FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Návrh a implementace aplikace pro prohlídku historického města ve virtuální realitě
Student:	Bc. Matěj Sedlák
Vedoucí:	Ing. Petr Pauš, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je návrh a implementace řešení pohybu a manipulace s předměty ve VR podle výsledků analýzy a Usability testování. Výsledkem práce bude balíček assetů pro Unity3D obsahující implementace výsledných řešení pro pohyb a manipulace s předměty. Výsledný balíček bude implementován a otestován v aplikaci určené k virtuálnímu procházení českých historických budov a lokalit, vytvářené ve spolupráci s Filosofickou fakultou UHK.

Cíle práce:
1. Analýza aktuálních řešení pohybu a manipulace s předměty ve VR aplikacích.
2. Návrh sady testovacích příkladů v závislosti na typu aplikace pro použití.
3. Implementace navržených testovacích příkladů.
4. Usability testování navržených testovacích příkladů.
5. Vyhodnocení testování.
6. Návrh a implementace výsledného balíčku pro pohyb a manipulaci s předměty ve VR.
7. Implementace a testování ve výsledné aplikaci.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 6. ledna 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Návrh a implementace aplikace pro prohlídku historického města ve virtuální realitě

Bc. Matěj Sedlák

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petr Pauš, Ph.D.

30. července 2020

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 30. července 2020 .....

Odkaz na tuto práci

Sedlák, Matěj. *Návrh a implementace aplikace pro prohlídku historického města ve virtuální realitě*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato diplomová práce je zaměřena na návrh, implementaci a testování řešení pohybu a manipulace s předměty ve virtuální realitě. Výstupem této práce je balíček pro vývojáře obsahující implementované způsoby pohybu a manipulace s předměty. V tomto balíčku jsou implementovány 4 způsoby pohybu. První dva využívají teleport, z nichž první je teleport přímý, jeho alternativou je teleport obloukový. Dále je dostupný pohyb pomocí joysticku a mávání rukama. Pro manipulaci s objekty jsou implementovány dva způsoby. První způsob využívá ke zvětšování, rotaci a pohybu joystick, druhý pohyb ovladačem. Tyto způsoby manipulace byly navrženy pro práci s historickými budovami.

Součástí práce je dále testování implementovaných funkcionalit a následná úprava implementace na základě výsledků testování. Při testování byly použity modely historických budov poskytnutých Filosofickou Fakultou UHK. Po sekci testování je popsán proces vytváření a nahrávání výsledného balíčku na Unity AssetStore, kde jsou funkcionality dostupné zdarma ostatním vývojářům.

Klíčová slova Virtuální realita, Unity, Pohyb, Manipulace

Abstract

The master's thesis you are having in front of you is focusing on movement solution design, implementation and testing, and object manipulation in virtual reality. Output of the thesis is a package for developers containing implemented movement solutions and object manipulation options.

The package is composed of four movement types. First two use teleport technique, direct teleport and arched teleport. Other two use joystick and physical arm swing motion. For object manipulation two solutions are implemented. First is using joystick for object size change, rotation and movement, the second one uses controller movement.

Part of this thesis are also usability tests of implemented functionalities and their subsequential adjustment on the basis of the test results. Historical buildings models used in the tests were provided by Philosophical Faculty UHK. Last thesis segment is dedicated to the final package creation and upload process on Unity AssetStore, where it's available for free to other developers.

Keywords Virtual reality, Unity, Movement, Manipulation

Obsah

Seznam obrázků

Úvod

V dnešní době se virtuální realita stává stále více populárnější a cenově dostupná. Pro virtuální realitu existuje mnoho technologií, od použití mobilního telefonu (Google cardboard) až po sofistikované technologie snímající pohyb člověka v prostoru. Aplikace pro virtuální realitu jsou především hry, ale i výukové aplikace. Možnosti využití jsou popsány v několika publikacích, např. [2].

Práce je zaměřena na návrh a implementaci různých způsobů pohybu a manipulace s předměty ve virtuální realitě podle výsledků analýzy a usability testování (uživatelské testování). Součástí této práce je implemetace balíčku pro Unity, který bude obsahovat implementované funkcionality pro použití na různých projektech ostatních vývojářů. Součástí tohoto balíčku budou i jednoduché prvky uživatelského rozhraní, potřebné pro výběr nástrojů pro manipulaci s objekty. Implementované způsoby pohybu a manipulace s předměty ve virtuální realitě budou testovány v aplikaci určené k virtuálnímu procházení historických budov a lokalit, která bude vytvářena s použitím 3D modelů památek, které poskytla Filosofická fakulta UHK.

Začátek práce se zabývá analýzou stávajících apliakcí pro virtuální realitu, zejména těmi se zajímavými způsoby pohybu. Tyto aplikace budu zkoumat z hlediska toho, v čem by se dalo inspirovat a čemu se vyvarovat. Zároveň je důležitá i analýza technologií, které se při implementaci použijí. Dále bude následovat návrh způsobů pohybu a manipulace s předměty na základě předchozí analýzy a další literatury. Po návrhu bude popsána implementace těchto způsobů a následné testování. Na závěr bude stávající implementace upravena podle výsledků testování a bude vytvořen výsledný balíček assetů pro Unity.

Toto téma jsem si zvolil, protože mě už od mala bavilo hrát hry na počítači a později i programovat hry a v budoucnu bych se tomuto chtěl věnovat. Použití virtuální reality ve hrách je poměrně nové téma a rád bych se touto metodou naučil nové věci. Zároveň je vývoj aplikací pro virtuální realitu poměrně odlišný od vývoje klasických desktopových 3D aplikací.

Kapitola 1

Cíl práce

Cílem práce je návrh a implementace různých způsobů pohybu a manipulace s předměty ve virtuální realitě na základě výsledků analýzy a Usability testování. To zahrnuje analýzu stávajících VR aplikací, které zajímavým způsobem řeší pohyb uživatele v prostoru a manipulaci s předměty. Dále je důležité analyzovat samotnou VR platformu a vývojové prostředí z hlediska toho, jaké všechny možnosti tyto platformy nabízí. Cílem návrhu je vymyslet sadu způsobů pohybu na základě předešlé analýzy aplikací a her a zároveň zohlednit stávající literaturu týkající se tohoto tématu. V části realizace je cílem implementace navržených řešení a následné otestování. Důležité je, aby byli testováni jak uživatelé, kteří mají jisté zkušenosti s virtuální realitou, tak ti, kteří virtuální realitu vyzkouší poprvé. Zároveň je důležité, aby byly vyzkoušeny různé způsoby pohybu pro různé příležitosti, například pohyb ve větším otevřeném prostoru a pohyb v menších prostorách. Pro účely testování dojde ke spojení s prací Bc. Petra Polívky, který řeší uživatelské rozhraní ve virtuální realitě a manipulaci s historickými artefakty. Po testování bude následovat úprava implementace na základě výsledků testování a vytvoření výsledného balíčku pro Unity, který bude dostupný ostatním vývojářům aplikací pro virtuální realitu.

Kapitola 2

Analýza a návrh

V této kapitole se budu zabývat analýzou dostupných platforem pro vývoj aplikací pro virtuální realitu. Budou zde rozebrány možnosti enginů pro virtuální realitu. Zároveň zde budu analyzovat stávající aplikace a hry pro virtuální realitu.

2.1 Analýza dostupných platforem

Součástí této části práce je analýza dostupných platforem pro vývoj výsledné aplikace. Jako první je důležité zvolit vývojové prostředí a engine pro vývoj. Ideální engine k vývoji aplikací pro virtuální realitu je ten, který je primárně určený na hry a 3D aplikace, protože pro virtuální realitu musíme vyvíjet ve 3D. Současně musíme zvolit engine, který podporuje nějaký framework pro jednoduché programování VR aplikací.

2.1.1 Herní enginy pro vývoj VR aplikací

Při výběru herního enginu máme několik možností. Prvním z nich je Unreal Engine[3]. Tento herní engine je dostupný zdarma, platí se pouze procenta, pokud za aplikaci vyděláme určité množství peněz za čtvrtletí. Další možností je Unity[4] engine. Ten je také zdarma pro menší vývojáře, pro větší týmy se platí měsíční poplatek. Třetí engine, který zde rozeberu je CryEngine[5]. Tento engine již není zdarma, jeho použití je zpoplatněno měsíčním poplatkem. Všechny zmíněné enginy jsou v dnešní době jedny z nejpoužívanějších pro vývoj her na PC.

CryEngine je z těchto tří enginů asi nejnáročnější pro nové uživatele, ale na druhou stranu nabízí obrovské grafické možnosti. Skripty pro tento engine se programují v jazyce LUA. Jako všechny tři tyto enginy nabízí obchod, ve kterém si můžeme různé komponenty stahovat, buď za poplatek, nebo zdarma.

Unreal Engine také nabízí obrovské grafické možnosti. Komponenty pro aplikaci si opět můžeme kupovat/stahovat v integrovaném obchodě. Progra-

movací jazyk pro skripty se zde používá C++. Pro začínající vývojáře je zde možnost řídit chování aplikace místo klasických scriptů pomocí grafického programování, tzv. blueprints.

Unity je z těchto tří enginů asi nejpřívětivější pro začínající vývojáře, i když nenabízí možnost plnohodnotného grafického programování, které je zde zatím ve vývoji. Na druhou stranu nabízí možnost psát skripty ve třech programovacích jazycích. Zároveň má z těchto tří enginů nejrozšířenější podporu různých cílových platforem, od mobilních telefonů, webových aplikací, přes počítače až po konzole. Obchod s komponentami je také větší, než u ostatních enginů dle článku[6].

Z těchto tří možností jsem zvolil pro vývoj engine Unity. Jedním z důvodů tohoto rozhodnutí je to, že je dostupné zdarma. Dále díky velké možnosti získávání komponent do aplikace z obchodu. Zároveň s tímto enginem už mám nějaké zkušenosti. Kromě výběru vhodného enginu je také důležité vybrat vhodný framework, který slouží k pohodlnému vyvýjení aplikací pro virtuální realitu. Asi nejrozšířenějším frameworkem je SteamVR, hlavně díky dostupnosti v obchodě Steam, který je asi nejrozšířenějším obchodem pro hry. Vyvíjení aplikací pomocí SteamVR má tu výhodu, že tento framework zajišťuje kompatibiliu s velkým počtem hardwaru pro virtuální realitu. Steam VR [7] je dostupný jako plugin pro Unity i Unreal Engine.

2.2 Analýza použitých platforem

V této kapitole se budu zabývat analýzou platforem, které budou v této práci použity. Jako vývojové prostředí a engine jsem zvolil Unity, protože je dostupný zdarma a v dnešní době je poměrně rozšířený obzvlášť v oblasti vývoje aplikací pro virtuální realitu. To znamená, že pro tento engine existuje mnoho návodů a veliká komunita vývojářů. Zároveň je pro něj dostupný balíček SteamVR, který zajišťuje ovládání hardwaru pro virtuální realitu a tudíž je vývoj aplikací jednodušší. Díky rozšíření tohoto enginu bude mít výsledný balíček dobré uplatnění. Pro psaní skriptů jsem si zvolil jazyk C#, protože už mám nějaké zkušenosti s vyvíjením aplikací v Unity za použití tohoto jazyka. Pro psaní skriptů budu používat program MS Visual Studio Community 2019, který je dostupný zdarma, a Unity má pro tento editor rozšiřující balíček, který velice usnadňuje psaní skriptů pro Unity.

2.2.1 Unity

Unity je jeden z nejrozšířenějších enginů v dnešní době a pro jednotlivce, nebo malé vývojové studia je zdarma. Primárně je určený k vývoji her, ale dají se v něm vyvíjet například i edukativní aplikace. Unity také obsahuje rozsáhlou dokumentaci, jak ze strany vývojářů, tak ze strany komunity a zároveň o vývoji her na tuto platformu vyšlo několik publikací, kde za zmínku stojí např. [8]. Jednou z jeho největších výhod je možnost vyvíjet aplikace na téměř všechny

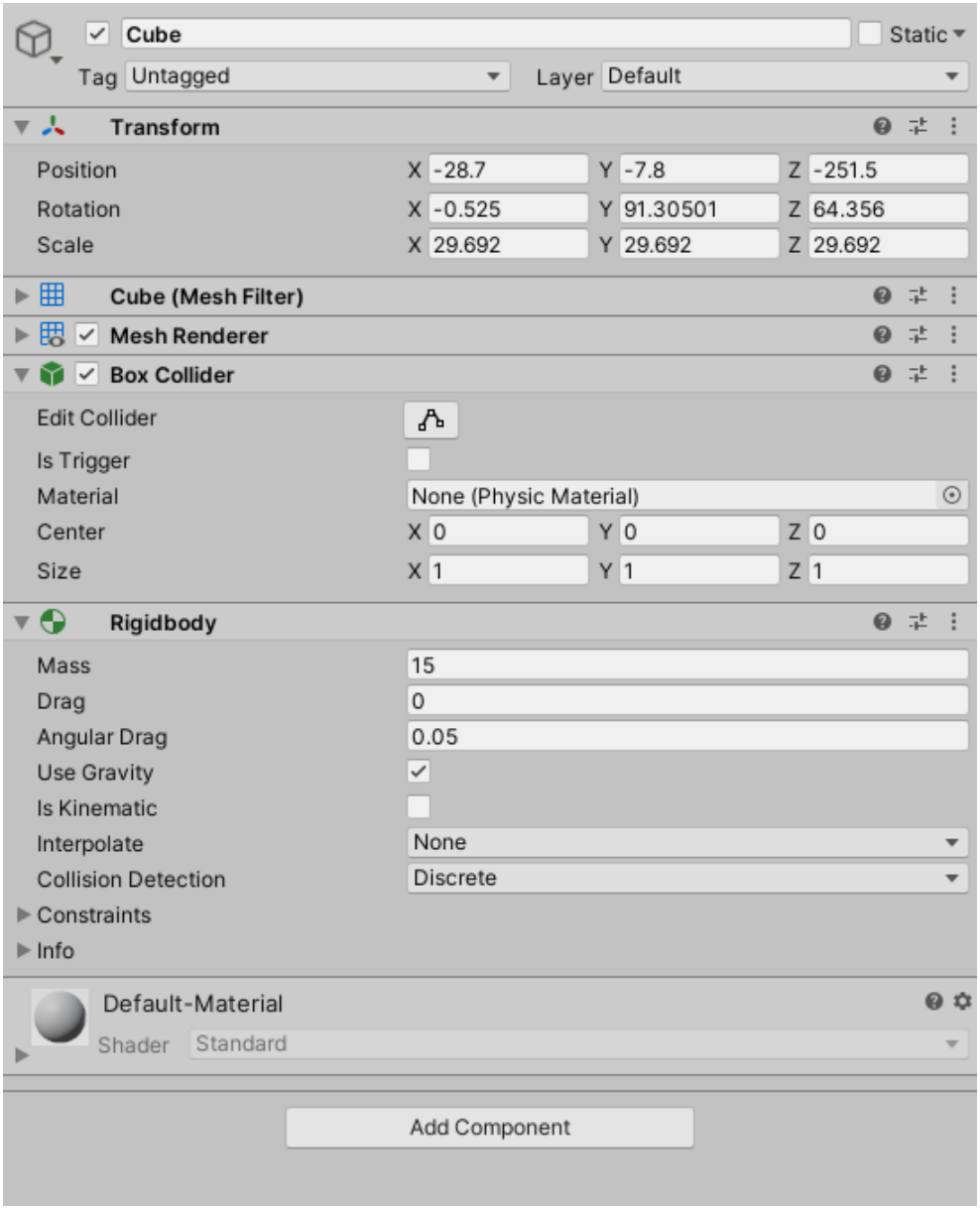
platformy (PC, iOS, Android, Linux, PlayStation, Xbox, atd.). Tento engine umožňuje vyvíjet 3D i 2D aplikace. Engine používá komponentovou architekturu, tzn. že chování aplikace se skládá z menších komponentů, které mohou fungovat nezávisle na sobě, např. komponenta renderování, pohybu, kolize, atd. Lze vidět na obrázku níže 2.1. Ukázkový objekt zde má několik komponent. Jednou z nich je komponenta *Box Collider*, která ohraničuje kolizní oblast tohoto objektu a zajišťuje, že s objektem lze kolidovat. Další komponentou je *Rigidbody*, která slouží k tomu, aby na tento objekt byla aplikovaná simulace fyziky. Například zajišťuje gravitaci, nebo pokud do sebe dva objekty s komponentami kolizí a *Rigidbody* narazí, vypočítají se odrazy objektů na základě jejich energie a váhy. Jako poslední komponenta tohoto objektu je materiál, což zajišťuje, jak bude objekt v aplikaci vypadat. Součástí materiálu je tedy kromě textury pro barvu objektu i textura pro odraz světla, nebo nerovností na povrchu. Dále je možné v materiálu určovat, jak bude objekt matný, nebo jestli bude emitovat světlo. Parametry všech komponent lze libovolně editovat, čímž se mění chování komponent. Kromě možnosti změny parametrů v editoru lze tyto parametry měnit i ve skriptech. K tomuto se použije metoda *GetComponent()*, kde uvedeme název požadované komponenty a můžeme její parametry měnit.

