude moct vyhýbat a ukrývat za sloupy, bude mít možnost vykouzlit ledovou stěnu nebo štít, aby se mohl účinně bránit jak le vidět na obrázku 5.8. Ledová stěna bude moci pohltit určitý počet ohnivých koulí stejně jako štít.

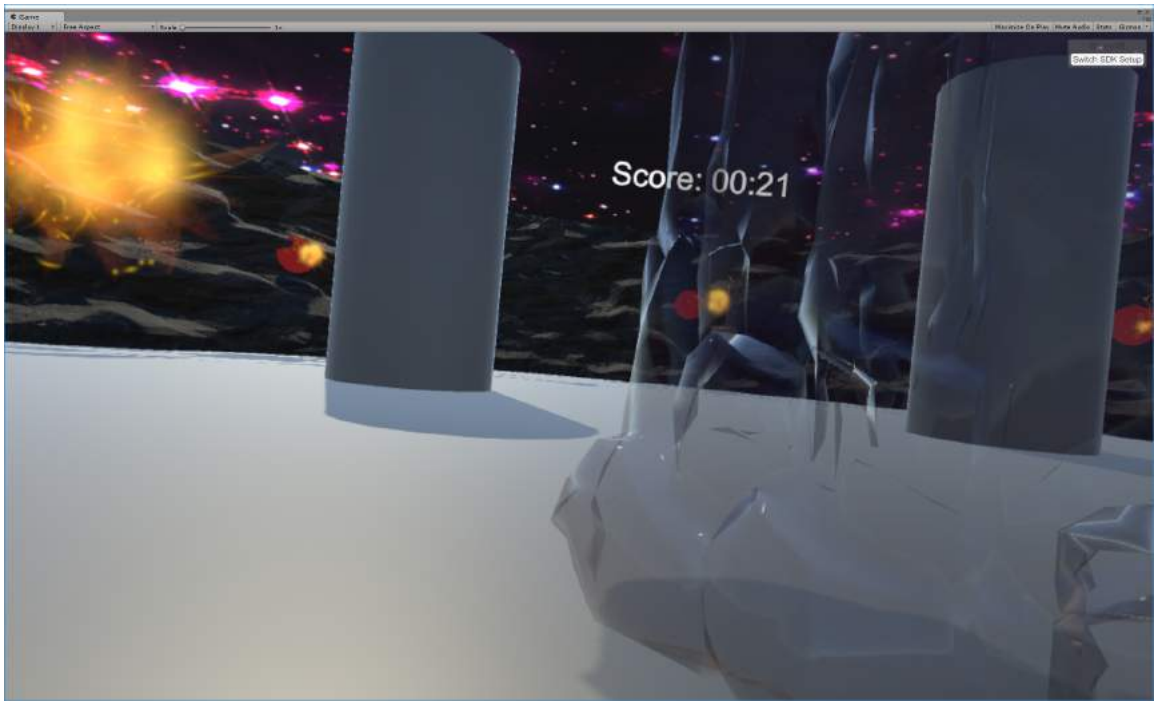
Opustit místnost půjde kruhem Exit, který hráče vrátí do tutoriálu.

5.3 Kouzlení

Stěžejní část hry je kouzlení hůlkou, protože to bude skoro jediná možnost, jak interagovat s okolním světem. Zde je dobré si představit film Harry Potter, ze kterého budeme volně čerpat některé principy, jako např. kouzla budou vyvolané pohybem hůlky neboli ruky v prostoru. Bude zde tedy nový a specifický způsob, jak zaznamenávat a rozpoznávat gesta.

Kouzlo budeme v této hře chápat jako pohyb hůlky, kterou hráč drží v ruce, v prostoru, který bude snímán, vyhodnocován a pokud bude vyhovující, tak se kouzlo aktivuje. Pokud hráč bude chtít kouzlit, zmáčkne tlačítko na ovladači a pohyb hůlky se začne snímat, tato cesta se vloží na vstup konvoluční neuronové síti a ta rozhodne, o jaké kouzlo jde. Síť bude co nejjednodušší, aby fungovala velmi rychle.

V této hře budou tři kouzla a to ohnivá koule, ledová zeď a štít na ruku. Každé z těchto kouzel bude muset mít svůj vlastní efekt, vlastní jedinečné gesto a datovou sadu.



Obrázek 5.8: Takto bude vypadat aréna z pohledu hráče, letící ohnivá koule na levé straně obrázku a vyčarovaná ledová stěna jako obrana.

5.3.1 Tvorba datové sady

Díky tomu, že zde bude využita neuronová síť, tak je nutné vytvořit datovou sadu. Toto zadání je totiž velmi specifické a není možné použít cizí datovou sadu gest.

Tvorba datové sady bude vyžadovat dvě scény, v první se bude pracovat ve VR, nastaví se jméno kouzla a spustí se ukládání. Po té se musí vykonat potřebný počet opakování pohybu kouzla v různých směrech, velikostech a rychlostech, aby se pak síť dostatečně dobře naučila rozpoznávat toto kouzlo. Druhá scéna nebude vyžadovat VR a bude zobrazovat nahraná data, nebo generovat data pro učení a testování.

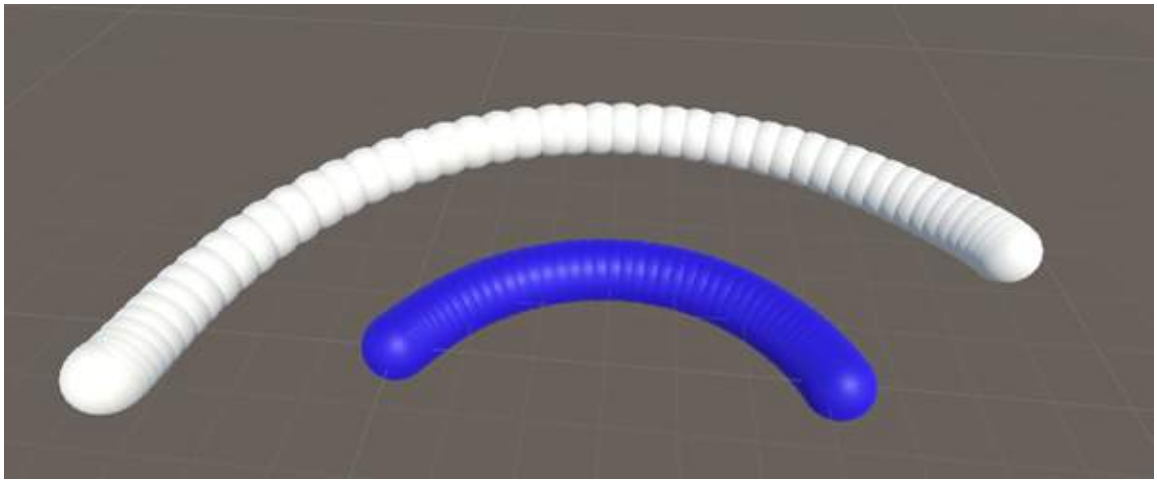
Datová sada bude tvořena takto:

- Nahrajeme gesto
- Normalizujeme jeho data
- Vytvoříme pomocí průměru další data
- Tyto data použijeme jako trénovací data pro síť
- Originální normalizované data použijeme jako testovací

Každé dato datové sady se bude skládat ze dvou polí pro jednoduchý přístup k hodnotám. První pole bude obsahovat body, které opisuje vrchní část hůlky a druhé pole bude spodní strana hůlky, jak lze vidět na obrázku 5.9. Tato struktura musí být serializovatelná, aby se dala uložit na disk. Toto ukládání bude probíhat v Unity a to do formátu json.

