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

VIRTUÁLNÍ REALITA: TECHNOLOGICKÉ DEMO S OCULUS RIFT

VIRTUAL REALITY: TECHNOLOGY DEMO USING OCULUS RIFT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **YURIY HLADYUK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JOZEF KOBRTEK**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19908/2016/xhlady00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Hladyuk Yuriy**
Obor: Informační technologie
Téma: **Virtuální realita: Technologické demo s Oculus Rift**
Virtual Reality: Technology Demo Using Oculus Rift
Kategorie: Počítačová grafika

Pokyny:

1. Obeznamte se s principy a fungováním headsetů virtuální reality
2. Seznamte se s Oculus Rift a jeho SDK.
3. Seznamte se s grafickým rozhraním OpenGL a jeho použitím pro virtuální realitu.
4. Navrhněte aplikaci demonstrující možnosti Oculus Rift v kombinaci s OpenGL.
5. Otestujte Vámi implementovanou aplikaci na několika scénách, sesbírejte zpětnou vazbu od uživatelů.
6. Sesbírané výsledky vyhodnoťte.

Literatura:

- Oculus Best Practices Guide https://developer.oculus.com/documentation/intro-vr/latest/concepts/book-bp/
- Unreal Engine - VR Best Practices Guide https://docs.unrealengine.com/latest/INT/Platforms/VR/ContentSetup/

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kobrtek Jozef, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá headsetem Oculus Rift a možnostmi integrace s OpenGL. Představí čtenáři virtuální realitu a její historii, princip fungování headsetů virtuální reality, teorii návrhu pro virtuální realitu a nástroj Oculus Rift SDK. Popisuje návrh a implementaci aplikace demonstrující možnosti Oculus Rift s OpenGL. V závěru se práce zabývá testováním a vyhodnocením sesbíraných výsledků. Aplikaci testovalo celkem 43 uživatelů.

Abstract

This thesis describes device Oculus Rift and how it can be integrated with OpenGL. It will present an introduction to virtual reality, head-mounted display, best practices and Oculus Rift SDK to the reader. The thesis describes design and implementation of application for Oculus Rift integrated with OpenGL. Additionally, measurements and performed tests will be discussed. The application was tested with 43 users.

Klíčová slova

Virtuální realita, Headsety pro virtuální realitu, Oculus Rift, Oculus Rift SDK, OpenGL, Technologické demo, C++

Keywords

Virtual reality, Head-mounted display for virtual reality, Oculus Rift, Oculus Rift SDK, OpenGL, Technology demo, C++

Citace

HLADYUK, Yuriy. *Virtuální realita: Technologické demo s Oculus Rift.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jozef Kobrtek

Virtuální realita: Technologické demo s Oculus Rift

Prohlášení

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

.....
Yuriy Hladyuk
17. května 2017

Poděkování

Děkuji vedoucímu práce Ing. Jozefovi Kobrtkovi za odborné vedení a ochotu při konzultacích.

Obsah

Kapitola 1

Úvod

Virtuální realita je známa již od 20. století, ale teprve v posledních letech se znovu dostává do popředí zájmu. Je to především díky vzniku nových technologií a značného zvýšení zájmu o pokrok v tomto odvětví. Mezi ně patří právě headset Oculus Rift od firmy Oculus, na který se tato práce zaměřuje.

Virtuální realita už od samého počátku trpěla různými nedostatky. Na počátku svého vývoje, tedy v 60. letech, virtuální realita trpěla hlavně na nedostatečně vyvinuté technologie, například nízkou kvalitou zobrazení, nedostatečným výkonem headsetů, a tedy i vysokou odezvou obrazů. V dnešní době se však vývoj neustále posouvá dopředu a kvalita zobrazení je značně lepší a odezva displeje, díky novým technologiím, je v reálném čase. Nedostatky spojené s výkonem jsou tedy značně minimalizovány a vývojáři se musí zabývat spíše jinými, mnohdy daleko závažnějšími problémy, jako je například nevolnost v simulátoru (*z angl. simulator sickness*). I přesto, že se jedná o velmi závažný problém, lze jej poměrně snadno omezit, případně úplně odstranit správným návrhem aplikace, která na zařízeních poběží. Nicméně, ani dnešní technologie nedovolují se naprosto ponořit do virtuálního světa a zapomenout tak na realitu skutečnou. Je to způsobeno především problémem ovládání, kdy je sice možné precizně a téměř bez problému snímat zařízení a herní ovladače, ale pohyb těla v prostoru je značně omezen. Dalším aspektem je nedostatečná simulace ostatních lidských vjemů. Pro dosažení pocitu naprostého vcítění do jiného světa bychom museli simulovat nejen obraz a zvuk, ale také hmat, čich, chuť atd.

Cílem práce je naimplementovat aplikaci s několika scénami a následně je otestovat. Práce je rozdělená na několik logických části. Nejdříve v kapitole 2 definujeme pojem virtuální realita, jaké jsou její druhy a ve stručnosti nastíníme její vývoj. Dále je podkapitola 2.2 věnována novodobému zařízení Oculus Rift, pro které je naimplementována naše aplikace. A protože v dnešní době je návrh a implementace největším problémem, budeme se dále v podkapitole 2.3 věnovat teorii vývoje pro virtuální realitu.

Kapitola 3 je věnována analýze a návrhu. Jsou zde zmíněné zásadní technologie, které nám pomohly při vývoji aplikace. Mezi tyto technologie patří například Oculus Rift SDK, které je věnována podkapitola 3.1, v níž je zmíněna i její funkcionalita. Další části této kapitoly jsou věnovány návrhu aplikace a scén, ale je zde zmíněný i návrh testování.

V další kapitole se věnujeme implementaci naší aplikace. V podkapitole 4.1 se nejdříve zaměříme na integraci Oculus Rift SDK s OpenGL. Následující část je věnována implementaci scén a jejich komponent. Detailně popíšeme, jakým způsobem vytváříme vodu, generujeme osvětlení a také jak jsme vyřešili ovládání celé aplikace.

V poslední částí, kapitole 5, se věnujeme testování práce. Byl vytvořen dotazník, který jsme využili při testování s uživateli a následně jsme dotazníky vyhodnotili. Testování bylo

zaměřeno hlavně na nevolnost v simulátoru, ale byly otestovány i celkové pocity z aplikace. Pro testování aplikací virtuální reality již existuje dotazník, který se často využívá. My si tento dotazník pouze zmíníme, ale nakonec bylo rozhodnuto, že si dotazník navrhneme sami. Náš dotazník nám umožní lépe vyhodnotit výsledky v závislosti na naší aplikaci, jelikož obsahuje více scén. Testování je plánováno i v průběhu vývoje, abychom mohli scény upravovat před výslednou verzí.

Kapitola 2

Teorie

V této kapitole si definujeme, co je to virtuální realita a povíme si ve zkratce o její historii. Dále se budeme věnovat zařízení Oculus Rift. Za velký nedostatek dnešní virtuální reality se považuje nevolnost v simulátoru [15]. Proto se v kapitole 2.3 budeme věnovat teorií správného návrhu aplikací, abychom se tomuto problému téměř zcela vyhnuli.

2.1 Virtuální realita

Virtuální realita je simulované prostředí, které vytváří iluzi skutečného nebo fiktivního světa. Tento název vymyslel a zpopularizoval ve druhé polovině 80. let 20. století Jaron Lanier, jeden z průkopníku této oblasti. Lanier také zavedl definici virtuální reality, která zní, „počítačem vytvořené interaktivní trojrozměrné prostředí, do něhož se člověk totálně ponoří“ [20]. Principem je zmást lidské smysly tak, aby si mozek myslel, že jde o realitu. Tohoto efektu lze docílit řadou různých způsobů a zařízení. My se budeme věnovat brýlím neboli zařízení HMD (*angl. head mounted displays*), které se připevňuje na hlavu [3]. Toto zařízení se zaměřuje na nejdůležitější lidský smysl a tím je zrak.

2.1.1 Zařízení HMD

HMD je pro nás důležité zařízení, které zprostředkovává obrazový vstup do virtuálního světa. Tento přístroj můžeme definovat jako výstupní zobrazovací zařízení využitelné současně pouze pro jednu osobu. Je to tedy zařízení v podobě brýlí, popřípadě helmy. Vizuální vjem je vytvářen pomocí jednoho nebo dvou displejů (CRT, LCD, LED atd.). V případě jednoho displeje je obraz rozdělený na polovinu a rozlišení je tedy sdílené na obě oči. Před každým obrazem je optická čočka, která je velmi důležitou součásti HMD. Tato optika zajišťuje širší zorné pole a dosahuje se až 120° ze 180° ve skutečnosti [11]. Virtuální svět je pak připraven pro každé oko zvlášť a každé oko vidí tento svět z lehce jiného úhlu. Dále se v brýlích často nachází snímače několika typů včetně akcelerometru[1], gyroskopu[2] a laserových snímačů polohy, díky kterým je zařízení schopno vypočítat výslednou polohu helmy v prostoru [3].

Jeden způsob vytvoření virtuálního prostoru je pomocí připojení k externímu zařízení např. počítač nebo konzole. Jiný, jednodušší způsob je pomocí mobilního telefonu, který zasuneme do přední částí helmy. Helma v tomto případě obsahuje pouze čočky a žádné jiné pomocné senzory. Ty jsou součástí telefonu [15].

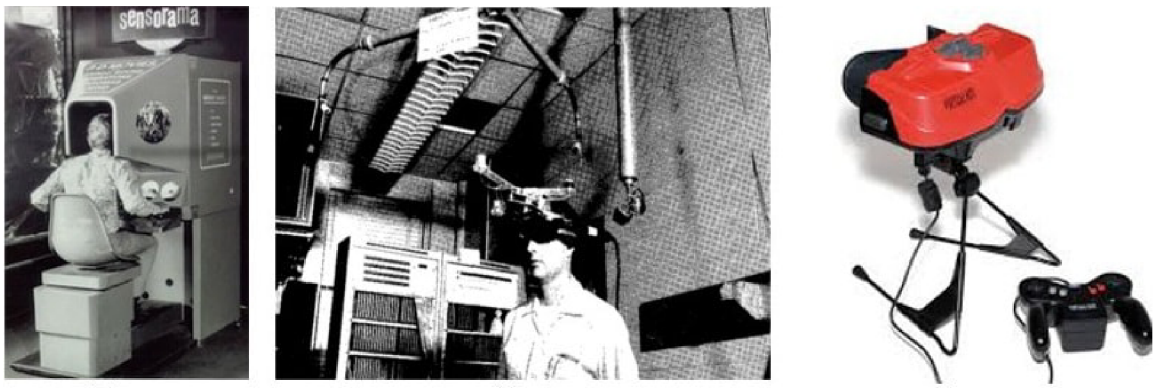
[1] Elektromechanické zařízení, které měří zrychlení sil.