Každá aplikace v Unity má assety, což jsou soubory, ze ktrých se aplikace skládá. Mezi assety patří 3D modely, textury, zvuky, skripty, atd. Součástí aplikace jsou i scény, do kterých umisťujeme herní objekty. Tyto objekty obsahují zmíněné assety jakožto komponenty. Chování jednotlivých objektů řídí skripty. V Unity lze psát skripty v jazyce JavaScript, C# nebo Boo. Zároveň je možné psát skripty, které nějakým způsobem upravují chování samotného enginu a editoru. Do assetů je možné i vkládat objekty, umístěné do scény jako tzv. *Prefabs*, což jsou assety, které už mají v sobě různé komponenty a nastavení. Toto se používá pokud např. chceme instancovat jeden konkrétní objekt s nastavenými komponentami.

Jednou z výhod Unity je přítomnost tzv. obchodu, kde je možné si stáhnout výše zmíněné assety, některé jsou zcela zdarma a jiné za poplatek. Cílem této práce je právě balíček assetů, který bude umístěn do tohoto obchodu pro ostatní vývojáře. Výhodou tohoto obchodu je, že lze naprogramovat hru, nebo aplikaci bez znalosti 3D modelování, protože potřebné modely, nebo UI prvky lze v tomto obchodě získat.

Při psaní skriptů pro herní objekty používáme třídu *MonoBehaviour*, od které potom dědí námi definované třídy. V námi definovaných třídách lze implementovat několik metod, které zajišťují ovládání a řízení objektů, které obsahují jako komponentu tuto třídu.

- Metoda *Update()*, která se vyvolá při každém snímku během běhu aplikace. Pokud tedy potřebujeme provádět nějakou akci, například detekci stisknutí tlačítka, lze tuto metodu použít. Pro akce po stisku tlačítka lze použít i tzv. Handeler, ale ty vyžadují větší režii, protože je musíme



Obrázek 2.1: Ukázka komponent objektu v Unity.

na začátku aplikace přidat a při změně scény opět odebrat. U této metody nevíme, kolikrát se provede, protože počet snímků za sekundu je určen výkonem cílového zařízení a výpočetní náročností aplikace. Pro náročnější fyzické kalkulace použijeme alternativní metodu *FixedUpdate()*, která se vyvolává současně s kalkulací fyzických objektů. Poslední alternativou k metodě *Update()* je metoda *LateUpdate()*, která se volá až po doběhnutí *Update()* metod všech objektů. Je tedy vhodná, pokud potřebujeme udělat nějakou akci po tom, co se nějaký objekt pohne, když má pohyb řešený v metodě *Update()*. Vzhledem k volání těchto tří metod v závislosti na snímcích za sekundu není možné v těchto metodách vlákno uspávat.

- *Start()* metoda se vyvolá, pokud je herní objekt s touto komponentou vytvořen poprvé. Je tedy vhodná pro inicializaci objektů při startu aplikace.

- Metoda *Awake()* je podobná metodě *Start()*. Liší se v tom, že metoda *Awake()* se vyvolává při každém vytvoření objektu a ne jen při prvním vytvoření. Toto může být vhodné, pokud potřebujeme objekt inicializovat při každém vytvoření, například nastavit rychlost pohybu vystřelenému projektilu.

- Dále jsou důležité metody pro řízení kolizí. Zde je důležitá metoda, která se vyvolává při startu kolize a metoda, která se vyvolá po ukončení kolize. Jako parametry těchto metod jsou reference na objekty, se kterými nastala kolize a bod v prostoru, kde kolize nastala.

2.2.2 SteamVR

SteamVR je součástí aplikace Steam, která slouží pro online nákup PC her a aplikací. SteamVR slouží jako ovladač ke komunikaci mezi výslednými aplikacemi a hardwarem pro virtuální realitu. Kromě SteamVR ovladače lze použít i ovladač od výrobce, ale SteamVR je použitelný téměř pro všechny headsety. Zároveň je již integrovaný v aplikaci Steam, kde si můžeme stahovat/kupovat hry a aplikace pro virtuální realitu. Ve výše zmíněném obchodě s assety pro Unity je volně dostupný balíček SteamVR, který umožňuje jednoduché vyvíjení aplikací pro virtuální realitu tím, že obsahuje propojení s hardwarem a zároveň nabízí v Unity implmentované funkcionality pro vývojáře, jako jsou modely ovladačů v herní scéně a systém pro detekci pohybu hlavy a ovladačů ve virtuální realitě. Vývoj VR aplikací v Unity za použití SteamVR je popsán v publikaci [9]. Zároveň tento balíček nabízí řadu základních implementovaných systémů, jako například teleportování pro pohyb v prostoru, nebo jednoduchou manipulaci s předměty. Pro účely této práce si ale budu všechny tyto funkcionality implementovat sám, abych se podrobněji naučil používání SteamVR knihovny. Takže z tohoto balíčku assetů použiji pouze

jeho jádro, tzn. systém pro detekci pohybu hlavy a ovladačů ve virtuální realitě.

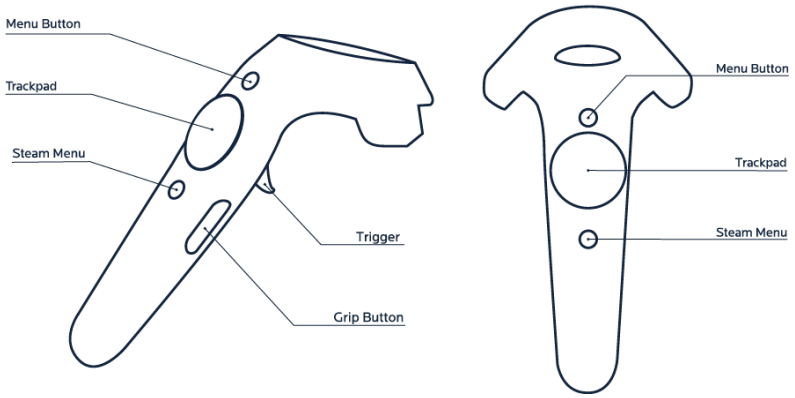
Při použití této knihovny je další výhoda ta, že výsledná aplikace půjde bez problému používat i na jiných sadách pro virtuální realitu, protože se pouze přemapují tlačítka u ovladačů. Toto přemapování může udělat vývojář aplikace, nebo i koncový uživatel, pokud by mu implicitní mapování akcí nevyhohvovalo. Při programování aplikace pro virtuální realitu pomocí balíčku SteamVR si vývojář nadefinuje akce, které aplikace používá. Ty mohou být typu *Boolean* pro tlačítka, popřípadě jednorozměrné i dvourozměrné vektory pro joysticky a páčky. Tudíž vývojář pracuje pouze s těmito akcemi a ve výsledné aplikaci pro konkrétní sadu ovladačů se přiřadí tyto akce na konkrétní tlačítka.

2.2.3 HTC Vive

Pro účely testování jsem zvolil sadu pro virtuální realitu HTC Vive. Jak je vidět na obrázku 2.3 sada se skládá z náhlavní soupravy, která obsahuje dva displeje, každý s rozlišením 1080 x 1200 pixelů a obnovovací frekvencí 90 Hz. Dále náhlavní souprava obsahuje jedno tlačítko. Součástí sady jsou i dva ovladače, které obsahují různá tlačítka pro interakci hráče s virtuálním prostředím. Pohyb hráče a ovladačů v prostoru snímají dvě stanice pomocí laseru. Tyto stanice emitují do prostoru paprsky, které ovladače a náhlavní souprava zachytávají pomocí mnoha světelných senzorů a tímto způsobem je určena jejich poloha a rotace v prostoru. Stanice tedy vůbec s PC a ostatními prvky nekomunikují a pouze emitují paprsky. Pokud se ale vyskytnou nějaké potíže se synchronizací mezi stanicemi, je nutné je připojit k PC pomocí synchronizačních USB kabelů. Náhlavní souprava je připojena k PC pomocí kabelu, který obsahuje HDMI a USB konektor a napájení náhlavní soupravy. Ovladače jsou připojeny bezdrátově a mají vlastní akumulátor, který je potřeba nabíjet. Existují i sady pro virtuální realitu, které nepoužívají stanice na určování polohy hráče v prostoru, ale používají kamery na náhlavní soupravě, které snímají pozici ovladačů. To má ale nevýhodu v tom, že pokud dá hráč ruce na místo, kam nevidí kamery náhlavní soupravy (např. za záda), tak není snímán pohyb ovladačů.

Vzhledem k tomu, že v návrhu, implementaci a testování se budu odkazovat na názvy tlačítek ovladačů, jsou zde popsány tlačítka, viditelné na obrázku 2.2.

- Menu Button – jedná se o obyčejné tlačítko, které se většinou používá pro vyvolání herní nabídky.
- Trackpad – vzhledem k tomu, že HTC Vive nemá oproti některým sadám pro virtuální realitu joystick, dá se simulovat tímto ovládacím prvkem. Celá plocha kruhového tlačítka snímá pohyb prstu a zároveň lze oblast stisknout. Můžeme tedy prstem po oblasti pohybovat, jako při pohybu joystickem.



Obrázek 2.2: Obrázek sady HTCVive [1].

- Steam Menu – toto tlačítko je primárně používáno na vyvolání menu aplikace Steam, kde je kromě jiného možnost měnit hlasitost, zavřít aplikaci, spustit jinou aplikaci, atd. Je tedy možné tyto úkony provádět, aniž bychom museli virtuální realitu opouštět.
- Trigger – zde se jedná o tlačítko spouště, které zaznamenává kromě toho, jestli je stisknuté i jak moc je stisknuté. Ve většině aplikací se toto tlačítko používá spolu s paprskem vycházejícím z ovladače pro ovládání uživatelského rozhraní a interakci s předměty.
- Grip Button – toto tlačítko je umístěné na obou bocích ovladače, ale jejich stisknutí vyvolává tu samou akci. Toto chování je z důvodu toho, že ovladače nemají pevně určeno, který je do levé ruky a který do pravé.

2.3 Analýza stávajících aplikací pro VR

V této sekci jsou popsány způsoby pohybu ve virtuální realitě, které jsou implementovány ve stávajících aplikacích a hrách. Jsou zde uvedeny příklady těchto aplikací se zajímavým způsobem pohybu, nebo manipulace s předměty a při návrhu budu z těchto aplikací vycházet. Nejrozšířenější druhy pohybu, které jsou použity téměř ve všech stávajících aplikacích jsou popsány v následujícím seznamu.

- Teleport – teleportování jako způsob pohybu je asi nejrozšířenější v aplikacích a hrách pro virtuální realitu, protože je pro uživatele nejjednodušeji použitelný. Dalším důvodem je to, že uživatelům při tomto způsobu pohybu není nevolno. Spočívá v tom, že uživatel drží stisknuté tlačítko na ovladači určené pro teleport a z virtuálního ovladače vidí rovný, nebo obloukový paprsek. Po puštění tohoto tlačítka je uživatel

Obrázek 2.3: Obrázek sady HTCVive [1].

teleportován na konec tohoto paprsku. Další alternativa tohoto způsobu pohybu je, že po herní ploše jsou umístěny určité body, na které se uživatel může teleportovat. Rozdíl je v tom, že jinam, než na tyto body se teleportovat nelze, takže uživatel nemá úplnou svobodu pohybu. Také se používá druh teleportu, který hráče teleportuje na místo, kam se dívá a ne na místo, kam míří ovladač.

- Joystick – další způsob pohybu je pohyb pomocí joysticku, jako známe u klasických herních ovladačů, uživatel určuje směr pohybu pomocí páčky, nebo prstem na dotykové ploše ovladače. Tento způsob pohybu není tak rozšířený jako teleport, protože někomu, může být při tomto způsobu pohybu nevolno.

2.3.1 Hra Hot Dogs, Horseshoes & Hand Grenades

Tato hra se je simulátor zbraní a střelby ve virtuální realitě, který obsahuje mnoho modelů reálných zbraní a mnoho scén, ve kterých si lze zbraně zkoušet. Z hlediska pohybu ve světě tato hra nabízí 3 hlavní styly pohybu, přičemž dva z nich mají i různé varianty, které každý z těchto způsobů pohybu trochu

poupravují. Celkem tedy hra nabízí výběr mezi sedmi způsoby pohybu. Tato hra má i zajímavé prvky z hlediska uživatelského rozhraní, například pokud jeden z ovladačů otočíme zadní stranou k sobě, tak se na něm zobrazí nabídka několika možností, lze vidět na obrázku 2.4 vlevo. V tomto malém menu na ruce lze upravovat způsoby pohybu, zároveň zde můžeme přejít do hlavní nabídky, nebo restartovat aktuální scénu. Další nabídky jsou řešeny pomocí tabletů, které ve virtuální realitě ovládáme ovladači a tím můžeme měnit různé nastavení. Tato hra byla také vyvíjena na platformě Unity s použitím balíčku SteamVR a je dostupná v obchodě Steam [10].

- Jako první způsob pohybu je zde implementován výše zmíněný teleport, který v této hře má 3 varianty. Varianty se liší rychlostí přesunu na požadované místo. První hráče teleportuje okamžitě, druhá varianta hráče rychle přesune na zvolené místo a třetí hráče přesune pomaleji. Tento způsob pohybu je nastaven jako implicitní, protože je pro většinu hráčů nejpříznivější. Lze vidět na obrázku 2.4 vpravo.

- Ve hře jsou dále různé varianty pohybu pomocí joysticku. Např. na levém ovladači ovládáme směr pohybu dopředu a dozadu a na druhém se můžeme otáčet. Nebo je otáčení i pohyb pouze na jednom ovladači. Tento způsob pohybu má oproti teleportu tu výhodu, že při pohybu můžeme dělat i další věci s ovladači, protože nemusíme stále mířit na místo, kam se chceme přesunout, ale má i nevýhody. Například lidem, kteří s virtuální realitou začínají se při tomto způsobu pohybu může dělat nevolno. Další nevýhoda je ta, že pokud se pohybujeme, tak máme omezené další možnosti tím, že joystick, nebo trackpad je vyhrazený pro pohyb a nelze ho použít k ničemu jinému.

- Třetí možnost pohybu je Arm Swing, neboli pokud se chceme pohybovat nějakým směrem, tak musíme máchat rukama, jako při reálném běhu a hra nás v tomto směru posouvá. Rychlost pohybu závisí na rychlosti máchání. Tento způsob pohybu eliminuje nevýhodu pohybu pomocí joysticku, protože při pohybu můžeme používat všechny tlačítka na ovladačích, kromě jednoho, který je vyhrazený k tomu, že pokud ho držíme stisknuté, tak se mácháním pohybujeme. Tento způsob pohybu mi přijde zajimavý a není úplně obvyklý v aplikacích pro virtuální realitu. Z tohoto důvodu se ho pokusím implementovat a z výsledků testování zjistím, jak bude mezi ostatními uživateli oblíbený.

V této hře je také pro mě zajímavý způsob ovládání zbraní a předmětů. Zbraně bereme do ruky pomocí tlačítka spouště, kde toto tlačítko stačí pouze zmáčknout a předmět držíme v ruce, dokud nezmáčkneme grip tlačítko (na straně ovladače). U většiny ostatních her je braní předmětů do ruky realizováno tak, že pokud držíme tlačítko spouště, tak předmět držíme a pokud toto tlačítko pustíme, tak tím pustíme i držený předmět. Toto chování má

Obrázek 2.4: Obrázek menu a pohybu pomocí teleportu.

tu výhodu, že nemusíme pořát držet tlačítko pro braní předmětu do ruky a může být využito k jiným účelům. Další zajímavá mechanika v této hře je zvedání předmětu na dálku. Pokud máme předmět, který je od nás vzdálený a nechceme k němu chodit a například ho zvedat ze země, můžeme na něj ovladačem zamířit, stisknout tlačítko na trackpadu a tím se nám předmět objeví v ruce. Toto se může hodit například v případě, že máme vyhrazený malý prostor v místnosti a nemůžeme se po ní tolik pohybovat. Tento způsob zvedání předmětů na dálku jsem nenašel v jiných analyzovaných aplikacích.

2.3.2 Neos VR

V této aplikaci má každý uživatel možnost vytvářet si vlastní světy a procházet ty, které vytvořili jiní uživatelé. Zároveň se můžeme setkávat s ostatními hráči v těchto světech, aplikace tedy podporuje interakci více hráčů.