Normalizace Dat

Data, která bychom jen nahráli a vytvořili z nich datovou sadu, budou mít velmi špatné výsledky, protože bude teoreticky nekonečné množství variací, jako je názorně ukázané na



Obrázek 5.9: Takto podobně vypadají všechny nahrané gesta pro ohnivou kouli. Modrá je spodní část hůlky a bílá je vrchní část.

obrázku 5.10. Všechna tato gesta jsou stejná kouzla zachycena ze stejného místa a to ohnivá koule, ale jak si můžete všimnout, jsou velmi odlišná. Musí se zde tedy aplikovat normalizace a co největším způsobem tuto variaci zmenšit.

Normalizace tedy bude pracovat ve třech krocích. První krok bude posouvat kouzlo do středu souřadného systému za pomocí odečtení prvního bodu od všech bodů v obou listech. Takto dostaneme cesty, jako jsou na obrázku 5.11. Stále je ale variabilita veliká a to kvůli rotaci kolem každé osy o 360 stupňů.

Ve druhém kroku je tedy nutné změnit rotaci všech, aby byly stejné. Toto vyřešíme tak, že budeme hledat rotaci na ose y, abychom minimalizovali tuto variaci. Toho docílíme tak, že si vypočteme, jakou rotaci potřebujeme, aby měl poslední bod souřadnici x = 0, pomocí pravoúhlého trojúhelníku a rovnice 5.1.

$$radY = arctg\left(\frac{abs(z)}{abs(x)}\right) \cdot \frac{\pi}{180} \tag{5.1}$$

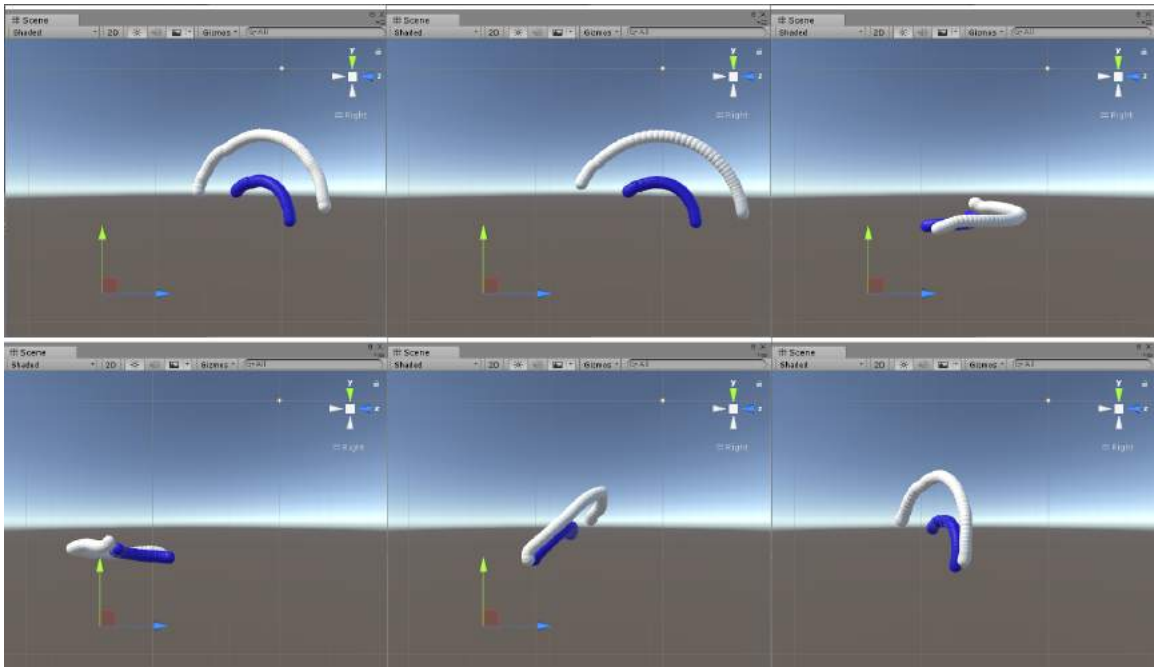
Takto získáme úhel v radiánech podle toho, o kolik otočíme všechny body, kolem osy Y. Otočení probíhá aplikováním rovnic 5.2a a 5.2b pro každý bod gesta, díky kterým získáme nové souřadnice x a z.

$$newX = x \cdot cos(radY) + z \cdot sin(radY) \tag{5.2a}$$
$$newZ = z \cdot cos(radY) - x \cdot sin(radY) \tag{5.2b}$$

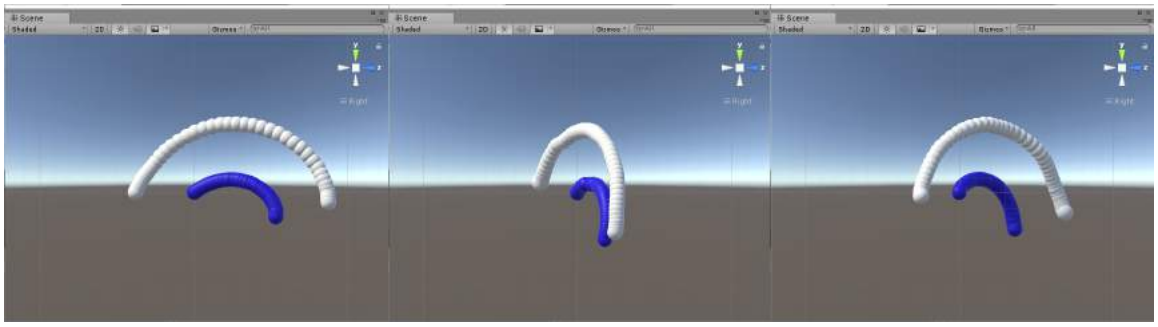
Tento proces aplikujeme i na osu Z, použijeme rovnici 5.3 pro získání úhlu a rovnice 5.4a, 5.4b pro získání nových souřadnic x a y.