[2] Senzor sloužící k určení naklonění a natočení zařízení

2.1.2 Vývoj virtuální reality

Samotný počátek virtuální reality, nebo alespoň její myšlenku, můžeme mapovat už od 19. století [15]. V 19. století se totiž poprvé objevují panoramatické obrazy, tedy obrazy zachycující celý prostor kolem malíře. V roce 1838 Charles Wheatstone demonstroval na stereoskopických fotografiích, jak fungují lidské oči. Ukázal tak, že pokud se člověk dívá zblízka na dvě fotografie ze stejného místa pořízené pod jiným úhlem, pozorovatel si připadá, jako by byl na místě dění. Vývoj dále pokračoval a v roce 1929 vznikl první letecký simulátor *Link Trainer*, který se nahýbal, nakláněl a otáčel. Za druhé světové války si výcvikem na *Link Traineru* prošlo více než půl milionu pilotů. Dalším historickým pokrokem byl vynález Mortona Heiliga. Přístroj zvaný *Sensorama* (Obrázek 2.1) nabízel divákovi pohyblivé obrazy, zvuk, vibrace, a dokonce i generátor pachů [15][17].

První pokus o virtuální brýle se objevil až v roce 1960 s názvem *Telesphere Mask*, který vynalezl také Morton Heilig. Brýle ale neměly žádné senzory pohybu a obraz se v brýlích nehýbal. O rok později inženýři ze společnosti Philco Corporation vynalezli první brýle se snímáním pohybu hlavy. Jedná se o první velký krok k virtuální realitě, jak jí známe dnes. Nicméně až v roce 1968 Ivan Sutherland postavil přístroj *Sword of Damocles* (Obrázek 2.1), který je považován za první headset pro virtuální realitu. V místnosti, kde byly brýle nainstalovány, dokázal počítač dopočítat a vykreslit jednoduché tvary, které měly připomínat nábytek. V roce 1987 Jaron Lanier definoval pojem virtuální realita a s kolegy začali vyrábět několik druhů brýlí. Začali vznikat první virtuální arkádové stroje, ke kterým měla přístup i veřejnost. Odezva těchto strojů už v té době byla pod 50ms. Inspirovat se nechali společnosti Sega a Nintendo a představili tak své vlastní virtuální brýle. Sega však své brýle kvůli technickým problémům nikdy nevydala a Nintendo (Obrázek 2.1) muselo svoji výrobu ukončit kvůli nedostatečné spokojenosti veřejnosti [15][17].



Obrázek 2.1: (a) Sensorama, 1957 (b) Sword of Damocles, 1968 (c) Nintendo Virtual Boy, 1995 [15]

2.1.3 Virtuální realita dnes

Současnou generaci virtuální reality lze datovat od roku 2012, kdy společnost Oculus spustila crowdfunding[3] kampaň na portálu Kickstarter[4] [12]. Kampaň byla úspěšná a o rok později Oculus vydává jejích první brýle pod názvem Rift, které mají dnes název DK1

[3]Způsob financování, při kterém větší počet jednotlivců přispívá menším obnosem k cílové částce.
[4]Portál pro crowdfundingové financování projektů. https://www.kickstarter.com

(*z angl. Development Kit 1*). Cílovou skupinou byly vývojáři a zařízení se vyprodalo během několika minut. Následník DK1 je DK2 (Obrázek 2.2), který nabízí vyšší rozlišení a obnovovací frekvenci, OLED displeje a lepší techniku snímání pozice. Poslední verze je v prodeji od počátku roku 2016 pod názvem Rift CV1 (*z angl. Consumer version*).

Hlavním konkurentem Oculus Rift je HMD od společnosti HTC s názvem Vive (Obrázek 2.2), který je v prodeji od roku 2016 [18]. Tyto brýle navíc nabízí dva ruční ovladače pro interakci s prostředím a dva senzory pro snímání prostoru o rozměru 4,6 x 4,6 metrů. Uživatel se může v tomto omezeném prostoru pohybovat a pomocí ovladačů virtuální svět ovládat. Jako další zajímavé zařízení HMD můžeme zmínit Fove od společnosti FOVE, Inc. Toto zařízení využívá technologii pro detekci pohybu oči a má zatím nejvyšší rozlišení (2560x1440). S touto technologii lze zamezit tomu, aby uživatel viděl okraje obrazovky rozmazané, jelikož lze detekovat, kam se uživatel dívá a toto místo vykreslit v plném rozlišení, zatímco ostatní místa v rozlišení menším. Technologii lze zároveň využít pro ovládání aplikací samotných [13].

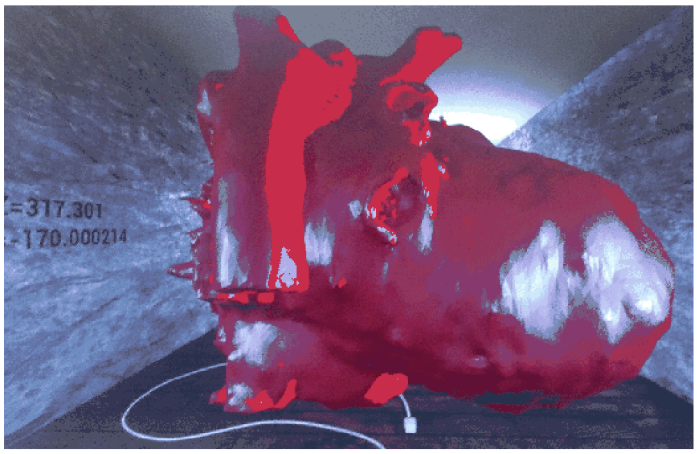
Obrázek 2.2: (a) Oculus Rift DK2 (b) HTC Vive s ovladači a senzory (Převzato z [15][18])

2.1.4 Druhy virtuální reality

Kdybychom měli rozdělit virtuální realitu na základě interakce s uživatelem, vyčleníme tři základní druhy a to pasivní, aktivní a interaktivní. Pasivní aplikace fungují podobně jako promítání filmu. Takové prostředí můžeme vidět, slyšet a určitým způsobem i cítit, ale nemáme žádnou možnost měnit chod aplikace. Uživatel je tedy pouze pasivním příjemcem virtuálních vjemů. Způsob užití jsou různá představení a filmy [11].

V aktivních aplikacích máme možnost virtuální prostředí libovolně zkoumat. Můžeme se pohybovat, rozhlížet se a slyšet odpovídající zvuk, ale nemáme možnost toto prostředí nijak měnit. Tento druh aplikací má potencionální využití např. v medicíně. Doktor má možnost si nechat zobrazit 3D orgán z naskenovaných dat pacienta, a následně si model prohlédnout (viz Obrázek 2.3). Jiná možnost využití je například virtuální prototyp libovolné konstrukce, např. kuchyně. Zákazník má možnost prohlédnutí návrhu ve virtuálním prostředí dříve, než bude konstrukce reálně postavena [15].

Nejdokonalejším a technicky nejnáročnějším je interaktivní druh aplikace. Tento druh aplikace dovoluje prostředí nejen zkoumat, ale také jej modifikovat. V tomto případě má uživatel možnost brát do ruky různé předměty, přemisťovat je nebo s těmito předměty pracovat. Jako příklad využití jsou opět různé návrhářské aplikace, ve kterých je uživatel schopný přemísťovat objekty, a přitom výsledek okamžitě pozorovat [11].



Obrázek 2.3: Vizualizace virtuálního srdce pomoci naskenovaných dat. Vyvinuto na Univerzitě Illinois. Zdroj: http://vr.cs.uiuc.edu/node20.html

2.2 Oculus Rift

V naši práci budeme využívat Oculus Rift DK2 (dále jen DK2), a proto si v této kapitole zmíníme více podrobností o tomto zařízení (Obrázek 2.2) a dále si přiblížíme techniky, které nabízí jeho SDK.

DK2 se skládá ze dvou částí. Tou hlavní jsou projekční brýle, které si uživatel připevní na hlavu. Druhou části je pohybový senzor, který se nejčastěji umisťuje na monitor a zaznamenává pohyb brýlí. Senzor neboli kamera slouží jako přijímač infračervených paprsků vysílaných z diod, které se nachází uvnitř helmy společně s procesorem ARM, který má tyto diody na starosti [16]. DK2 brýle tedy obsahuji gyroskop, akcelerometr, a navíc oproti brýlím HTC Vive obsahují magnetometr[5].

Prostor, ve kterém kamera dokáže snímat je poměrně malý, ale lze ho však rozšířit druhou kamerou. Ve spotřebitelské verzi CV1 můžeme celkový počet snímacích kamer rozšířit ještě třetí kamerou, která slouží pro sledování ovladačů *Oculus Touch*. Tyto ovladače jsou bezdrátové a jejích detekce funguje na stejném principu jako brýle. Pomocí ovladačů, které mají několik tlačítek, uživatel může interagovat s virtuálním světem [16].

2.2.1 Čočky a výsledný obraz

Při nasazení HMD jsou procesy akomodace[6] a konvergence očí odděleny, protože LCD obraz je stále ve stejné vzdálenosti od oka. Z tohoto důvodu čočky v DK2 navozují dojem, že je uživatel vzdálen 1,3m od monitoru, aby byla zmenšena zátěž očí uživatelů a naopak byl zvýšen čas strávený ve virtuálním světě [2]. Dosud v této verzi jsou čočky akrylové, ale ve poslední verzi CV1 se Oculus rozhodl pro Fresnelovy typ čoček.

DK2 se pyšní nízkou latencí, kterou definují jako čas mezi pohybem hlavy a novým snímkem v brýlích. Tento čas zahrnuje odezvu senzoru, fúzi senzoru, vykreslení, přenos obrazu do brýlí a odezvu zobrazení jednotlivých displejů. Brýle navíc využívají technologii předpovídání pohybu (*angl. predictive tracking technology*), která je součástí SDK [2].

[5] Magnetometr měří ve třech osách složku lokálního magnetického pole

[6] Akomodace je proces, při němž se mění optická mohutnost oka. Díky tomu zaostřujeme na různě vzdálené předměty

Výsledná projekce DK2 má rozlišení 1920 x 1080 s obnovovací frekvencí 75 Hz a zorným polem 100 stupňů. Doporučená hodnota snímků za vteřinu je stejná jako obnovovací frekvence, tedy 75 [5].

2.2.2 Oculus Rift SDK

Kromě OpenVR je Oculus Rift SDK jediný vývojový nástroj umožňující integraci s brýlemi Oculus Rift. V následujícím textu si zmíníme teoretické poznatky o tomto nástroji, například techniky, díky kterým brýle snižují odezvu obrazů nebo zvyšují rychlost zpracování [8].

