Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Užívateľské rozhranie vo virtuálnej realite

Bakalárska práca

2019
Marek Fedák

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Užívateľské rozhranie vo virtuálnej realite

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	Mgr. Andrej Mihálik, PhD.

Bratislava, 2019
Marek Fedák

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Marek Fedák
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Užívateľské rozhranie vo virtuálnej realite
Virtual reality user interface design

Anotácia: Novým smerom, ktorý odráža pokrok v počítačovej vizualizácii a animácii, je virtuálna realita. Virtuálna realita je v súčasnosti nová technológia, ktorá sa zrodila v ére informatiky zo syntézy viacerých odborov, počnúc robotikou, chémiou, matematikou, vedou o počítačoch až po umenie. Vo virtuálnej realite dochádza k sprostredkovanej interakcii medzi užívateľom a vlastným virtuálnym prostredím. Človek môže fyzické objekty vytvorené počítačom nielen pozorovať zo všetkých strán, ale tiež s nimi ľubovoľne manipulovať. Je však potrebné vytvoriť vhodné ovládanie, kde manipulácia s prvkami virtuálneho prostredia by mala byť jednoduchá a intuitívna.

Cieľ: Preskúmať existujúce riešenia a navrhnúť vlastné intuitívne užívateľské prostredie pre manipuláciu s prvkami vo virtuálnej realite. Implementovať a vyhodnotiť navrhnuté užívateľské rozhranie v jednoduchom virtuálnom prostredí.

Literatúra: https://unity3d.com/learn/tutorials/topics/virtual-reality/interaction-vr

Kľúčové slová: užívateľské rozhranie, virtuálna realita

Vedúci: Mgr. Andrej Mihálik, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 29.09.2017

Dátum schválenia: 17.04.2019 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Ďakujem vedúcemu bakalárskej práce Mgr. Andrejovi Mihálikovi, PhD. za jeho cenné poznatky, rady a pripomienky, ktorými mi bol nápomocný pri tvorbe tejto práce.

Abstrakt

V tejto bakalárskej práci sme sa venovali skúmaniu implementácií užívateľských prostredí vo virtuálnej realite. Analyzovali sme ich a následne navrhli vlastnú implementáciu. Sústredili sme sa hlavne na zlepšenie častí prostredia, v ktorom sme videli možné zlepšenie pre určité použitia. Tento návrh sme plne implementovali v programe Unity. Naše prostredie obsahovalo plne implementovaný, dynamicky sa správajúci kurzor, rozoznávanie tvarov a gest. Následne sme našu implementáciu nechali otestovať niekoľkými respondentami. Pri testovaní sme skúmali efektivitu, intuitívnosť, jednoduchosť a viacúčelovosť našej implementácie v rôznych častiach programu. Po otestovaní sme vyhodnotili výkony ľudí, čo potvrdilo výskyt vlastností, ktoré sme sa snažili pridať.

Kľúčové slová: virtuálna realita, užívateľské prostredie, kurzor, gesto, vzor

Abstract

In this bachelor's thesis we focused our attention on discovering different kinds of implementations of user interface in virtual reality. We analyzed them and later we suggested our own implementation. We focused mainly on improving different parts of interface, in which we saw possible improvements, which were visible in certain types of uses. We fully implemented this design in Unity. Our interface contained fully implemented, dynamically changing cursor, pattern and gesture recognition. We asked couple of respondents to test the implementation. During testing period we examined efficiency, simplicity, multifunctionality and intuitiveness of our implementation in different parts of program. After tests we evaluated performances of people, which confirmed presence of properties we tried to provide.

Keywords: virtual reality, user interface, cursor, gesture, pattern

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Väčšina z nás pravdepodobne nikdy nenavštívi 7 divov sveta, nebude lietať balónom, nenavštívi iné planéty, či sa nepozrie do najhlbších končín oceánu. Avšak s rozmachom virtuálnej reality, každý z nás bude môcť okúsiť čaro všetkých týchto momentálne nedosiahnuteľných zážitkov.

Prístroje, ktoré vedia sprostredkovať virtuálnu realitu si našli miesto v srdciach mnohých ľudí. Rozšírenie týchto prístrojov na každodenné použitie znamenalo aj pokročenie v softvérovej časti tohto sektoru. Ich používanie musí byt plynulejšie, pohodlnejšie a každým dňom aj zaujímavejšie. Aby si niekto mohol naplno užiť všetko, čo virtuálna realita ponúka, je potrebné mu sprostredkovať vhodné nástroje na manipuláciu v tomto prostredí. Užívateľské prostredie by preto malo ponúkať širokú škálu interakcií s prostredím a objektami v tomto prostredí. Malo by byť nenáročné používať toto prostredie hocikým.

Cieľom tejto práce bolo vytvoriť presne takéto prostredie, ktoré by ponúkalo užívateľom hlbšie sa ponoriť do sveta virtuálnej reality. Vyvinúť prostredie, s ktorým bude možne poskytnúť pestrejšiu manipuláciu so všetkým, na čo môžeme vo virtuálnej realite natrafiť. Prostredie, ktoré by sa po chvíľke používania stalo intuitívnym.

V prvej kapitole si povieme niečo o histórii virtuálnej reality. Povieme si, ako sa vyvíjal dizajn zariadenia a ako sa táto technológia zdokonaľovala až do dnešných čias. Tiež si povieme niečo o užívateľských prostrediach, aké prvky využívajú ako medzivrstvu medzi virtuálnou realitou a človekom.

Nasledovať bude kapitola, kde sme sa pozerali, ktorým smerom chceme ísť s vývojom nášho prostredia. Preskúmať ich zákutia a zistiť, v čom by sa dali zmeniť alebo zlepšiť. Zistiť, čo iné by sme vedeli ponúknuť od prostredí, ktoré už existujú.

Ďalším z našich cieľov je samozrejme naimplementovať nami navrhnuté užívateľské prostredie. Podeliť sa, ako sme museli riešiť určité problémy, ktoré sa počas vývoja tohto prostredia vyskytli.

Nakoniec sme nami implementované prostredie otestovali s ľuďmi. Niektorí z nich sa s virtuálnou realitou už stretli, no pre pár z nich to bolo prvé stretnutie s virtuálnou realitou. Spýtali sme sa na ich názory, ako si to predstavovali a aké zmeny by radi privítali.

Kapitola 1

Virtuálna realita - VR

V tejto kapitole si ukážeme, že virtuálna realita síce vyzerá ako novinka z 21. storočia, no jej koncept pochádza z omnoho starších čias. Zoznámime sa s pojmom virtuálnej reality, vysvetlíme si pojmy nepochybne sa spájajúce s virtuálnou realitou a užívateľským prostredím.

1.1 Čo je to virtuálna realita

Virtuálna realita je prostredie generované počítačom, ktoré má za úlohu buď napodobňovať skutočnosť, alebo môže vytvárať celkom nové, vymyslené prostredie. Týmto vzniká u používateľa vizuálny zážitok, ktorý sa vykreslí na obrazovke počítača alebo ho uvidí cez špecifické stereoskopické okuliare. Tam potom nasleduje interakcia medzi používateľom a objektami umiestnenými v prostredí, respektíve interakcia so samotným prostredím.

1.2 História virtuálnej reality

Keď sa spomenie virtuálna realita, veľa ľudí sa myšlienkami obráti k modernej technológii, vzniknutej v posledných desaťročiach, ktorá sa v dnešnej dobe používa v rôznych počítačových alebo mobilných zariadeniach. Opak je však pravdou a samotný koncept virtuálnej reality siaha až do tridsiatych rokov minulého storočia. S určitými medzerami, vývoj smerujúci k dnešnej technológii, ktorý virtuálna realita používa, trval vyše 60 rokov [8], pričom jeho hlavné etapy sa odohrávali v nasledujúcich rokoch:

- 1930 - spomenutý prvý koncept VR okuliarov
- 1960 - vytvorenie prvého systému používaného v armáde
- 1990 - vytvorenie prvého systému pre domáce používanie

- 2014 - plne funkčný VR systém
- dnešná doba - vytváranie nových doplnkov a technológií pre virtuálnu realitu

Vývoj v 1930 - v príbehu Pygmalions Spectacles, Stanley G. Weinbaum ukazuje po prvýkrát koncept hrania hier či ukazovania filmov pomocou zariadenia pripomínajúceho okuliare. Rozpráva tam o tom, ako si jednotlivec mohol užívať celkom iný svet, nachádzajúci sa za sklíčkami okuliarov. Tam si hlavný hrdina mohol vychutnávať nielen vizuálny obraz, ale taktiež zvuk, chuť, čuch a hmat. Aj keď nie je jednoduché implementovať všetky tieto zmysly do virtuálnej reality, udržujeme si to ako cieľ, ku ktorému sa chceme jedného dňa dostať, aby sme dosiahli plný zážitok pri používaní virtuálnej reality.

Vývoj v 1960 - už o 30 rokov od spomínaného prvého konceptu sa začalo s vývojom prvých aparátov pre virtuálnu realitu. Ivan Sutherland, americký vedec a priekopník v odbore počítačovej grafiky, vytvoril prvý aparát, ktorý mal slúžiť pre armádne účely. Použitím špeciálneho softvéru a platformy na riadenie pohybu, sa začal tento aparát využívať v armáde na tréningové cvičenia. V dnešnej dobe sa používanie zariadení virtuálnej reality stalo v armáde štandardom, aby vedeli pripraviť vojakov na boj v teréne v bezpečných, no napriek tomu realistických podmienkach. Na rozdiel od prístrojov, ktoré sú nám bežne dostupné, armáda má k dispozícií oveľa pokročilejšie technológie. Tie sa budú aj naďalej vyvíjať v rýchlejšom tempe ako produkty určené pre bežných ľudí.

Vývoj v 1990 - v deväťdesiatych rokoch prichádzajú Nintendo a Sega s prvými systémami pre domáce použitie. Nintendo prišlo so systémom Virtual Boy, ktorý sa považuje za prvý domáci systém, ktorý bol vysoko rozšírený. Jeho nedostatkami však boli hry vytvorené exkluzívne v červeno-čiernych farbách a limitovanému počtu zariadení, ktoré sa dali k tomuto systému pripojiť. Aparát na hlave bol nepohodlný a kvôli nedostatku hier Nintendu nepriniesol zisk ako jeho iné konzoly v tom čase. Systém, ktorý priniesla Sega, obsahoval LCD obrazovku, systém sledovania hlavy a stereo. S danými technickými poznatkami sa cena vývoju projektu a produktu vyšplhala až príliš vysoko, čo znamenalo pre Segu úplný prepadák.