Světy se tvoří pomocí umísťování různých předmětů a aplikováním textur, nebo materiálů na tyto předměty. Tyto objekty lze různě zvětšovat, zmenšovat, otáčet, atd. Manipulace s předměty se provádí pomocí paprsků z ovladačů, aby bylo možné ovládat předměty, které jsou od uživatele vzdálené, nebo je možné převážně malé předměty vzít do ruky. Zároveň je možné vzít do ruky určité nástroje, například nástroj pro kreslení, nebo fotoaparát. Tyto předměty pak fungují, jako reálné předměty.

Způsoby pohybu v této aplikaci jsou tři. Všechny tři způsoby umožňují pohyb téměř kamkoliv, není zde tedy žádná restrikce pohybu. Lze vidět na obrázku 2.5

- První způsob pohybu je teleport. Ten je zde obloukový a funguje na principu vystřelení objektu z ovladače a vykreslování jeho trajektorie. Pomocí puštění tlačítka je možné se přesunout na místo kolize vystřeleného objektu s jiným objektem.

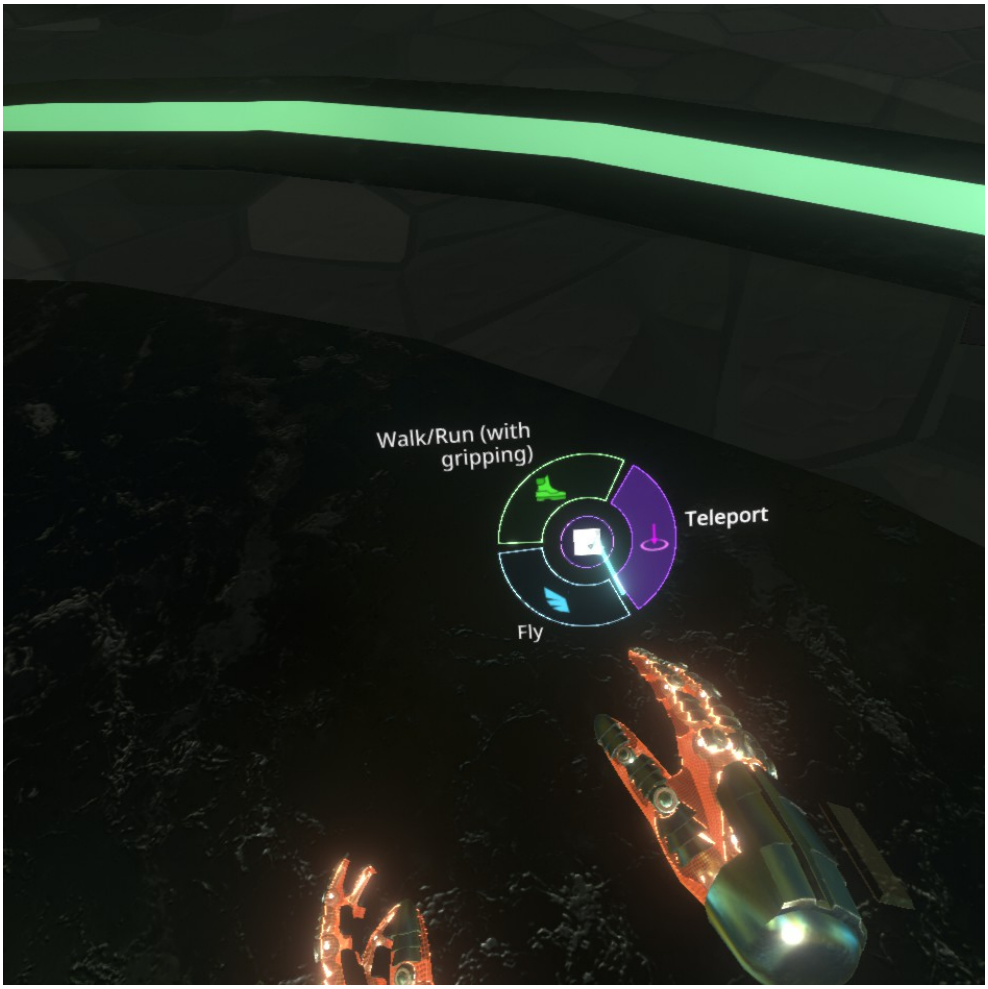
- Dalším způsobem pohybu je chození pomocí joysticku. U sady HTC Vive se používá pro simulaci joysticku trackpad a u tohoto způsobu pohybu musíme trackpad držet stisknutý ve směru dopředu/dozadu, abychom se mohli pohybovat. Pokud stiskneme na trackpadu směry vlevo vpravo, otočíme se na místě o 90 stupňů. Směr pohybu je závislý na tom, kam ovladačem míříme. Zde je rozdíl oproti jiným aplikacím v tom, že u nich je směr pohybu závislý na rotaci kamery.

- Poslední způsob pohybu je opět pohyb pomocí joysticku a princip je stejný, jako při chození popsaném výše. Rozdíl je v tom, že zde se nepohybujeme pouze po zemi, ale je umožněno i létání. Tím, že aplikace nemá restrikce v tom, kde je pohyb umožněn, je možné prolétnout skrze objekty a stěny. U tohoto způsobu pohybu bývá, dle mých zkušeností, mnoha uživatelům nevolno.

U této aplikace je zajímavé, že lze spouštět jak ve virtuální realitě, tak v klasické desktop verzi. Můžou se tedy potkávat i uživatelé, kteří VR hreadset nevlastní. Tato aplikace je také dostupná v obchodě Steam [11]

2.3.3 SteamVR Home

Tato aplikace je součástí SteamVR aplikace a je to výchozí prostředí, kam se dostanete, po zapnutí SteamVR aplikace. Jedná se o místnost, ze které je možné spouštět aplikace pro virtuální realitu stažené přes Steam obchod. Kromě toho je zde možné interagovat a manipulovat s předměty. Do této místnosti lze i přizvat ostatní lidi, jako je možné v aplikaci Neos VR. Stisknutím menu tlačítka se vyvolá obrazovka, přes kterou je možné měnit vzhled místnosti, dále je možné vyvolávat nové instance různých objektů. S těmito objekty je možné manipulovat a interagovat různými způsoby. Můžeme tyto objekty brát do rukou a tím předměty rozmisťovat po místnosti. Pokud držíme nějaký objekt v rukou, tak jejich oddálením můžeme objekt zvětšovat a přiblížením zmenšovat. Zajímavý způsob manipulace je, že pokud držíme grip tlačítko na ovladači, můžeme si vybrat nástroj pro manipulaci, který je jako fyzický objekt, který držíme v ruce. Na obrázku 2.6 jsou zobrazeny tyto nástroje. Například, pokud chceme změnit barvu objektu, vezmeme do ruky barvící nástroj, na kterém si zvolíme barvu a namířením a stisknutím spouště můžeme



Obrázek 2.5: Ukázka výběru způsobu pohybu v aplikaci Neos.

předmět obarvit. Dále je možnost si do ruky vzít štětec a kreslit ve 3D prostoru libovolný objekt. S tímto objektem je poté možné manipulovat úplně stejně jako s ostatními předměty.

Po místnosti je možné se pohybovat pomocí teleportu. Teleport je zde umožněn ve vyhrazené oblasti a nebo na předem určená místa. Kromě teleportu není možné si vybrat jiný způsob pohybu. Zajímavý prvek aplikace je to, že pokud držíme tlačítko pro teleport, tak se na zemi zeleně rozsvítí oblast, na kterou se můžeme teleportovat, plus body, které nejsou součástí této oblasti, ale lze se na ně teleportovat.

Obrázek 2.6: Obrázek nástrojů pro manipulaci s předměty ve VR Home.

2.4 Rozdíl ve vývoji 3D aplikací pro VR oproti standardním aplikacím

Pokud mluvíme o pohybu ve virtuální realitě, máme na mysli pohyb uživatele ve 3D prostoru nějaké aplikace. Zde je velký rozdíl oproti 3D aplikacím, které ve virtuální realitě nejsou. Jedním z těchto rozdílů je to, že uživatel 3D svět vnímá mnohem reálněji. Vzhledem k tomu, že aktuální sady pro virtuální realitu neumí pracovat na neomezené ploše, je nutné nějakým způsobem vyřešit pohyb uživatele ve virtuálním světě. Ve standardních (desktopových) aplikacích je pohyb uživatele v prostoru realizován nejčastěji pomocí tlačítek na klávesnici, v případě her standardně tlačítky WASD, nebo šipkami. Popřípadě je možné využít joystick, máme-li k dispozici nějaký herní ovladač. Pokud ale budeme chtít stejný způsob využít ve virtuální realitě, narazíme na několik problémů. Jedním z nich může být to, že při pohybu pomocí joysticku se uživatel v reálném světě nehýbe, ale ve virtuálním ano a to může u někoho způsobovat nevolnost. Vzhledem k tomu není možné tento způsob pohybu využívat pokaždé. Z tohoto důvodu je jedním z nejrozšířenějších způsobů pohybu pohyb pomocí teleportu. Ten umožňuje uživateli pohyb po 3D scéně, ale nevyvolává pocit pohybu, protože se uživatel přemisťuje okamžitě.

Dalším problémem ve vývoji aplikací pro virtuální realitu je ten, že musíme pracovat pouze s ovládacími tlačítky, které jsou k dispozici na ovladačích.

Tyto ovladače mají standardně menší počet ovládacích prvků, než například klávesnice, nebo herní ovladače. Pokud tedy potřebujeme v aplikaci pro virtuální realitu používat komplexnější ovládací prvky, musíme je nějak simulovat. Dobrým příkladem této simulace je to, že pokud ve VR po uživateli požadujeme zadání nějakého textu, objeví se před ním klávesnice, se kterou interaguje namířením ovladače na požadovaný znak a stisknutím nějakého tlačítka. Použití HW klávesnice pro ovládání aplikace ve virtuální realitě ovšem nepřipadá v úvahu, protože pokud má uživatel náhlavní soupravu na hlavě, tak už na klávesnici nevidí. Zároveň, pokud by používal klávesnici, ztratil by volnost pohybu po místnosti, protože by musel zůstat stát u klávesnice.

Pokud vyvíjíme 3D aplikace, nebo hry na klasické desktopové prostředí, tak máme poměrně velkou volnost v tom, kam umístíme kameru, kterou uživatel vidí svět. Zde je na výběr například pohled z první osoby, nebo pohled ze třetí osoby, pohled shora, atd. U aplikací pro virtuální realitu bereme v úvahu pouze pohled z první osoby.

2.5 Návrh způsobů pohybu ve virtuální realitě

Při návrhu různých způsobů pohybu budu vycházet z analýzy ostatních aplikací a z osobních zkušeností s aplikacemi ve virtuální realitě. Důležité je navrhnout větší množství způsobů pohybu ve virtuální realitě, protože každému uživateli může vyhovovat jiný způsob pohybu a ve výsledném balíčku pro ostatní vývojáře by měli mít velký výběr. Důležité bude vyzkoušet při testování, který způsob pohybu je vhodný pro přesun na velkou vzdálenost a který na menší vzdálenost, například uvnitř budovy a tyto výsledky poté napsat do dokumentace. Níže jsou uvedené návrhy na způsoby pohybu ve virtuální realitě. Ve výsledném balíčku by měly být jednotlivé způsoby pohybu používány v Unity jako komponenty a zapínáním/vypínáním jednotlivých komponent určuje programátor, jaký způsob se právě použije. Zároveň by měl mít vývojář možnost upravovat různé parametry těchto způsobů pohybu, například ryhlost pohybu pomocí joysticku. Tyto komponenty půjdou přepínat i za běhu aplikace, takže si programátor může implementovat v rámci uživatelského rozhraní způsob přepínání mezi způsoby pohybu a zároveň bude moci měnit nastavení atributů. Atributy, které budou typu SteamVR akce, použité převážně na tlačítka, bude moci programátor upravit dle svých preferencí. Zároveň i koncoví uživatelé budou schopni přemapovat tyto tlačítka (pokud např. používají jinou sadu pro virtuální realitu) díky kompatibilitě SteamVR. Atributy komponent pro pohyb budou následující:

- Tlačítko, kterým aktivujeme pohyb pomocí teleportu a máchání rukama. Atribut bude typu SteamVR akce.
- SteamVR akce typu dvourozměrného vektoru pro použití při pohybu joystickem.

- U pohybu pomocí joysticku a máchání rukama bude možné nastavit rychlost pohybu.
- Při pohybu teleportem bude možné upravit maximální vzdálenost, na kterou se uživatel může teleportovat.
- Dále u pohybu pomocí teleportu (obloukového i přímého) bude nutné dát referenci na paprsek, který se bude při pohybu zobrazovat, aby uživatel viděl, kam s ovladačem míří.
- U pohybu pomocí obou teleportů musí programátor dát referenci na objekt, který se zobrazí na místě, kam uživatel míří ovladačem a pokud je pozice validní, přesune se právě na tento bod.
- Posledním atributem u pohybu pomocí teleportu je doba trvání zatmavení obrazovky. Toto zatmavení slouží k lepší plynulosti přesunu a uživateli nenarušuje pocit z virtuálního světa.

Všechny implementované způsoby pohybu by měly přijímat zprávy pro aktivaci a deaktivaci pohybu. Toto může programátor použít, pokud nechce, aby se uživatel v určité chvíli mohl pohybovat.

2.5.1 Teleport

Ve výsledné aplikaci pro testování budou dvě varianty na pohyb pomocí teleportu. První varianta je nejpoužívanější v ostatních a aplikacích a spočívá v tom, že po stisknutí a držení tlačítka pro teleport se objeví na konci ovladače obloukový paprsek, na jehož konec se uživatel po uvolnění tlačítka teleportuje. Díky obloukovému paprsku se uživatel může pohodlně dostat i na místa, která jsou nějakým způsobem vyvýšená. Druhá varianta teleportu bude také na principu paprsku, ale tento paprsek nebude obloukový, ale rovný. Uživatel tedy nebude mít možnost dostat se pomocí tohoto paprsku na vyvýšená místa, ale bude vhodnější pro pohyb po rovném povrchu. Oba tyto způsoby teleportují hráče během zlomku sekundy, při které se zatmaví obrazovky a poté zase odtmaví, aby byl přechod plynulejší. Zde se chci při testování zaměřit na optimální čas tohoto přechodu, protože pokud bude hodnota moc velká, tak to uživateli může přijít obtěžující a naopak, pokud bude hodnota malá, tak už nebude přesun tak plynulý. Dalším parametrem při implementaci pohybu pomocí teleportu je maximální možná vzdálenost, na kterou se uživatel může teleportovat. Pokud je vzdálenost moc malá, bude pohyb po větším prostoru pomalý a bude vyžadovat hodně krátkých přesunů. Na druhou stranu, pokud bude maximální vzdálenost moc velká, tak bude přesun na větší vzdálenosti rychlejší, ale při míření paprsku na větší vzdálenost může docházet k nepřesnostem a nebo, pokud uživatel omylem zmáčkne tlačítko pro teleport, tak se může dostat nechtěně na vzdálené místo.

Teleport má oproti ostatním způsobům pohybu tu výhodu, že je jednoduchý na používání a nezpůsobuje nevolnost uživateli, protože se ve virtuálním světě nehýbe, ale pouze se přemisťuje. To je nejspíš důvod, proč je toto nejpoužívanější způsob pohybu. Jeho nevýhoda je, že uživatel při pohybu musí paprsek nějakým způsobem směrovat, takže je pro něj těžší dělat během pohybu další věci, například v nějakých akčnějších hrách.

Ve výsledném balíčku tedy budou implementovány dva druhy teleportu, jeden přímý a druhý obloukový. Přímý teleport pouze vykreslí rovný paprsek, na jehož konec se uživatel po puštění tlačítka pro teleport přesune. U obloukového teleportu je důležitý výpočet oblouku, v článku [12] jsou popsány tři druhy výpočtu křivky pro obloukový teleport. První způsob používá 3 body v prostoru (startovní pozice paprsku, koncová pozice paprsku a ovládací bod), které spojí a vytvoří oblouk pomocí výpočtu Beziérovy křivky. Dalším způsobem je takzvané vystřelení objektu z ovladače, který má gravitaci relativní k rotaci ovladače a vykresluje se cesta tohoto objektu, čímž docílíme obloukového tvaru. Třetí způsob pracuje s bodem, který je umístěn nad uživatelem a vystřeluje paprsek pod nějakým úhlem ve směru dolů a směru, kterým uživatel míří ovladačem. Výsledný paprsek z tohoto bodu se vyhladí tak, aby vypadal jako oblouk z ovladače, na což lze například použít Beziérovu křivku.

Z těchto způsobů implementace pohybu pomocí teleportu jsem se rozhodl použít první způsob, protože se mi zdá z hlediska implementace nejjednodušší a oproti druhému způsobu i výpočetně méně náročnější. Vzhledem k tomu, že druhý způsob modelování oblouku pracuje s fyzickým enginem, mohlo by opakované vystřelování objektu snižovat plynulost běhu aplikace.