- **Asynchronous TimeWarp** (ATW) je technika pro snížení odezvy a chvění. Bez této techniky se nám může stát, že se budou opakovat předchozí snímky kvůli vysoké odezvě zařízení. Když uděláme pohyb hlavou a v brýlích není zobrazován odpovídající obraz, aplikace může vyvolat dezorientaci, a proto je tato technika velice užitečná. ATW je automaticky povoleno a není nutné tuto techniku ručně nastavovat.
- **Adaptive Queue Ahead** (AQA) zvyšuje rychlost zpracování obrazu. Pomoci této techniky SDK zpracovává vždy další snímek už během vykreslování předešlého snímku. Bez této techniky by procesor začal zpracovávat další snímek až bychom zobrazili předchozí snímek, což je značně pomalejší. AQA umožňuje tedy zpracovávání dalšího snímku daleko dříve a snížit tak odezvu aplikace. SDK má tuto techniku automaticky nastavenou s výchozí hodnotou jeden obraz.
- **Asynchronous SpaceWarp** (ASW) je technika, která predikuje další snímek pomocí zpracování animace a posunutí kamery předchozího snímku. Pomoci této techniky je výsledný obraz plynulejší a aplikace tedy mohou běžet i na hardwaru, který nesplňuje doporučené požadavky. Od verze 1.10 je tato technika opět automaticky povolena.
- **Oculus Guardian System** je systém který nám dokáže v aplikaci zobrazovat indikátory na stěně nebo na zemi, pokud se přiblížíme k určitým hranicím, které si sami definujeme.
- **Oculus Debug Tool** je nástroj pro zobrazení informací o naší aplikaci jako je například odezva obrazu.

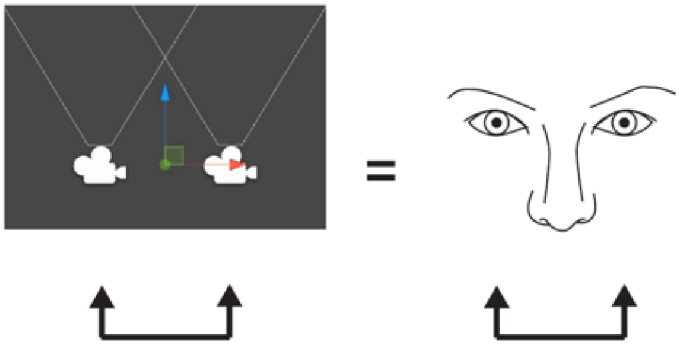
2.3 Teorie vývoje pro virtuální realitu

Přestože se vývoj technologií posunul vpřed, stále se projevují některé problémy, které jsou s virtuální realitou spojené od jeho počátku. Za jeden z největších nedostatků lze považovat zmiňovanou nevolnost v simulátoru. My se v této kapitole budeme věnovat teorii, jak se tomuto zásadnímu problému téměř zcela vyhnout.

Nevolnost v simulátoru byla pravděpodobně hlavní příčinou úpadku zájmu o virtuální realitu v 90. letech. Nevyhneme se jí pouze rychlejším hardwarem, ale musíme dbát i na správný návrh softwaru [5]. Z čehož plyne, že některé žánry, které vypadají na první pohled ideální pro virtuální svět, mohou být naopak nevhodné, např. letecké simulátory. Hlavní příčinou je pocit zrychlení, což nemusí dělat dobře lidskému mozku.

2.3.1 Virtuální kamery a jejích zorné pole

Výsledný obraz je pro každé oko vygenerovaný pomocí virtuální kamery, která se nachází ve scéně. Tyto kamery mají mezi sebou malou vzdálenost ICD (*z angl. inter-camera distance*), díky které vzniká výsledný 3D stereoskopický obraz. Je vhodné zvolit ICD tak, aby odpovídala vzdálenosti, jakou mají čočky v zařízení DK2 a ta je 63,5mm, což odpovídá průměrné vzdálenosti zornic IPD (*z angl. interpupillary distance*). Zvýšením vzdálenosti virtuálních kamer neboli ICD, můžeme zapříčinit přehnanou hloubku obrazu a větší konvergenci oči, která směřuje k většímu namáhání zraku. Naopak snížením vzdálenosti může být výsledný obraz příliš plochý a uživatel si bude připadat menší. Proto je vhodné zvážit, jestli opravdu chceme hodnotu ICD měnit a nenechat jí shodnout s IPD, kterou nám poskytuje Oculus Rift SDK [5].



Obrázek 2.4: Vzdálenost a zorné pole kamer musí odpovídat nastavení profilu uživatele (Převzato z: [5])

V běžné aplikaci si můžeme zvolit libovolné zorné pole kamery cFOV (*z angl. camera field of view*) a nemá to téměř žádný následek nevolnosti. Periferně jsme totiž schopni vidět zbytek místnosti a monitor nijak nereaguje na pohyb hlavy. I když obraz monitoru nás může pohltit svým obsahem, náš mozek není nijak oklamán, že je obraz reálný. Při nasazení brýlí nám však není umožněno vidět své okolí a vše, co vidíme periferně je obsah virtuálního světa. Z tohoto důvodu je velmi důležité zvolit cFOV stejné jako zorné pole displeje dFOV (*z angl. display field of view*). Tyto informace nám opět poskytuje samotné Oculus SDK a není doporučeno je modifikovat, abychom nezapříčinili zkreslení scény nebo nevyvolali uživateli nevolnost [15][5].

2.3.2 Pozice a orientace

Pomocí senzoru Oculus Rift dokážeme zjistit orientaci hlavy uživatele v reálném světě a synchronizovat se s tím virtuálním. Dohromady všechny senzory vytváří systém šest stupňů volnosti 6DoF (*z angl. Six degrees of freedom*), který nám poskytuje reálný pocit rozhlížení se kolem sebe. Díky tomuto systému může uživatel např. uhýbat střelám nebo se naklonit a dívat se jedním okem za roh [15].

Díky této technologii má uživatel možnost pozorovat objekty z různých úhlu a více se ponořit do virtuálního světa. Na druhou stranu vznikají nové neočekávané stavy, které bychom neměli zanedbávat. Pozorovatel se díky volnému pohybu hlavy může dostat do konfliktu s virtuálním prostředím např. dostat pozici virtuální kamery dovnitř objektů nebo stěny a vzniká otázka, jak tento stav řešit. Nejčastějším východiskem daného problému je

dodržovat minimální vzdálenost od objektů, ale na druhou stranu, tím uživatele distancujeme pomyslnou bariérou od virtuálního světa. Dalším řešením je upozornit uživatele hlášením, že je mimo snímací prostor a postupně ztmavovat scénu, i když uživatel je pořád v zorném poli snímací kamery. Toto řešení je vhodné využít i v situaci, kdy uživatel je příliš nakloněný na jednu ze stran a snímací prostor reálně opustil [5].

2.3.3 Vykreslování a pozorování objektů

Před samotným vykreslováním scény je důležité vhodně zvolit měřítko virtuálního světa ku realitě. Doporučuje se zvolit měřítko 1:1, aby si uživatel nepřipadal jako příliš velký nebo naopak malý vůči scéně. Navíc můžeme všechny vzdálenosti referovat v metrické soustavě.

Objekty, které bude uživatel pozorovat delší dobu, je vhodné umístit do vzdálenosti přibližně 0,75m až 3,5m, abychom se vyhnuli namáhaní zraku. Nicméně společnost Oculus se věnuje neustálému vylepšování optiky, aby bylo možné mít objekty téměř v libovolné vzdálenosti od uživatele. Přesto bychom menu nebo grafické uživatelské rozhraní GUI (*angl. graphical user interface*) měli vykreslovat ve vzdálenosti přibližně 2,5m od uživatele. Jsou však jedinci, kteří chtějí pozorovat objekty z menší vzdálenosti, a přitom se více vnést do virtuálního světa. Na druhou si tímto sami zapříčiňuji namáhání zraku [5][10].

Další faktor, který může znepříjemňovat pozorování virtuálního světa je tzv. *flickering*, který můžeme přeložit jako blikání nebo mihotání. Výskyt tohoto blikání na okrajích scény je pro uživatele daleko více nepohodlný, než jeho přítomnost uprostřed zorného pole. Tomuto problému se však můžeme vyhnout, pokud budeme vytvářet tmavší scény, které jsou o poznání méně náchylné na blikání jako světlejší scény. Pokud však hodláme navrhovat světlejší scény, je vhodné ztmavovat barvy alespoň na okrajích zorného pole. Pochopitelně blikání není zapříčiněné pouze světlými scénami, ale i různými vykreslovacími technikami jako je například *normal mapping*[7] [10].

Technika *normal mapping* přidává do výsledného modelu o poznání lepší detaily a nerovnosti bez změny jeho geometrie. Různé podobné techniky můžou ale zapříčiňovat právě zmiňované blikání. Příčinou je výsledné nesladění dvou obrazů. Naší povinností je poskytovat uživateli pro každé oko správný obraz, aby je mozek mohl správně sloučit dohromady a nevznikala nesladěnost.

Zásluhou stereoskopických obrazů můžeme vnímat hloubku a vzdálenost objektů mezi sebou. S rostoucí vzdálenosti však tuto schopnost ztrácíme a už nejsme schopni rozeznat vzdálenosti mezi objekty. Toho můžeme využít a raději vykreslovat plochý obrázek místo celých 3D objektů a zvýšit tak výkon výsledné aplikace [5].

2.3.4 Pohyb ve virtuálním prostředí

Pohyb ve virtuálním světě budeme rozumět jako jakákoliv změna pozice, která nesouhlasí s pohybem uživatele v reálném světě. Přitom se nejedná pouze o změnu pozice, ale i o změnu úhlu pohledu nebo jakékoliv otřesy zobrazení. Zatímco na monitoru různé otřesy kamery mohou vypadat efektivně např. zásah střelou protivníka, ve virtuální realitě tomu tak bohužel není. Tyto neočekávané pohyby vyvolávají nevolnost v simulátoru, kvůli protichůdným signálům do našeho mozku. Náš zrak totiž vysílá signál do mozku, že jsme v pohybu, ale rovnovážné ústrojí ve středním uchu správně detekuje, že je tělo v klidu a vzniká tak zmatení mozku [15][5].

[7] http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

Pokud se rozhodneme pohyb využít, a to zpravidla chceme, je doporučeno zachovávat rychlost pohybu ve virtuálním světě přibližně 1,4 m/s, což odpovídá průměrné rychlosti chůze člověka [4]. Pohyb okolo 3 m/s je ještě zcela přijatelný, ale při vyšších rychlostech už je důležité aplikaci řádně otestovat na několika uživatelích. Pohyb by měl směřovat zároveň dopředu, jak je to v reálném životě. Ve hrách se často můžeme pohybovat do stran nebo dozadu. V reálném světě se však pohybujeme vyloženě dopředu a v málokteré situaci děláme pohyb zpátky nebo do strany. Zároveň pohyb do schodů může být velmi nepříjemný. Nejenže tento pohyb vyvolává vertikální zrychlení, ale navíc nám může ve speciálních případech připadat, že se schody pohybují, zatímco okolí je stacionární. V některých okolnostech může být vhodnější, a dokonce i snadnější, zvolit teleportaci neboli přemístění mezi dvěma místy. Při teleportaci však není vhodné měnit směr pohledu uživatele, v opačným případě bude dezorientovaný [10][14].