Vývoj v 2014 - v roku 2014, keď Facebook odkúpil Oculus VR systém, znamenalo to veľkú revolúciu pre virtuálnu realitu. Oculus Rift začal ako kickstarter projekt a dohoda s Facebookom znamenala lepšie financovanie projektu a taktiež záujem svetových vývojárov o túto technológiu. Oculus uzavrel niekoľko dohôd s inými firmami a spolu s podporou, ktoré svojmu projektu dodávali, to bol veľký krok pre virtuálnu realitu. Po tom, ako získali svetové uznanie od viacerých veľkých firiem, ukázali tým záujem o túto technológiu a tým prispeli k vzniku iných firiem, ktoré začali vyrábať vlastné VR systémy. Na rozdiel od predchádzajúcich VR systémov, ktoré boli technicky zložité pre vtedajšie počítače a ich využitie bolo limitované, Oculus upravil svoj systém tak,

aby bol kompatibilný s dnešnými počítačmi a priniesol širokú paletu aplikácií pre tento systém.

Vývoj v dnešnej dobe - veľký počet firiem sa snaží prísť na trh s novými, inovatívnymi príslušenstvami, ktoré rozšíria zážitok používateľa vo virtuálnej realite. Prichádzajú nové produkty, založené na Oculus systéme, ktoré sa chcú dostať do popredia buď svojou cenou alebo pridaním nových funkcií do systému. Komu už nestačí zážitok s VR okuliarmi, môže vyskúšať jeden z najnovších prístrojov pre virtuálnu realitu.

VirZoom, aj napriek jeho vzhľadu obyčajného stacionárneho bicykla, vám umožní počas vášho cvičenia zabudnúť na tréning, keď budete počas neho ovládať helikoptéru či formulu 1. Alebo si jednoducho môžete vychutnávať výhľad zapadajúceho slnka pri mori počas vašej jazdy popri pláži [6].

KOR-FX prišlo na trh s vestou, ktorá vám vie ponúknuť spätnú väzbu. Táto vesta našla svoje uplatnenie hlavne v strieľačkách. Tak môžete cítiť každý zásah guľky alebo neďaleký výbuch granátu [3].

S toľkými zariadeniami prichádzajúcimi každú chvíľu na trh, virtuálna realita je trend, ktorý sa môže spoľahnúť na ešte väčší rozvoj v nasledujúcich rokoch.

1.3 Užívateľské prostredie - UI

Užívateľské prostredie nám umožňuje interakciu medzi človekom a počítačovým prostredím. Za cieľ má sprostredkovať ovládanie systému človekom, ktorý sa rozhoduje, aké procesy sa majú v danom čase vykonať. Ďalej by toto prostredie malo byť užívateľsky prívetivé, jednoduché na ovládanie a efektívne. Preto sa často pri vytváraní takéhoto prostredia kladie veľký dôraz na štúdium disciplín ako sú ergonomika a psychológia.

- Ergonomika - veda, zaoberajúca sa riešením, ako navrhovať produkt s prihliadaním na pohodlnosť človeka, ktorý daný produkt bude používať.
- Psychológia - veda, študujúca kognitívne procesy, vedomé a nevedomé podnety ako aj myšlienky človeka.

Užívateľské prostredie by preto malo vyžadovať čo najmenej práce používateľa na dosiahnutie jemu žiadanej akcie. V minulosti bolo použitých viacero typov prostredí, ako napríklad dierne štítky či príkazový riadok. Neskôr prišlo grafické rozhranie založené na prácu s oknami, tzv WIMP (window, icon, menu, pointer)

Prvky užívateľského prostredia používané vo virtuálnej realite by sa dali rozdeliť do troch skupín, a to:

- Non-diegetic
- Spatial UI

- Diegetic UI

1.3.1 Non-diegetic

Patria tu prvky, ktoré má používateľ momentálne v danom prostredí k dispozícii, ako napríklad život, skóre, typ zbrane, koľko munície mu zostáva a podobne. Takéto prvky užívateľského prostredia sa inak nazývajú aj HUD elementy (Heads up display elements). Tieto elementy sú väčšinou mimo vygenerovaného prostredia, viditeľné iba daným používateľom, čo dáva zmysel, lebo to sú väčšinou informácie potrebné pre používateľa. Často sú tieto prvky umiestnené na vrchu obrazovky.

Treba si dávať pozor, aby takýchto prvkov nebolo príliš veľa, respektíve aby neboli príliš veľké, aby nenarušili vizuálny pôžitok, ktorý máme z prostredia, keďže tieto elementy sú viditeľné vždy a vidíme ich všade kam sa pozrieme a teda zakrývajú určitú časť našej viditeľnej plochy.

1.3.2 Priestorové UI

Vo všeobecnosti prvky užívateľského prostredia musíme zakomponovať do priestoru samotného. Takéto umiestňovanie však potrebuje zvážiť rôzne faktory, hlavne vo virtuálnej realite. Prvok užívateľského prostredia, ktorý je príliš blízko, nám môže vytvárať dvojitý obraz, lebo naše oči nebudú môcť zaostriť na tak blízku vzdialenosť, alebo ak je objekt príliš ďaleko, môže sa javiť, že sa nepozeráme na daný objekt, ale až za obzor vygenerovaného sveta. Preto treba veľkosti a vzdialenosti prvkov nakalibrovať a dynamicky meniť vzhľadom na prostredie, v ktorom sa momentálne nachádzame. Najprirodzenejšie je nájsť vzdialenosť, ktorá je príjemná ľudskému oku na čítanie a následne to potom škálovať.

1.3.3 Diegetic UI

Na rozdiel od priestorového UI, kde ukladáme prvky do prostredia, do skupiny diegetic UI patria prvky, ktoré patria jednotlivým objektom v hre, ktoré ukazujú nejakú dôležitú vlastnosť objektu. Môže to byť napríklad život nejakého stvorenia, ktoré sa pohybuje nad jeho hlavou, obrazovka televízora, kompas ukazujúci správny smer a podobne.

Kapitola 2

Návrh implementácie

V tejto kapitole budeme rozoberať, čo všetko nám bežne používané užívateľské prostredia ponúkajú. Zamerať sa, v čom majú nedostatky a pokúsiť sa zamerať práve na vytvorenie prostredia, ktoré by tieto nedostatky dokázalo odstrániť a bolo intuitívne. Budeme tu rozoberať, čo všetko sme brali do úvahy pri navrhovaní nášho užívateľského rozhrania a kompromisy, ktoré jednotlivé návrhy prinášajú pri manipulácii s objektami nachádzajúcimi sa vo virtuálnej realite.

2.1 Výber technického vybavenia

Predtým, ako sme začali rozmýšľať, aké užívateľské prostredie chceme vytvoriť, bolo dôležité zistiť, k akému technickému vybaveniu spojenému s virtuálnou realitou máme prístup. Najčastejšie z techník používaných na interakciu s objektami (či už výber objektu alebo manipuláciu s nimi) predstavujú nasledovné metódy:

- Sledovanie pohybu hlavy - založená na tom, v akom smere má užívateľ otočenú hlavu. Používajú sa pri tom hlavne informácie ako zmena pozície VR okuliarov (napríklad, keď sa zo sedu postavíme alebo naopak skloníme) a ich natočenie v 360 stupňoch.

- Sledovanie pohybu očí - dôležitosť sa kladie na snímanie pohybu očí samotných na manipuláciu s objektami. Kvôli technickej náročnosti sa táto funkcionalita dostáva na trh v najnovších modeloch a práve preto ešte nie je tak rozšírená, ako metóda sledovania pohybu hlavy.

- Ovládače do rúk - na výber a interakciu sa používajú okrem okuliarov nasadených na hlave ešte ďalšie ovládače, jeden do každej ruky, na ktorých sa nachádzajú tlačidlá, dotykové podložky alebo iné funkcie, líšiac sa od výrobcu produktu.

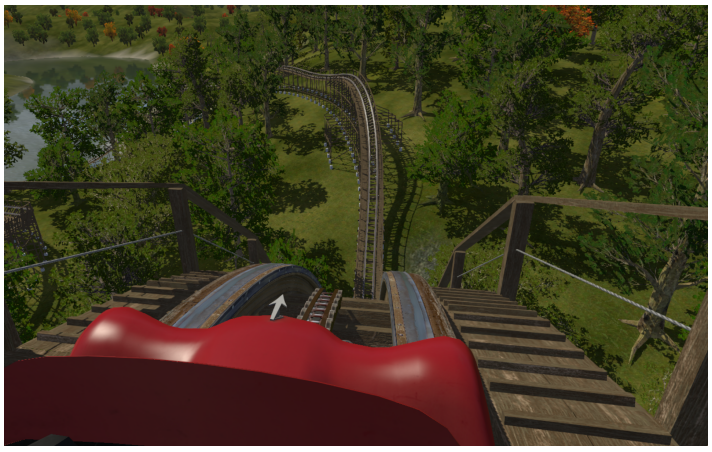
Priestor na realizáciu tejto bakalárskej práce sme dostali v našom školskom laboratóriu, kde sa nachádzali dve zariadenia na prácu s virtuálnou realitou - HDK2 OSVR

a Oculus rift. Ani jeden z týchto prístrojov nemal pribalené externé ovládače do rúk a nepodporoval funkciu sledovania očí. Zamerali sme sa teda na vymýšľanie návrhu s metódou sledovania hlavy, ktorý je momentálne bežnému publiku najbližší.

2.1.1 Sledovanie pohybu hlavy

Sledovanie pohybu hlavy je najrozšírenejšia a najzákladnejšia funkcionalita VR okuliarov. Už od začiatku vyrábania VR okuliarov sa pohyb hlavy používal na otáčanie v 360 stupňovom priestore. Na zisťovanie orientácie a otočenia okuliare používajú senzor, ktorý funguje na základe princípu gyroskopu. Čisto na tejto funkcionalite funguje veľa VR aplikácií vytvorených pre mobilné telefóny a teda nie je potrebné míňať stovky eur na špeciálne zariadenie pre VR alebo dostatočne silný počítač, ktorý by toto zariadenie podporoval.

Ako teda vyzerajú užívateľské prostredia, ktoré sú založené iba na tejto jednej funkcionalite? Kvôli absencii tlačidiel, veľa aplikácií sa rozhodlo dať väčší dôraz na vykreslenie daného prostredia, ako na interakciu s ním. Preto sa často stáva, že užívateľ nemá žiadnu alebo veľmi malú kontrolu nad pohybom samotnej kamery, či manipulácií objektov. Ako príklad typov takýchto aplikácií by sa mohli uviesť rôzne scény, kde sme jednoducho postavení do stredu scény, ktorá sa vzhľadom na čas mení a vyvíja, no umožňuje nám iba pozerať sa dookola a obdivovať krásu vytvorenej fiktívnej reality. Ďalším príkladom je scenár, kde je naša kamera uložená do pohyblivého vozíka (obrázok 2.1) a počas prehrávania scény sa tentokrát nemení iba scéna samotná, ale aj naša kamera. Tu však nastáva spomínaný problém - dráha kamery je predom zvolená a užívateľ ju nevie nijako ovplyvniť.