$$radZ = arctg\left(\frac{abs(y)}{abs(x)}\right) * \frac{\pi}{180} \tag{5.3}$$

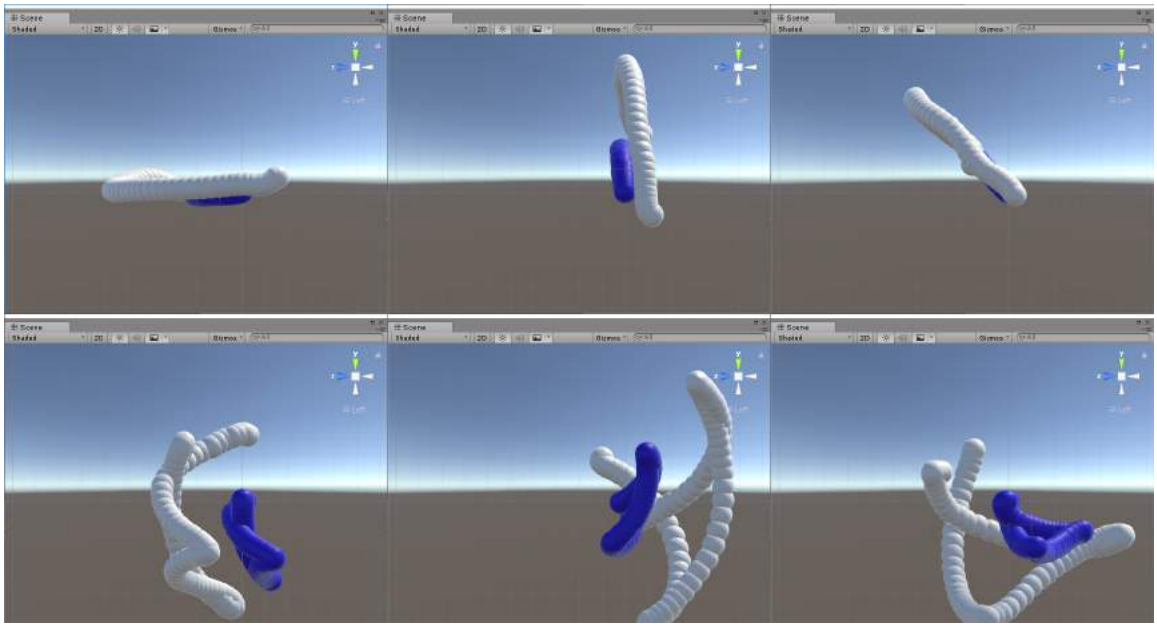
$$newX = x \cdot cos(radZ) - y \cdot sin(radZ) \tag{5.4a}$$
$$newY = y \cdot cos(radZ) + x \cdot sin(radZ) \tag{5.4b}$$



Obrázek 5.10: Ukázka dat jak vypadají před normalizací. Všechny jsou jinak natočeny a umístěny v prostoru.



Obrázek 5.11: Takto vypadají data, které začínají v počátku souřadného systému.



Obrázek 5.12: Takto vypadají data, které mají první a poslední bod zarovnaný na ose x (horní je gesto ohnivé koule, spodní štítu, na kterém je zde lépe vidět toto zarovnání).

Takto budou všechna data mít první bod v počátku souřadného systému a poslední bod ležící na ose X. Stále ale tato data obsahují variaci, znázorněnou na obrázku 5.12, v rotaci kolem osy X, kterou nelze odstranit jako předchozí dvě. Poslední bod je již podle X zarovnán, ale ostatní nejsou, nelze ani použít jiný jeden bod gesta pro otočení, protože by na tomto bodu velmi záleželo, tedy pokud bychom uprostřed gesta udělali v tomto místě lehkou změnu, otočení by bylo velké. Z tohoto důvodu bude použito otočení podle těžiště. Těžiště se spočítá pomocí rovnice 5.5 a dále použijeme stejnou techniku jako u předchozích dvou otáčení.

$$X = \frac{\sum_{i=0}^{N}(x_n)}{N} \qquad Y = \frac{\sum_{i=0}^{N}(y_n)}{N} \qquad Z = \frac{\sum_{i=0}^{N}(z_n)}{N} \tag{5.5}$$

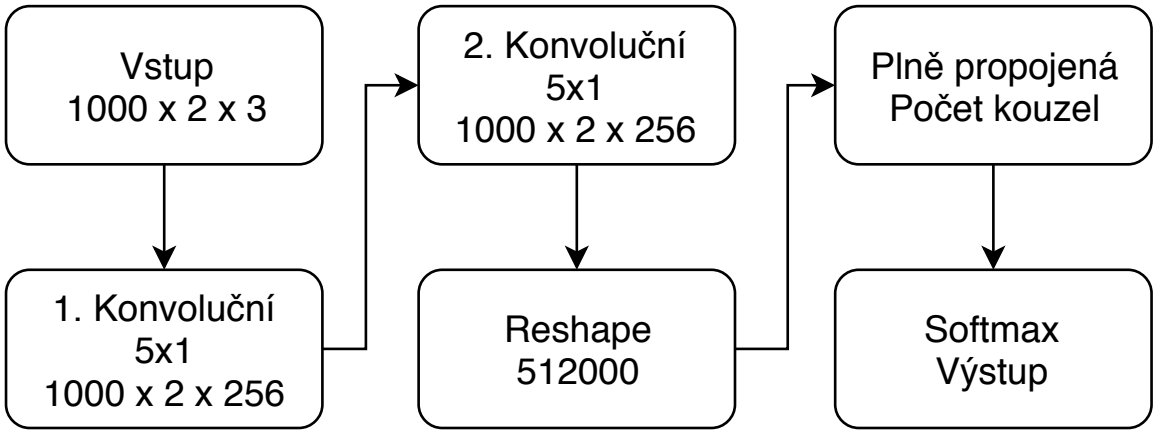
kde N je počet bodů v gestu, x_n je hodnota x bodu n a obdobně pro y a z.

Generování dat

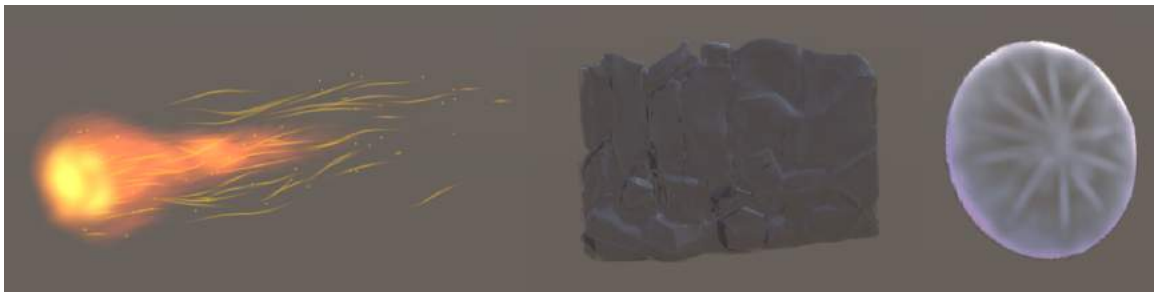
Samozřejmě není v mých silách, abych dokázal sám nahrát dostatek dat pro naučení sítě (z mých zjištění a testování je zapotřebí tisíce dat), takže data musejí být částečně generována. Zde bude zapotřebí vytvořit takové prostředí, které vezme originální gesto, které jsem nahrál a lehce ho pozmění. Tato operace bude vytvořena tak, že vezmu gesto a k němu jiné náhodné, ne stejné, gesto vytvoří z nich průměr a uloží. Takto pro každé gesto, které bude nahráno, se vytvoří x dalších, které jsou jiné.

5.3.2 Detekce kouzla

Síť bude mít čtyři vrstvy, jak lze vidět na obrázku 5.13, kde vstup bude mít velikost 1000 x 2 x 3 což znamená, že velikost kouzla bude mít maximálně 1000 bodu. Konvoluční sítě nemají velkou podporu v proměnlivém vstupu, takže bylo nutno vymyslet nějaké omezení. Body se budou seskupovat vždy na začátek, takže vzniknou nepoužité body na konci a ty budou mít hodnotu 0. Takto se síť naučí tyto 0 ignorovat, protože z nich nebude žádná informace,



Obrázek 5.13: Struktura sítě, kde lze vidět vstupní data, které jdou do 1. konvoluční vrstvy po té do 2. konvoluční vrstvy, následně do vrstvy reshape pro změnu tvaru a jako poslední plně propojená vrstva a softmax pro normalizaci.