Pro uživatele je pohodlnější pohyb, který sami iniciují, oproti pohybu, který provede aplikace samotná. Můžeme to přirovnat jízdě autem, kde řidič je značně méně náchylný na nevolnost jako spolujezdci. V některých přídech je tedy vhodnější umožnit uživateli pohyb pomocí ovladačů nebo jiného zařízení. Pohyb ve virtuálním prostředí je jedním z největších příčin nevolnosti v simulátoru, a proto je velmi důležité si na veškerý pohyb s virtuální kamerou dávat pozor a nezapomínat na skutečnost, že pozorovatel téměř vždy sedí pouze na židli a tyto pohyby necítí [5].

2.3.5 Obsah virtuálního světa

V běžných aplikacích se často vykresluje tzv. HUD (*z angl. heads-up display*), tedy informace, které uživateli například sdělují, kde se nachází na mapě nebo kolik zbývá munice ve zbrani. Ve virtuálním prostředí bychom se tomu však měli vyvarovat, protože výsledný obraz může mást uživatele. První důvod je zapříčiněný nulovou odezvou těchto objektů na jakýkoliv pohyb hlavy [9]. Druhým důvodem je obtížné sloučení obrazu virtuálních kamer kvůli binokulární disparitě[8]. Zároveň se nedoporučuje HUD vykreslovat do periferního zorného pole, jinak přinutíme uživatele daleko více konvergovat očima a po čase mu způsobíme jejích únavu [5]. Proto je pro uživatele příjemnější vykreslit počet nábojů například na samotnou zbraň (viz Obrázek 2.5). Tomuto problému se můžeme vyhnout, pokud budeme vykreslovat tyto objekty v minimální vzdálenosti 75 cm, ale na druhou stranu tímto opět vytvoříme pomyslnou bariéru a distancujeme uživatele od virtuálního prostředí.

[8] Jemný rozdíl v úhlu pohledu

Obrázek 2.5: Ukázka vhodného způsobu zobrazení HUD[9]

Uživatele můžeme vnést více do virtuálního světa díky tzn. „Avataru“. Avatar je virtuální reprezentace těla pozorovatele, který naznačuje pohyby a gesta uživatele ve virtuálním světě. Virtuální tělo přináší do virtuálního světa spíše výhody než nevýhody. Hlavní výhodou je vcítění se do scény díky přítomnosti jakéhosi měřítka ve virtuálním prostoru. Za nevýhodu můžeme považovat zobrazení nevlastního těla, které nás může rozptylovat např. opačné pohlaví nebo jiná barva pleti [19]. Dalším problémem může být zobrazení různých nástrojů, který bude mít uživatel zobrazený před sebou. Tyto nástroje na popředí aplikace jsou běžným řešením na monitoru, ale ne ve virtuální realitě. Ve virtuálním světě může opět docházet k namáhaní oči kvůli častému přeostřování mezi scénou a nástrojem před námi. Navíc nástroj je velmi blízko obrazu a je obtížnější obrazy spojit do výsledného trojrozměrného obrazu. Současné doporučení je tyto nástroje vykreslovat spolu s avatarem [5].

2.3.6 Možnosti interakce s virtuálním světem

Je důležité si uvědomit, že jakmile uživatel nasadí brýle, neuvidí klávesnici ani žádné jiné podobné zařízení. Výjimkou jsou zmiňované ovladače *Oculus Touch*, které mají své snímací senzory pomoci, kterých můžeme ovladače vykreslovat do virtuálního světa [16]. Tyto ovladače nejčastěji slouží jako virtuální ruce, díky kterým má uživatel možnost brát nebo přesouvat předměty.

Pokud však ovladače nemáme k dispozici a chceme dát uživateli možnosti výběru z menu jiným způsobem jako klávesnici, můžeme využít paprsek nebolí tzv. *ray-casting*. Díky této technice máme k dispozici možnost výběru pohledem. Pokud uživatel míří správně na nějaký objekt, je zpravidla vhodné zobrazit nějakou zpětnou vazbu jako například ztmavení objektu. Nevýhoda tohoto způsobu je nepřesnost kvůli nechtěnému pohybu hlavy [5].

[9]Převzato z: https://www.youtube.com/watch?v=CLOPOAGz-Ew

Kapitola 3

Analýza a návrh

V této kapitole se budeme věnovat analýze knihoven pro naší aplikaci. V prvé řadě se detailněji podíváme na Oculus Rift SDK, které poskytuje integraci s OpenGL. Podíváme se na výběr knihoven pro implementaci naší aplikace. Další část je pak věnována návrhu aplikace a testování.

3.1 Integrace s Oculus Rift

Jak bylo zmíněno v kapitole 2.2.2, Oculus Rift SDK není jediným dostupným nástrojem pro vývoj s brýlemi Oculus Rift. My se však budeme držet zadání a použijeme tento nástroj. Toto SDK má zatím plnou podporu pouze pro platformu Windows. Do budoucna je však plánovaná podpora i pro OSX a Linux. Tento nástroj se vyvíjí poměrně rychle, a tudíž je důležité sledovat jeho vývoj. Dokonce již během vývoje této práce byl SDK aktualizován nejméně o čtyři verze.

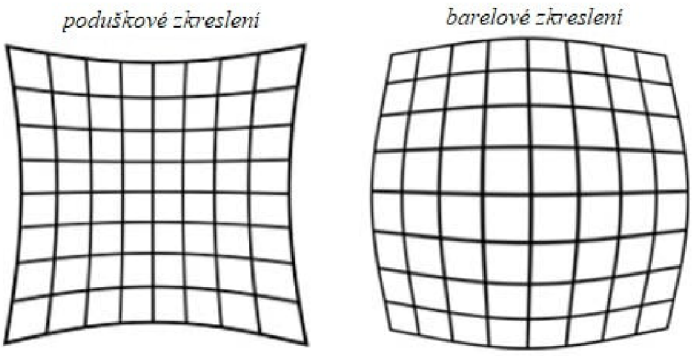
Oculus nabízí jednoduchou integraci HMD s profesionálními nástroji jako je Unity nebo Unreal Engine. Tato integrace ušetří vývojářům spoustu času a přebytečný kód jako výpočet polohy a tomu podobné. My však budeme vyvíjet bez těchto nástrojů, abychom zjistili, jaké jsou možnosti této knihovny s OpenGL [8].

3.1.1 Funkcionalita SDK

Oculus SDK je navrhnutý tak, aby byl uživatelsky pohodlný pro používání a v současnosti nabízí integraci pouze s jazyky C a C++. Toto SDK nám dává k dispozici informace o headsetu, který je připojený k počítači. Informace zahrnují typ HMD tzn. zda se jedná například o DK2 nebo CV1. Dále tyto informace zahrnují další doplňující položky o připojeném headsetu. Pro naše potřeby jsou podstatné jen položky výchozí hodnota dFOV a rozlišení displeje. Po inicializaci knihovny se zařízením Oculus Rift, máme zpřístupněné informace o poloze helmy v prostoru. Tato poloha a orientace odpovídá poloze brýlí v reálném čase. Existují dvě možnosti snímání pozice. První možnost je snímání pozice na úrovní podlahy (*z angl. floor-level*), kterou bychom měli použít v případě, že uživatel používá headset ve stoje a interaguje s objekty, které leží na zemi. Druhá možnost je snímání pozice na úrovní očí (*z angl. eye-level*), která je vhodnější, pokud uživatel sedí [8].

Jak bylo zmíněno v kapitole 2.1.1, některé čočky v HMD zvětšují zorné pole až na 120°. Má to však za následek deformaci obrazů, protože se obraz sbíhá do středu a vytváří poduškové zkreslení (*angl. pincushion distortion*) (viz Obrázek 3.1). Právě SDK má na starosti výslednou korekci obrazu. Před zobrazením výsledné scény SDK aplikuje barelové

zkreslení (*angl. barrel distortion*), díky kterému vyruší zkreslení čoček a dohromady se vytvoří čistý obraz.



Obrázek 3.1: Druhy zkreslení obrazu

3.2 OpenGL

Knihovna OpenGL (*Open Graphics Library*) byla navržená firmou SGI v roce 1992 a po krátké době se stala nejpoužívanějším standardem. Knihovna byla navržena tak, aby byla nezávislá na použitém operačním systému, grafickém ovladači a správci oken. Proto knihovna neobsahuje žádné funkce pro práci s okny nebo pro zpracování události. Navíc programátorské rozhraní knihovny bylo navrženo tak, aby byla knihovna použitelná v téměř libovolném jazyce [7].

Vykreslování scény se provádí procedurálně tzn. voláním funkcí OpenGL se vykreslí výsledný rastrový obrázek. Tento obrázek je uložený v tzv. framebufferu[1], kde jsou každému pixelu přiřazené atributy. OpenGL však nezaručuje, že stejná aplikace na různých platformách nebo různých grafických akcelerátorech zobrazí vždy stejný výsledek. Může to být způsobeno například odlišnými algoritmy pro interpolaci barvy. Celková geometrie a barevnost scény by měla být zachována [7].

Existuje i jiné grafické rozhraní, konkrétně DirectX, které můžeme integrovat s Oculus Rift SDK. Abychom však splňovali zadání naší práce, použijeme knihovnu OpenGL a pro její prvotní inicializaci použijeme knihovnu GLEW.

3.3 Ostatní knihovny

V této kapitole se budeme věnovat výběru dalších knihoven, které nám pomůžou při vývoji aplikace. Musíme vybrat knihovnu pro správu oken a vstupů a následně knihovnu pro zpracování modelů a textur.

3.3.1 Správa oken a vstupů

Protože OpenGL nenabízí správu oken, je nutné vybrat knihovnu, která toto umožňuje. Pro správu oken byla vybrána knihovna SDL (*Simple DirectMedia Layer*), která je dostupná

[1] https://www.khronos.org/opengl/wiki/Framebuffer

pro více operačních systému [1]. SDL je software s otevřeným zdrojovým kódem a je licencovaný pod LGPL[2]. Knihovna umožňuje práci se zvukem, klávesnicí, myší, ovladači a 2D i 3D počítačovou grafikou přes OpenGL. Tato knihovna je vytvořena v jazyce C a tedy díky interoperabilitě jazyka C, ji můžeme použít i v jazyce C++. Lze jí však propojit s téměř libovolným existujícím programovacím jazykem. Důvod výběru této knihovny je její jednoduchost a my jí budeme používat jako doplněk OpenGL k nastavení grafického výstupu a poskytnutí vstupu z klávesnice. Jako jinou alternativu bychom mohli zvolit knihovnu GLFW nebo GLUT, které nám taktéž poskytují okno pro grafický výstup a vstup z klávesnice.