Obr. 2.1: Kamera je na pevno umiestnená v pohyblivom vozíku. Vozík sa síce pohybuje, ale dráha je predom daná a jediné, čo vieme robiť je pozerať sa dookola.

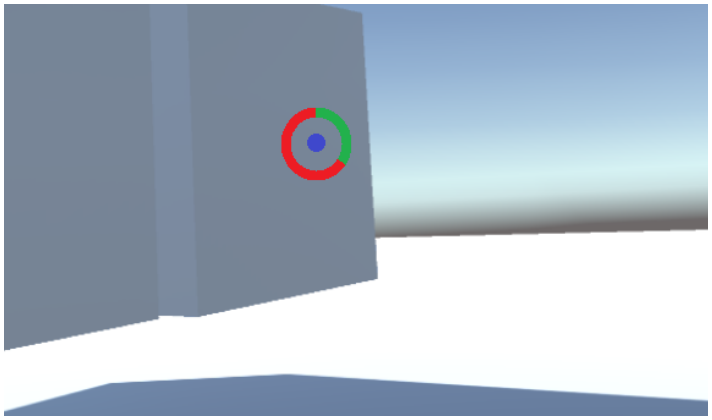
Tretí a najzaujímavejší typ je ten, ktorý nám naozaj ponúka interakciu s okolím vo virtuálnej realite a tak nám zaručí väčšie podvedomé vnorenie a tým aj lepší zážitok.

2.1.2 Interakcia pomocou sledovania pohybu hlavy

Pri tejto metóde sa používa kurzor. Kurzor je malý indikátor v strede obrazovky, ktorý nám umožňuje identifikovať, kam sa v danom momente pozeráme vo virtuálnej realite. Takto vieme identifikovať aj objekt, na ktorý sa pozeráme. Jeden z najpoužívanejších spôsobov interakcie s objektami, na ktoré sa pozeráme, je využívanie časovača. Funguje nasledovne: po tom, ako náš kurzor umiestnime tak, aby ukazoval na nejaký objekt, spustí sa časovač, ktorý plynie, pokiaľ daný objekt zostane nami zameraný. Po uplynutí určitého času sa na danom objekte vykoná akcia. Ak s kurzorom z objektu odídeme skôr, než na časovači uplynie celkový čas, žiadna akcia sa nevykoná. Takéto správanie časovača je väčšinou sprevádzané grafickým znázornením plynúceho času (obrázok 2.2). Ďalším spôsobom je jednoduché zameranie objektu pomocou kurzora a následného stlačenia tlačidla na klávesnici alebo magnetického tlačidla na okuliaroch, ak ním je daný typ okuliarov vybavený. Keďže našu implementáciu chceme navrhnúť bez externých ovládačov či klávesnice, ďalej sa budeme baviť viac o prvej metóde s časovačom.

Za nevýhodu tejto implementácie považujeme to, že vo väčšine prípadov aplikácií, ktoré sme takto otestovali, sme si nemohli vybrať, akú akciu chceme s objektom spraviť. Jednoducho sa po uplynutí časomiery spustí jedna akcia, ktorú má každý objekt predom priradenú. Aby sa akcia na objekte nevykonávala samovoľne, aj v prípade jednoduchého prejdenia kurzorom cez objekt, musí byť časomiere priradený dostatočne dlhý čas. Čakanie na uplynutie tohto času sa nám môže niekedy zdať zdĺhavé, pretože tam nastáva moment nečinnosti. Musíme udržať kurzor na danom objekte dlhší čas, čo znamená nehybnosť hlavy a teda určitú limitáciu.

Ďalšou nevýhodou je obmedzenie pohybu vo virtuálnej realite, ktorý nám tieto implementácie umožňujú bez používania externých tlačidiel, iba za pomoci VR okuliarov. Implementácie, ktoré sme našli, buď žiadnu podporu pohybu nemali, alebo bola veľmi



Obr. 2.2: Pri prejdení na objekt, s ktorým chceme interagovať sa spustí časomiera, ktorá graficky začne napĺňať kruh okolo kurzora. Po jej dokončení sa spustí akcia na vybranom objekte.

limitovaná. Buď sa dalo teleportovať na vyhradené miesta tým istým spôsobom, ako pri interakcii vysvetlenej v predošlom odstavci (teda pozeranie sa na objekt na určitý čas, po ktorom uplynutí sa kamera presunula na nové miesto), alebo celá podlaha fungovala ako jeden veľký objekt a dokázali sme sa teleportovať na miesta na podlahe, na ktoré sme sa pozerali určitý čas. Ďalšia z možností bola plynulý pohyb, ktorý sa začal v momente, keď sme sa začali pozerať na zem pod veľkým sklonom. Tieto riešenia vyžadujú, aby sme sa pozerali na zem namiesto pred seba. Toto správanie je pre ľudí neprirodzené, keďže pri pohybe sa na zem pozeráme iba periférne a namiesto toho sa pozeráme okolo seba. Uberá to tiež na zážitku z virtuálnej reality. Z veľkej časti sa kladie dôraz na prostredie, na ktoré sa nemôžeme plne sústrediť, pretože sa pri pohybe musíme pozerať smerom na zem.

Táto implementácia užívateľského prostredia pomocou časovača má aj veľké výhody, čo sa týka interakcie s objektami, a tou je napríklad intuitívnosť a odľahčenie námahy hlavy. V momente, keď chceme s niečím manipulovať, tak sa na danú vec podvedome začneme pozerať. Po krátkej dobe pozerania sa akcia automaticky vykoná a netreba pri tom čakať na ďalší zásah užívateľa.

2.2 Žiadané zmeny oproti existujúcim riešeniam

Pri dizajne nového užívateľského prostredia sme sa chceli pokúsiť eliminovať tri veci.

1. Nechať užívateľa vnoriť sa viac do prostredia tým, že na spustenie interakcie s objektom bude musieť vykonať nejaký pohyb a nielen nečinne stáť.

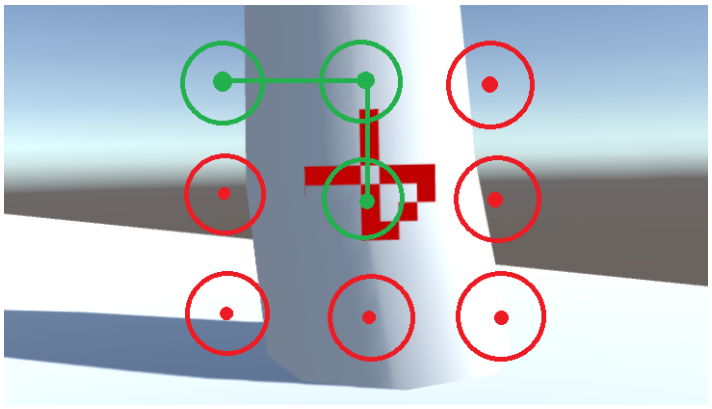
2. Následne mu dať možnosť vybrať si z viacerých možností, akú akciu na objekte vykonať.

3. Pridať gesto, ktorým by sa užívateľ vedel voľne pohybovať okolo. Aby pri tom nemusel pozerať do zeme, ale mohol sa pritom pozerať okolo seba a tak si mohol vychutnávať estetický zážitok.

2.3 Vzory na maticovej mriežke 3x3

Predstavme si pred nami maticu s troma riadkami a troma stĺpcami. Pre väčšinu z nás známy prístup zo zamykacej obrazovky na mobiloch, kedy musíme v správnom poradí spojiť určité body matice na to, aby sa systém odomkol (obrázok 2.3). Týmto spôsobom možno vygenerovať tisícky rôznych vzorov.

Implementácia tohto nápadu by umožnila nespočetný počet akcií s rôznymi objektami, ktoré by sa dali v aplikácii vykonávať. Avšak zapamätanie si desiatky takýchto vzorov by nebola vôbec jednoduchá záležitosť. Pre neskúsených užívateľov by to mohla



Obr. 2.3: Nachádza sa pred nami matica 3x3, ktorej pospájané body vytvárajú vzor.

byť úloha na niekoľko dní, dokonca by mohli nastať prípady, kedy by ani nebolo možné si všetky tieto vzory zapamätať ani po dňoch využívania aplikácie. My však chceme, aby ovládanie bolo intuitívne a zvládnuteľné v krátkom čase, dokonca aj pre ľudí, ktorí nemali predošlú skúsenosť s danou aplikáciou, či dokonca virtuálnou realitou samotnou. Zapamätanie si množstva vzorov by bolo pre veľa ľudí odradzujúce, čiže sme museli zvoliť inú metódu.

2.4 Gesto s otvorením menu

Na vyriešenie problému, ktorý mal maticový prístup, sme museli nájsť spôsob, ktorým eliminujeme zdĺhavú časť učenia sa vzorov. Použiť iba jedno gesto, ktorým by sme otvorili menu. Nasledovne by si v menu jednoduchým pohybom vedel užívateľ vybrať, ktorú z akcií chce na danom objekte vykonať. Pritom treba riešiť viaceré otázky, ako napríklad, akým vzorom by malo byť gesto otvárajúce menu reprezentované, aby bolo ľahko zapamätateľné. Pri vytváraní gesta užívateľ musí pohybovať hlavou, teda nemalo by byť príliš zdĺhavé. Ľahko zapamätateľné a krátke, no zároveň komplikované natoľko, aby sa neotvorilo aj v momentoch, kedy si to užívateľ nežiada (napríklad pri obzeraní scény dookola by sa mohlo stať, že by nejaké jednoduché gesto vedel náhodne napodobniť). Nemalo by predstavovať otočenie len v jednom smere, aby sa nestalo to, že sa bude z toho točiť hlava už po pár použitiach.

Vymysleli sme teda pohyb, pri ktorom sa hlava pohybuje vo všetkých smeroch - dole, hore, doprava a doľava - aby sa nestalo, že by nastala nevoľnosť pri dlhšom používaní. Daný vzor sa skladá z troch úsečiek. Z nich vytvárame ľahký tvar krížika, ktorý sa veľmi ľahko pamätá. Vykonanie gesta pozostáva z troch etáp, ktoré je možné vidieť na obrázku 3.1. Zvislý pohyb smerom nadol, diagonálny pohyb smerom dohora a doprava a následne priamy horizontálny pohyb doľava. Tento pohyb môže ale byť celkovo väčší, ako niektoré objekty. Preto je dôležité zapracovať, aby kurzor pri geste

mohol vyjsť aj mimo objektu. Bolo potrebné vyriešiť, ako naznačiť, s ktorým objektom chceme manipulovať. Nie s každým objektom sa musí dať manipulovať, preto sme implementovali funkciu vyberania objektu.