Obrázek 5.14: Ukázka jak vypadají kouzla, zleva ohnivá koule, ledová stěna a štít.

ale bude se soustředit na vyplněné body a jejich počet. Tedy jako vstup budou dvě pole o velikosti 1000 bodů, každý bod bude složen z 3 čísel a to jsou hodnoty v kartézském souřadném systému x, y, z. Tento vstup se vloží do první konvoluční vrstvy, která má 256 filtrů. Výstup této vrstvy tedy bude 1000 x 2 x 256 a vloží se na vstup druhé konvoluční vrstvy. Její výstup bude stejný jako u první vrstvy. Předposlední vrstva reshape jen pozmění uspořádání bodů, aby lépe vyhovovali plně propojené vrstvě a to tak, že vezme 2000 bodů a jeho 256 hodnot filtrů a vloží ho do jednoho pole. Takto vznikne pole o velikosti 512000 bodů a vloží se do poslední vrstvy. Tato vrstva rozhodne o tom, o jaké kouzlo se jedná a jako výstup bude mít pole o počtu kouzel plus „nekouzlo“. Jako poslední se tento výstup převede na hodnoty 0-1 pomocí softmax. Na konci sítě tedy máme pole, kde jsou jednotlivé hodnoty, jak moc si je síť jistá tím, že jde o dané kouzlo. Na základě tohoto výsledku se pak podle určitého prahu rozhodne, o jaké kouzlo jde.

5.3.3 Modely, efekty a reakce

Každé kouzlo musí mít jedinečný, charakterizující a rozpoznatelný efekt, aby hráč hned poznal, co vykouzlil a případně, co vykouzlil protivník, aby se mohl účinně bránit. Efekty budou dělány především jiným objektem kouzla, ale také pomocí světla, případně zvukem. Také bude mít každé kouzlo své parametry, jako je jeho útočná nebo obranná síla, rychlost letu, délka trvání, atd. Na obrázku 5.14 lze vidět, jak tato kouzla vypadají.

5.4 Modelování objektů

Abych se vyhnul právnímu porušení nějakých licenčních podmínek, budu se snažit o to, modely si vytvářet sám. Aby vše vypadalo co možná nejreálněji a působilo to dobrou atmosféru, bude toto velká část projektu, ale myslím si, že je tato část stejně důležitá jako programová část.

5.4.1 Programy

Modelovat budu v programu 3DS Max, protože již mám jeho základní znalost. Jedná se o velmi kvalitní nástroj pro 3D modelování. Tento program vyvíjí společnost Autodesk Media and Entertainment již od roku 1988 kdy vyšel první prototyp. Hra tedy bude obsahovat modely a jejich textury z tohoto programu.

Pro sculpting, neboli sochaření, bude využit program Blender, což je open-source software, protože má pro tento způsob modelování daleko lepší nástroje.

5.4.2 Ostatní

Modely musí splňovat co nejmenší počet polygonů, aby byla zátěž na grafickou kartu co možná nejmenší. Tvorba „low-poly“ objektů není jednoduchá a bude složité vytvářet textury pro tyto modely z důvodu, že nemají všechny vlastnosti vymodelovány. Díky tomu, že scény budou obsahovat dost světelných efektů, je těžké odhadnout kolik je maximální počet polygonů na scénu.

5.5 Ovládání

Ovládání hry bude velmi jednoduché a co nejvíce podobné ostatním aplikacím. Pohyb v prostoru bude možný pomocí dvou způsobů a to jak pohyb osoby v reálném prostoru, tak také teleport, na jaký jsme zvyklí pomocí stisku trackpadu ovladače a míření na místo, kam chceme. Pro uchopení nebo interakci s objekty budeme používat buďto dotek, nebo dotek a stisk tlačítka. Pro kouzlení již zmíněný pohyb hůlky v prostoru.

5.5.1 Možnost pohybu ve VR

Pohyb ve VR je velmi náročná operace oproti normálním 2D/3D hrám na PC a to z důvodu, že špatně naprogramovaný pohyb může u některých hráčů způsobit nevolnost nebo závratě, proto tato část u VR aplikace vyžaduje správný postup a pečlivost.

Nevolnost ve VR aplikacích není nic neobvyklého a je způsobena primárně díky pohybu hráče ve virtuálním světě, přičemž jeho reálné tělo se nehýbe a je stacionární. Existuje tedy dobré pravidlo, že se nikdy nehýbe s kamerou bez pohybu uživatele, protože pokud se kamera pohne a hráč stojí, jeho oči zaznamenají pohyb, kdežto jeho uši říkají, že stojí. Tento rozpor ve smyslech vyvolá v těle pocit, že bylo otráveno, a proto je zde nevolnost jako ukazatel. Z tohoto důvodu jsou velmi populární hry se „seated VR“, při které hráč jen sedí a ve hře je stacionární a jen se otáčí. [5]

Existuje pár přístupů, jak se pohybovat ve VR, jedním z nich je právě používání klávesnice nebo gamepadu, pro pohyb hráče pomocí tlačítek. Tento typ pohybu ale způsobuje nevolnost a je dobré se mu vyhnout. Pokud ale nastane situace, kdy je tento pohyb zapotřebí, je dobré vytvořit něco statického, co uživatel uvidí po celou dobu pohybu, něco jako kokpit letadla. [5]

Nejvíce populární metoda je pomocí instantního přesunu pomocí teleportace. Nejlépe během jednoho snímku se vše, co vidí, změní. Tuto možnost jsem zvolil v této aplikaci, protože byla nejvhodnější. Další variantou je, že mezi změnou obrazu je postupné (velmi rychlé) ztmavení a po té postupné objevení nového obrazu, stejně jako bychom mrkali. [5]

Dále tu je možnost deskových her, kdy kamera je na stejném místě a otáčí se jen prvky ve hře. Musí ale zde být nějaké pozadí, které zůstává statické. [5]

Kapitola 6

Implementace

V této kapitole se seznámíme s implementací aplikace pro VR. Jako první si vysvětlíme základy a proč využívám VRTK, když bych mohl využívat jen Steam VR. Dále zde budou implementační detaily některých zajímavých částí této aplikace. A to hlavně cesta jakou jsem musel jít, abych přišel na správné řešení některých částí. Toto se týká hlavně tvorby datové sady a rozpoznání gest. Na závěr zde bude pár slov k neoficiálnímu testování, které odhalilo některé nedostatky, přičemž část z nich již byla opravena.

6.1 Základy programování VR

Abych mohl pro VR vyvíjet, musel jsem správně nastavit prostředí. Pokud bych používal jenom Steam VR, využil bych CameraRig prefab, ten bych přesunul do scény a aplikace by šla spustit. Touto cestou jsem se vydal při mé první zkušenosti s VR, ale rozhodně to nebyla cesta správná, protože tato aplikace pak nejde bez VR spustit. Díky tomuto kroku jsem si implementaci propojil se zařízením a to je krok, který znemožní testování a spouštění na jiném PC, kde není VR. Takže jsem zvolil jinou cestu a to pomocí VRTK, který vytvoří „Dependency inversion“, tedy rozhraní mezi zařízením a programem. Ve VRTK k tomuto slouží VRTK_SDK_Manager, který dokáže spolupracovat se Steam VR a dalšími různými VR pluginy. Dále ale obsahuje simulátor VR, takže aplikace s tímto prvkem lze spustit na PC, kde není zařízení VR.