3.3.2 Nahrávání modelů a textur

Pro nahrávání různých modelů do naší aplikace budeme používat knihovnu Assimp (*Open Asset import Library*). Tato knihovna umožňuje nahrávání několik různých formátu 3D modelů [6]. Knihovna byla vytvořena v jazyce C++ a je dostupná pod licencí BSD[3]. Kromě nahrávání modelů knihovna nabízí výpočet různých typů vektorů jako například normálový vektor. Důvodem výběru této knihovny je opět její jednoduchost zavedení do projektů. Při použití stačí pouze uvést cestu k modelu a příznaky načítání a následně už můžeme tento model zpracovávat.

Jelikož OpenGL nepodporuje nahrávání obrázků, budeme využívat knihovnu SOIL (*Simple OpenGL Image Loader*)[4]. Tato knihovna umožňuje jednoduché nahrávání textur do aplikace a dále už je na uživateli její další využití. Knihovna nepodporuje nahrávání přímo do OpenGL objektu tzn. s nahranými daty můžeme dělat další modifikace a teprve pak nahrát do objektu OpenGL.

3.4 Návrh aplikace

V následující části si navrhneme předlohu naších scén, abychom byli v obraze, jaké možnosti bude muset naše vykreslovací komponenta poskytovat. Následně se podíváme na návrh ovládání.

3.4.1 Návrh scén

Do aplikace jsme se rozhodli zahrnout tři scény, aby si uživatel mohl vyzkoušet různé varianty pohybů. Po spuštění aplikace je tedy automaticky zobrazena první scéna, která zároveň zobrazuje grafické uživatelské rozhraní přímo před uživatelem. Představa je taková, že tato scéna bude sloužit jako rozcestí pro další scény naší aplikace. Výběrem první scény zmizí grafické uživatelské rozhraní a scéna bude ihned připravená k pozorování. V této scéně nebude žádný pohyb po scéně, ale pouze statické pozorování kolem sebe. Bylo rozhodnuto, že tématem scény bude loď na moři, kde bude možné pozorovat pohyb hladiny a odraz krajiny kolem uživatele.

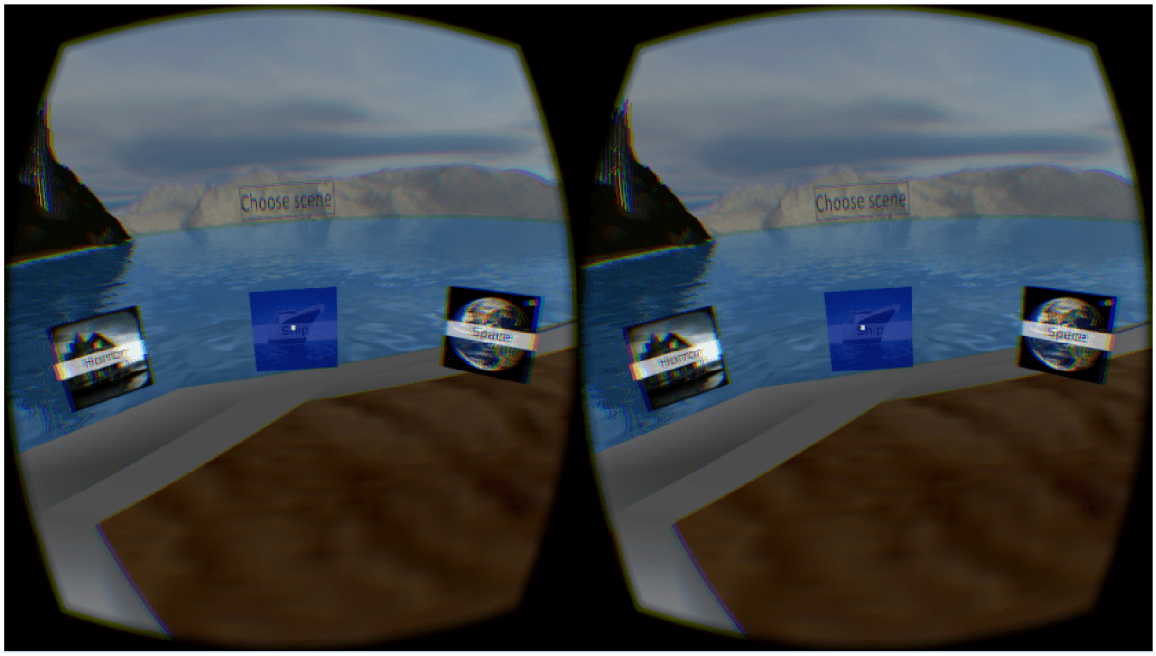
Výběrem druhé scény se přesuneme do kamenné budovy. Tato scéna bude na téma hororu a budou tomu odpovídat i její barvy. V této scéně se bude průběžně měnit světlo, a tedy budeme moci otestovat blikání textur. Na rozdíl od první scény zde bude krátký pohyb řízený aplikací. Třetí scéna se bude odehrávat ve vesmíru, kde budeme moci pozorovat

[2]LGPL je licence svobodného softwaru. http://wiki.knihovna.cz/index.php/GNU_LGPL
[3]BSD je licence svobodného softwaru. Volně šiřitelný ale musíme uvést autora a informaci o licenci
[4]https://www.khronos.org/opengl/wiki/Image_Libraries

neúplnou sluneční soustavu. V této scéně bude pohyb opět řízený aplikací ale oproti druhé scéně, bude tento pohyb velmi rychlý. Na těchto dvou scénách budeme moci otestovat vliv pohybu v aplikaci na uživatele.

3.4.2 Ovládání

Po spuštění aplikace bude k dispozici možný výběr z několika scén. Při návrhu ovládání si však musíme uvědomit, že uživatel bude mít připevněné zařízení HMD na hlavě. Z tohoto důvodu bude výběr scény spíše umožněn pomoci zmiňované techniky *ray-casting* a ne pomocí klávesnice. Takto zvolené ovládání bude pro uživatele nejpřirozenější. Uprostřed scény uživateli zobrazíme směr paprsku pomocí jednoduchého tvaru. Pokud směřuje správným směrem a protne námi definované uživatelské rozhraní, postupně toto rozhraní budeme ztmavovat a přibližně po dvou vteřinách se výběr potvrdí. Uživatelské rozhraní bude v podobě tlačítek, která budou horizontálně vedle sebe. Na každém tlačítku bude obrázek a nápis o jakou scénu se jedná. Návrat zpátky bude umožněn naopak pomocí klávesnice, a ne pomocí paprsku z důvodu, že nebude vhodné náhodně do scény vykreslovat jedno tlačítko.



Obrázek 3.2: Ukázka výběru scény pohledem v naší aplikaci

3.5 Návrh testování

Testovat aplikaci budeme i v průběhu vývoje, protože se chceme nejlépe zcela vyhnout nevolnosti v simulátoru. Tento druh nevolnosti má několik příznaku, mezi které patří dezorientace, porucha koordinace pohybů, nevolnost z pocitu zrychlení a bolestí oči. Někteří uživatele mohou pocítit tyto příznaky během krátké chvíle používání virtuálních brýlí, zatímco jiní se nemusí s příznaky setkat vůbec. Zpětná vazba je důležitá, protože si vývojáři mohou na zařízení HMD zvyknout a mohou vykazovat snížené příznaky oproti uživateli,

který se setká s virtuální realitou poprvé. V další části se budeme věnovat návrhu dotazníku, na který budou uživatele odpovídat po zkušenosti s naší aplikaci.

3.5.1 Návrh dotazníku

V této části si vytvoříme dotazník, který nám jednoduše odhalí, zda naše aplikace nevyvolává nějaké příznaky nevolnosti. Jedním z nejznámějších a zároveň nejvíce používaným dotazníkem je SSQ (*angl. simulator sickness questionnaire*). Tento dotazník má však nevýhodu, že dotazy zjišťují pocity uživatelů a někteří si nemusí být jistí svojí odpovědí. Tento dotazník je i časově náročný. Uživatel ho vyplňuje celkem pětkrát (před aplikací, po 10min, po 30min, po ukončení aplikace, 60min od ukončení aplikace). Nicméně oproti jiným metodám testování to má spíše své výhody. Stačí nám papír a tužka a nemusíme vlastnit žádná zařízení jako například metoda EEG (*angl. electroencephalogram*) [15]. Otázky dotazníku SSQ se ptají celkem na šestnáct symptomů, které se mohou objevit po krátké zkušenosti s virtuální realitou [15]. My však nakonec navrhli vlastní dotazník, který se skládá ze dvou částí. Vždy po prohlédnutí každé scény budou položeny tři otázky, na které účastník odpoví pouze ano nebo ne. Přitom však virtuální brýle neodkládají. Podle odpovědi bychom měli být schopni částečně vyhodnotit, zda scéna nevyvolává nevolnost. Nakonec po prohlédnutí všech scén, položíme účastníkovi doplňující otázky ohledně ovládání a celkového pocitu z virtuální reality. Po každé scéně se tedy zeptáme na následující otázky, jejichž odpovědi si zapíšeme.

- Vyvolává tato scéna jakýkoliv nepříjemný pocit? Může to být podobný pocit jako při jízdě automobilem nebo nepříjemný pocit oči.
- Měla scéna dostatečnou odezvu? Tzn. váš pohyb hlavy v realitě odpovídal pohybu ve scéně a nebyl žádný příznak trhání, a tudíž přeskakování snímku.
- Byla scéna po grafické stránce postačující? Neobjevují se ve scéně blikající textury?

Po prohlédnutí všech scén a po odložení zařízení, položíme doplňující otázky, které se budou týkat celkového hodnocení aplikace. Jedná se o tyto otázky.

- Máte nějaké příznaky nevolnosti po sundání brýlí? Jedná se o bolesti hlavy nebo břicha ale může to být například i pocení.
- Nebolí vás oči nebo nemáte rozmazané vidění?
- Vyhovovalo vám ovládání aplikace?
- Která scéna se vám líbila nejvíce?