Každý objekt, s ktorým by mala byť povolená manipulácia bude mať špeciálny štítok, podľa ktorého bude aplikácia vedieť, či s ním má vedieť manipulovať alebo nie. Túto informáciu treba sprostredkovať aj užívateľovi. Ak prejde kurzorom na daný objekt so správnym štítkom, daný objekt sa graficky odlíši zvýraznenou farbou a do stredu kurzora sa napíše názov objektu. Následne po odídení kurzora z objektu sa objekt vráti do pôvodného stavu a z kurzora sa vymaže daný názov. Výnimkou je, ak užívateľ začne robiť gesto na otvorenie menu. Vtedy sa po prvom pohybe, ktorý je smerom nadol, výber objektu uzamkne, a teda aj po následnom odídení kurzora z objektu si program zapamätá, na ktorom objekte daný obrazec začal vytvárať. V podstate to znamená to, že naša kamera na moment uzamknutia prestane brať do úvahy, ktoré z objektov zasiahnu lúče z nej vysielané. Následne sa odomkne buď po vypršaní maximálneho času, ktorý je pridelený na vykonanie gesta, alebo až po výbere niektorej z akcií z menu. Po odomknutí zase berieme ohľad na to, na aký objekt sa momentálne pozeráme.

Týmto by sme vyriešili prvé dve podmienky. Teraz pri interakcii s objektom musíme aktívne vynaložiť prácu (čím sa ešte viac ponoríme do virtuálneho sveta) a následne môžeme mať na výber z viacerých možností, akú akciu na objekte chceme vykonať. Zostáva preto vymyslieť, ako spraviť náš pohyb tak, aby nebol limitovaný miestom a aby sme sa pri ňom nemuseli pozerať do zeme.

Problém pri plynulom pohybe vo virtuálnej realite je ten, že naše telo sa v realite nehýbe, ale sedí na stoličke alebo stojí. Toto náš rozum nevie poriadne spracovať, a preto pri tomto postupe môže ľudom, ktorí na niečo podobné nie sú zvyknutí, prísť zle. Aj z tohto dôvodu sa často preferuje metóda teleportu na vykonávanie pohybu vo virtuálnej realite. Pri teleportovaní sa kamera v momente presunie na nové, nami špecifikované miesto a teda nenastáva v rozume rozpor medzi statickým telom a dynamicky sa meneným pohľadom.

Čo by sme mohli spraviť je, že ako jedna z položiek nášho menu by bola položka pohybu v smere, v ktorom sme menu otvorili. Avšak ak zoberieme na vedomie to, že predsa len otvorenie menu nie je najkratšia záležitosť, presunúť sa z jedného bodu do nami zvoleného druhého bodu by mohlo trvať príliš dlho. Tiež, ak by sme museli v krátkom čase opakovane veľakrát robiť dané gesto, mohlo by sa to prejaviť tým, že by sme sa fyzicky unavili.

Preto sme vymysleli ďalší, omnoho kratší pohyb, ktorý by mohol slúžiť exkluzívne na pohybovanie sa v priestore. Hlavným gestom budeme nazývať dlhšie gesto, po ktorom vykonaní sa nám otvorí menu, s ktorým môžeme manipulovať. Vedľajšie gesto bude v porovnaní s hlavným omnoho kratšie a bude slúžiť na vykonanie pohybu. Malo

by byť možné tento pohyb vykonávať veľakrát v krátkom čase. Musí mať jednoduchý vzor. Zvolili sme si preto pohyb, ktorý sa vykonáva iba v dvoch smeroch, najprv krátky zvislý pohyb smerom dohora a ihneď za tým nasleduje krátky zvislý pohyb nadol.

Aj pri tomto vedľajšom vzore musíme riešiť spôsob, ako zaručiť to, aby sa kamera samovoľne nepohybovala pri obzeraní si scény. V hlavnom vzore sme to vyriešili vybraním dostatočne komplexného vzoru na to, aby sa všetky pohyby vykonali v správnom poradí, dĺžkach a času. Toto bohužiaľ nemôžeme spraviť aj pri vedľajšom geste, keďže využívame omnoho jednoduchší vzor. Čo sme teda vymysleli je, že vždy tesne predtým, ako sa chceme pohnúť, musíme na malý moment s kurzorom zastaviť. Tiež sme na vykonanie tohto pohybu dali veľmi krátky časovač, ktorý ma ďalej pomáhať tomu, aby sme sa nepohli aj vtedy, keď nechceme. Po vykonaní vedľajšieho gesta je na programátorovi, aký typ pohybu implementuje vo svojej aplikácií. Môže tam implementovať prístup, ktorý po prvom vykonaní môže spustiť plynulý pohyb smerov vpred. Následným opakovaným vykonaním gesta sa pohyb zastaví. My sme implementovali spôsob teleportovania (instantného presunu) kamery, kedy sa po vykonaní gesta presunieme na súradnice nachádzajúce sa pred nami.

Predošlé popísané prvky by mali tvoriť základ nášho užívateľského prostredia. Daná funkcionalita interakcií s objektami, respektíve ako sa má vykonávať pohyb kamery, záleží čisto na implementácií programátora, ktorý si to môže zmeniť podľa vlastných potrieb.

Kapitola 3

Implementácia

V tejto kapitole sa budeme zaoberať implementáciou užívateľského rozhrania v Unity3D, ktorá pozostávala z niekoľkých etáp:

1. vytvorenie kurzora,
2. rozpoznávanie pohybu kurzora,
3. vytvorenie menu a interakcia s ním.

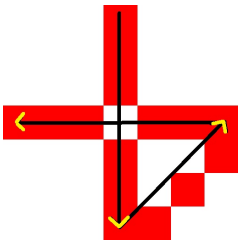
3.1 Kurzor

V dnešnej dobe technológia bežne dostupných VR systémov ešte nedokáže zaznamenať a spracovať, kam sa naše oči pozerajú. Výnimkou sú najnovšie modely svetoznámych značiek v oblasti virtuálnej reality. Príkladom je napríklad novinka na trhu HTC Vive pro eye, ktorého jednou z mnohých funkcií je aj sledovanie pohybu očí. Kým sa ale toto nestane normou pre väčšinu zariadení pre virtuálnu realitu, jednou z hlavných položiek užívateľského prostredia zostane kurzor.

Kurzor zabezpečuje používateľovi informáciu, kde si zariadenie myslí, že sa používateľ pozerá. Väčšinou má pozíciu v strede obrazovky a indukuje funkciu zameriavania, ktorá je známa hlavne v strieľačkách, kde sa využíva na to, aby mierenie bolo presnejšie.

Vo virtuálnej realite je kurzor o to potrebnejší, lebo naše oči sa môžu pozerať na rôzne časti obrazovky, a teda pri pokuse o interakciu s objektom, na ktorý sa síce pozeráme, no nemierime, môže zostať veľa ľudí v rozpakoch.

V budúcnosti pravdepodobne kurzory zaniknú, keď bude možná širšia interakcia vo virtuálnom prostredí, ktorá nám umožní ukazovať na jednotlivé objekty rukami či prstom a tým nahradí používanie hlavy. Vzhľadom na to, že naša implementácia má byť založená hlavne na tom, že používateľ bude vedieť interagovať s objektami bez požívania externých ovládačov do rúk, bude kurzor v našej implementácii o to dôležitejší.



Obr. 3.1: Textúra s vyznačeným pohybom hlavného gesta.

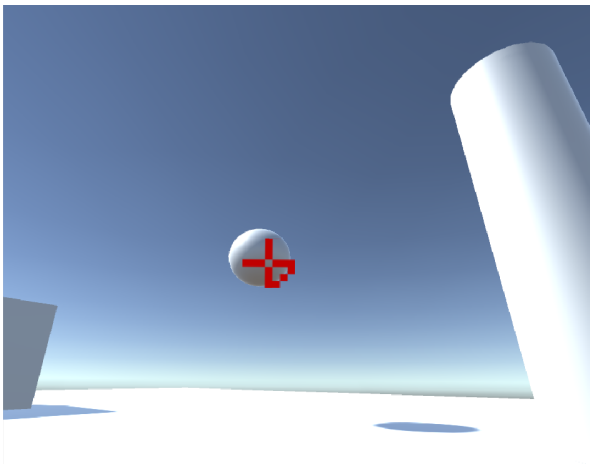
3.1.1 Implementácia kurzora

Na začiatok si pridáme do scény nový objekt - štvorec, na ktorý budeme zobrazovať textúru - Quad. Štvorec má veľkosť 1x1 a skladá sa z 2 trojuholníkov, na rozdiel od roviny, ktorá má veľkosť 10x10 a skladá sa z 200 trojuholníkov. Taktiež na rozdiel od roviny, ktorá je vykreslená v XZ súradniciach, quad je vykreslený v XY súradniciach, a teda sa nám javí už ako stojaci pred kamerou a netreba ho dodatočne otáčať v priestore okolo osi x, ako by to bolo nutné s rovinou. Je zaužívané, že pri zobrazovaní objektov, ktoré majú byť jednoducho obrázky alebo videá, sa používa quad namiesto roviny. [4]

Potom, ako sme vytvorili tento štvorec, pridali sme si textúru, ktorú sme si dopredu vytvorili v grafickom softvéri. Zvolili sme si zaužívanú textúru krížika, obohateného o diagonálnu čiaru, spájajúcu dolnú a pravú časť krížika. Toto spojenie sme zvolili preto, aby tvar kurzora pripomínal pohyb, ktorým sa neskôr bude zobrazovať menu (obr. 3.1). Tiež je dôležité si uvedomiť, že vzhľadom na to, že viditeľná časť kurzora má zaberať len malý podiel z plochy nami vytvoreného štvorca, treba danú textúru uložiť vo formáte, ktorý podporuje priesvitnosť (napríklad PNG alebo BMP). Ďalej sme textúre zmenili farbu na červenú, aby ho bolo ľahko vidieť v každom momente. Keďže textúry sú aplikované na objekty pomocou materiálu (ktorý používa shader na vykresľovanie textúr na povrch mnohouholníkovej siete objektu), treba vytvoriť materiál pre náš kurzor, v ktorom použijeme našu textúru. Následnou aplikáciou materiálu na náš štvorec by sa mal zmeniť jeho vzhľad. Kurzor možno vidieť na obrázku 3.2.

Kurzor je v tomto momente nehybne umiestnený v priestore. Teraz mu musíme pridať funkcionalitu, aby mohol robiť prácu kurzora v našom prostredí. Túto funkcionalitu budeme pridávať tomuto objektu pomocou skriptu v C# a meniť vlastnosti nášho kurzora budeme hlavne cez transform API [5], ktorej funkcionalitu dostane pri vytvorení každý z objektov. Prvú vec, ktorú môžeme spozorovať je, že kurzor je viditeľný iba z jednej strany. Preto je dôležité mu ako prvé nastaviť transformácie v priestore tak, aby bol stále otočený smerom ku kamere. To v unity dokážeme spraviť pomocou funkcie LookAt, ktorá otočí daný objekt tak, aby jeho normála smerovala k pozícii objektu použitom v parametri, čo je v našom prípade pozícia kamery.

```
transform.LookAt(VRCamera.transform.position);
```



Obr. 3.2: Ukážka kurzora. Kurzor má červenú farbu a tvar vyznačujúci pohyb, ktorým sa bude spúšťať menu (viac o tomto pohybe bude v kapitole 3.2.1 Implementácia rozpoznávania hlavného vzoru). Farba a tvar sa dá veľmi jednoducho zmeniť a prispôsobiť.