2.5.2 Joystick

Další způsob pohybu bude pomocí joysticku, kterým ale nedisponují všechny ovladače na trhu. Např. HTC Vive, na kterém budu testovat. Na těchto ovladačích se použije tzv. trackpad, což je dotyková plocha se směrovými tlačítky, kde se dá zmíněný joystick simulovat. Díky platformě SteamVR bude aplikace fungovat i na ovladačích, které joystick mají. Při testování tohoto způsobu pohybu bych rád vyzkoušel dynamické zmenšování zorného pole jakožto prevenci proti nevolnosti z pohybu ve virtuální realitě. Některým uživatelům může být při pohybu např. pomocí joysticku nevolno, protože ve virtuální realitě se hýbou, ale fyzicky ne. Columbia University dělala ohledně zmenšování zorného pole (field of view, neboli FOV) výzkum [13]. Podle jejich výsledků se uživatelům spíše nedělá nevolno, pokud se při pohybu sníží jejich zorné pole, nebo se zatmaví okraje obrazovek. Tuto techniku bych chtěl otestovat v praxi a pokud by se ukázala jako účinná, tak by byla zahrnuta ve výsledném balíčku. Při testování bych se zaměřil na optimální hodnotu zatmavení, nebo snížení FOV.

Pohyb pomocí joysticku má oproti teleportu tu výhodu, že uživatel může během pohybu dělat další věci a pouze hýbe joystickem. Jeho nevýhoda je, že

někomu se může dělat nevolno při pohybu tímto způsobem. Další nevýhoda je, že ne všechny sady pro virtuální realitu mají na ovladačích joystick, proto tento způsob pohybu nemusí být i přes kompatibilitu SteamVR použitelný se všemi ovladači bez nutnosti zásahu do editoru tlačítek. Zároveň při pohybu joystickem používáme celý joystick/trackpad, takže ho není možné používat k jiným účelům.

Pro implementaci tedy bude použit joystick/trackpad k určování směru pohybu a rychlosti. Zároveň směr pohybu bude relativní k aktuální rotaci uživatele, aby se při pohybu joysticku dopředu pohyboval ve směru, kam se právě dívá. Druhou možností je, že se uživatel nebude hýbat směrem relativním k rotaci kamery, ale směr pohybu bude fixní na rotaci objektu *CameraRig* (objekt obsahující kameru, ovladače, atd.). Pokud bude chtít uživatel změnit rotaci pohybu, použil by tlačítka na straně trackpadu pro otočení tohoto objektu o 90 stupňů. Osobně se mi zdá první možnost jako uživatelsky přívětivější, proto implementuji tuto možnost, a pokud by se při testování ukázalo, že druhá možnost by byla lepší, tak by se tento způsob pohybu eventuelně implementoval také, nebo by ve výsledném balíčku byla možnost volby mezi těmito dvěma způsoby. Rychlost pohybu bude záviset na vzdálenosti prstu od nulového bodu joysticku/trackpadu (bod uprostřed).

2.5.3 Arm swing

Jako poslední způsob pohybu budu implementovat tzv. arm swing, neboli způsob pohybu založený na tom, že uživatel máchá rukama jako při běhu a tím se pohybuje ve virtuálním světě. Čím větší intenzita máchání, tím rychleji se uživatel ve virtuálním světě pohybuje. Tento způsob jsem zvolil, protože mi přijde zajimavý a neobjevuje se v mnoha ostatních aplikacích pro virtuální realitu. Jeho výhoda je, že nechává možnost používat joystick, nebo trackpad k jiným účelům, například pro ovládání uživatelského rozhraní. Zde by se také dala použít technika zmiňovaná výše se snížením zorného pole uživatele, protože se opět pohybujeme ve virtuální realitě, ale v reálném světě ne. U tohoto způsobu pohybu bude důležité vyladit různé odchylky a tolerance při máchání rukama a zajistit co nejlepší plynulost pohybu uživatele. Zajímalo by mě u tohoto způsobu pohybu, jak ho budou vnímat uživatelé, kteří mají malé, nebo žádné zkušenosti s VR, vzhledem k neobvyklosti tohoto způsobu pohybu.

Tento způsob pohybu bude fungovat na principu rozdílu aktuálních pozic individuálních ovladačů vzhledem k jejich předešlé pozici při předchozí kalkulaci. Na základě těchto rozdílů bude vypočítána intenzita máchání a uživatel se bude ve virtuálním světě pohybovat. Chození různými směry se bude určovat podle toho, jakým směrem uživatel máchá. Rychlost pohybu bude přímo úměrná intenzitě máchání, pro rychlý pohyb tedy bude nutné máchat ryhleji.

2.6 Návrh manipulace s předměty ve virtuální realitě

Při návrhu manipulace s předměty se budu primárně zaměřovat na manipulaci s velkými objekty, například budovami, tudíž uživatel nebude mít možnost tyto objekty brát přímo do ruky (jako u malých předmětů), ale manipulace bude probíhat na dálku pomocí paprsků. Ve výsledném balíčku budou čtyři druhy manipulace s předměty.

- Přesun předmětu v globálních osách.
- Otočení předmětu po vertikální ose.
- Změna velikosti předmětu.
- Měření vzdálenosti mezi dvěma body.

Pro manipulaci s objektem si uživatel zvolí požadovaný nástroj a namíří ovladačem na objekt, kterým chce manipulovat. Poté podrží tlačítko pro manipulaci a tím se zahájí manipulace pomocí požadovaného nástroje. Manipulovat bude možné pouze s předem určenými objekty. Tyto objekty budou muset být označeny tagem, který tyto předměty bude odlišovat od ostatních, se kterými manipulace není možná. Budou implementovány dva způsoby manipulace.

- Manipulace pomocí joysticku.
- Manipulace pomocí pohybu rukou.

V obou případech se manipulace ukončí uvolněním tlačítka pro manipulaci, nebo změnou nástroje. Měření vzdálenosti, mezi dvěma body bude pro oba způsoby manipulace probíhat totožně. To znamená, že z obou ovladačů povede paprsek a bude se měřit vzdálenost, mezi kolizními body těchto paprsků.

Každý z těchto způsobů manipulace s objekty bude implementován jako samostatná komponenta, která se bude umisťovat na ovladač, pomocí kterého budeme chtít manipulaci provádět. Tyto komponenty půjdou zapínat/vypínat během běhu aplikace. Tímto způsobem bude možné mezi těmito způsoby přepínat. Atributy, které budou mít oba způsoby manipulace a vývojář je bude moci upravit jsou následující:

- Tlačítko, kterým se při stisknutí a držení bude manipulace aktivovat. Toto tlačítko si programátor nastaví podle svých preferencí a zároveň se bude jednat o SteamVR akci typu boolean.
- Dalším atributem bude opět SteamVR akce typu dvourozměrného vektoru. Tento vektor se použije při manipulaci s objekty pomocí joysticku k samotné manipulace s objektem. Zároveň při manipulaci za použití pohybu rukou se tato akce použije pro přiblížení a oddálení objektu při přesunu.

- Pro zobrazování paprsků z obou ovladačů bude potřeba mít jako další atribut referenci na druhý ovladač, aby se mohly paprsky zapínat a vypínat. Zároveň při manipulaci s objekty pomocí pohybu rukou, bude probíhat výpočet vzdáleností mezi ovladači, např. pro zvětšování objektů.

- Dále je potřeba, aby programátor uvedl referenci na objekt paprsku, který se bude zapínat při namíření na objekt, se kterým je možné manipulovat.

- Posledními atributy budou akce, které si programátor může libovolně zvolit a tyto akce se provedou, pokud uživatel zahájí manipulaci s objektem a po ukončení této manipulace. Toto půjde použít například pro zamezení pohybu uživatele při manipulaci, jelikož by mohlo dojít ke kolizi akcí joysticku, pokud by byl aktivní způsob pohybu pomocí joysticku a zároveň manipulace s objektem také pomocí joysticku a obě tyto komponenty by byly na stejném ovladači.

Kromě dvou komponent pro každý druh manipulace bude implementovaná ještě jedna pomocná třída, která bude zajišťovat referenci na objekt, se kterým chce uživatel manipulovat. Dále bude uchovávat právě zvolený druh manipulace s objektem. Komponenty pro manipulaci budou pouze počítat velikost změny transformace ale samotnou transformaci bude provádět tato pomocná třída pomocí definovaného rozhraní. Většina metod tohoto bude statická, aby nemusely mít objekty referenci na tuto třídu. Dále bude mít tato pomocná třída referenci na objekt textového typu, aby bylo možné zobrazovat naměřenou vzdálenost mezi body. Toto textové pole si programátor zvolí a referencuje sám, aby nebyl nucen používat přednastavené textové pole, které by třeba jeho aplikaci nevyhohvovalo. Přesto bude základní textové pole implementováno.

Aby bylo možné při běhu přepínat mezi různými druhy manipulace s objekty, musí se ještě implementovat jednoduché uživatelské rozhraní, kterým si uživatel bude moci přepínat mezi druhy manipulace. Při vytváření testovací aplikace by k tomu šlo použít uživatelské rozhraní, které bude implementovat Bc. Petr Polívka. Zpočátku ale bude nutné navrhnout a implementovat nějaké vlastní řešení.

Tento výběr nástrojů by měl být jednoduchý a přehledný. Vzhledem k tomu, že se pro pohyb a manipulaci s objekty nepoužívá joystick na levé ruce, bude použit na výběr nástrojů. Použití bude jednoduché, každému ze 4 směrů joysticku bude přiřazen jeden nástroj pro manipulaci a pohybem joysticku budeme moci mezi těmito nástroji přepínat.

2.6.1 Manipulace pomocí joysticku

Při tomto druhu manipulace se pro určení os k otáčení, rotování a zvětšování objektu použije joystick, nebo v případě HTC Vive pozice prstu na trackpadu. Díky tomu bude uživateli k manipulaci stačit pouze jedna ruka. Tento způsob manipulace by měl být oproti manipulaci pomocí pohybu rukou přesnější.

- Pohyb objektu bude probíhat pomocí držení tlačítka pro manipulaci a současným pohybem joysticku ve směru, kterým chceme objekt přesouvat. Rychlost přesunu bude závislá na vzdálenosti joysticku od nulového bodu (uprostřed). Pohyb objektu bude probíhat po globálních osách, aby bylo možné s objekty pohybovat přesně a nehrála roli rotace hráče, nebo rotace objektu.
- Při změně velikosti objektu bude použita pouze jedna osa joysticku a sice vertikální. Opět jako při pohybu bude záležet na vzdálenosti od nulového bodu. Podle této vzdálenosti bude vyhodnocena rychlost zvětšování, nebo zmenšování.
- Dalším způsobem manipulace s objekty je rotace. Zde se taktéž využije pouze jedna osa joysticku a to horizontální osa. Stejně, jako při změně velikosti se rychlost rotace bude odvíjet od vzdálenosti od nulového bodu. Směr rotace bude udávat to, jakým směrem pohybujeme joystickem. Rotace bude probíhat po ose Y.
- Měření vzdálenosti mezi body joystick nevyužívá, protože při měření nijak vlastnosti objektů neměníme, pouze měříme vzdálenost, mezi dvěma body. Způsob měření bude pro oba styly manipulace totožný.

Výběr os joysticku pro rotaci a zvětšování jsem zvolil tímto způsobem, protože při zvětšování objektu je přirozenější používat vertikální osu a při rotaci rotujeme objektem po ose Y, tudíž je pro joystick zvolena osa horizontální. Budeme-li tedy chtít objektem otočit směrem doprava, přesuneme joystickem do pravé části. Pokud by byly osy zvoleny jinak, mohlo by to být pro uživatele matoucí.

2.6.2 Manipulace pomocí pohybu rukou

U tohoto způsobu se pro manipulaci používá pohyb rukou. Bude se zde dynamicky pracovat se vzdáleností mezi ovladači, na základě které se bude počítat intenzita transformace objektu např. rychlost zvětšování objektu. Kromě pohybu s objektem zde budeme obdobně jako při manipulaci pomocí joysticku využívat statických metod pomocné třídy. Pohyb objektem zde nebude probíhat pomocí určení směru pohybu, což je hodnota, se kterou bude pomocná třída pracovat. Při pohybu objektu tedy budeme přímo nastavovat pozici tohoto objektu v závislosti na pozici ovladače.

Manipulace s objekty bude probíhat následujícím způsobem.

- Pro pohyb předmětu se při držení tlačítka pro manipulaci bude předmět pohybovat stejným způsobem jako ovladač, takže objekt přesouváme pomocí pohybu ovladače. Objekt se bude chovat, jako bychom ho drželi v ruce s tím rozdílem, že musíme brát v potaz vzdálenost, kterou má objekt od ovladače.

- Při zvětšování bude uživatel oddalovat ovladače od sebe a naopak pro zmenšování bude ovladače přibližovat. Při změně velikosti pomocí pohybu ovladači nebude možné dosáhnout požadované velikosti jedním pohybem a bude nutné těchto pohybů provést více.

- Rotace předmětu bude probíhat pomocí posunutí ovladače vlevo pro rotaci po směru hodinových ručiček a při posunutí ovladače vpravo bude objekt rotovat proti směru hodinových ručiček. Ve výsledné implementaci by měl tento způsob rotace navozovat pocit, že držíme objekt v jednom místě a pohybem ruky s ním otáčíme. Vzhledem k tomu, že se výsledný směr rotace bude počítat podle vzdálenosti ovladačů, bude možné objektem rotovat oběma rukama.

Tento způsob manipulace jsem navrhl na základě zkušeností z ostatních aplikací, kde se například tímto způsobem manipuluje s předměty, které držíme v ruce. Například v analyzované aplikaci SteamVR Home (2.3.3) zvětšujeme, nebo zmenšujeme objekt tak, že ho uchopíme do rukou a pohybem ovladačů od sebe / k sobě měníme velikost objektu. Tento způsob jsem tedy upravil tak, aby šel aplikovat i na větší objekty, které nelze jednoduše uchopit do ruky.

Kapitola 3

Realizace

V této kapitole se budu zabývat implementací návrhů na způsoby pohybu a manipulace s předměty ve virtuální realitě. Implementace bude probíhat pomocí programu Unity verze 2019.3.11f1 a pro psaní skriptů bude použit program Microsoft Visual Studio Community verze 2019. Vzhledem k tomu, že výsledkem této práce je balíček funkcionalit pro ostatní vývojáře, bude implemetace probíhat v testovací scéně v Unity, která bude součástí aplikace určené pro testování implementovaných funkcionalit.

Pro implementaci testovací aplikace jsme se s Bc. Petrem Polívkou museli předem domluvit na tom, jaká tlačítka a ovládací prvky na ovladačích budeme používat, aby následné spojení implementací do testovací aplikace proběhlo bez problémů. Rozvržení tlačítek je následující (popis toho, kde se tlačítka nachází na ovladačích je popsán v kapitole analýza na obrázku 2.2).

- Menu Button – toto tlačítko slouží na obou ovladačích pro volbu nástroje pro manipulaci s předměty, kterou implementoval Bc. Petr Polívka.

- Trackpad – Na levé ruce slouží tento ovládací prvek pro výběr nástroje na manipulaci s budovami. Bere se zde dvourozměrný vektor pozice prstu na ploše tlačítka a zároveň stisknutím tlačítka se výběr potvrzuje. Na pravé ruce slouží tento ovládací prvek k pohybu uživatele, nebo k pohybu s objektem. Při pohybu uživatele se používá akce na stisknutí (u teleportů a pohybu pomocí máchání rukama) a nebo pozice prstu na ploše pro pohyb pomocí joysticku/trackpadu.

- Trigger – toto tlačítko na obou ovladačích slouží pro výběr možností z UI (implementovaném Bc. Petrem Polívkou). Zároveň toto tlačítko na levé ruce slouží pro vyvolání menu, pomocí kterého se dají přepínat způsoby pohybu.

- Grip Button – tlačítko je použito pro zahájení veškeré manipulace s předměty v aplikaci.

Jak je ze seznamu zřejmé, využili jsme v testovací aplikaci všechny ovládací prvky ovladačů HTC Vive.