Kapitola 4

Implementace

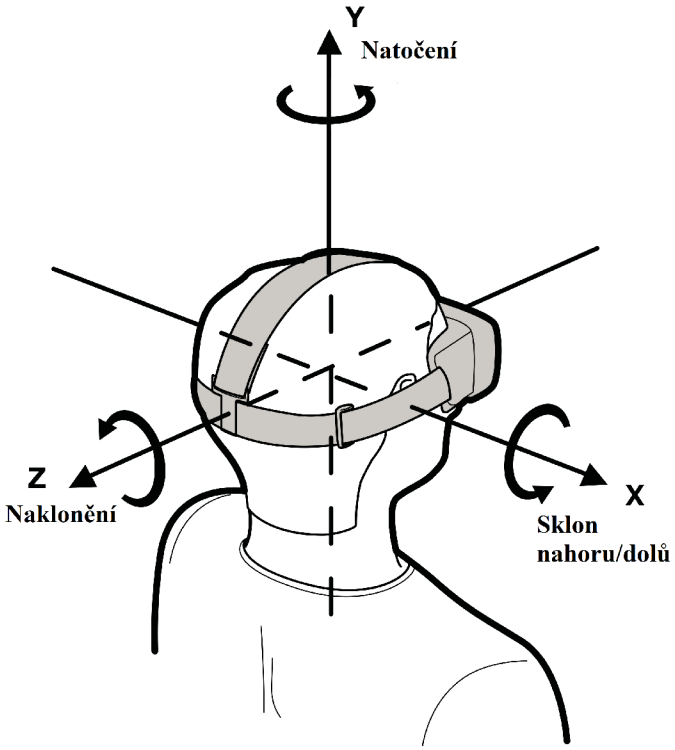
V této kapitole se podíváme na implementaci zajímavých komponent naší aplikace. Bude zde řešena inicializace knihovny Oculus Rift SDK a následně její použití s OpenGL. Dále nás čeká implementace vykreslovací komponenty, která obsahuje všechny atributy potřebné k vykreslení celé scény. Aplikace je vytvořena v jazyce C++ pouze pro platformu Windows, protože Oculus Rift SDK jinou platformu nepodporuje.

4.1 Zakomponování Oculus Rift SDK

Všechno, co je součástí Oculus Rift se nachází ve třídě `TrackerOculus`, aby byla aplikace oddělena od implementace rozhraní Oculus Rift SDK. V konstruktoru této třídy inicializujeme knihovnu pomocí metod `ovr_Initialize` a `ovr_Create` a následně musíme zavolat naši metodu `InitVRBuf`, která inicializuje další potřebné objekty ke správnému běhu aplikace. V této metodě inicializujeme struktury, které obsahují framebuffer, barevnou texturu a černobílou hloubkovou texturu. Protože se v naší aplikaci chceme vyhnout blikání a chceme dosáhnout vyšší kvalitu obrazu, aplikujeme metodu vyhlazování multisampling. Musíme tedy barevnou a hloubkovou texturu vygenerovat pomoci metody `glTexImage2DMultisample`, kterou nám poskytuje rozhraní OpenGL. Inicializování tohoto SDK tedy není tak přímočaré, jak bychom očekávali.

Každý snímek scény musíme vykreslit dvakrát, jednou pro levé oko a jednou pro pravé oko. Musíme tedy nejdřív zvolit displej, do kterého chceme scénu vykreslovat. Toho dosáhneme pomocí naší metody `prepareEye`, která nám připraví námi zvolený displej pro renderování. V tuto chvílí můžeme vykreslovat naši scénu a v případě, že jsme hotoví, musíme renderování pro daný displej ukončit a toho dosáhneme pomocí metody `onRenderFinish`. Až jsme hotoví s renderováním obrazů pro obě oči, musíme obrazy zobrazit v brýlích a toho docílíme pomocí naší metody `appendLayers`, která ve své kostře volá metodu `ovr_SubmitFrame`. Tato metoda je součástí Oculus Rift SDK a předává výsledné data ke zpracování, kde je na obraz nejdříve aplikovaný barelové rozmazání (zmiňovaný v kapitole 3.1.1) a následně je obraz zobrazen v brýlích.

Pro zjištění polohy a orientace hlavy musíme nejdříve zavolat metodu `ovr_GetEyePoses`, která nám vypočítá polohu v absolutním čase a výsledek uloží do struktury. Z této struktury následně můžeme extrahovat 3D vektor pozice a rotace, které jsou v pravotočivém souřadném systému. To znamená, že osa z směřuje dozadu, osa Y směřuje nahoru a osa X směřuje doprava (viz Obrázek 4.1).



Obrázek 4.1: Souřadný systém Oculus Riftu[1]

4.2 Implementace komponent

Všechny položky scény, tedy modely, pozadí, časovač a tomu podobné, se nachází ve třídě `Scene`. Tato třída je tedy jakýmsi kontejnerem pro všechny prvky v naší scéně a zároveň vykreslovací třídou. Nejdříve tedy musíme vytvořit obsah scény a následně můžeme vše přidat do této třídy. Z tohoto důvodu máme k dispozici třídu `SceneBuilder`, ve které vytváříme scénu.

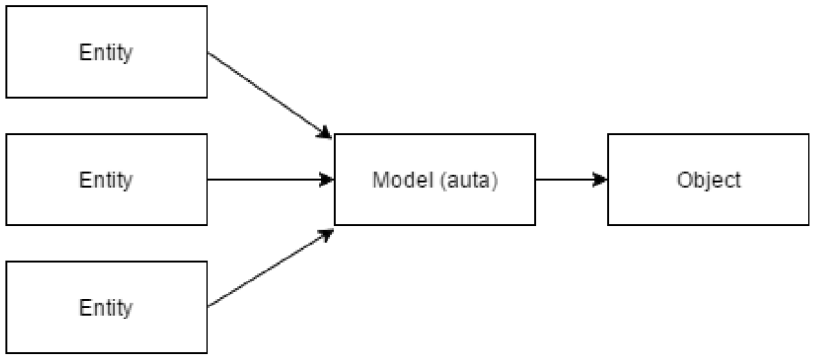
V této kapitole se tedy budeme věnovat implementaci komponentám ze scény, a nakonec konečnému vykreslování scény.

4.2.1 Modely

Pro přidání modelu do scény máme k dispozici třídu `Entity`. Tato třída nám dává k dispozici informace o modelu, a navíc poskytuje různé metody pro jejích modifikaci např. metoda `changePosition` změní aktuální pozici modelu. Mezi tyto informace patří například pozice, rotace nebo škálování a v neposlední řadě tato třída má ukazatel na třídu `Model`. A protože například model lidského těla může být složen z více části např. hlava, ruce, tělo a nohy, má třída `Model` vektor ukazatelů na třídu `Object`, která nese informace o konkrétní části modelu a její textury. Celé toto řešení je navrhnuté a implementované takto z několika důvodu. První důvod je přehlednost a druhý důvod je znuvupoužitelnost toho samého modelu a zvýšení výkonu aplikace. Například konkrétní model auta můžeme vykreslit třikrát, ale

[1] Převzato a přeloženo z: https://developer3.oculus.com/documentation/pcsdk/latest/concepts/dg-sensor/

pokaždé na jiné pozici a s jinou rotaci, protože tři instance třídy `Entity` mohou nést jiné údaje o rotaci, pozici a tomu podobné, ale všechny instance mají ukazatel na stejný model auta (viz Obrázek 4.2).



Obrázek 4.2: Příklad použití tříd Entity a Model

Pro nahrávání modelů byla využita zmiňovaná knihovna Assimp, která nám usnadnila práci s načítáním různých formátu modelů. Funkčnost této knihovny je obalena ve třídě `ModelLoader`, která má ve svém rozhraní důležitou metodu `loadModel`, pomocí které získáme instanci třídy `Model`. Tato třída je vázaná na další dvě třídy, které pracují s funkcemi OpenGL. Konkrétně se jedná o třídy `ObjectManager` a `TextureLoader`, která navíc pracuje s knihovnou SOIL. Jak jejich název napovídá, třídy se starají o nahrávání textur a konkrétních části modelů do OpenGL objektů (*Buffer Objects*)[2].

4.2.2 Osvětlení

Do scény můžeme přidat až tři druhy světla a liší se podle toho jakou mají intenzitu a jaký je jejích zdroj. První druh světla je směrové světlo (*z angl. directional light*) a zdrojem takovéhoto světla je například slunce. V tomto případě můžeme předpokládat, že všechny paprsky dopadají na objekt pod stejným úhlem a se stejnou intenzitou. Ve fragment shader programu je tento druh světla vypočítáván pomocí funkce `CalcDirLight`. Dalším druhem je světlo, které má zdroj v nějakém bodě a vyzařuje paprsky do všech směrů např. žárovka. V tomto případě máme možnost určit intenzitu světla, která se zmenšuje s rostoucí vzdáleností. Intenzita je vypočítána pomocí rovnice 4.1, kde d je vzdálenost fragmentu od zdroje světla, hodnota K_c je konstanta rovna jedné a hodnoty K_l a K_q jsou vhodně zvolené hodnoty[3]. Pro tento druh světla je implementována funkce `CalcPointLight`, která je opět ve fragment shaderu.

$$x = \frac{1.0}{K_c + K_l \cdot d + K_q \cdot d^2} \tag{4.1}$$

Posledním typem je světlo vyzařující jedním směrem a slouží například jako baterka. Tento typ světla je implementován podobně jako světlo vycházející z jednoho bodu.

Aby bylo naše světlo co nejvíce reálné a uživatel byl co nejvíce pohlcen virtuálním světem je výsledné světlo pixelu vypočítáno pomocí čtyř různých textur. Pro nás je nejvíce zajímavá

[2] Více informací zde: https://www.khronos.org/opengl/wiki/Buffer_Object
[3] http://www.ogre3d.org/tikiwiki/tiki-index.php?page=-Point+Light+Attenuation

textura zobrazující normálový vektor pro každý pixel (*z angl. normal map*), protože výsledek může způsobit blikání výsledné scény a my to budeme moci otestovat.

Pro světlo je tedy implementována třída `Light`, která obsahuje atributy jako je pozice, barva, intenzita a tomu podobné. Do scény můžeme přidat všechny tři typy světla zároveň.