Touto transformáciou kurzor získa vlastnosť, že sa bude stále otáčať smerom ku kamere. Keďže na začiatku je predná stena odvrátená od kamery, treba mu ďalej pridať transformáciu, kde ho otočíme o 180 stupňov okolo osi y.

```
transform.Rotate(0.0f, 180.0f, 0.0f);
```

Tým pádom bude kurzor stále otočený smerom ku kamere viditeľnou stranou pri ľubovoľnom pohnutí kamery.

To, aby sa kurzor otáčal smerom ku kamere, ale nestačí. Potrebujeme, aby mal stále pozíciu v strede obrazovky, aby mohol určovať, kam užívateľ mieri. Čo teda chceme spraviť je, že kurzoru budeme meniť jeho pozíciu na pozíciu kamery, s tým rozdielom, že mu musíme pridať určitú vzdialenosť od kamery. Túto zmenu dosiahneme vynásobením vektora pohľadu kamery vyhovujúcou vzdialenosťou, ku ktorej výpočtu sa dostaneme neskôr v tejto kapitole.

```
transform.position = VRCamera.transform.position +
+ VRCamera.transform.rotation * Vector3.forward * distance;
```

V tomto momente už vieme využívať náš kurzor na zameriavanie objektov a vieme rozpoznávať, na ktorý objekt sa práve pozeráme. Ďalej tu budeme riešiť problémy ako prelínanie sa kurzora s ostatnými objektami, zdvojovanie obrazu kurzora alebo objektu, na ktorý sa pozeráme, škálovanie veľkosti kurzora vzhľadom na vzdialenosť, do ktorej sa pozeráme a podobne[2].

Distance je číselná hodnota, reprezentujúca vzdialenosť, ako ďaleko by sa mal náš kurzor nachádzať od kamery. Ak by sme si za premennú distance zvolili hocijakú náhodnú konštantu, kurzor by sa vznášal pred kamerou v centre obrazovky vo vzdialenosti rovnajúcej sa tejto konštante. Čo by toto začalo spôsobovať je, že pri zaostrení zraku

na objekty v rôznych vzdialenostiach, by sa nám začal zdvojovať obraz. Taktiež, ak by premenná distance bola konštanta, kurzor by nevedel dynamicky meniť vzdialenosť kurzora od kamery a v momente, kedy by sme prišli veľmi blízko k nejakému objektu alebo stene, kurzor by sa zobrazil až za týmto objektom a vytratil by sa z nášho zorného poľa. Keď však kurzor zmizne z našej obrazovky, nemôžeme ho použiť na zameriavanie nášho pohľadu.

Začneme tým, že opravíme dvojitý obraz. Na to použijeme unity funkcionalitu Physics.Raycast[5], pomocou ktorej vieme získať informáciu, či lúč vysielaný z našej kamery zasiahne nejaký objekt. Ak akýkoľvek objekt v scéne bude zasiahnutý, z funkcie Raycastu vieme zistiť, ako ďaleko sa daný objekt od kamery nachádza. Teda jediné, čo nám treba spraviť je, že v tomto momente nastavíme do premennej distance práve túto hodnotu.

```
distance = hit.distance;
```

Tu si treba dávať pozor, aby náš kurzor nemal zapnutú detekciu kolízií, pretože to by znamenalo, že náš lúč by stále zasiahol kurzor na miesto objektu, na ktorý sa v danom momente pozeráme. Ak náš lúč netrafí žiaden objekt (napríklad v danom smere žiaden objekt alebo stena nie je umiestnená, alebo sa môžeme pozerať smerom na oblohu, kde sa nemusia nachádzať žiadne objekty), musíme mu nastaviť vzdialenosť trochu menšiu, ako je hodnota FarClipPlane, ktorá nám určuje maximálnu vzdialenosť, v ktorej sa objekty ešte budú zobrazovať.

Po aplikovaní tejto zmeny sa nám veľkosť kurzora začala dynamicky meniť podľa vzdialenosti objektu, na ktorý sa práve pozeráme. Čo môže byť problematické v tomto momente je to, že ak je objekt veľmi ďaleko, náš kurzor bude tiež veľmi malý. To opravíme pomocou zmeny v škálovaní, aby sme dosiahli to, že náš kurzor bude zaberať stále rovnakú veľkosť z obrazovky. Na to jednoducho budeme meniť škálovanie nášho kurzora podľa premennej distance.

```
transform.localScale = originalScale * distance;
```

Teraz kurzor zaberá stále rovnakú časť obrazovky. Ak sa ale pozeráme na objekt, ktorý je veľmi blízko, aj keď kurzor má správnu veľkosť, podvedome sa nám javí ako veľmi malý. To sa deje z dôvodu, pretože si spájame dokopy to, že náš kurzor je veľmi blízko, ale zdanlivo zaberá veľmi malú časť obrazovky oproti objektu. To upravíme tak, že premennú distance vynásobíme hodnotou $1 + 4.4^{(-Distance)}$ v momente, keď sa budeme nachádzať príliš blízko objektu.

Napriek tomu, že sme nastavili, aby sa náš kurzor nevnáral do objektov, ale namiesto toho sa zobrazoval na povrchu objektov, časť z neho sa stále môže skryť do textúry pri pohľade na vybrané objekty. To sme napravili pomocou zmeny v nastaveniach shaderu, ktorý používame pre zobrazovanie nášho kurzora. Na to, ako sa majú

správne veci zobrazovať, slúži hĺbkové testovanie, ktoré sa stará o to, aby sa zobrazovali iba najbližšie objekty. To vieme zmeniť v premennej ZTest, slúžiacej na upresnenie, ako má daný shader vyhodnotiť daný objekt pri tomto hĺbkovom teste, a teda ak ju zmeníme na hodnotu Always, znamená to, že daný objekt bude vždy vykresľovať na vrchu do okna[4].

Týmto sme dokončili jednu z dôležitých častí nášho užívateľského prostredia - kurzor. Ten nám bude pomáhať nielen na zameriavanie objektov, ale aj pri koordinácii ako vzor pre gestá na spúšťanie nášho menu v užívateľskom prostredí.

3.2 Rozpoznávanie vzorov

Druhou dôležitou kapitolou v implementovaní nášho užívateľského prostredia je rozpoznávanie vzorov. Pohyb v tvare vzoru, ktorý užívateľ bude vytvárať pomocou hlavy, budeme nazývať gesto.

Aplikácia rozpoznáva momentálne dva vzory. Jeden hlavný vzor, ktorým budeme spúšťať naše menu, v ktorom sa nachádzajú tlačidlá na ďalšiu manipuláciu s objektami. A potom druhý, vedľajší vzor, ktorý bude slúžiť na vykonávanie rýchlej akcie, ktorú musí užívateľ opakovať častejšie počas behu programu.

3.2.1 Implementácia rozpoznávania hlavného vzoru

Na implementovanie vykonávania hlavného vzoru použijeme troj etapový postup. To znamená, že užívateľ sa nielen musí držať v líniách daného vzoru, ale musí sa postupne po jednom dostať do správnych zón v správnom poradí. V našom vzore sa budú nachádzať tri zóny. Prvá sa bude nachádzať na spodku vzoru, presnejšie na mieste, kde by sme mali skončiť priamy vertikálny pohyb.

Druhá zóna sa bude nachádzať napravo-hore od prvej zóny, na konci krátkeho diagonálneho pohybu, ktorý musíme vykonať po správnom ukončení prvého zvislého pohybu.

Tretia a zároveň posledná zakončujúca zóna sa nachádza naľavo od druhej, na konci ľavotočivého horizontálneho pohybu. Ako tieto zóny vyzerajú môžeme pozorovať na obrázku 3.3.

V tomto momente by sa nám malo otvoriť menu a nasledovným prejdením na nejakú z jeho položiek by mala nastať interakcia. Po splnení každej z etáp nastáva grafická zmena textúry kurzora, ktorá ma za úlohu pomôcť užívateľovi určiť, ktorú etapu práve dokončil. Na vykonanie jednotlivých pohybov sme vybrali časový limit - dve sekundy. Je to z toho dôvodu, že nechceme, aby textúra zaseknutá v inom ako pôvodnom stave, pokiaľ sa užívateľ naozaj nerozhodol vykonať pohyb na otváranie menu.

V akom otočení sa naša kamera v danom momente nachádza, dokážeme dostať z vlastnosti eulerAngles z triedy transform. Ak ju použijeme na kameru, vie poskytnúť informácie o tom, ako je kamera otočená okolo všetkých troch osí - X, Y a Z. Toto bude pre nás kľúčové pri kontrole počas detekcie pohybu.

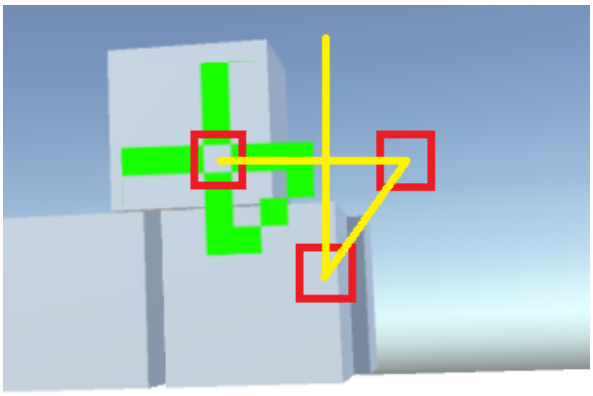
```
transform.rotation.eulerAngles;
```

Zmena rotácie premennej X bude pre nás znamenať vertikálnu rotáciu, zatiaľ čo premenná Y bude udávať horizontálnu rotáciu.

Začneme teda s detekciou prvého pohybu, čo je zvislý pohyb smerom nadol. Tu si musíme pamätať poslednú najvyššiu hodnotu rotácie X, aby sme podľa nej mohli určiť, kde sa ma nachádzať končiaca zóna tohoto pohybu. Stále preto pri pohybe nahor aktualizujeme našu hraničnú hodnotu uhla okolo osi X, ktorá nám určuje maximálnu, respektíve minimálnu hodnotu nášho uhla. Zapamätáme si pritom aj hodnotu rotácie Y, aby sme mohli kontrolovať, či zvislý pohyb prebiehal naozaj smerom kolmo nadol. Pri každej zmene tejto hraničnej hodnoty reštartujeme náš časovač na vykonávanie jednej etapy, keďže pri pohybe hore ešte nepredpokladáme, že naozaj už chceme začať vykonávať náš pohyb hlavou, ktorým otvoríme menu. V momente, keď začneme pohyb smerom nadol, celý čas kontrolujeme, či sa nachádzame v zvislom pásme, ktoré ma rezervu desiatich stupňov na každú stranu.