3.1 Implementace způsobů pohybu

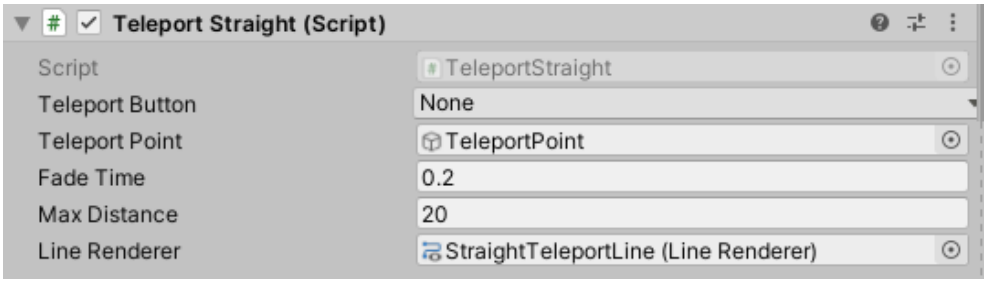
Každý způsob pohybu ve výsledném balíčku je samostatný skript, nebo skupina skriptů. Pokud tedy chceme používat jeden konkrétní způsob pohybu, aktivujeme jej jako komponentu, která je součástí herního objektu ovladače. Podle toho, kterým ovladačem chceme pohyb ovládat, musíme komponentu na patřičný ovladač umístit. Různé způsoby pohybu lze aktivovat i při běhu aplikace a to tak, že komponenty pro pohyb vypínáme/zapínáme pomocí skriptů.

3.1.1 Pohyb pomocí přímého teleportu

Jako první způsob pohybu byl implementován přímý teleport. Jedná se o samostatný skript a třídu *TeleportStraight*, kde lze nastavovat různé atributy, jak je vidět na ukázce 3.1.

- *teleportButton* je typu *SteamVR_Action_Boolean* a jedná se o přiřazení tlačítka, které bude ve výsledné aplikaci použito k ovládání teleportu. Programátor tedy může zvolit libovolné tlačítko pro teleport.
- Dalším atributem typu *GameObject* je *teleportPoint*, což je reference na objekt, který slouží jako vizuální reprezentace bodu, na jehož místo se teleportujeme. Opět je možné objekt volit libovolně. Jediné omezení je to, že daný objekt nesmí mít komponentu *Collider*, protože potom by se při jeho zobrazení paprsek mířící z ovladače zastavil o tento bod a ne o plochu, která je k pohybu určená.
- *fadeTime* je atribut typu *float* a udává čas v sekundách, po který se při teleportu zatmaví obrazovka. Tento čas je také použit pro odtmavení, takže pokud ho nastavíme například na 0,25, musíme počítat s tím, že celkový čas teleportu bude 0,5 s.
- Následuje atribut *maxDistance*, což je nastavení maximální možné vzdálenosti pro teleport. Pokud tedy uživatel míří na místo, které je vzdálenější, než tato hodnota, pozice bude vyhodnocena jako nevalidní.
- Posledním atributem je reference na komponentu typu *LineRenderer*, která slouží k vykreslování paprsku, který se zobrazí po stisknutí tlačítka pro teleport, pokud uživatel míří na místo, na které je možné se teleportovat.

Tato třída dále obsahuje několik privátních atributů, které převážně řeší stav při teleportování, například jestli uživatel míří ovladačem na validní pozici, atd. Dalším důležitým privátním atributem typu *SteamVR_Behaviour_Pose*



Obrázek 3.1: Ukázka atributů přímého teleportu.

je *pose*. Tento atribut je nastavený při inicializaci objektu v metodě *Awake()* a slouží pro určení, na kterém ovladači je teleport aktivní (pravá/levá ruka). Používá se při dotazování, jestli bylo stisknuto tlačítko pro teleport na této ruce.

Třída dále reaguje na broadcast dvou zpráv, které slouží pro zamezení pohybu a pro opětovné povolení pohybu. Toto se může použít k tomu, když programátor nechce, aby se uživatel v nějakou chvíli mohl pohybovat. V Unity se tyto zprávy posílají pomocí příkazu:

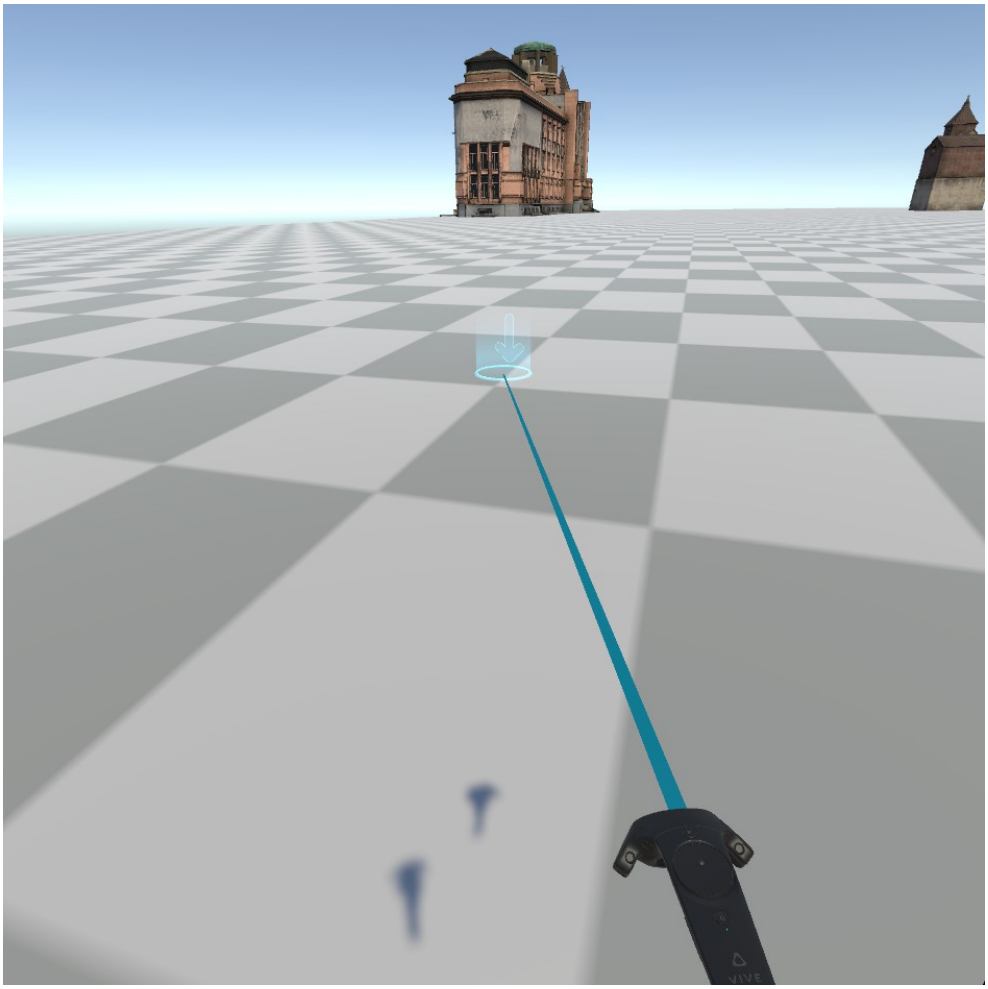
```
hand.SendMessage("stopMovement");,
```

kde *hand* je reference na ovladač, na kterém pohyb probíhá. Tato ukázková zpráva zamezí uživateli možnost se pohybovat. Pro opětovné povolení pohybu se použije obdobná konstrukce se zprávou *startMovement*.

V metodě *Update()* této třídy se řeší jednak odchytávání akce stisknutého tlačítka pro teleport, dále se zde řeší případné vykreslování paprsku a přemisťování a aktivace objektu *teleportPoint*. Toto samozřejmě probíhá pouze tehdy, není-li pohyb zakázaný. Dále se zde pracuje s akcí puštění tlačítka pro teleport, na základě čehož je uživatel přesunut na zvolené místo, pokud je toto místo validní pro teleport.

Pokud jsou všechny tyto podmínky splněny, vyvolá se metoda *MoveRig()*, která hráče na požadované místo přesune. Tato metoda vrací hodnotu typu *IEnumerator*, protože v této metodě potřebujeme čekat kvůli zatmavování a nesmí tedy běžet na hlavním vlákně. Poslední metodou této třídy je metoda *updatePointer()*, která vrací hodnotu typu *Boolean* a určuje, jestli je místo, na které uživatel míří validní. K tomu se používá třída *Raycast*, která umožňuje vyslat paprsek v určitém směru a do určité vzdálenosti a pokud paprsek trefí nějaký objekt, se kterým je možné kolidovat, tak tento objekt vrátí a na základě toho určíme, jestli a kam se lze v tomto směru přesunout. Návratová hodnota této metody se tedy použije při aktivaci/deaktivaci směrového paprsku a objektu *teleportPoint*, který ukazuje, kam se uživatel přesune.

Na ukázce 3.2 je znázorněno, jak tento způsob pohybu ve výsledku funguje. Z ovladače míří paprsek a pokud je namířeno na validní lokaci, objeví se na

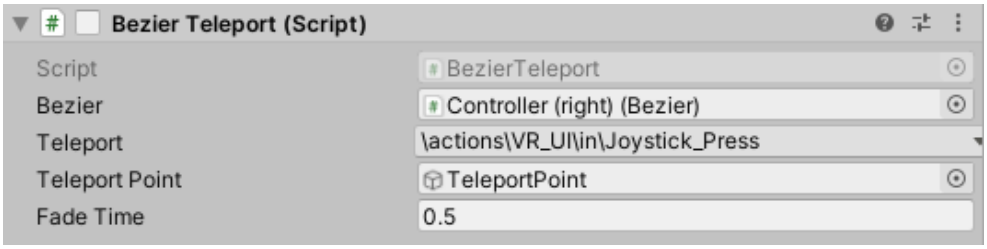


Obrázek 3.2: Ukázka implementovaného přímého teleportu.

konci paprsku modrý kruh, který značí, kam se uživatel přesune. Pokud se tento kruh na konci paprsku neobjeví, znamená to, že uživatel míří na nevalidní lokaci.

3.1.2 Pohyb pomocí obloukového teleportu

Při implementaci tohoto způsobu pohybu budu vycházet z výše popisovaného přímého teleportu, protože jsou si podobné. Jak jsem uvedl v sekci návrhu, pro výpočet oblouku se použije Beziérova křivka. Tento způsob pohybu se skládá ze dvou skriptů (komponent), kde první je *BezierTeleport*, který se stará o detekci stisknutí tlačítka pro teleport, vykreslování bodu, kam se uživatel přesune a samotné přemisťování uživatele. Jak je na obrázku 3.3 vidět, jeho



Obrázek 3.3: Ukázka atributů obloukového teleportu

atributy jsou velmi podobné přímému teleportu s tím rozdílem, že obloukový teleport neobsahuje referenci na *LineRenderer*, jelikož o vykreslování paprsku se stará druhý skript *Bezier*, který výslednou křivku počítá. Z tohoto důvodu tento skript potřebuje referenci na komponentu *Bezier*. Další rozdíl oproti přímému teleportu je ten, že zde není maximální vzdálenost, na kterou se uživatel může přemísťovat, protože tato hodnota je udávána výpočtem křivky.

U druhého skriptu, který se stará o výpočet křivky jsem vycházel z implementace dostupné na serveru GitHub [14]. Zde jsem ve skritpu autora upravil konstanty udávající maximální vzdálenost oblouku, hodnotu zakřivení a způsob práce s objektem typu *LineRenderer*, aby vše fungovalo v mém projektu. Při práci s tímto skriptem jsou tedy při běhu aplikace důležité dva atributy a to *endPointDetected* typu *Boolean*, který má hodnotu *true*, pokud oblouk míří na validní místo pro teleport. Dalším atributem je *EndPoint*, což je bod, na který oblouk míří a na toto místo se lze teleportovat.

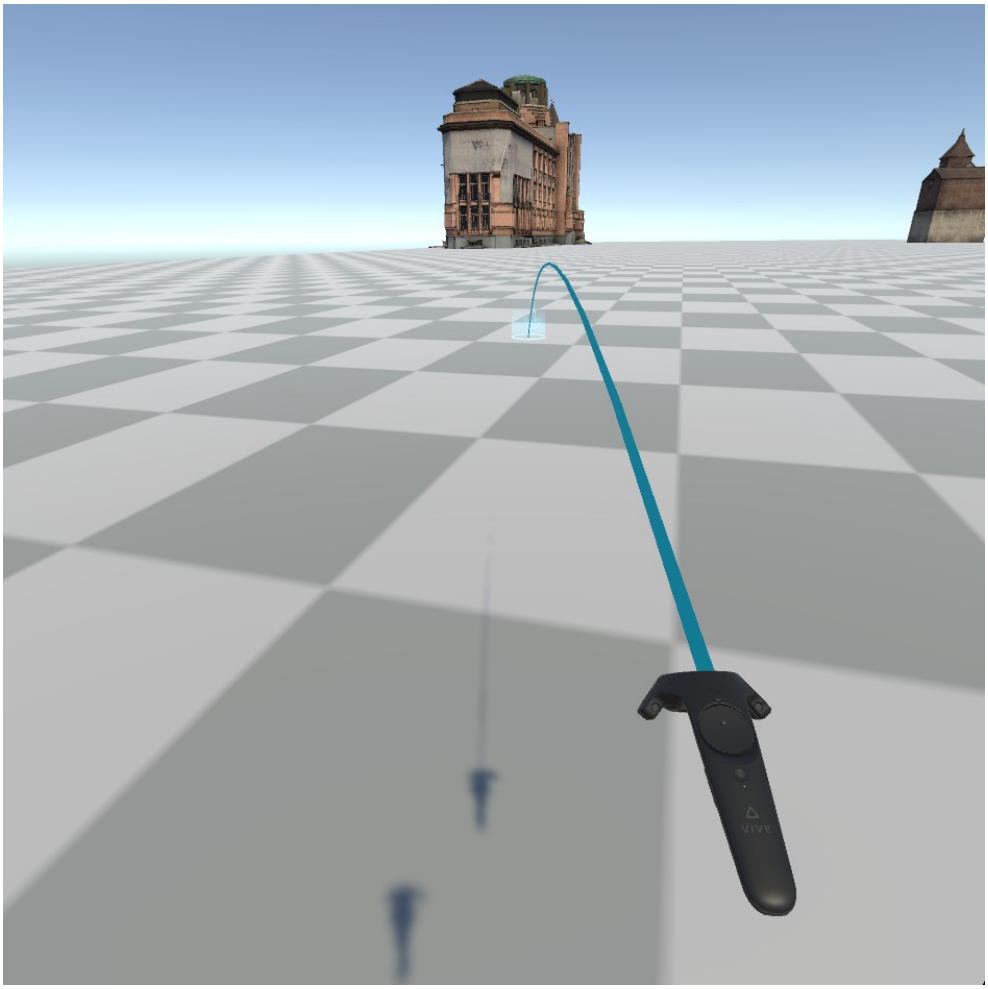
Stejně jako přímý teleport i obloukovému teleportu je možné zasílat zprávy pro aktivaci a deaktivaci pohybu. Opět zde jsou metody na přesunutí uživatele na konečné místo oblouku a pro validaci lokace. V metodě *Update()* se opět odchytávájí akce pro stisknutí a puštění tlačítka určeného pro teleport. A místo upravování pozic komponenty *LineRenderer* probíhá komunikace s objektem *Bezier* pro vykreslování obloukové křivky.

Na obrázku 3.4 je znázorněno, jak probíhá pohyb pomocí obloukového teleportu. Stejně, jako u přímého teleportu je zde paprsek, který vede z ovladače, v tomto případě je ale obloukový. Na konci paprsku je opět modrý kruh, který indikuje validní lokaci pro přesun.

3.1.3 Pohyb pomocí joysticku/trackpadu

Pohyb tímto způsobem zajišťuje třída *JoystickMovement*. Opět je umožněno programátorovi upravovat různé parametry, jak je vidět na ukázce 3.5.

- *JoystickAxis* je atribut typu *SteamVR_Action_Vector2*, což znamená, že obsahuje dvě hodnoty v intervalu (0,1), které udávají pozici joysticku,



Obrázek 3.4: Ukázka použití obloukového teleportu.

nebo trackpadu na dvou osách. Podle této hodnoty se vypočítává směr a rychlost pro pohyb.

- Dalším atributem je *CameraRig*, což je reference na *GameObject* hráče a dalších objektů, které pod něj spadají, v případě použité knihovny SteamVR obsahuje ovladače, kameru a další modely a objekty. Tato reference slouží k tomu, abychom mohli s tímto objektem (hráčem) pohybovat.
- Poslední atribut je *Speed*, kterým lze upravovat rychlost pohybu pomocí joysticku/trackpadu.

Dále má tato třída dva privátní atributy, první je jako u pohybu pomocí

Obrázek 3.5: Ukázka atributů pohybu pomocí joysticku.

teleportu typu *SteamVR_Behaviour_Pose*, abychom zajistili, že bude uživatel moci ovládat pohyb pouze na ovladači, kde se tato komponenta nachází. Druhým atributem je *canMove*, což je hodnota typu *Boolean*, která slouží k zamezení/povolení pohybu pomocí výše zmíněných zpráv. Zde je dobrý příklad použití zamezování pohybu uživatele, když manipuluje s nějakým objektem pomocí manipulace joystickem/trackpadem. Pokud by tu toto omezení nebylo implementované, hýbal by se uživatel zároveň se zvoleným objektem.