Při správném nastavení prostředí jsem mohl začít vyvíjet, aniž bych potřeboval zařízení VR. Ale jen do doby než jsem začal potřebovat ovladače, v ten moment jsem vždy musel mít reálný ovladač. VRTK má další výhodu a to tu, že skripty ovladače jsou jen navázány na VRTK_SDK_Manager, který je aplikuje na ovladač příslušného pluginu, v mém případě na Steam VR.

6.1.1 Ovladače

Ovladače představují hlavní možnost jak interagovat s okolím pomocí úchopu a pohybu. Pohyb jsem implementoval s pomocí VRTK a funguje za pomocí Bezierovy křivky. Pomocí této křivky je daleko jednodušší pohyb v prostoru než s prostou přímkou [7]. Na základě této křivky jsem zobrazil barevnou čáru pro ukázání cesty. Tam kam míříme, nebo tam kam je dovoleno, je zobrazeno kolečko, na kterém se objevíme. Pohyb je stejně jako u ostatních aplikací aktivovaný za pomocí stisknutí trackpadu.

Dále uchopování předmětu bylo nepříjemné a to z důvodu, že v mé aplikaci je kouzlení zapínáno pomocí 7 tlačítka 2.8 neboli triggeru stejně jako v mnohých aplikacích je toto

tlačítko použito pro uchopení věci. Díky tomu jsem tuto funkčnost přesunul na tlačítko 1. Každý objekt, který lze uchopit, musí mít VRTK skripty pro „interactable object“.

6.2 Kouzlení

Kouzlení je hlavní část mé aplikace, základem je ovladač jako spouštěč a neuronová síť jako mechanismus pro rozhodnutí. Jak již bylo zmíněno výše, kouzlo se započne snímat ve chvíli, kdy zmáčkneme tlačítko 7, až do chvíle, kdy ho pustíme, tedy za předpokladu, že držíme v ruce hůlku. Pro identifikaci hůlky jsem využil schopnost Unity a hůlce jsem dal specifický tag, díky kterému lze jednoduše identifikovat, jaký předmět je držen. Pokud je tedy držena hůlka a stisknutý trigger, započne snímání.

Snímání pohybu bylo velmi problematické a to z důvodu, že nelze do funkce update, která je volána každý snímek, vložit kód, který by zjistil pozici a uložil si ji. Tento způsob byl otestován a způsoboval nepříjemné vlastnosti, jako je přeskakování bodů a nedotahování pohybu do konce. Zvolil jsem tedy cestu za pomocí Trail rendereru, který zaznamená pohyb v prostoru a dokáže vrátit pole bodů, které cestou opsal. Obrovská výhoda je zde díky optimalizaci, která zapisuje body podle vzdálenosti od sebe a ne na základě času stráveného na místě. Takže variace v rychlosti kouzlení je mírně zmenšena, ale stále existuje, protože pokud kouzlíme výrazně rychle, Unity nezvládne tyto body snímat a aproximuje cestu velice špatně. Proto zde je u kouzlení stále variace v rychlosti provedení kouzla.

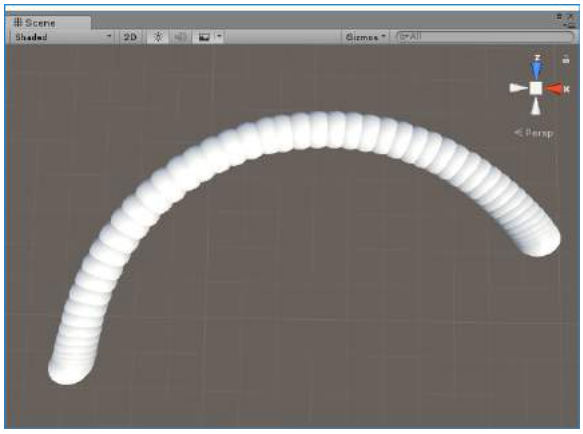
Další problém nastává při konci kouzla, kdy data musím normalizovat a odeslat do sítě, aby vyhodnotila gesto a popřípadě vytvořit kouzlo. Tento proces je časově velmi náročný a nemůže pracovat v hlavním vláknu aplikace, protože docházelo k mírnému zasekávání obrazu. V normálním případě by jen stačilo vytvořit vlákno, dát mu pole gesta a počkat na výsledek. Ale díky tomu, že musím vytvořit instanci GameObject, musím být v hlavním vláknu, nastává zde problém. Tento problém jsem vyřešil za pomocí sdílené paměti, kdy vedlejší vlákno provede výpočet sítě a nastaví sdílenou proměnnou na hodnotu, že již provedlo výpočet. Hlavní vlákno tuto proměnnou kontroluje, a pokud je nastavená, vyčte si výsledek a vytvoří příslušné kouzlo.

6.3 Rozpoznání gest

Cesta k tomu, abych aplikaci naučil rozpoznávat gesta, byla velmi náročná. Protože teorii sítí, která je zde použita, jsem se musel naučit od základu, abych pochopil, jak funguje. Jsem přesvědčen, že tato práce může být dobrým základem pro ostatní, kteří budou vyvíjet podobné aplikace, uvedu zde špatné přístupy, které jsem vyzkoušel. Po té se pokusím zhodnotit, jaké výsledky s jednotlivými pokusy byly.

Jako první jsem se rozhodl pohyb hůlky snímat jen na jednom bodě hůlky (jen špičku hůlky), jak lze vidět na obrázku 6.1, tento krok se ukázal jako velmi chybný, protože pohyb hůlky není jednoznačný. Stejné gesto lze takto vytvořit nekonečně možnými tahy hůlkou. Tedy základ je, aby gesto bylo co nejvíce jednoznačné, proto jsem přidal i spodní část hůlky.

Pokusy o rozpoznání jsem začal za pomocí jednoduchého PCM (Point cloud matching), protože mi to na první pohled přišlo vhodné z důvodu, že mám pole bodů a chci je porovnat s jiným polem bodů. Bohužel i tato cesta má své úskalí a to ve škálovatelnosti a přesnosti. Úkolem rozpoznání je říct jak moc je blízko danému gestu, ale co se týká PCM, tak tato blízkost není uživatelsky příjemná. Co se týká škálovatelnosti, zde je problém, pokud bych chtěl mít kouzlo ve více velikostech, a těchto kouzel chtěl mít více, musel bych pomocí



Obrázek 6.1: Data zaznamenána jen za pomocí špičky hůlky.