Pásmo je priestor, ktorý je ohraničený maximálnymi hodnotami v určitom smere. Ak z tohto priestoru odídeme, budeme to považovať za nekorektné vykonanie gesta. Všetky tri pásma môžeme vidieť na obrázku 3.4 v poradí, v akom treba vykonávať etapy. Pásmo je staticky umiestnené do priestoru a kontroluje, či nevyjdeme mimo jeho hranice.

Dohromady má naše zvislé pásmo šírku dvadsiatich stupňov, z ktorých ak počas tohoto pohybu vyjdeme, reštartuje sa nielen naša časomiera, ale aj celkový postup s hodnotami rotácií. Ďalej kontrolujeme vykonanú dĺžku smerom nadol, ktorá by mala



Obr. 3.3: Môžeme tu vidieť vyznačené zóny, do ktorých sa používateľ musel trafiť pri robení gesta na to, aby dokončil všetky etapy.

zodpovedať piatim stupňom otočenia okolo osi X. Keď sa nám tento pohyb úspešne podarí dokončiť, zmeníme textúru kurzoru, ktorá sa líši tým, že má strednú vertikálnu časť vyfarbenú na zeleno. To naznačuje užívateľovi, že prvú etapu vytvárania gesta zvládol úspešne a môže pokračovať v druhej etape. Ďalej nastavíme nové hraničné hodnoty rotácie X a Y, určujúce našu momentálnu rotáciu kamery. Toto je potrebné na správne vykonanie pohybu v etape číslo dva. Reštartujeme našu časomieru. Tiež nastavíme hodnotu zámku, ku ktorého funkcii sa dostaneme neskôr.

V etape dva - diagonálny pohyb - vykonávame kontrolu veľmi podobne, ako v etape jeden, len s rozdielom kontrolovania pásma, ktoré nám obmedzuje priestor, v ktorom sa musíme v danom momente vytvárania gesta nachádzať. Tu treba meniť hodnoty hraníc nášho pásma podľa toho, ako veľmi vľavo (respektíve hore), sme sa doposiaľ pohli a tak upravili ich X-ovú hranicu uhla (respektíve Y-ovú). Analogicky, ak počas pohybu vyjdeme z určeného pásma, reštartuje sa celkový postup, pri ktorom sa znovu nastavia nové hraničné hodnoty rotácie X a Y, textúra kurzora sa zmení na pôvodný a hodnota zámku sa zmení na otvorenú.

Etapu tri - horizontálny pohyb - spravíme analogicky s tým rozdielom, že teraz bude hodnota rotácie Y označovať šírku pásma a hodnota rotácie X bude určovať jeho dĺžku. Po správnom dokončení tohoto pohybu považujeme vykonanie vzoru za korektné a kompletné a môže nastať spustenie akcie, ktorá sa má v tomto prípade zapnúť. V našom prípade to znamená otvorenie menu, kde môže pokračovať výber jednej z možných interakcií.

3.2.2 Implementácia rozpoznávania vedľajšieho vzoru

Na rozdiel od hlavného gesta, kde bolo pre nás dôležité mať väčší výber interakcií (čoho dôsledok bol komplikovanejší vzor), vo vedľajšom vzore budeme dbať na to, aby bolo možné vykonať ho rýchlo a bez väčšieho úsilia. Tento pohyb bude mať zodpovedať krátkemu pohybu smerom hore a následne krátkym pohybom smerom dole. Pripomínať bude malé kývnutie hlavy. Takýto pohyb nie je ničím výnimočný. Mohlo by sa stať, že by užívateľ vedel náhodne a nechtiac tento pohyb napodobniť pri obyčajnom pohybe, napríklad pri prezeraní si prostredia vo vnútri virtuálnej reality, čo by nebolo chcené správanie.

Obr. 3.4: Tu môžeme vidieť, čo reálne pásmo znamená. Pásmo sú hraničné hodnoty, cez ktoré nemôžeme prejsť, ak chceme korektne vykonať naše gesto.

Ošetríme to tak, že predtým, ako daný pohyb začneme vykonávať, je potrebné, aby sme na krátky moment zastavili akýkoľvek pohyb a teda dali najavo programu, že momentálne sa len tak neobzeráme. Pri kontrole, či sme zastali, sme si zase vytvorili pásmo, ktoré má teraz tvar skôr malého štvorca (na rozdiel od hlavného gesta, kde to boli skôr vertikálne a horizontálne obdĺžniky a rovnobežníky), ktoré má za úlohu pridať rezervu na malé pohyby hlavy, ktoré nevykonávame vedome. Tiež je tam nastavený časovač, ktorý má limit 300 milisekúnd, ktorých ak vydržíme nečinne stáť, program to vyhodnotí ako pokus na zastavenie pohybu.

```
const float limiter = 0.3f;
```

Tu zase zmeníme farbu textúry kurzoru, na naznačenie toho, aby užívateľ vedel, že v tomto momente už môže vykonať pohyb špecifikovaný vo vedľajšom vzore. Na vykonanie jednotlivých pohybov hore a dole má daný limit 400 milisekúnd.

```
if (t > limiter + 0.1f)
{
    ResetFastMovement();
}
```

Po dokončení celého pohybu sa naša kamera pohne o päť metrov dopredu v smere, v akom sa pozeráme. Prebleskne farba textúry kurzora naznačujúc korektné ukončenie vedľajšieho vzoru. Následne sa farba textúry kurzora vráti do pôvodného stavu. Táto akcia však môže byť vývojárom pozmenená a prispôsobená jeho požiadavkám.

Prítomnosť zámku a jeho vplyv na vykonávanie vedľajšieho vzoru

Pri hlavnom vzore zastavujeme na krátky čas po každej úspešne vykonanej etape, pričom následne pokračujeme pri vytváraní vzoru. Vedľajší pohyb sme spravili natoľko minimalistický, že po chvíľkovom zastavení pri hlavnom geste by to program mohol vyhodnotiť ako zastavenie nutné pred vykonaním vedľajšieho pohybu a tak by sa prerušilo vytvárané gesto.

Tu prichádza na rad zámok. Zámok má za úlohu spraviť to, že pri hlavnom pohybe sa vedľajší pohyb nemôže vykonať. Po prvej etape hlavného pohybu zámok aktivujeme a ten zostane aktívny až do ukončenia gesta. Existujú dve možné uvoľnenia zámku.

Prvý spôsob uvoľnenia nastáva po úspešnom vykonaní hlavného vzoru a vybratia si akcie z menu. Druhý spôsob nastáva pri neúspešnom vykonaní vzoru, ako napríklad pri vyjdení z niektorého z pásiem alebo pri vypršaní dvoj sekundového časového limitu na vykonanie etapy z hlavného vzoru. V momente, keď sa zámok uvoľní, vráti sa kurzor do pôvodného stavu. V tomto stave môže kurzor znova prejsť do stavu zastavenia, ktoré bude nasledovať vykonaním vedľajšieho gesta slúžiaceho na pohyb.

3.3 Menu a interakcia s ním

Menu je jedna z nevyhnutných komponentov v našej implementácii. Umožňuje nám možnosť výberu z viacerých možných interakcií s objektom. Aby menu správne fungovalo a naozaj vykonávalo vybranú akciu na objekte, ktorý sme si zvolili, je treba sprostredkovať užívateľovi viacero informácií.

Je veľmi dôležité dať nejakým spôsobom najavo, s akým objektom chceme vykonať akciu. Ďalej je dôležité, aby sme v menu mali základné popisky akcií, ktoré môžeme v danej aplikácií využívať. Musia byť jasne čitateľné a zrozumiteľné. Keď všetko z tohoto listu splníme, môžeme dané menu pridať ako koncovku ku nášmu hlavnému gestu.

3.3.1 Implementácia vyberania objektov

Pri vyberaní objektu je dôležité si uvedomiť, že nemusíme chcieť interagovať s každým objektom v scéne rovnako. Napríklad nemusíme chcieť, aby sme so stenou izby vedeli manipulovať rovnako, ako so stolíkom v izbe. Treba tu rozlišovať objekty, ktoré majú byť stredobodom našej pozornosti a objektami, ktoré slúžia na dotváranie pozadia scény.

Vyberanie objektov preto funguje nasledovne. Kamera neustále vypúšťa smerom dopredu vektor - takzvaný lúč. Pri vyberaní objektu kontrolujeme, s ktorým objektom sa daný lúč preťal. Používame na to rovnakú metódu raycastingu, ako v kapitole 3.1.1 Implementácia kurzora.

Objektom, s ktorými chceme povoliť špeciálnu manipuláciu, vieme pridať štítok SelectableTag, ktorý kontrolujeme pri tomto prieniku.

```
if (selection.CompareTag(SelectableTag))
```

Ak objekt, na ktorý sa kurzorom pozeráme tento štítok má, vieme túto informáciu podať ďalej užívateľovi a tiež si uchovať referenciu na daný objekt. Aby užívateľ vedel rozpoznať, ktoré objekty v scéne sú tie hlavné, s ktorými vie manipulovať, môžeme týmto objektom meniť farbu textúry. Farbu textúry zmeníme objektu na zvýraznenú v momente, kedy prejdeme kurzorom na daný objekt, a zasa naopak na pôvodnú, ak z daného objektu s kurzorom odídeme.

Ďalej môžeme názov objektu zobrazovať v strede kurzora. Tento výpis užívateľovi uľahčí rozpoznávanie, či pri vykonávaní gesta hlavného pohybu hlavou, má naozaj označený daný objekt. Pridali sme kurzoru text, ktorý spolu s farbou textúry meníme na meno objektu, ktorý má štítok SelectableTag a ktorý práve kurzorom označujeme. Toto správanie môžeme vidieť na obrázku 3.5.

Obr. 3.5: Na obrázku je vidno, ako pri prejdení s kurzorom nad objekt, ktorý má štítok SelectableTag sa objekt zvýrazní a jeho názov sa napíše do kurzora.

3.3.2 Implementácia menu

Vytvoríme si v našom prostredí objekt plátno. Rovnako ako v kapitole 3.1.1 Implementácia kurzora, potrebujeme zakaždým pri otvorení menu meniť rotáciu tohoto plátna tak, aby bolo stále otočené smerom ku kamere. To dosiahneme analogicky použitím transform funkcií na toto plátno.

Naše menu momentálne pozostáva z ôsmich tlačidiel. Tie sme rovnomerne rozmiestnili po obvode elipsy tak, aby medzi nimi bol dostatočný odstup. Nechceme aby sa mohlo stať, že by užívateľ pri výbere zvolil iné tlačidlo, ako by si žiadal.