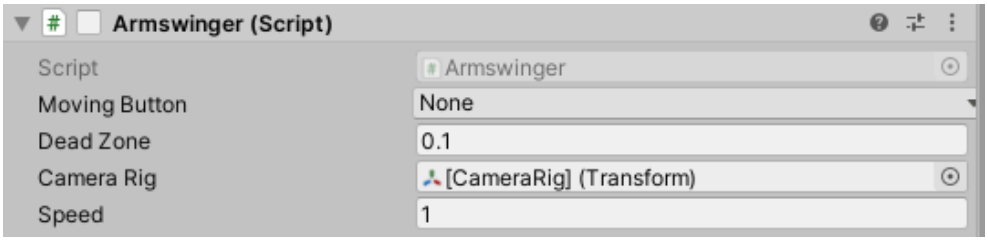
Tato třída má opět způsob pohybu řešený v metodě *Update()*. Zde se v prvním kroku použije hodnota typu *Vector2*, ve které je uložená aktuální pozice joysticku, nebo pozice prstu na trackpadu a pokud tato hodnota není rovná nulovému vektoru a zároveň není zakázaný pohyb, tak se přechází k samotnému pohybu uživatele.

Při výpočtu směru a rychlosti pohybu se bere v úvahu i směr, kterým se uživatel dívá, tedy rotace kamery na vertikální ose Y, jak bylo popsáno v návrhu. Touto rotací je vynásoben trojrozměrný vektor, jehož první a poslední složka odpovídá pozici z dvourozměrného vektoru *JoystickAxis*. Druhá složka vektoru, neboli vertikální pozice je nulová, protože pomocí joysticku se uživatel po této ose nemůže pohybovat. Po tomto výpočtu následuje přesunutí objektu *CameraRig*, který probíhá pomocí příkazu

```
cameraRig.transform.Translate(directionVector);
```

Metoda *Transform.Translate()*, přijímá jako parametr (*Vector3*) zmíněný výše, který určuje, jakým směrem má být objekt přesunut. Tento vektor se následně vynásobí konstantou *Time.deltaTime*, ve které je uložený čas od předchozího snímku. Toto slouží jako korekce problému způsobeného tím, že pokaždé nelze renderovat stejný počet snímků za sekundu (FPS) a při kolísavé hodnotě FPS by se uživatel nepohyboval konstantní rychlostí, ale rychlost by na snímcích za sekundu byla závislá.

Po přesunutí uživatele následuje úsek, který řeší korekturu vertikální osy pro případ, že se uživatel pohybuje například do kopce. Zde se z pozice kamery vystřelí paprsek (pomocí třídy *Raycast*) směrem dolů a pokud je bod kolize paprsku se zemí nad úrovní pozice objektu *CameraRig*, znamená to, že se uživatel pohybuje směrem do kopce a musí být ještě přemístěn po vertikální ose směrem nahoru, aby stál opět na úrovni země. Toho docílíme posunutím



Obrázek 3.6: Ukázka atributů pohybu pomocí máchání rukama.

objektu *CameraRig* na místo bodu kolize se zemí opět použitím metody *Transform.Translate()*. Obdobně se řeší i přesunutí uživatele směrem dolů, pokud jde z kopce.

Jak bylo uváděno v návrhu tohoto způsobu pohybu 2.5.2, chtěl jsem vyzkoušet snižování FOV (field of view), nebo zatmavování okrajů obrazovek, aby se uživatelům při tomto způsobu pohybu nedělalo nevolno. Bohužel se danou funkcionalitu nepovedlo implementovat. Důvodem je to, že nelze za běhu aplikace upravovat parametr FOV, protože to nepovoluje grafická karta v PC. Poté jsem zkoušel zatmavení okrajů obrazovky, to se ovšem také nepovedlo, protože v aplikaci je použita kamera z knihovny SteamVR, která toto neumožňuje. Tato navrhovaná funkcionalita tedy nebude součástí testování ani výsledné aplikace.

3.1.4 Pohyb pomocí máchání rukou

Tento způsob pohybu je implementován ve třídě *Armswinger*. U tohoto způsobu pohybu je nutné, aby skripty/komponenty byly součástí obou ovladačů, protože uživatel může máchat oběma rukama. Je možné použít pohyb pouze pomocí jedné ruky, tento pohyb však působí nepřirozeně. Zde, jako u předchozích způsobů pohybu máme opět parametry, které může programátor měnit 3.6.

- *MovingButton* slouží k detekci akcí zmáčknutí a puštění tlačítka určeného na tento způsob pohybu.
- *DeadZone* je hodnota typu *float*, která určuje minimální rozdíl pozic ovladačů. Toto slouží k tomu, aby se uživatel nemohl pohybovat nechtěně, tzn. pohyb začne pouze pokud uživatel mácháním překoná tuto hodnotu.
- Opět je jako atribut reference na objekt *CameraRig*, abychom s tímto objektem mohli pohybovat.
- Jako poslední atribut je rychlost, kterou se bude uživatel pohybovat.

Jako u předchozích tříd implementujících způsob pohybu je zde několik privátních atributů, například atribut na znemožnění/povolení pohybu. Dále je zde atribut typu *SteamVR_Behaviour_Pose* určující, na které ruce je objekt umístěn. Atribut *lastPos* slouží k výpočtu intenzity máchání rukou a obsahuje hodnotu *Vector3* udávající pozici ovladače v předchozím snímku. Tato hodnota je při pohybu každý snímek přepočítávána. Tato třída opět umožňuje naslouchat zprávám pro povolení/zakázání pohybu.

V metodě *Update()* se odchytávají akce pro zmáčknutí/puštění tlačítka pro pohyb a pokud není pohyb zakázán, tak se vypočítá rozdíl mezi pozicí ovladače v předchozím snímku pokud není rovna nulovému vektoru, což znamená, že se uživatel právě začal pohybovat. Následně, pokud je hodnota rozdílu pozic větší než hodnota atributu *DeadZone*, je zahájen vlastní pohyb uživatele. Zároveň je uložena aktuální pozice ovladače pro výpočet v dalším snímku. Vypočítaná hodnota rozdílu pozic se před dalším použitím pro pohyb musí ještě vynásobit konstantou, protože vzhledem k tomu, že tuto hodnotu počítáme každý snímek a za sekundu je možné mít desítky, nebo i stovky snímků za sekundu, byl by mezi pozicemi minimální (až zanedbatelný) rozdíl.

Pokud tedy zaregistrujeme pohyb, voláme metodu *movePlayer()*, která bere jako parametr vzdálenost aktuální pozice ovladače od předchozí pozice. Zde posuneme uživatele směrem, kterým s ovladačem míří rychlostí, která je vypočítána z parametru rozdílů pozic a z atributu ovlivňujícího rychlost pohybu. Opět je po pohybu uživatele provedena korektura pozice na vertikální ose. Tato korektura je provedena stejně, jako při pohybu pomocí joysticku-/trackpadu, popsaném výše.

3.2 Implementace manipulace s předměty

Jak bylo uvedeno v kapitole zabývající se návrhem, jsou implementovány dva způsoby manipulace s objekty. První umožňuje manipulaci pomocí pohybu prstem po trackpadu, nebo joysticku. Druhý způsob používá pro manipulaci pohyb ovladačů. Každý z těchto způsobů má tedy implementovanou vlastní komponentu, které se mohou libovolně aktivovat/deaktivovat. Tyto komponenty se přidávají k objektu ovladače, podle toho, kterým chceme s objekty manipulovat.

Další důležitou pomocnou třídou je *ObjectManipulator*, která zajišťuje uchovávání právě zvoleného objektu pro manipulaci, aby bylo například možné měnit způsob manipulace s objekty a přitom se neztratila reference na aktuálně zvolený objekt. Dále tato třída uchovává právě zvolený druh manipulace (pohyb, rotace, změna velikosti, měření vzdálenosti). Pokud tedy chceme v aplikaci používat implementované způsoby manipulace s předměty, musí být tato komponenta aktivní.

Tato třída má pouze jeden atribut *DisplayText*, což je reference na objekt typu *UI.Text*, který bude zajišťovat zobrazení naměřené vzdálenosti. Třída

Obrázek 3.7: Ukázka atributů třídy ObjectManipulator.

má dále statické metody, které se používají pro manipulaci s předměty.

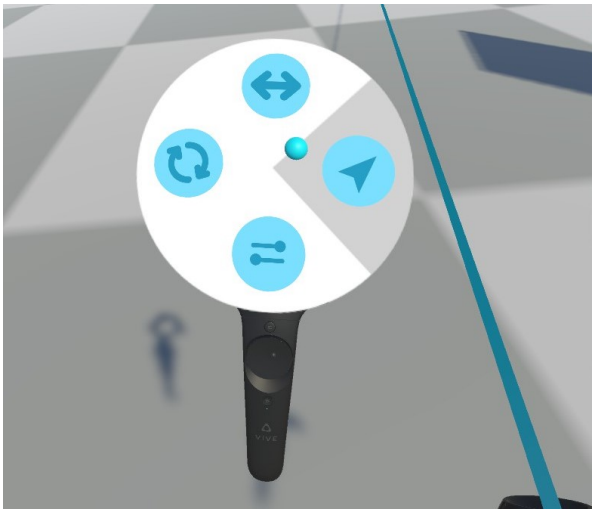
- *Move(), Scale(), Rotate()* – tyto metody složí pro pohyb, změnu velikosti a rotaci vybraného objektu.
- *ChangeMode()* – tato metoda se používá pro změnu druhu manipulace.
- Metodu *DisplayMeasure* použijeme, pokud chceme zobrazit naměřenou hodnotu.

Aby bylo možné při běhu aplikace měnit druhy manipulace s předměty, bylo implementováno jednoduché pomocné uživatelské rozhraní, skrze které je možné měnit druhy manipulace. Jedná se o jednoduché kruhové menu, které je součástí levého ovladače, protože pomocí pravého ovladače probíhá samotná manipulace s předměty. Toto menu bylo implementováno pomocí video návodu na stránce YouTube [15]. Jedná se o kruhové menu, kde kurzor kopíruje pozici prstu na trackpadu u ovladače pro HTC Vive. Kruhové menu má na čtyřech stranách možnosti, které se po najetí kurzoru zvýrazní. Stisknutím trackpadu výběr potvrdíme. Tímto způsobem uživatel mění požadované nástroje jak pro manipulaci pomocí joysticku, tak pro manipulaci pomocí pohybu rukou. Zde se při zvolení požadovaného nástroje volá statická funkce třídy *ObjectManipulator* určená pro změnu právě zvoleného nástroje.

3.2.1 Manipulace s objekty pomocí joysticku/trackpadu

Manipulaci s objekty tímto způsobem zajišťuje třída *ObjectSelection*. Tato třída má některé veřejné atributy, které může programátor upravovat.

- *Activation Button* je *Boolean* akce, kterou po namíření na objekt, zahájíme manipulaci.
- Dalším atributem je *JoystickAxis*, což je obdobně jako při pohybu joystickem dvourozměrný vektor, udávající pozici joysticku/trackpadu. Tento vektor slouží pro ovládání manipulace.
- Atribut *OtherHand* je reference na druhý ovladač a slouží k tomu, abychom mohli z druhého ovladače vysílat paprsky při měření.



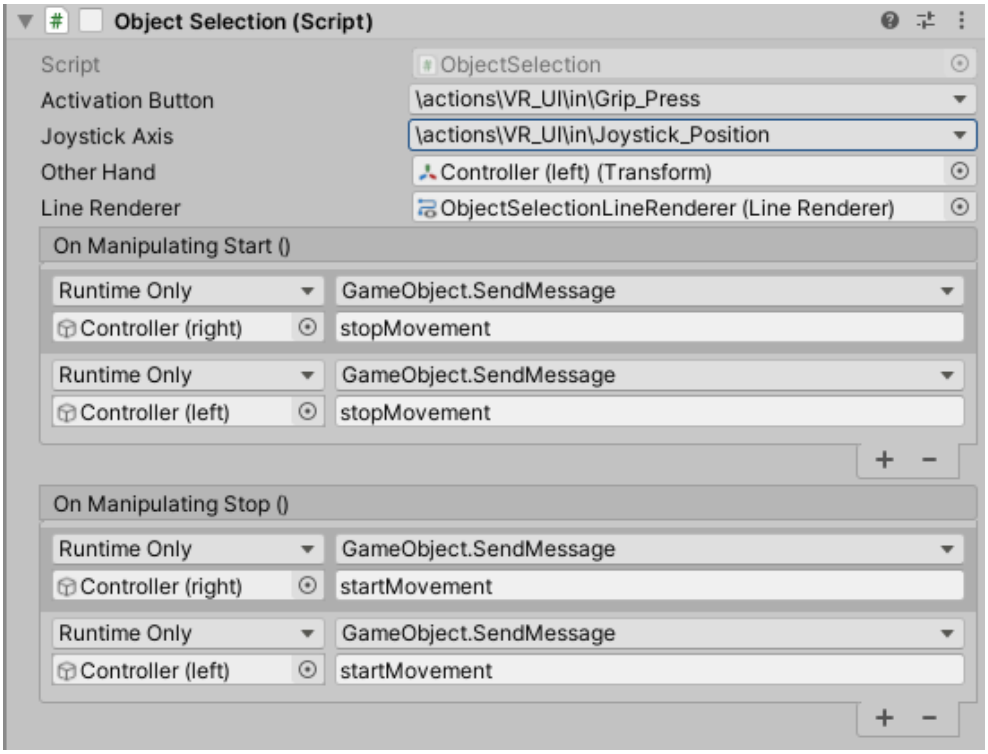
Obrázek 3.8: Kruhové menu na volbu nástroje pro manipulaci s objekty.

- Další atribut *LineRenderer* je reference na objekt s komponentou *LineRenderer*. Tato reference slouží k tomu, abychom mohli viditelný paprsek zapínat a vypínat.
- Poslední dva atributy jsou typu *UnityEvent* a slouží k tomu, že programátor může přidat své vlastní akce, které se vyvolají při začátku, resp. při konci manipulace. Na ukázce 3.9 je toto použito pro vypnutí a zapnutí možnosti pohybu. Samozřejmě lze přidat i více akcí.

Privátní atribut třídy je pouze jeden typu *SteamVR_Behaviour_Pose*, který slouží k určení, jestli manipulujeme levou, nebo pravou rukou, stejně jako při pohybu.

V metodě *Update()* zkoušíme, jestli uživatel míří na objekt, se kterým může manipulovat a pokud ano, tak vykreslíme paprsek, aby uživatel věděl, že s daným objektem, na který míří může manipulovat. Objekty, se kterými je umožněna manipulace musí být označeny tagem „Interactable“. Pokud uživatel míří na objekt, se kterým může manipulovat, stiskne a drží tlačítko pro manipulaci. Zároveň se přiřadí reference na tento objekt ve třídě *ObjectManipulator* a poté se vyvolá akce pro zahájení manipulace a v závislosti na zvoleném způsobu manipulace se vyvolá požadovaná metoda.

V metodě *Measure()*, která se používá na měření vzdálenosti mezi konci dvou paprsků z ovladačů se řeší vykreslování těchto paprsků, pokud tedy míří na nějaký objekt. Zároveň se zde vyvolává metoda třídy *ObjectManipulator*, která zobrazuje naměřenou hodnotu do zvoleného textového pole. V metodě *Rotate()* se testuje pozice prstu na trackpadu, nebo pozice joysticku. Zde se použije pouze jedna osa a to horizontální osa, protože objekty rotujeme pouze po jedné ose. Pokud je tedy testovaná hodnota nenulová, voláme metodu *Ob-*



Obrázek 3.9: Ukázka atributů manipulace s objekty.

jectManipulator.Rotate(), kde jako parametr posíláme hodnotu vstupní osy. Téměř totožně funguje i metoda *Scale()*, s tím rozdílem, že nepoužívá horizontální osu pro určování změnu velikosti objektu, ale osu vertikální. Metoda *Move()* funguje obdobně jako metoda pro rotaci, ale zde neposíláme třídě *ObjectManipulator* jednu osu, ale obě pro zajištění pohybu objektu po dvou osách.

3.2.2 Manipulace s objekty pomocí pohybu ovladačů

Manipulace pomocí pohybu ovladačů má totožné atributy, jako manipulace pomocí joysticku, ale jejich využití je rozdílné. Atribut *OtherHand* je kromě vykreslování paprsků při měření objektu použit zároveň k tomu, aby se z reference na objekt druhého ovladače mohla získat i jeho pozice. Atribut pro akci joysticku/trackpadu je použit na to, aby se při pohybu objektem pomocí ovladače mohl objekt přibližovat, nebo oddalovat. Atributy pro vyvolávání akcí při začátku a při konci manipulace zde mají stejnou funkci, jako při manipulaci prvním způsobem. Rozdíl je v privátních atributech, kde v této třídě potřebujeme uchovávat vzdálenost mezi ovladači z předchozího snímku, abychom mohli počítat, jak se tato vzdálenost mění.