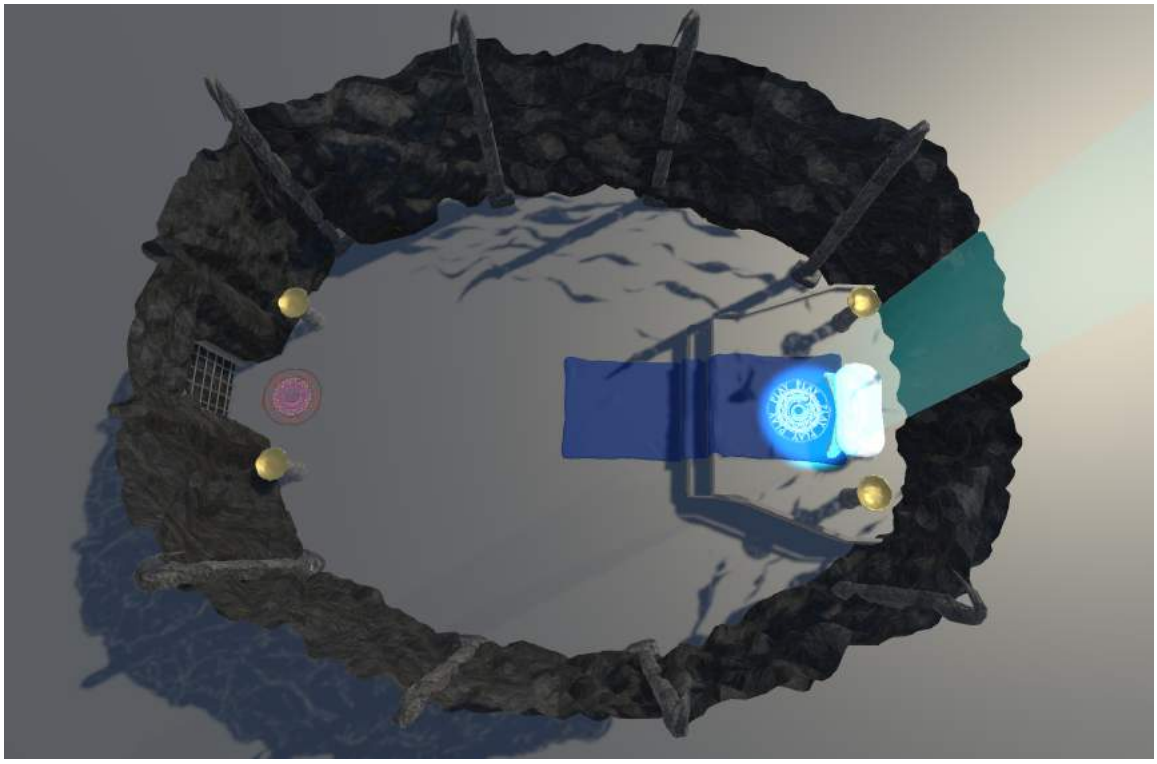
PCM porovnávat obrovské množství dat, což rozhodně nebude dobré pro odezvu hry. Tento přístup jsem tedy na začátku také zavrhl.

Jako poslední jsem použil konvoluční neuronovou síť. Její implementace nebyla jednoduchá, protože jsem s tímto prvkem neměl moc zkušeností, ale dokázal jsem nakonec postupem pokus omyl vytvořit síť představenou v návrhu. Cesta to ale byla dlouhá, protože síť nebyla schopná brát libovolně velký vstup. Jako první jsem se tedy rozhodl, že zvolím určitý počet bodů (průměr všech gest, které jsem měl) a větší kouzla lehce podvzorkoval, a menší jsem aproximoval novými body. Tato cesta nebyla vyloženě špatná, ale u malých kouzel docházelo k 0% detekci, tedy síť se tyto kouzla nedokázala naučit. Takže jsem zvolil cestu zmíněnou v návrhu a to omezení kouzla na 1000 bodů a prázdné body vyplnit hodnotou [0,0,0]. Tato cesta se již ukázala jako správná.

Další problém nastal u normalizace, na začátku jsem nevěděl, že něco takového potřebuji, takže jsem se rozhodl normalizovat jen do počátku souřadnicového systému a datovou sadu generovat. To ale znamenalo, že jsem se snažil generovat data s nekonečnou variací možností. To sice není nic těžkého, ale pracoval jsem jen s jedním kouzlem a k němu jsem vygeneroval přes 200 000 vzorků a stejně tato síť neuměla rozpoznávat vůbec dobře, i když trénování trvalo podstatně dlouho. Je jasné, že nekonečně možností nejsem schopný pokrýt za pomocí 200 000 vzorků. Kdežto při použití normalizace, kterou používám aktuálně vysvětlenou výše, je jasně patrné, že tato cesta je ta správná.

Jedním z dalších problémů s datovou sadou vznikl nedostatkem lidí, kteří by mi pomohli nahrát data. Síť má příjemnou vlastnost, že jaké data jim dám, takové bude umět. Horší je, že pokud tyto data nahraji jen já, tak se síť naučí kouzla podle mě. Na začátku vývoje jsem tomu nepřikládal mnoho váhy, ale ve fázi testování jsem pochopil, že musím přidat data i od někoho jiného. Nakonec se mi podařilo přemluvit alespoň jednoho kamaráda, který mi nahrál pár gest, a síť měla při testování podstatně lepší výsledky. V tuto chvíli tedy mám průměrně 200 nahraných gest mou rukou a 150 gest od kamaráda. To znamená, že na každé kouzlo mám 350 gest a z nich vytvořím 3500 dat pro trénování sítě. Samozřejmě kdyby toto číslo bylo větší, byla by síť do jisté míry lepší.

Nejlepší metoda, na kterou jsem přišel, je snímat vrchní i spodní stranu hůlky, počet bodů gesta omezenou na 1000, normalizovat data a generovat pomocí průměru. Takto jsem na testovací sadě dosáhl 91% úspěšnosti, pocitová úspěšnost je 66% (zde velmi záleží na tom, jak dlouho uživatel trénoval). Oproti tomu výsledky bez normalizace skoro 0%, jen s



Obrázek 6.2: půdorys místnosti menu, ze jsou dva kruhy Play a Exit, pomocí Play se spustí tutoriál a pomocí Exit se ukončí hra.

vrchní stranou hůlky byl výsledek 80%, ale byla zde použita menší datová sada, přičemž mé ruční testování dopadlo daleko hůře (a byl zde problém s pozicí hůlky).