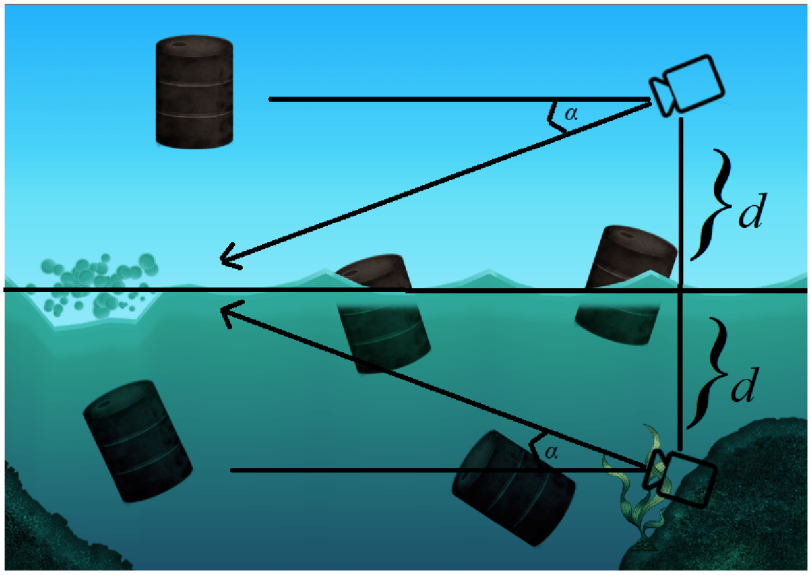
4.2.3 Vodní plocha

V jedné z naších scén se vyskytuje voda, a tudíž se budeme věnovat i její implementaci. Voda je vykreslována jen pomocí šesti vrcholů, tedy dvou trojúhelníku, které spojením vytváří čtverec. Na výslednou plochu čtverce vykreslujeme dvě textury přes sebe. Jedna textura zobrazuje odraz scény nad vodou (*angl. reflection texture*) a druha textura zobrazuje scénu pod vodou tedy refrakční textura (*angl. refraction rexture*). Abychom takovéto textury získali, vykreslujeme scénu vícekrát. Jednou pro získání odrazové textury a jednou pro získání refrakční textury. A jelikož scéna je zobrazena ve virtuálních brýlích, celkem jí vykreslujeme šestkrát.

Nejdříve si připravujeme scénu pro vykreslení odrazové textury. To zahrnuje připravení framebuffer objektu, posunuti kamery pod vodu, invertování úhlu a konečně vykreslení scény (Obrázek 4.3). Ve vertex shader programu pak pomoci vestavěné funkce `gl_ClipDistance` zabráníme renderování modelů, které ve výsledné textuře být nemají. To znamená, že pokud vykreslujeme scénu z pod hladiny, výsledná textura neobsahuje modely, které jsou pod vodou.

Refrakční texturu získáváme vykreslením scény z původního místa a jediným rozdílem je, že v tomto případě nevykreslujeme objekty, které jsou nad hladinou nýbrž pod hladinou.

Pomocí DuDv textury[4] je zkreslena textura na hladině a vytvořena iluze pohybu hladiny Nakonec aplikujeme zjednodušený Fresnelův efekt tzn. pokud se díváme do vody přímo ze shora, nevidíme žádný odraz, ale pouze hladinu nebo dno. Pro vodu je tedy naimplementována třída `Water`, která nese informace pouze o velikosti vody a texturách.



Obrázek 4.3: Změna pozice kamery pro získání odrazové textury.[5]

[4]Další informace o DuDv: http://wiki.polycount.com/wiki/DuDv_map

4.2.4 Navigace ve scéně

V aplikaci jsou implementovány dva způsoby ovládání. První způsob ovládání je pomocí klávesnice a její implementace se nachází ve třídě `Window`, která mimo jiné vytváří i hlavní okno aplikace. Události zachytáváme pomocí metody `handleInput`, ve které testujeme stisk kláves.

Druhý způsob ovládání je pomocí třídy `PointPicker`, ve které testujeme, zda paprsek protíná objekt. Pomocí této třídy jsme schopni vybrat pohledem scénu z menu. Testujeme pomocí metody paprsku a koule (*angl. ray-sphere intersection*)[6], která nezaručuje stoprocentní přesnost, ale pro naše účely plně vystačuje. Protnutí paprsku a koule je vypočítáváno pomocí rovnice 4.2, kde testujeme, zda je větší jako nula.

$$b^2 - c \geq 0 \tag{4.2}$$

$$b = \vec{d} \cdot \vec{v} \tag{4.3}$$

$$b = \vec{v} \cdot \vec{v} - r \tag{4.4}$$

Vektor $\vec{d}$ je směr paprsku, tedy směr pohledu uživatele. Vektor $\vec{v}$ je vzdálenost objektu od uživatele a r je poloměr objektu.

4.2.5 Vykreslování

Pro vykreslování scény je vytvořena abstraktní třída `Renderer`, která vytváří rozhraní pro komponenty tím, že vynucuje implementaci metody `render`. Navíc tato třída má ukazatel na třídu `Scene`, pomocí které vykreslujeme celou scénu. Renderování je naimplementováno tímto způsobem z důvodu, že například třída `SceneWithWaterRenderer`, potřebuje scénu vykreslovat vícekrát kvůli získání textur pro hladinu, zatímco třída `BasicSceneRenderer` vykresluje scénu pouze jednou pro každý displej.

Celé toto řešení zastřešuje třída `SceneChanger`, která má ukazatel na všechny scény a pomocí metody `getActualRenderer` získáme aktuální vykreslovací třídu se scénou, kterou máme vykreslit. Tato třída navíc rozšiřuje abstraktní třídu `ChangeSceneListener`, která vynucuje implementaci metody `onChange`. Tímto způsobem naslouchám požadavkům na změnu scény[7].

[5] Převzato a upraveno z: http://answers.unity3d.com/questions/1263793/apply-overlay-effect-in-an-area-on-specific-object.html

[6] http://antongerdelan.net/opengl/raycasting.html

[7] Observer Pattern. https://www.tutorialspoint.com/design_pattern/observer_pattern.htm

Kapitola 5

Testování

V této kapitole se podíváme na výsledky testů, které by nám měli sdělit, zda aplikace byla správně navržena, a naimplementována a jestli způsoby nevolnost v simulátoru. Aplikace byla testována i v průběhu implementace a byla neustále vylepšována.

Poprvé byla testována pouze první scéna, která simulovala pohyb na lodi. Očekávali jsme však nepříznivé výsledky, protože jsme se záměrně neřídili ověřenými způsoby návrhu aplikací pro virtuální realitu. Jak jsme tedy očekávali, výsledky byly velmi nezdařilé a kvůli tomu byla tato scéna nakonec upravená a pohyb lodi byl odstraněn. Loď je nyní ve scéně pouze staticky na moři. Bez výjimky totiž všechny uživatelé začala bolet hlava a už nechtěli podobný zážitek znovu zažít. Je to z důvodu, že jsme programově zasahovali uživateli do pohybu hlavy, abychom tím simulovali pohyb vln. Uživatel pohyb vln ve skutečnosti necítil, pouze se mu měnil obraz ve virtuální realitě, a tudíž jsme vyvolali nevolnost.

Další testování během implementace bylo zdařilé, a tudíž žádné další zásadní změny aplikace nebyly provedeny a vše tedy pokračovalo podle návrhu.

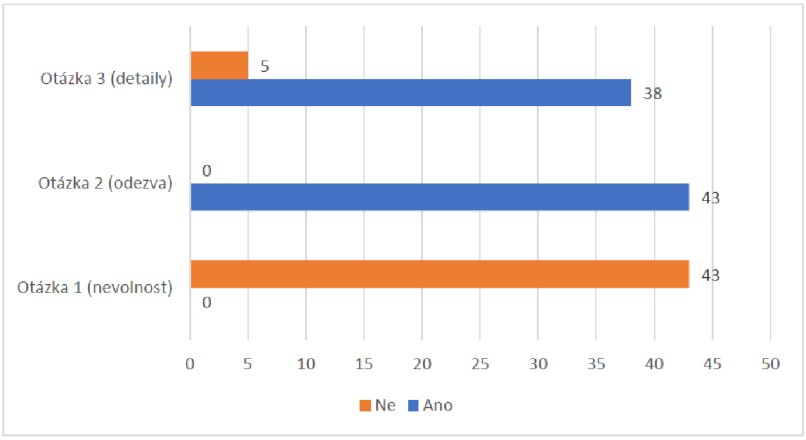
5.1 Závěrečné testování a vyhodnocení

Závěrečné testování bylo provedeno podle navrženého dotazníku. Celkem byla aplikace otestována na čtyřiceti třech lidech. Každý z účastníku si vyzkoušel postupně všechny scény, během kterých odpovídali na otázky a následně všichni vyplnili nachystaný dotazník. Podle vyhodnocení testů se účastníci nesetkali s žádným vážným problémem vyvolávající nevolnost v simulátoru.

Výsledky testů jsou tedy velmi kladné, a tudíž téměř žádné další zásahy do aplikace po konečném vyhodnocení neproběhly. Hlavním důvodem může být jak dodržování teorie navrhování aplikace pro virtuální realitu, tak i průběžné testování scén. Výsledek testování však nemusí být naprosto přesný, důvod může být například, že uživatelé byli krátkou dobu v aplikaci, v průměru do šesti minut. Někteří jednotlivci však scény zhlédli několikrát, a i přesto je nevolnost v simulátoru nepostihla. Ostatní uživatelé si prohlédli všechny scény aplikace sice pouze jednou, ale aby mohli vyplnit dotazník a ohodnotit naši aplikaci to stačilo, protože testujeme pouze naší aplikaci, a ne virtuální realitu celkově. Další důvod nepřesnosti může být, že se symptomy nevolnosti v simulátoru objeví až později, tedy po zodpovězení dotazníku.

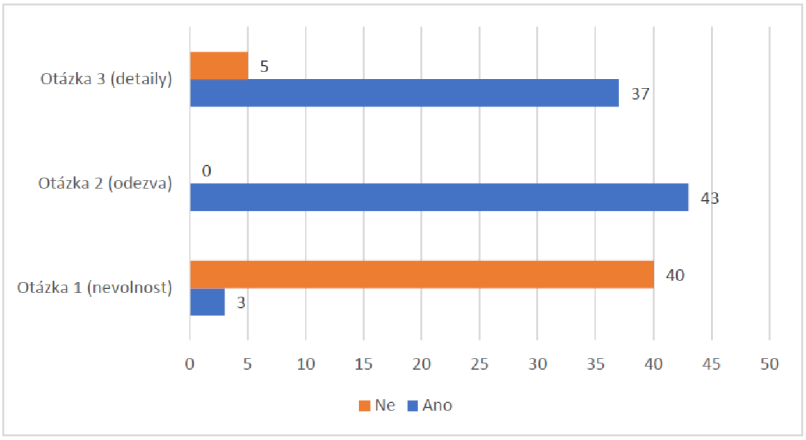
První scéna nevyvolávala nepříjemný pocit u žádného pozorovatele a odezva obrazů odpovídala pohybu hlavy v realitě, tudíž v této scéně k žádným změnám po testování nedošlo. Z grafu 5.1 můžeme vyčíst, že pět uživatelů nebylo spokojeno s grafickou stránkou

první scény. Nicméně po diskuzi s uživateli jsme zjistili, že kladou příliš vysoké nároky na všechny aplikace.