Metoda *Update()* se zde moc neliší od metod v předchozím způsobu manipulace. V této metodě se opět řeší, jestli uživatel míří na objekt, se kterým lze manipulovat, vykreslení paprsku, popř. aktivace manipulace.

Měření vzdáleností pomocí metody *Measure()* je totožná s předchozím způsobem manipulace. Tato metoda se opět stará o vykreslování paprsků z druhého ovladače a zobrazování naměřené hodnoty.

V metodě *Rotate()*, se vyhodnocuje rotace zvoleného objektu. Při tomto způsobu manipulace každý snímek počítáme, jak jsou od sebe ovladače vzdálené a porovnáme tuto hodnotu s hodnotou v předchozím snímku. Na základě rozdílu těchto hodnot objektem rotujeme. Díky tomu, že počítáme vzdálenost mezi ovladači, je možné objektem rotovat při pohybu jakýmkoliv ovladačem, nebo i oběma zároveň.

Při přesouvání objektu v metodě *Move()* nastavíme zvolený objekt, jako potomek ovladače a tím zajistíme, že se bude pohybovat zároveň s ovladačem v té vzdálenosti, ve které byl původně. Při ukončení manipulace s předmětem se objekt vrátí na původní pozici v hierarchii.

Pokud chceme objekt zvětšovat či zmenšovat, využíváme k tomu atribut *prevDistance*, stejně jako při rotaci. Zde podle vypočítaného rozdílu vzdáleností ovladaču objekt buď zvětšujeme, nebo zmenšujeme. Pokud tedy pohybujeme ovladači od sebe, tak objekt zvětšujeme a pokud ovladače přibližujeme, objekt se zmenšuje.

Kapitola 4

Testování

V této kapitole se budu zabývat vytvořením testovací scény v Unity, která bude použita pro testování implementovaných návrhů. Dále bude následovat vyhodnocení testování a úprava implementace na základě výsledků testování. Testování bude probíhat na mém HW, protože nikdo z testovaných uživatelů nevlastní headset pro virtuální realitu. Pro testování bude použita sada HTC Vive a počítač s procesorem AMD Ryzen 7 2700X, grafickou kartou NVIDIA GeForce GTX 1060 3G a 24GB RAM. Tyto komponenty by měly poskytnout dostatečný výkon, aby se výsledná aplikace nesekala a tím nerušila uživatele při testování.

4.1 Vytvoření aplikace pro testování

Vzhledem k tomu, že kolega Bc. Petr Polívka pracuje na diplomové práci na téma uživatelské rozhraní a manipulace s předměty ve virtuální realitě, jsme se rozhodli na vytvoření testovací aplikace spolupracovat. Výhodou této spolupráce je možnost použití uživatelského rozhraní Bc. Petra Polívky, například pro výběr způsobu pohybu. Zároveň Bc. Petr Polívka může použít mé způsoby pohybu, aby se v jeho části mohli uživatelé pohybovat. Další výhodou spojení implementací do jedné testovací aplikace je možnost provádění uživatelského testování obou diplomových prací nezávisle na sobě a výsledky poté sdílet. Zároveň spojením implementací jsme vyzkoušeli, jak jdou naše funkcionality propojit, abychom objevili nedostatky, které mohou nastat programátorům, kteří by používali naše výsledné balíčky ve své aplikaci. Modely do první scény, kde testujeme způsoby pohybu a manipulace s menšími předměty, byly staženy z Asset Store [16].

V druhé scéně byly použity 3D modely budov poskytnuté Filosofickou fakultou UHK.

Průběh testovací aplikace jsme zvolili takový, aby bylo možné aplikaci testovat bez nutnosti naší přitomnosti. Z tohoto důvodu jsme napsali ke každé části návod, který podrobně popisuje, co a jak má uživatel dělat. Díky tomuto

můžeme aplikaci poslat někomu, kdo naše implementace otestuje, bez nutnosti fyzické přítomnosti, například historikům z Filosofické fakulty v Hradci Králové, od nichž máme poslané 3D modely historických budov, které jsou v aplikaci použité. Bohužel kvůli koronavirové situaci nebylo možné tuto spolupráci na testování zrealizovat.

Testovací aplikaci jsme nakonec rozdělili do dvou scén, kde v první scéně testujeme způsoby pohybu a manipulaci s malými předměty implementovanou Bc. Petrem Polívkou. V druhé scéně testujeme manipulaci s většími předměty, konkrétně budovami, což je zaměření mojí diplomové práce. Rozdělit testovací aplikaci jsme se rozhodli, protože v implementaci manipulace s objekty oba používáme velkou část tlačítek, které ovladače nabízejí a při implementaci do jedné scény by docházelo ke kolizím funkcí.

4.2 Průchod testovací aplikací

Testovací aplikace se skládá ze tří částí, kde každá část obsahuje určitý počet úkolů, které musí uživatel splnit, aby bylo prověřeno pochopení ovládání dané funcionality. Postup, jak má uživatel danou část splnit, je vždy podrobně popsán na tabulích, které vysvětlují cíl úkolu, postup a tlačítka používané pro interakci v dané části.

- První část aplikace je zaměřená na způsoby pohybu. Zde si uživatel postupně vyzkouší všechny způsoby pohybu, které aplikace nabízí. Jako první vyzkouší přímý teleport, protože je nejjednodušší. Poté vyzkouší obloukový teleport a na závěr pohyb pomocí joysticku a máchání rukama. Tato část probíhá ve virtuálním prostředí na cestě, aby bylo jasné, kudy se má uživatel pohybovat. Na této cestě jsou označeny záchytné body, na které se uživatel přesune pomocí určeného způsobu pohybu a poté je mu na tabuli vysvětleno, jak si má přepnout na další způsob a je instruován, aby se přesunul k dalšímu záchytnému bodu. Samozřejmě že se uživatel nemusí pohybovat pouze na vyznačené trase, ale v rámci testování pohybových funkcí se může pohybovat po celé scéně.

- Po otestování všech způsobů pohybu následuje část, která je zaměřená na testování manipulace s předměty implementovanou Bc. Petrem Polívkou. Opět je zde instruktážní tabule, která uživateli vysvětluje, jak má daný úkol splnit a jakým způsobem přepínat na požadované nástroje a jak je ovládat. Zde si vyzkouší přesuny předmětů, změnu jejich rotace, textury, atd. Po dokončení úkolů si uživatel může ještě libovolně tyto nástroje zkoušet. Objeví se mu stůl s předměty, kde některé mají aktivovanou fyziku, takže může testovat manipulaci pomocí uchopování předmětů a objekty házet. Až si nástroje dostatečně vyzkouší, přesune se ke kontrolnímu bodu, kde se mu otevře nabídka, ve které si může zvolit, jestli si chce tuto scénu zopakovat, nebo pokračovat do další scény.

- V poslední části uživatel testuje manipulaci s budovami. Zde je opět instruktážní tabule, která popisuje úkoly, které má uživatel vykonat a jakým způsobem se nástroje ovládají. Uživatel začíná manipulaci s objekty pomocí joysticku/trackpadu a má za úkol postupně přesunout, zvětšit, otočit budovu. Poté se přepne způsob manipulace na manipulaci pomocí pohybu rukou a opět má uživatel úkoly, aby objekt přesunul, zmenšil, ... Po dokončení úkolů má opět možnost si nástroje libovolně zkoušet. Jsou zde modely budov skenované Filosofickou fakultou UHK, takže je možné si tyto budovy detailněji prohlížet.

V průběhu celé aplikace má uživatel možnost přepínat mezi způsoby pohybu, aby nebyl nucen se pohybovat způsobem, který mu nevyhovuje, nebo aby je mohl dále zkoušet i v jiných fázích testovací aplikace. Součástí aplikace je i testování prvků uživatelského rozhraní implementovaných Bc. Petrem Polívkou. Ty jsou použity například na změnu způsobu pohybu, nebo jako instruktážní tabule.

Během testování jednotlivých uživatelů jsem se snažil s nimi nekomunikovat, aby se pokusili ovládání aplikace pochopit sami, k čemuž jim pomáhají instruktážní tabule. Pokud ale měli s úkoly velké problémy, nebo nevěděli jak nástroje ovládat, byla jim poskytnuta pomoc. Po splnění všech úkolů v dané sekci byli ještě tázáni na různé otázky ohledně používání pohybu a manipulace s předměty, například jaký způsob pohybu se jim zdál nejlepší.

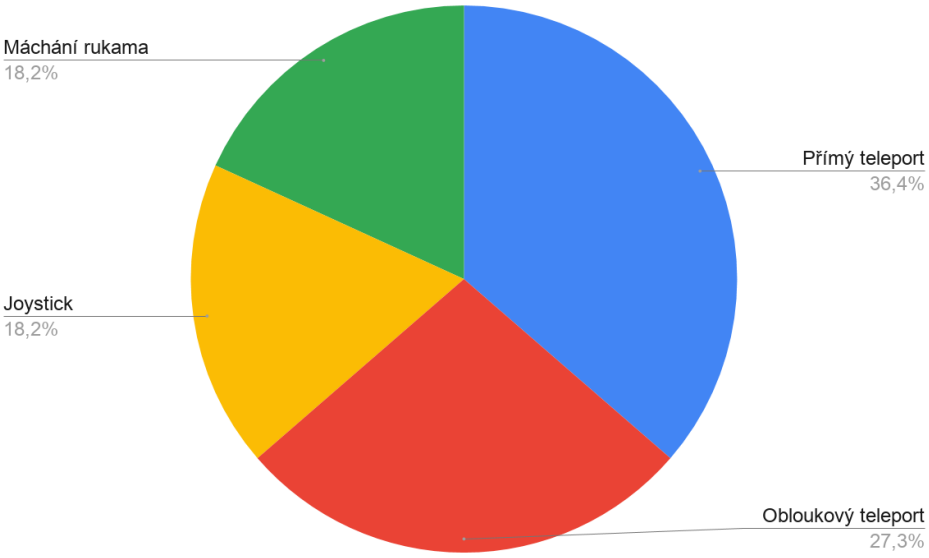
4.2.1 Výběr uživatelů pro testování

Jako dobrovolníky pro testování jsem oslovil uživatele, kteří mají už nějaké zkušenosti s aplikacemi ve virtuální realitě i uživatele, kteří mají malé, nebo žádné zkušenosti s virtuální realitou. Díky tomu bude pokryto větší spektrum uživatelů z různými zkušenostmi s virtuální realitou. Uživatele pro testování jsme volili mezi známými a rodinnými příslušníky.

4.3 Výsledky testování

Testovací aplikaci jsme testovali na patnácti uživatelích, kteří měli za úkol projít tuto aplikaci a splnit úkoly, kterými si vyzkouší implmentaci navrženého řešení. Tím, že se v aplikaci testuje jak moje implementace způsobů pohybu a manipulace s předměty, tak manipulace s předměty a uživatelské rozhraní Bc. Petra Polívky, mohli jsme oba nezávisle na sobě testovat obě implementace a poté jsme si předali výsledky.

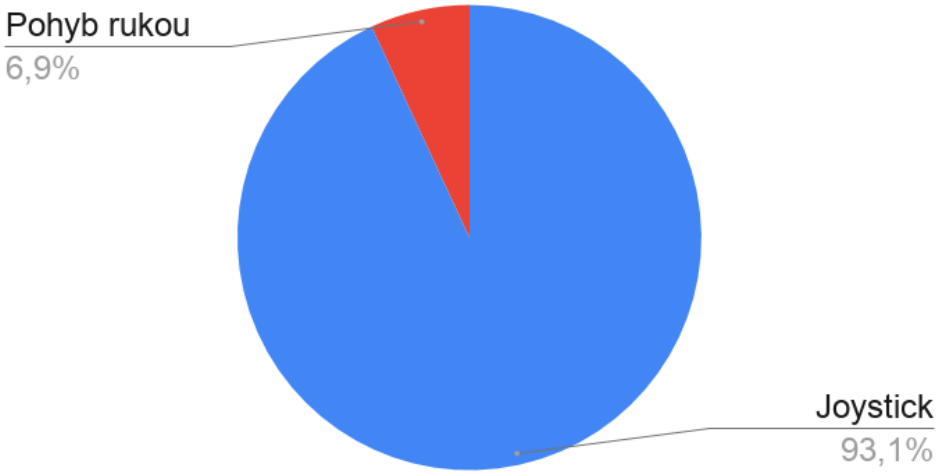
Dle předpokladu, při testování způsobů pohybu pomocí joysticku a máchání rukama bylo několika uživatelům nevolno, ale některým uživatelům tento způsob pohybu přišel jako nejlepší z implementovaných. Oblíbenosti způsobů pohybu jsou znázorněny na grafu 4.1 níže. Při dotazu na preferovaný způsob pohybu po malém prostoru byly odpovědi různorodé. Naopak při dotazu na



Obrázek 4.1: Graf oblíbeností způsobů pohybu.

preferenci způsobu pohybu na větší vzdálenost se valná většina testovaných uživatelů shodla na přímém teleportu. Osobně s tím souhlasím, protože při přímém teleportu můžeme opakovaně mačkat tlačítko pro teleport a mířit směrem, kterým se chceme pohybovat a jsme takto schopni se rychle přesunout na velkou vzdálenost. Z grafu je tedy patrné, že jako celkově nejoblíbenější způsob pohybu zvolili uživatelé pohyb pomocí teleportu. Zajímavé ale je, že přímý teleport byl více oblíbený než obloukový, i když je ve většině ostatních aplikací pro virtuální realitu běžnější pohyb pomocí obloukového teleportu. U teleportu jsem dále testoval ideální čas zatmavení při přesunu. Z testů vyplynulo, že ideální hodnota byla 0, 25s, což byla předem nastavená implicitní hodnota. Když je hodnota menší, tak není přesun tak plynulý. A naopak, když je hodnota větší, tak je přesun až moc pomalý. Zároveň většině uživatelů přišel způsob pohybu pomocí teleportu, ať už obloukového nebo přímého, přesnější, protože předem vidí, kam se přesunou a můžou se tak přemístit přesně na to místo, kam chtějí.

Při testování manipulace s objekty jsem narazil na určité chyby a připomínky, které jsou popsány níže. Uživatelům přišel lepší způsob pro manipulaci pomocí joysticku oproti pohybu ovladači, což je vidět na grafu 4.2. Tento způsob zvolili jako lepší například proto, že k němu lze použít pouze jednu ruku. Zároveň bude nutné upravit citlivost u manipulace s objekty, protože manipulace pomocí joysticku se několika uživatelům zdála pomalá. Na druhou stranu, pokud je manipulace s předměty pomocí joysticku pomalejší, může být



Obrázek 4.2: Graf oblíbeností způsobů manipulace s předměty.

přesnější, než manipulace pomocí pohybu ovladači.

Zajímavé bylo pozorovat rozdíly mezi uživateli, kteří již mají zkušenosti s virtuální realitou, a těmi, kteří ji zkoušeli poprvé. Jedním z těchto rozdílů je ten, že při manipulaci s předměty se v testovací aplikaci používalo tlačítko grip. Uživatelé, kteří se již s virtuální realitou setkali, měli s tímto způsobem ovládání problém, protože u většiny aplikací se předměty berou do rukou pomocí tlačítka spouště. Uživatelé, kteří zkoušeli virtuální realitu poprvé naopak s tímto ovládáním neměli problém. Další pozorovanou skutečností je, že rychlost průchodu aplikací a pochopení ovládání nezávisí na zkušenosti s virtuální realitou. Někteří uživatelé, kteří s virtuální realitou neměli žádné zkušenosti, ovládání aplikace pochopili mnohem rychleji, než někteří uživatelé, kteří již mají s virtuální realitou zkušenosti. Zároveň uživatelé, kteří měli virtuální realitu poprvé, měli zpočátku problém pochopit systém teleportu, protože při pohybu přímým teleportem jim hned nebylo jasné, že mají namířit ovladačem na místo na zemi, kam se chtějí přesunout, ale mířili směrem, kam se chtěli přesunout. Následně ale systém teleportu pochopili poměrně rychle.

4.3.1 Nejčastější problémy

V průběhu testování se ukázaly problémy, které měli téměř všichni uživatelé. Z tohoto důvodu se v úpravě implementace musí všechny tyto problémy vyřešit. Tyto problémy vznikly převážně z důvodu, že jsem při implementaci vycházel z vlastních zkušeností a preferencí, ale uživatelé, kteří nemají takové zkušenosti s virtuální realitou, mají jiné preference.

- Způsob teleportu pomocí obloukové křivky má mnohem menší dosah,

než přímý teleport. Takže musí být upravena maximální vzdálenost, na kterou se lze pomocí obloukové křivky teleportovat. Zároveň vzdálenost obloukového teleportu byla pro většinu uživatelů nedostatečná při pohybu na větší vzdálenost.