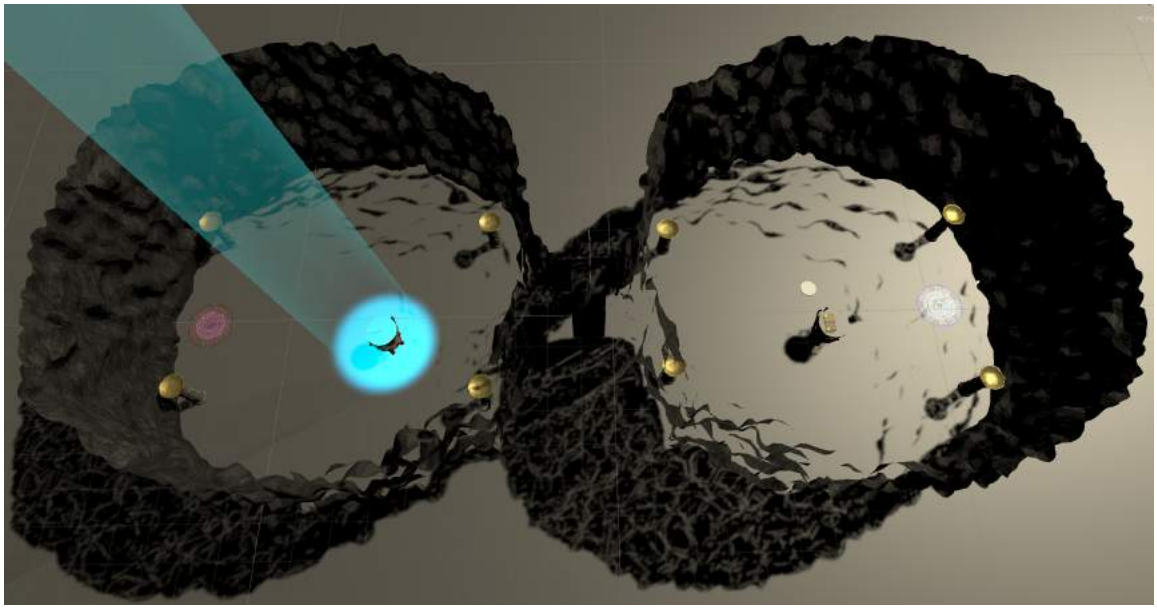
6.4 Menu

Menu jsem z větší části modeloval pomocí programu Blender, je vytvořeno jako jedna velká místnost s oltářem. Na obrázku 6.2 lze vidět kruhy „Exit“ a „Play“, které mají skript, aby s nimi otáčel. Dále je zde kopule jako noční obloha, tu jsem vytvořil za pomoci „seamless“ neboli bezešvé textury, vytvořenou pomocí programu Photoshop z normálního obrázku. U této textury se v Unity mění offset textury a díky tomu navodí atmosféru, že letíme vesmírem. Dále jsem použil SpotLight, který je namířený z vrchu na oltář s kolem „Play“, aby uživatel věděl, že zde má jít. Ve scéně jsem také použil Particle system pro vytvoření ohně jako osvětlení.

Kruh „Play“ a „Exit“ jsem vytvořil tak, že mají svůj bounding box, a skript, který reaguje na průnik jiným bouding boxem. Dále k Headset je připojený také bounding box, od vrchu hlavy až na zem. Takto pokud vstoupíme na kruh, spustí se příslušný skript a v případě „Play“ se spustí místnost s tutoriálem, a při „Exit“ se aplikace vypne.

6.5 Tutoriál

Tato místnost je velmi podobná předchozí místnosti s tím rozdílem, že je zde dvakrát a propojena krátkým tunelem, jako na obrázku 6.3. Tento půdorys má opodstatněný důvod,



Obrázek 6.3: půdorys místnosti tutoriálu, vlevo první místnost kde se hráč objeví a má za úkol vzít hůlku ze stolu nebo možnost kruhem Exit opustit hru a na pravé části je místnost s knihou, kde se hráč bude učit kouzlit a poté může vstoupit kruhem Play do Arény.

a to že rozděluje úlohu, jakou zde musí uživatel provést. První místnost kde se při spuštění objevíme, jsem osvětlil pomocí Particle System ohně, a druhou místnost jsem co možná nejvíce ztmavil. Doprostřed místnosti jsem vložil stůl, na kterém je hůlka, tento stůl jsem osvětlil pomocí Spot Light aby uživatel věděl, že zde musí hůlku vzít. Hůlce jsem dal skript, který implementuje rozhraní VRTK a spouští se pokud je tato hůlka zvednuta. Pokud se hůlka uchopí, celá místnost ztmavne a rozsvítí se následující místnost. Toto je řešeno pomocí odstranění Spot Light, a vypnutí/zapnutí Particle system ohně.

Další místnost obsahuje důležitější část tutoriálu a to nauku kouzlení. Umístil jsem zde stůl s knihou kouzel, která obsahuje tři kouzla a lze v ní listovat. Listování probíhá za pomocí šipek, které mají úlohu tlačítka. Pokud tedy hůlkou nebo ovladačem stiskneme šipku, otočí se stránka. Tato interakce probíhá za pomocí bounding boxů a kolize, tlačítko reaguje na bouding box posunem, jako v reálném světě a pokud se dostane za určitou hranici tak spustí event kliknutí. Ojediněle zde dochází ke špatné kolizi a tlačítko se vyhne špatným směrem, k tomuto dochází při prudkém stlačení, kdy fyzika Unity lehce aproximuje pohyb a tlačítko se dostane do knihy. Event stisknutí spustí skript změny stránky, který změní texturu na příslušnou texturu kouzla. Pokud pak na příslušnou stránku najedu hůlkou, tedy proběhne průnik bouding boxů, tak stránka dostane jiný nádech barvy, aby uživatel věděl, že může provést interakci, jak lze vidět na obrázku 6.4. Zde nastal problém kvůli konfliktu s VRTK, protože pokud něco pomocí VRTK označíme jako „InteractableObject“, tak tato změna barvy pro určení interakce je řízena tímto skriptem. Bohužel pokud s materiálem pracujeme i jinde, v tomto případě měním list stránky, dojde k tomu, že barva označení zůstane. Tento problém jsem musel řešit tak, že jsem VRTK odstranil z této interakce. Musel jsem tedy implementovat změnu barvy a notifikaci příslušného objektu, že dochází k interakci s tímto objektem. Tedy notifikaci, že došlo ke kliknutí na stránku.

Jako poslední je zde ukázka kouzla vytvořena po kliknutí na stránku. Tento objekt je tvořen za pomocí Tube renderer ze stránek Unity. Zde byl hlavní problém jak vybarvovat

Obrázek 6.4: Na tomto obrázku lze vidět, že stránka, ve které je hůlka, je zvýrazněna.

gesto, které se snažíme provést. V Unity každý bod a mesh, má svou barvu, jen s tím nepracuje žádný defaultní shader. Musel jsem tedy z Unity Asset Store stáhnout shader, který to podporuje a lehce ho upravit (z větší části smazat), aby byl jednoduchý. Po té jen po stisknutí triggeru vypočítat vzdálenost bodů hůlky a bodů gesta v prostoru. Zde byl ale problém, že body gesta jsem nebyl schopný dostat přímo z Unity a musel jsem si je tedy dopočítat za pomocí 6.1, kde $point_i$ je bod v prostoru, který hledáme, gestureParentRotation je rotace rodiče, $estурePoint_i$ je normalizovaný bod gesta a gestureParentPosition je pozice rodiče v prostoru. Po té již vzdálenost lze vypočítat pomocí rovnice 6.2, a pokud je tato vzdálenost v rámci limitu tak dochází k vybarvení. Při vybarvování se berou určité body gesta a porovnávají se, aby nedocházelo k tomu, že se uživatel může v gestu vracet. Existuje zde tedy plovoucí okno, ve kterém se musíme pohybovat směrem dopředu, a při tomto pohybu se okno pohybuje také dopředu. Pokud tedy vybarvíme celé gesto, správnou rychlostí (nesmí to být přiliš pomalé ani rychlé), tak se příslušné kouzlo provede.

$$point_i = gestureParentRotation \cdot gesturePoint_i + gestureParentPosition; \tag{6.1}$$

$$distanceAB = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2}; \tag{6.2}$$

6.6 Testování

Po implementování aplikace (z části i při implementaci) proběhlo testování, kde jsem zkoumal na uživatelích, jak jsou schopni tuto aplikaci ovládat. Při prvním testování jsem zjistil, že mnou natrénovaná síť je velmi specifická pro mou ruku, protože lidé, kteří zkoušeli kouzlit, měli veliké problémy provést kouzlo alespoň jednou. Z tohoto důvodu se mi povedlo

domluvit si jednoho kamaráda, který mi pomohl nahrát více dat, jak bylo uvedeno v kapitole 6.3.