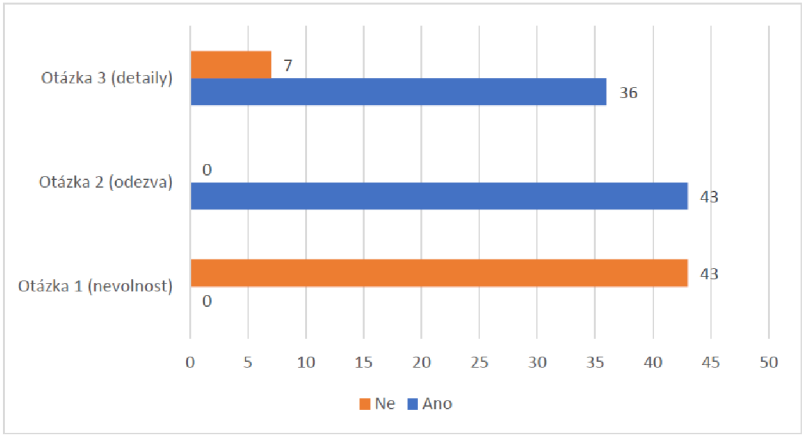
Graf 5.1: Vyhodnocení dotazníků z první scény

V hororové scéně byla po testování nepatrně snížena rychlost pohybů z důvodu, že tři uživatelé odpověděl první otázku pozitivně (viz graf 5.2), a tudíž měli nepříjemný pocit z pohybu. Pro nás je to téměř sedm procent a náš požadavek je odstranit ideálně všechny nepříjemné stavy. Jak bylo zmiňováno v kapitole 3.5, někteří mohou pocítit nevolnost během krátké chvíle ve virtuální realitě a naším cílem je nevyvolat nevolnost nejlépe u žádného z uživatelů. Další výsledky testování dopadly obdobně jako v první scéně. Odezva byla opět v pořádku a nikomu se nestal případ, že by se zobrazení ve virtuálním světě zobrazovalo trhaně.



Graf 5.2: Vyhodnocení dotazníků z druhé scény

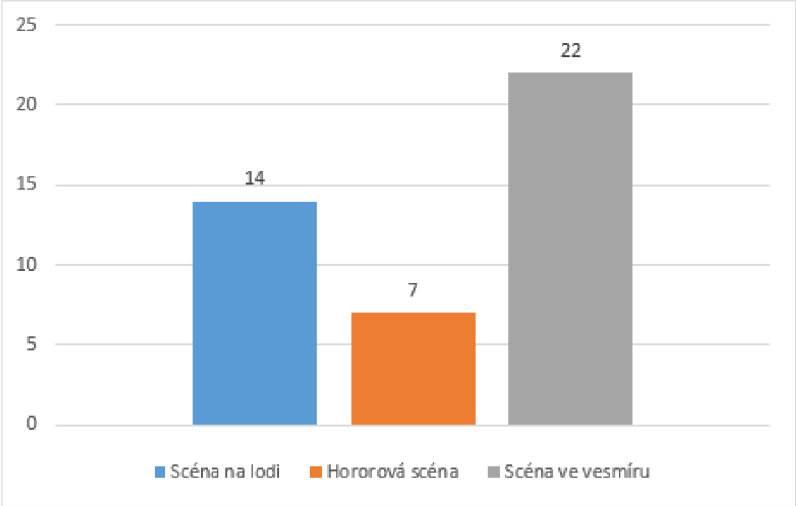
Ve vesmírné scéně opět nebyla provedena žádná změna po testování. Co se týče grafické stránky scény, někteří uživatelé si povšimli občasného probliknutí textur na planetách, pokud byly tyto objekty ve větší vzdálenosti od pozorovatele. I přesto se scéna uživatelům po grafické stránce líbila a tohoto občasného probliknutí si povšimli, až jsme je na to upozornili. Na výsledný graf to tedy nemělo téměř žádný vliv. V ostatních bodech byla tato scéna hodnocena velmi pozitivně.



Graf 5.3: Vyhodnocení dotazníků z třetí scény

Po prohlédnutí všech scén jsme pokládali doplňující otázky, které můžeme vyhodnotit velmi pozitivně. Nikdo z uživatelů neměl potíže se zrakem ani nepociťoval žádné příznaky nevolnosti v simulátoru po sundání brýlí i přesto, že pohyb ve druhé scéně vyvolal nepříjemný pocit u třech uživatelů. Co se týče ovládání, většina pozorovatelů neměla problém s ovládáním pomocí pohledu. Někteří však očekávali, že budou mít k dispozici ovladače *Oculus Touch* a také volný pohyb prostorem, bohužel jsme ani jedno neměli k dispozici.

Uživatelům se nejvíce líbila třetí scéna, tedy scéna ve vesmíru, ačkoli byla po grafické stránce hodnocená nejhůře. Naopak nejméně se uživatelům líbila druhá scéna, která byla navržena v hororovém duchu. Předloha této scény měla velký vliv při konečném výběru, protože nikdo z uživatelů tento typ neupřednostňoval.



Graf 5.4: Nejoblibenější scéna

Kapitola 6

Závěr

Na začátku práce bylo nutné nastudovat virtuální realitu a seznámit se s Oculus Rift a jeho SDK. Následně bylo cílem bakalářské práce bylo navrhnout aplikaci pro Oculus Rift v kombinaci s OpenGL. Nicméně tato integrace nebyla zcela jednoduchá kvůli nedostatku zdrojů. Byly nalezeny pouze dva důvěryhodné zdroje, jak využít SDK s OpenGL a jedním z nich byla domovská stránka Oculus Rift. Je to nejspíše z důvodu, že Oculus Rift SDK lze jednoduše integrovat s herními enginy jako je Unity nebo Unreal a málokdo využívá integraci s OpenGL.

V průběhu práce jsme se seznámili s pojmem virtuální realita a headsetem Oculus Rift, které toto prostředí vytváří. Jednou z hlavních komplikací pro minulé i stávající headsety je nevolnost v simulátoru, se kterou jsme se museli vypořádat i v naši aplikaci. Proto jsme se seznámili s lehce odlišným způsobem návrhu aplikací, který se snaží omezit nevolnost v simulátoru, který mohou uživatelé pociťovat během používání aplikace. Díky těmto poznatkům byla navržena aplikace obsahující tři scény, které by podle návrhu neměly vyvolávat nevolnost v simulátoru. Jelikož bylo nezpůsobovat nevolnost během používání tak důležitým požadavkem, byla aplikace testována nejen po dokončení vývoje, ale také v průběhu vývoje. Díky studiu příčin nevolnosti a detailnímu návrhu zaměřenému na její omezení, proběhlo závěrečné testování velmi kladně. Z výsledků je patrné, že naše aplikace nezpůsobovala žádné potíže a uživatelé byli převážně spokojeni.

Aplikace je navržena tak, aby ji bylo možné lehce rozšířit. Do aplikace lze jednoduše přidat další scény a umožnit jejich výběr nově přidaným tlačítkem z menu. Mezi další možná rozšíření patří zvuková složka, která by zajisté obohatila celkový požitek z virtuální reality.

Literatura

[1] *About SDL* . Libsdl, [Online; navštíveno 08.04.2017].
URL https://www.libsdl.org/

[2] *Binocular Vision, Stereoscopic Imaging and Depth Cues* . Oculus VR, [Online; navštíveno 21.02.2017].
URL https://developer3.oculus.com/documentation/intro-vr/latest/concepts/bp_app_imaging/

[3] *Head-mounted Displays (HMDs)*. Virtual Reality Society, [Online; navštíveno 12.01.2017].
URL https://www.vrs.org.uk/virtual-reality-gear/head-mounted-displays/

[4] *Informativní tabulka rychlosti* . [Online; navštíveno 10.03.2017].
URL vorce.webgarden.cz/rubriky/vzorečky-fyzika/pohyb/informativni-tabulka-rychlosti

[5] *Oculus Best Practices* . Oculus VR, [Online; navštíveno 22.02.2017].
URL https://developer3.oculus.com/documentation/intro-vr/latest/concepts/book-bp/

[6] *Open Asset Import Library* . Assimp, [Online; navštíveno 08.04.2017].
URL http://assimp.sourceforge.net/

[7] *OpenGL Overview*. OpenGL, [Online; navštíveno 07.04.2017].
URL https://www.opengl.org/about/

[8] *PC SDK Getting Started Guide* . Oculus VR, [Online; navštíveno 01.02.2017].
URL https://developer3.oculus.com/documentation/pcsdk/latest/concepts/book-gsg/

[9] *User Interfaces for VR* . Unity Corporation, [Online; navštíveno 15.03.2017].
URL https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr

[10] *Virtual Reality Best Practices* . Unreal Engine, [Online; navštíveno 23.02.2017].
URL https://docs.unrealengine.com/latest/INT/Platforms/VR/ContentSetup/index.html

[11] *Virtuální realita*. [Online; navštíveno 12.01.2017].
URL http://it.pedf.cuni.cz/~bobr/ucspoc/virtreal.htm

[12] Kumparak, G.: *A Brief History Of Oculus*. 2014, [Online; navštíveno 27.01.2017].
URL https://techcrunch.com/2014/03/26/a-brief-history-of-oculus/

[13] Lang, B.: *Fove Eye-tracking Headset Gets Final Specs and Pre-order Date* . 2016, [Online; navštíveno 20.02.2017].
URL http://www.roadtovr.com/fove-0-eye-tracking-vr-headset-final-specs-pre-order-date/

[14] Martindale, J.: *How should we move around in vr? Nobody has figured it out yet* . 2016, [Online; navštíveno 11.03.2017].
URL https://www.digitaltrends.com/virtual-reality/vr-locomotion-movement-omni-hover-junkers/

[15] M.LaValle, S.: *Virtual Reality*. 2017, [Online; navštíveno 08.01.2017].
URL http://vr.cs.uiuc.edu/vrbook.pdf

[16] Nield, D.: *How Oculus Rift works: Everything you need to know about the VR sensation*. 2016, [Online; navštíveno 20.02.2017].
URL https://www.wareable.com/oculus-rift/how-oculus-rift-works

[17] Petřík, J.: *Virtuální realita pod lupou: cizí svět na dosah* . 2016, [Online; navštíveno 15.01.2017].
URL https://doupe.zive.cz/clanek/virtualni-realita-pod-lupou-cizi-svet-na-dosah

[18] Prasuethsut, L.: *HTC Vive: Everything you need to know about the Steam VR headset* . 2016, [Online; navštíveno 13.02.2017].
URL https://www.wareable.com/vr/htc-vive-vr-headset-release-date-price-specs-7929

[19] Reynolds, E.: *Virtual Reality makes avatars more important than ever* . [Online; navštíveno 16.03.2017].
URL https://motherboard.vice.com/en_us/article/vr-makes-avatars-more-important-than-ever

[20] Steve, A.; Blatner, D.: *Reálně o virtuální realitě : umění a věda virtuální reality*. Jota, 1994, ISBN 80-85617-41-2.

Přílohy

Příloha A

Obsah CD

Přiložené CD obsahuje:

- Zdrojové soubory textu bakalářské práce
- Text bakalářské práce ve formátu .pdf
- Přeložené binární soubory
- Zdrojové kódy aplikace