Všetkým tlačidlám sme pridali štvorcovú detekciu zrážky (takzvaný box collider) po ich obvode. Znamená to, že v momente, keď prejdeme na jedno z týchto tlačidiel, náš skript na vyberanie objektov popisovaný v minulej kapitole zachytí zrážku kurzora s daným tlačidlom. Vtedy už len treba spustiť funkciu správania pridelenú danému tlačidlu. Napríklad do tlačidla štyri, nachádzajúceho sa priamo dole sme pridali funkcionalitu Cancel, ktorá má za úlohu pri prejdení kurzora nad týmto tlačidlom menu vypnúť. Takto môžeme pridať ľubovoľnú funkcionalitu každému tlačidlu iba pomocou toho, že pri prejdení nad konkrétne tlačidlo vieme detekovať názov tlačidla.

```
if (hit.collider.name == "Button4")
{
    CloseMenu();
}
```

Pre naše testovanie sme si zvolili funkcionalitu nasledovne. Päť z tlačidiel sa budú používať na zmenu farby objektu, čo budeme využívať ako úlohu na kontrolných bodoch našej testovacej scény. Spodné tlačidlo využijeme ako tlačidlo na vypínanie menu, ak by sme sa po vykonaní pohybu rozhodli, že nechceme vykonať žiadnu z ponúkaných akcií. Posledné dve tlačidlá nachádzajúce sa na pravej a ľavej strane menu použijeme na otáčanie orientácie kamery o 90 stupňov vpravo, respektíve vľavo.

Každé z tlačidiel má vlastný text objekt, do ktorého môžeme vložiť krátky popis akcie, ktoré dané tlačidlo vykonáva. Tu by nastal problém, ak by sme využívali metódu

sledovania pohybu očí a nie sledovania pohybu hlavy. Problém s tým, že ak by sme prešli nad tlačidlo iba zrakom a nie kurzorom, pri snahe prečítať si daný popis tlačidla, detekovali by sme prienik lúča a nášho zraku. Vtedy by sa vykonala akcia predtým, ako by sme si mohli prečítať daný popis. Pri používaní sledovania pohybu hlavy sa toto ale nestáva, pretože hlava nemusí smerovať rovnakým smerom ako náš zrak. Najprv si popisy vieme prečítať a následne po rozhodnutí, na ktoré tlačidlo chceme prejsť, tak vieme urobiť pohybom hlavy a presunutím kurzora na dané tlačidlo.

Pre plne korektné ukončenie stlačenia každého tlačidla by mala okrem vykonania jeho konkrétnej funkcionality nastať aj akcia zatvorenia menu a reštartovania zámku na odomknutú hodnotu, aby bola umožnená ďalšia interakcia s prostredím. Toto je posledný krok pri používaní gesta v našom užívateľskom prostredí.

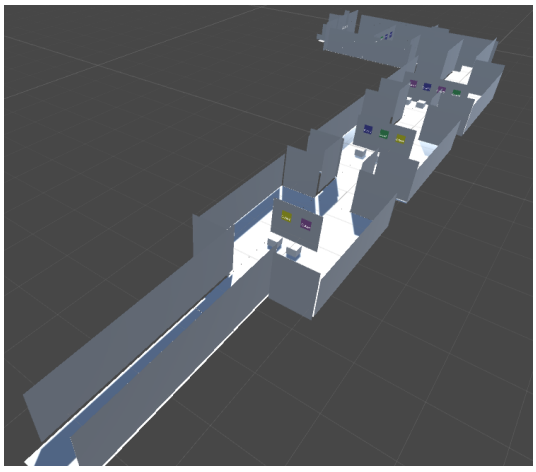
Kapitola 4

Testovanie a výsledky

Na testovanie nášho užívateľského prostredia sme si vytvorili jednoduchú scénu s niekoľkými prekážkami, ktoré musíme prekonať. Budeme pri tom merať čas respondentov, za ktorý stihli prejsť jednotlivými kontrolnými bodmi. Tiež budeme merať, koľko krát sa im pri celkovom teste stalo, že sa naše menu snažili gestom otvoriť, no z nejakého dôvodu sa im to nepodarilo.

Takto spravíme dokopy tri testy, každý s mierne pozmenenými konštantami, ako sú napríklad dĺžka pohybu smerom nadol pri vykonávaní gesta smerom nadol, zvýšenie či zníženie limitu časovača pri vykonávaní vedľajšieho gesta.

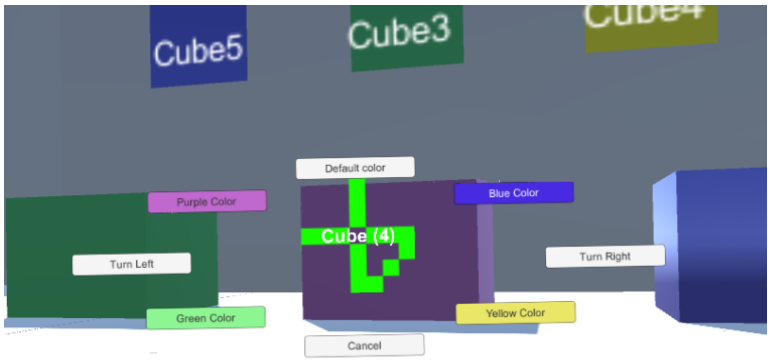
V tejto scéne sa nachádza jednoduchý tunel (vid obrázok 4.1), ktorým respondent musí prejsť. V tuneli sa bude nachádzať päť kontrolných bodov, na štyroch z nich bude musieť respondent meniť farby kociek podľa vzoru ukázaného na stene za kockami. Pritom si musí dávať pozor, pretože farby na stene nie sú uložené v poradí, ale je na nich napísané, ktorej kocke patrí určená zo štyroch farieb. Musí si teda stále kontrolovať správnosť kocky v momente, keď na ňu presunie kurzor. Názov kocky zobrazený na kurzore bude musieť zodpovedať farbe na stene, kde je tento názov napísaný tiež (pozri



Obr. 4.1: Scéna s tunelom, v ktorom sú prichystané úlohy pre testovacie účely.

obrázok 4.2).

Jedno zo stanovíšť, nachádzajúce sa tesne pred koncom, je zamerané na rýchlosť pohybu a na 90 stupňovom otočení. Respondenta musí teda napadnúť, že v menu má možnosť tohto otočenia a snažiť sa na to rýchlo zareagovať.



Obr. 4.2: Jedna z úloh v testovacej scéne. Na obrázku vidíme tri zafarbené kocky. Ďalej vidíme predlohu farieb na stene, ktoré značia akou farbou majú byť jednotlivé kocky zafarbené. Vidíme, že kocka číslo štyri je zafarbená nesprávne. Mala by byť zafarbená na žlto a nie na fialovo, preto to respondent musel znovu nafarbiť.

Pred každým testom boli respondentom vysvetlené a ukázané všetky vzory. Tiež im bolo povedané, koľko času na každý vzor majú. Boli im vysvetlené úlohy, ktoré ich čakali. Pred tým, ako sme ich pustili do testovacej scény a začali merať časy, nechali sme ich behať po miestnosti s pár objektami, aby si mohli krátko na minútku vyskúšať dané vzory. Bolo to z dôvodu porovnávania jednotlivých testov. Ak by sme im tento čas nenechali, ich časy by boli veľmi skresané v druhom a treťom teste. Dôvod skresania výsledkov by bol zapríčinený tým, že tieto vzory by už mali vyskúšané z prvého testu, kde by si museli od nuly zvykať na dané pohyby.

Respondent číslo 2, 3 a 5 vraveli, že nejakú skúsenosť s virtuálnou realitou už v minulosti mali, zvyšní traja (respondenti s číslom 1, 4 a 6) zasa, že okuliare pre virtuálnu realitu nikdy nevyskúšali. Očakávaný výsledok je, že s predošlou skúsenosťou s virtuálnou realitou, by tento test mal byť úrovňou ľahší a teda vykonaný rýchlejšie. Pre nás je ale najdôležitejšie to, aby s ovládaním nemal problém žiaden z nich a aby sa po krátkom používaní stávalo čoraz menej, že sa im nepodarí spraviť gesto správne.

Prvá konfigurácia bola pôvodná, nami určené hodnoty, ktoré sme si mysleli, že by mohli vyhovovať. Po každom vyskúšaní sme sa spýtali, čo im robilo problémy, respektíve, ako by dané pohyby pozmenili. Po krátkom vyskúšaní v miestnosti s niekoľkými objektami sme ich nechali vojsť do testovacieho tunela.

Pri zmene konfigurácií sme sa zamerali hlavne na tieto hodnoty:

1. Dĺžka časovača, detekujúceho nehybnosť hlavy. Úzko spojené s dĺžkou času potrebného na vykonávanie pohybu kamery vpred, pretože vedľajšie gesto vyžaduje

zastavenie kurzora na krátky čas pred tým, ako ho môžeme vykonať.

2. Dĺžka pohybu smerom nadol pri hlavnom vzore. Táto hodnota súvisí so zameriavaním objektu, na ktorom chceme otvoriť menu, pretože zámok spúšťame až po vykonaní tohoto pohybu. Čím väčšia je hodnota, tým náročnejšie je objekt vybrať. Príliš malá hodnota by však znamenala neustále zamykanie zámku a následné znemožnenie vykonania pohybu kamery (na to, aby sme detekovali zastavenie kurzora musí byť zámok odomknutý).

3. Diagonálna dĺžka pri hlavnom pohybe a upravenie veľkosti jeho pásma, z ktorého užívateľ nemôže vyjsť pre korektné dokončenie pohybu.

Po prvom teste sa všetkým z respondentov (v tabuľkách ďalej už iba R) zdalo, že pohyby a časy sú príliš zdĺhavé (časy z testu jeden sú zapísané v tabuľke 4.1), preto sme pre test číslo dva upravili v konfigurácii hodnoty. Keď sa im nepodarilo otvoriť menu, bolo to väčšinou z dôvodu diagonálneho pohybu, ktorý bolo treba vykonať viac smerom doľava ako hore, čo sa im nezdalo úplne intuitívne. Druhý z najčastejších dôvodov zlyhania bolo netrafenie objektu pri hlavnom geste smerom nadol. Počet zle otvorených menu možno vidieť v tabuľke 4.2.

Preto sme nasledovnú konfiguráciu pre test číslo dva upravili nasledovne. Pohyb smerom nadol pri hlavnom geste sme zmenšili o päť stupňov (pôvodná dĺžka bola urobenie otočenia o desať stupňov, teda sme potrebnú dĺžku zmenšili na polovicu). Hodnota časovača pri vedľajšom geste bola znížená zo 400 milisekúnd na 300 milisekúnd. Končiacu zónu pri diagonálnom pohybe sme posunuli o dva stupne doľava, čím sme viac vyvážili rozdiel dĺžok, ktoré musíme prejsť v smere doprava a hore.