Druhé testování probíhalo již po implementaci celé aplikace. Na místo toho abych nechával subjekty, vyplňovat dotazník, jsem hlavně zkoumal a ptal se jich během hraní na chyby, které cítí. Spousta těchto chyb se vztahovala na zpětnou vazbu. Protože jsem sám viděl, že neví, co mají dělat tak jsem s nimi musel souhlasit a tuto část napravit. Z tohoto důvodu jsem zakomponoval daleko lepší zpětnou vazbu hlavně pomocí vibrování ovladače. Dále jsem vypozoroval, že se velmi liší doba naučení kouzel. Samozřejmě nejrychleji se to naučili lidé, kteří již měli s VR předešlou zkušenost a také je určitá výhoda pokud hrají jakékoli počítačové hry. Tato doba byla v rozmezí 1-5 minut na naučení všech tří kouzel, což je podle mého názoru velmi příjemná doba. Protože aplikace není nastavená tak aby byla uživatelsky co nejpříjemnější, ale tak, aby donutila kouzlo provádět pořádně. Použití kouzel v aréně již bylo pro některé problém z důvodu tlaku vyvíjeného ohnivými koulemi letícím na hráče a neschopností si pamatovat gesta. S konceptem aplikace byli všichni velmi spokojení, protože něco takového není běžné. Také byli všichni přesvědčeni, že taková aplikace má budoucnost, a rádi by si tuto hru zahráli i v budoucnu.

Kapitola 7

Závěr

Zadáním práce bylo vytvořit interaktivní aplikaci demonstrující VR, toto zadání není specifické a dávalo mi dost prostoru pro mé nápady. Toho jsem využil k tomu, abych vytvořil koncept kouzlení a postavil na něm aplikaci demonstrující tuto schopnost. Takže zadání bylo splněno, protože aplikace je ucelená a funkční. Největší předností této práce je koncept rozpoznání kouzla, kterému jsem věnoval podstatnou část času, ale výsledkem jsou informace, popřípadě objekty v Unity, které jsou nejen důležité pro tuto aplikaci, ale lze je použít i mimo ni.

Další velkou předností je, že se podařilo naprogramovat neuronovou síť pro takovýto druh problému. Ne že by neuronová síť nebyla v tomto směru používaná, ale po dlouhém hledání na Steam jsem nebyl schopný najít jedinou aplikaci využívající síť takovým způsobem. Tato aplikace, nebo spíše koncept, je jedinečný a dle mého názoru je zde veliký potenciál. Dalším úspěchem je, že aplikace jako taková je díky dobrému programátorskému přístupu velmi otevřena rozšíření, hlavně co se týká nových kouzel, protože je zde vytvořeno celé rozhraní pro nahrávání, generování a učení sítě. Dále díky použití Unity je možné velmi pohodlně změnit grafické prostředí aplikace, nebo její chování.

Na základě neoficiálního testování jsem tedy schopen tvrdit, že tento koncept se lidem líbí a byla by veliká škoda ho v budoucnu nevyužít. Rozhodně bych chtěl s tímto projektem v budoucnu pokračovat a rozvinout již nalezené principy.

Jak již bylo zmíněno, koncept má budoucnost i potenciál, je velká škoda, že jsem ho nebyl schopný plně využít z důvodu, že rozpoznání kouzel zabralo daleko více času, než bych si byl přál. Samozřejmě s každou zkušeností se člověk někam posune a tento projekt, kdybych dělal podruhé, tak přesně informace z kapitoly 6 bych využil a mohl se plně věnovat implementaci aplikace. Dále použití sítě mě svazovalo díky nedostatku dat pro datovou sadu. Tato síť bude tím lepší, čím více bude mít dat (do určité míry), takže data od dalších uživatelů by mi velmi pomohla.

Literatura

[1] URL http://www.virtuix.com/accessories/

[2] About the VIVE controllers.
URL https://www.vive.com/eu/support/vive/category_howto/about-the-controllers.html

[3] Convolutional Neural Networks (CNNs / ConvNets).
URL http://cs231n.github.io/convolutional-networks/

[4] History Of Virtual Reality.
URL https://www.vrs.org.uk/virtual-reality/history.html

[5] Movement in VR.
URL
https://unity3d.com/learn/tutorials/topics/virtual-reality/movement-vr

[6] SteamVR Plugin.
URL https://assetstore.unity.com/packages/templates/systems/steamvr-plugin-32647

[7] VRTK - Virtual Reality Toolkit.
URL https://vrtoolkit.readme.io/

[8] Virtual Reality Menu Design - Part 1. Sep 2017.
URL https://www.re-flekt.com/blog/virtual-reality-menu-design-part1

[9] Pro Playstation VR bude do konce roku k dispozici 280 her. Jan 2018.
URL http://www.gamebro.cz/playstation-vr-bude-konce-roku-k-dispozici-280-her/

[10] Chalupník, V.: Jun 2012.
URL https://www.root.cz/clanky/biologicke-algoritmy-5-neuronove-site/

[11] Dolejš, J.: Google Daydream View: Povedený headset, který potřebuje kvalitní aplikace (recenze). Feb 2017.
URL https://www.svetandroida.cz/google-daydream-view-recenze-201702

[12] Fulton, W.: Slipping and sliding around virtual worlds in the latest Virtuix Omni. Jan 2016.
URL https://www.digitaltrends.com/gaming/virtuix-omni-hands-on

[13] Kumparak, G.: A Brief History Of Oculus. Mar 2014.
URL https://techcrunch.com/2014/03/26/a-brief-history-of-oculus

[14] Martindale, J.: Oculus Rift and HTC Vive head to head: Prices are lower, but our favorite remains the same. Apr 2018.
URL
https://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive

[15] Martín Abadi, J. C. Z. C. A. D. J. D. M. D. S. G. G. I. M. I. M. K. J. L. R. M. S. M. D. G. M. B. S. P. T. V. V. P. W. M. W. Y. Y., Paul Barham; Xiaoqiang Zheng, G. B.: TensorFlow: A System for Large-Scale Machine Learning. *TensorFlow*, , č. ISBN 978-1-931971-33-1.

[16] Pino, N.: PlayStation VR review. Apr 2018.
URL
http://www.techradar.com/reviews/gaming/playstation-vr-1235379/review

[17] Point, T.: Artificial Neural Network Basic Concepts. Jan 2018.
URL https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm

[18] Technologies, U.: Collider.
URL https://docs.unity3d.com/ScriptReference/Collider.html

[19] Technologies, U.: Rigidbody.
URL https://docs.unity3d.com/ScriptReference/Rigidbody.html

[20] Webster, A.: The PS4 has sold 70 million units, while PSVR tops 2 million. Dec 2017.
URL https://www.theverge.com/2017/12/7/16745752/playstation-4-70-million-sales

Příloha A

Obsah DVD

Přibalené DVD obsahuje složky:

- **Application** - obsahuje celý projekt v Unity 3D (skripty, obrázky, materiály, atd.), společně se sítí a daty pro trénování.
- **Video** - obsahuje video demonstrující aplikaci
- **Documentation** - obsahuje tex soubory a pdf verzi tohoto dokumentu