Kontrolný bod	R1	R2	R3	R4	R5	R6
1	1:22	0:44	0:30	1:02	0:37	1:14
2	2:55	1:36	1:02	2:28	1:21	2:37
3	4:13	2:39	1:41	3:59	2:09	4:02
4	5:07	3:16	2:06	4:40	2:45	4:37
5	7:15	4:30	2:58	5:57	3:56	5:30

Tabuľka 4.1: Výsledky časov z testov pri prvej konfigurácií hodnôt

	R1	R2	R3	R4	R5	R6
Počet zlyhaní	7	2	1	4	2	2

Tabuľka 4.2: Počet zlyhaných gest respondentov pri prvej konfigurácii hodnôt

Následne sme znovu nechali respondentov odskúšať si vykonané zmeny na pár minút v miestnosti s objektami a následne sme ich zase pustili do tunela spraviť test. Výsledky z druhého testu možno vidieť v tabuľkách 4.3 a 4.4.

Väčšina respondentov si novú konfiguráciu pochvaľovala, až na respondenta číslo šesť, ktorý mal problémy s vykonávaním vedľajšieho gesta na pohyb. Vraveli, že menu išlo otvárať ľahšie, rýchlejšie a zameriavanie objektov pri spúštaní menu bolo presnejšie. Vykonanie pohybu sa im zdalo plynulejšie, pretože sme skrátili možný čakací čas medzi jednotlivými posunmi. Zodpovedajú tomu aj výsledky z druhého testu, kde všetci dosiahli rapídne zlepšenie oproti prvému testu. Treba tu brať do úvahy, že časť zlepšenia výkonu mohla nastať aj kvôli tomu, že po skúšaní prostredia pred prvým a druhým testom a zároveň aj po vykonaní prvého testu si mohli viac zvyknúť na dané pohyby. Tie potom samozrejme robili presnejšie ako v prvom teste, keď ešte úplne netušili, čo sa od nich očakáva.

Kontrolný bod	R1	R2	R3	R4	R5	R6
1	0:50	0:24	0:20	0:52	0:31	1:05
2	1:53	0:57	0:53	1:42	1:02	2:02
3	2:46	1:39	1:33	2:58	1:52	2:58
4	3:26	2:01	1:48	3:38	2:20	4:12
5	4:26	2:46	2:28	4:43	3:10	5:20

Tabuľka 4.3: Výsledky časov z testov pri druhej konfigurácií hodnôt

	R1	R2	R3	R4	R5	R6
Počet zlyhaní	2	1	0	0	1	2

Tabuľka 4.4: Počet zlyhaných gest respondentov pri druhej konfigurácii hodnôt

Pre test číslo tri sme opäť zvolili zmenšiť hodnoty, ktoré prispeli k zlepšeniu času v druhom teste. Pohyb smerom nadol v hlavnom geste sme zmenšili na polovicu - 2,5 stupňa. Diagonálny pohyb sa piatim zo šiestich respondentov zdal v druhom teste primeraný, teda túto hodnotu sme viac neposúvali. Nechceme, aby diagonálny pohyb mal veľmi rozdielne dĺžky, ktoré musíme prejsť v smere doprava, ako aj v smere dohora. Časovač pri vedľajšom geste sme skrátili o ďalších 100 milisekúnd.

Keď sme ich nechali pred testom vyskúšať zmeny v malej miestnosti s objektami, prišli prvé sťažnosti. Pohyb smerom nadol v hlavnom pohybe bol až príliš krátky a často sa stávalo, že sa vykonal samovoľne, aj keď si to neželali. Keďže po vykonaní tohto pohybu aktivujeme zámok, ktorý dočasne uzamkne vedľajšie gesto, často to znamenalo

znemožnenie pohybu. Ďalej to znemožňovalo rýchle zameriavanie objektov, pretože tam tiež potrebujeme, aby bol zámok uvoľnený.

Aj napriek nežiaducemu správaniu užívateľského prostredia sme ich nechali zbehnúť test s treťou konfiguráciou. Ukázalo sa, že táto konfigurácia je ešte viac nepoužiteľná, ako sme si mysleli. Museli sme po desiatich minútach test stopnúť, pretože sa respondenti nemohli poriadne hýbať, nehovoriac o zameriavaní objektov. Počet nekorektne spravených hlavných gest sa pohyboval rádovo v desiatkach (vzhľadom na výsledky z prvého a druhého testu sme to prestali rátať v momente, kedy počet zle vykonaných gest presiahol hodnotu 20 - tabuľka 4.6). U respondenta s číslom jeden by sme dokonca povedali, že sa toto číslo pohybovalo v stovkách, vzhľadom na to, že nedokázal prekročiť ani prvý kontrolný bod po desiatich minútach. Ostatní síce prešli týmto kontrolným bodom, no tiež sa nedostali ďaleko a tiež sme ich po desiatich minútach zastavili (vid tabuľka 4.5).

Nemá zmysel si voliť ďalšie hodnoty, ktoré by sa nachádzali medzi hodnotami z druhej a tretej konfigurácie, keďže rozdiel by tam bol minimálny až zanedbateľný. Ukázalo sa, že menšie hodnoty sú pre človeka ľahšie na zopakovanie a sú menej náchylné na chyby. Netreba ale ísť do úplných extrémov, kedy naše telo nebude schopné na dané rýchlosti reagovať.

Pýtali sme sa na grafické vylepšenia, ktoré by mohli prispieť ku kvalite nášho užívateľského prostredia. Niektorým trochu vadilo, že aj po otvorení menu nám kurzor zostal vo vzdialenosti objektu, na ktorý ukazuje. Preto sa prišlo s návrhom pridania obrázku tesne za naše menu. Tomuto obrázku sme zapli detekciu kolízie, čo znamená to, že v momente otvorenia menu sa kurzor zobrazí na vrchu daného obrázku. Traja zo šiestich ľudí povedali, že po tejto zmene ide ľahšie vyberať jednotlivé položky z menu. Dvom respondentom sa zdalo, že v tomto prípade je kurzor až príliš blízko, takže sa im táto zmena nepáčila. Poslednému vyhovoval aj jeden aj druhý spôsob. Ten tvrdil, že aj

Kontrolný bod	R1	R2	R3	R4	R5	R6
1	99:99	2:21	9:55	6:15	8:52	7:36
2	99:99	5:36	99:99	9:24	99:99	99:99
3	99:99	99:99	99:99	99:99	99:99	99:99

Tabuľka 4.5: Výsledky časov z testov pri tretej konfigurácii hodnôt

	R1	R2	R3	R4	R5	R6
Počet zlyhaní	20+	20+	20+	20+	20+	20+

Tabuľka 4.6: Počet zlyhaných gest respondentov pri tretej konfigurácii hodnôt

keď je kurzor väčší kvôli nárazu na obrázok v menu. Náraz znamená, že lúč zaznamenal prienik s obrázkom a teda kurzoru sa nastavila iba vzdialenosť po tento obrázok. Nepáči sa mu nárazový skok, ktorý kurzor spraví pri otvorení menu. Vravel, že bez tohto obrázka mal kurzor stále konzistentnú veľkosť, ktorá sa menila plynule.

Ďalším zaujímavým poznatkom bolo to, že po dokončení testovania sme sa ich spýtali, či si pamätajú rozloženie tlačidiel a ich funkcionalitu. Piati zo šiestich nám presne povedali, ktoré tlačidlo sa kde nachádzalo a akú malo úlohu. Posledný respondent vymenil tlačidlá, ktoré menili farbu na zeleno a žlto.

Spýtali sme sa ich, ktoré z troch nastavení sa im zdalo najlepšie. Tu bola jednotná a jednoznačná odpoveď konfigurácia číslo dva. Respondent číslo tri povedal: Övládanie bolo jednoduché a vôbec ma nemiatlo. Po malej zmene v hodnotách sa to dokonca správalo ešte lepšie, ako som očakával.

Záver

V tejto práci sa nám podarilo splniť náš hlavný cieľ, čím bola implementácia užívateľského prostredia, ktoré by bolo jednoduché a intuitívne. Prostredie, v ktorom sme povolili jednoduchú, no rozmanitú manipuláciu s prvkami vo virtuálnej realite.

Splneniu tohto hlavného cieľa predchádzalo splnenie čiastkových cieľov, ako napríklad preskúmanie hotových, bežne dostupných riešení. Tam sme objavili nedostatky hlavne pri manipulácii s prvkami virtuálnej reality pri absencii externých ovládačov. Preto sme sa pri jeho navrhovaní snažili sústrediť na miesta, ktoré sme si mysleli, že by sa dali zlepšiť. Vyvinuli sme teda užívateľské prostredie, ktoré nepotrebuje na interakciu žiadne ďalšie externé prístroje, okrem okuliarov samotných.

Pomocou pohybu okuliarov rozpoznávame pohyb užívateľa, ktorý si môže vyberať z viacúčelového menu, aké akcie chce vykonávať. Hlavnou prioritou bolo do našej implementácie zakomponovať možnosť výberu a rozmanitosť príkazov.

Sekundárnym cieľom bolo toto prostredie otestovať aj s ľuďmi, ktorí sa s virtuálnou realitou zatiaľ nestretli. Spýtali sme sa ich na názory k našej implementácii a snažili sme sa ich pripomienky ďalej zapracovať v našom prostredí. Po absolvovaní viacerých testov s našou implementáciou sme sa dozvedeli, že naozaj nemali žiadny problém s jej využívaním. Jej používanie nazvali intuitívnym a dobre reaktívnym.

Literatúra

[1] HTC Vive pro eye. URL: https://www.vive.com/eu/product/vive-pro-eye/. [Online, Citované 16-5-2019].

[2] Interaction in VR. https://unity3d.com/learn/tutorials/topics/virtual-reality/interaction-vr/. [Online, Citované 28-11-2017].

[3] KOR-FX gaming vest. URL: http://korfx.com/products/#Features. [Online, Citované 4-3-2018].

[4] Unity Manual. https://docs.unity3d.com/Manual/. [Online, Citované 30-11-2017].

[5] Unity scripting API. https://docs.unity3d.com/ScriptReference/. [Online, Citované 30-11-2017].

[6] VirZoom VR Bike. URL: https://get.virzoom.com/. [Online, Citované 4-3-2018].

[7] Doug A. Bowman and Larry F. Hodges. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *Visual Languages and Computing*, 10:i37–i53, 1999.

[8] Liza Brown. A brief history of virtual reality. URL: https://filmora.wondershare.com/virtual-reality/history-of-vr.html, Nov 2017. [Online, Citované 4-3-2018].

[9] Armin Zahirovic et al. *The Encyclopedia of Human-Computer Interaction*, chapter 32. The Interaction Design Foundationl, 2 edition, 2013.

[10] Steven M. LaValle. *Virtual Reality*. Cambridge University Press, 2017.

[11] Rob Miles. *C# Programming Yellow Book*. Department of Computer Science at the University of Hull, 2016.

Appendix A

Na priloženom CD sa nachádzajú zdrojové súbory projektu spolu s testovacím prostredím, ktoré sme v práci používali.