- Při pohybu pomocí přímého teleportu se zobrazí paprsek z ovladače pouze ve chvíli, když uživatel míří na místo, kam se může teleportovat. Toto chování bylo pro uživatele, kteří měli virtuální realitu poprvé, trochu zmatečné, protože nevěděli, jestli postupují správně, protože po držení tlačítka pro pohyb není vidět žádná změna. Asi by bylo vhodné zobrazovat paprsek při držení tlačítka, ale zobrazit značku, kam se uživatel teleportuje, až když míří paprskem na místo, kam se může teleportovat.

- Při pohybu pomocí joysticku/trackpadu je nastavená malá maximální rychlost.

- U pohybu pomocí máchání rukama je nutné upravit pohyb do kopce a z kopce, v aktuálním řešení je při pohybu například ke stolu uživatel přemístěn na tento stůl, nebo při chůzi ze strmějšího kopce, je uživatel hodně rychle přemístěn směrem dolů, až má pocit, že se teleportoval. Je tedy nutné upravit v implementaci řešení stoupání a klesání při pohybu.

- Při pohybu objektem pomocí joysticku byl problém s pohybem podle globálních souřadnic, které je potřeba změnit na souřadnice relativní k pohledu kamery.

- Při manipulaci s budovami pomocí pohybu ovladači nelze objekt přibližovat a oddalovat.

4.3.2 Návrhy a připomínky k implementaci

Při dotazování testovaných uživatelů jsme shromáždili jisté připomínky k implementaci, které jsou zajímavé, a v úpravě implementace by se daly zohlednit. Popřípadě by se dala do výsledného balíčku dát možnost pro výběr, jestli danou funkcionalitu využít, nebo ne.

- Několik uživatelů navrhlo, že při pohybu joystickem by bylo dobré postupné zrychlování pohybu, pokud uživatel rovnou nastaví maximální rychlost joysticku. To by mohlo pomoci lidem, kterým se dělá nevolno při pohybu joystickem.

- Při manipulaci s objekty není nikde vidět, který nástroj je aktuálně zvolený. Bylo by dobré přidat označení právě aktivního způsobu manipulace.

- Dalším návrhem bylo přidat vizuální označení aktivace manipulace s objektem. Protože momentálně tato indikace chybí a někteří uživatelé si nebyli jisti, jestli postupují správně.

4.4 Úprava implementace na základě výsledků testování

Na základě proběhlého testování jsem se v implementaci rozhodl změnit, nebo přidat funkcionality popsané v této sekci. Z části se jedná o drobné nedostatky, u kterých jsem nečekal, že budou pro uživatele velkým problémem, ale při testování se ukázalo, že problémem jsou. Dále jsem upravil implementaci některých funkcionalit na základě návrhů, nebo podnětů od uživatelů. Zjištěné problémy ve smyslu úpravy parametrů, například problém s pomalým pohybem pomocí joysticku, zde nebudou rozebrány, protože se jedná pouze o přepsání jedné hodnoty. Zároveň výsledný balíček budou používat ostatní vývojáři a ti mají možnost tyto hodnoty měnit dle svých preferencí.

4.4.1 Úprava obloukového teleportu

Pro většinu uživatelů byla maximální vzdálenost obloukového teleportu nedostatečná. Osobně si myslím, že to je způsobeno tím, že v testovací aplikaci měli uživatelé za úkol se přesouvat na větší vzdálenost a pokud by se měli pohybovat na menší vzdálenost, například v místnosti, tak by tento problém nenastal. Nicméně pro použití tohoto způsobu pohybu i na větší vzdálenost jsem poupravil hodnoty pro výpočet obloukové křivky tak, aby paprsek dosahoval dál, než doposud. Zároveň jsem přidal zakřivení oblouku, protože někteří uživatelé na první pohled nepoznali rozdíl mezi obloukovým a přímým teleportem. Tyto úpravy byly provedeny ve třídě *Bezier*, která řeší výpočet obloukové křivky. Maximální vzdálenost je tedy ve výsledku zhruba stejná, jako maximální vzdálenost přímého teleportu a zároveň je obloukový paprsek více zakřiven.

4.4.2 Řešení stoupání a klesání u pohybu pomocí joysticku a máchání rukama

Při pohybu joystickem/trackpadem a máchání rukama byl problém při stoupání a klesání. To je způsobené systémem pro přesouvání uživatele po ose Y, který při detekci stoupání/klesání uživatele přemístí po vertikální ose tak, aby vždy stál na zemi. Zde byl problém při prudkém klesání, protože toto instantní přesunutí bylo na velkou vzdálenost a tudíž úplně nesimulovalo reálnou chůzi. Dalším problémem bylo to, že detekce stoupání a klesání se provádí pouze pomocí jednoho bodu v prostoru, tudíž je tento systém náchylný na malé nerovnosti v terénu. Pokud by se tento problém měl vyřešit úplně, musel by se

kompletně přepracovat systém pro tuto detekci, například přidáním více bodů aby ohraničovaly oblast 0,5x0,5 m v místě, kde uživatel stojí ve virtuálním světě. Toto by ale bylo poměrně složité na implementaci.

Rozhodl jsem se tedy, že upravím zmiňované přemísťování při stoupání a nebo klesání. Zde jsem zaměnil obyčejné přiřazování polohy na vertikální ose za metodu *Lerp()*, která má tři parametry a to dvě hodnoty typu *Vector3*, kde první jsou souřadnice bodu, odkud se chceme přemístit, a druhý prametr je, kam se chceme přemístit. Třetím parametrem je hodnota mezi 0 a 1 (v tomto případě jsem zvolil hodnotu 0,25), udávající procentuální vzdálenost požadovaného bodu od první souřadnice na přímce mezi těmito dvěma body. Jedná se tedy o lineární interpolaci. Díky tomuto se razantně zmenší vzdálenost, o kterou je při stoupání/klesání uživatel přemístěn a tento pohyb je mnohem plynulejší a tím pádem uživatel nemá pocit, že se teleportuje.

4.4.3 Úprava oblasti pro teleportování

V testovací aplikaci bylo možné se teleportovat na všechna místa, která měla aktivní komponentu pro detekci kolize. Což znamená, že se uživatel může teleportovat téměř na všechna místa, protože většina objektů má komponentu pro detekci kolize. Provedl jsem úpravu implementace, nyní je možné se teleportovat pouze na herní objekty, které mají tag „CanMove“, což zamezuje teleportaci na všechny objekty obsahující komponentu *Collider*.

Někteří uživatelé měli podnět, že by mohla být oblast, na kterou se dá teleportovat, vizuálně označena při držení tlačítka pro teleport, jako je tomu například v aplikaci SteamVR Home. Toto by určitě mohlo být implementováno, ale vzhledem k tomu, že výsledkem této práce je balíček funkcionalit pro ostatní vývojáře, nebude tato funkcionalita součástí implementace výsledného balíčku. Díky předešlé úpravě si může vývojář jednoduše sám implementovat vlastní styl této oblasti. Jednoduchý způsob, jak by toho šlo docílit je, že se na výslednou oblast pro pohyb umístí objekt, který bude mít požadovaný vizuální styl a požadovaný tag, aby se na něj dalo teleportovat. Tento objekt by se zobrazil pouze při držení tlačítka pro teleport.

4.4.4 Změna způsobu přesouvání objektů pomocí joysticku

Při pohybu objektem pomocí joysticku byl problém v tom, že se objekt pohyboval podle globálních os. Problém způsobilo to, že pokud měl uživatel jinou rotaci, tak pohyby joysticku neodpovídaly směru, kterým se objekt pohyboval. Tento způsob manipulace s objektem byl upraven tak, že se nyní objekt pohybuje v závislosti na rotaci ovladače, kterým s objektem manipulujeme.

Původně byl tento systém pohybu po globálních osách navržen, jelikož by měl být přesnější, z důvodu toho, že k pohybu není připočítána rotace

ovladače. Ovšem při testování se ukázalo, že žádným uživatelům tento systém nevyhovoval.

4.4.5 Změna způsobu přesouvání objektů pomocí pohybu rukou

Zde byl hlavní problém v tom, že při hýbání objekty pomocí pohybu rukou nelze držený objekt přibližovat a oddalovat. Toto je nyní možné pomocí joysticku, nebo pohybu prstu na trackpadu tak, že pohybem prstem nahoru se objekt od uživatele oddaluje a pohybem prstu k sobě se objekt přibližuje. Rychlost tohoto pohybu byla ponechána stejná, jako při pohybu objektem pouze pomocí joysticku/trackpadu.

Zároveň byla upravena implementace systému pro tento způsob manipulace, protože mnoha uživatelům nevyhovovalo, že během pohybu objektu se může měnit i rotace. To je způsobeno tím, že při aktivaci manipulace se zvoleným objektem, se objekt nastaví jako potomek herního objektu ovladače, tudíž při pohybu ovladačem pohybujeme i zvoleným předmětem. Toto má za následek, že kromě pozice se upravuje i rotace, takže pokud chceme objektem pouze pohnout, musíme být přesní, abychom nezměnili i rotaci. V nově předělaném systému už se objekt nenastavuje jako potomek, ale dynamicky se mu mění pozice v závislosti na rotaci ovladače na vertikální ose a vzdálenosti ovladače od objektu. Objektem už tedy nelze při pohybu rotovat.

4.4.6 Úprava označení objektu při manipulaci

Při testování mělo mnoho uživatelů problém s tím, že není vizuálně znázorněno, jestli byla interakce zahájena a zároveň nebylo nikde vizuálně znázorněno, který nástroj pro manipulaci je právě aktivní. Tyto funkcionality budou ve výsledném balíčku hrát roli pouze v ukázkové scéně, předpokládám, že programátor, který bude výsledný balíček používat, bude mít vlastní styl UI a zobrazování nástrojů si implementuje sám.

Označení předmětu při manipulaci bylo upraveno tak, že při aktivaci manipulace paprsek mířící na objekt změní barvu, aby bylo jasné, že došlo k zahájení manipulace. Po puštění tlačítka pro manipulaci se barva paprsku opět změní na původní, aby bylo zřejmé, že se manipulace ukončila.

Přidáno bylo textové pole u ovladače, na kterém je textově znázorněno, který nástroj je momentálně aktivní. Uživatel tedy vždy vidí, jaký nástroj se momentálně používá.

4.5 Vytvoření výsledného balíčku pro Unity Asset Store

Výsledný balíček by měl obsahovat ukázkovou scénu, kde bude ukázáno praktické využití funkcionalit. Pro tuto scénu jsem zvolil jednoduché prostředí, kde

je umístěný pouze objekt *CameraRig*, který reprezentuje uživatele a ovladače. Na objektu pravého ovladače jsou umístěny skripty pro pohyb s nastavenými atributy na základě výsledků z testování. Programátor může mezi nimi přepínat, aby mohl všechny jednoduše vyzkoušet.

Zároveň je před ním objekt ve tvaru krychle, který má požadovaný tag, aby s ním šlo manipulovat pomocí implementovaných nástrojů. Nástroje jsou opět komponenty, které používají pravý ovladač k manipulaci, a lze mezi nimi přepínat. Součástí balíčku je i jednoduché kruhové menu na levé ruce, aby šlo přepínat mezi druhy manipulace. Předpokládá se ale, že si programátor UI a design implementuje vlastní.

Důležitou součástí této scény je i informační tabule, kde je popsáno jednoduché používání balíčku. Podrobněji je návod popsaný v souboru README.txt, který je součástí balíčku. Do dokumentace jsem také přidal odkaz na server YouTube, kde je ve videu ukázáno nastavení skritptů [17].

Postup pro publikování balíčku v Unity Asset Store byl následující. Výsledný balíček v AssetStore je na ukázce 4.3.

- Po přihlášení do Unity účtu bylo potřeba vytvořit publisher účet. Zde jsme museli zadat kontaktní údaje, název společnosti, atd.

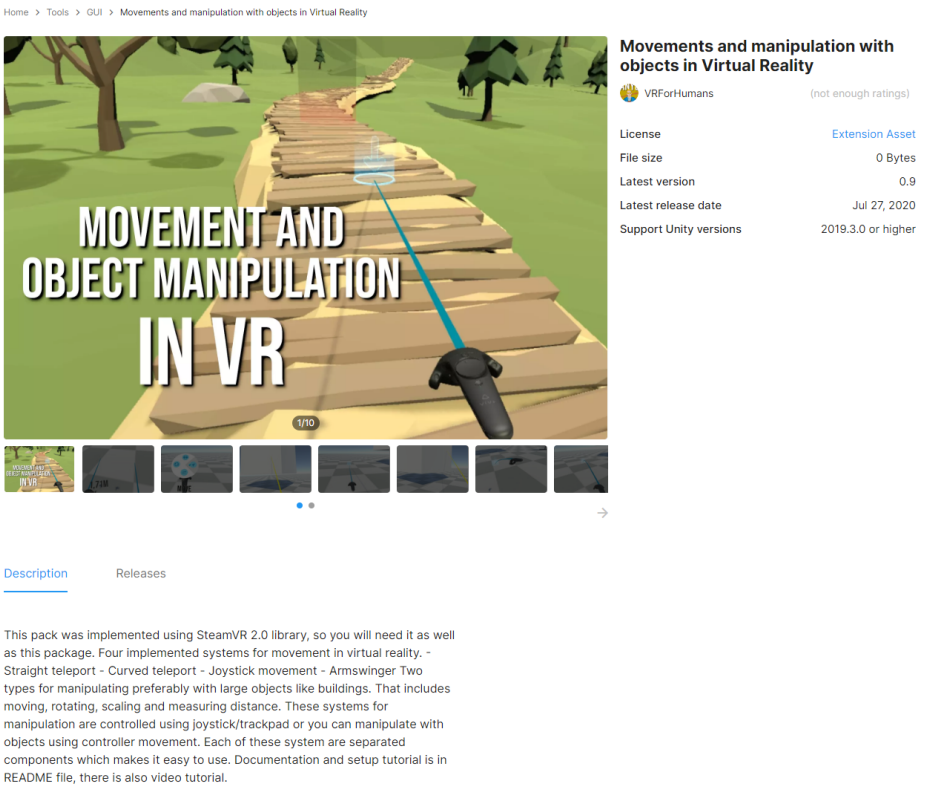
- Dále jsme si z asset store museli stáhnout balíček, který do editoru přidává možnosti vytvoření balíčku. Důležité je, že tento balíček musí být v projektu, z kterého chceme náš balíček vytvářet. Zároveň jsme v Unity provedli validaci balíčku. Zde validátor ověřuje například, jestli máme přiloženou dokumentaci ve formátu PDF, nebo RTF. Po úspěšné validaci balíčku jsme zvolili možnost upload.

- Další důležitá součást vytváření balíčku je, že musíme uvést název, popis balíčku, klíčová slova a zvolit kategorii, do které náš balíček patří a jak bude v obchodě dohledatelný. Zároveň je možné stanovit poplatek, za který bude balíček dostupný, v našem případě je balíček dostupný ke stažení zdarma.

- Poté jsme museli nahrát ikony pro balíček a snímky obrazovky, které budou ukazovat použití funkcionalit. Zde je možné i nahrát video.

- Dále je důležité označit, které verze Unity jsou podporovány, v našem případě to jsou Unity verze 2019, protože na této verzi byl balíček vyvíjen. Zároveň je nutné označit, závislosti balíčku, pro naše balíčky se jednalo pouze o plugin SteamVR.

- Poté, co jsme měli výše zmíněné kroky hotové, museli jsme podvrdit zaškrtávácím políčkem, že nahrávané assety máme právo tímto způsobem šířit. Poté jsme potvrdili publikování balíčku. Před samotnou publikací je balíček zkontrolován ze strany Asset Store. Tato kontrola spočívá



Obrázek 4.3: Výsledný balíček v AssetStore.

v tom, že se náš balíček otestuje, jestli neobsahuje věci podléhající autorským právům. Toto může trvat řádově i několik dní. Poté již bude náš balíček dostupný v obchodě pro ostatní programátory. Jelikož jsme nepředpokládali, že kontrola bude trvat tak dlouhou dobu, nebyl balíček zkontrolován před termínem odevzdání diplomové práce a proto neobsahuje odkaz na tento balíček. Podle názvu obsaženého v obrázku 4.3 bude po dokončení kontroly balíček jednoduše dohledatelný.

4.6 Možnosti dalšího vývoje způsobů pohybu a manipulace s předměty

Výsledný balíček způsobů pohybů a manipulace s předměty lze ještě vylepšit a doplnit různými dalšími funkcionalitami nad rámec této diplomové práce. Kromě přidávání dalších způsobů pohybu by šlo také přidat různé možnosti ke stávajícím způsobům. Například při pohybu joystickem by mohlo jít přepínat mezi pohybem po globálních osách a nebo pohybem relativním k rotaci ka-

mery. U způsobu pohybu pomocí teleportu by šlo vizuálně ohraničit oblast, po které se dá teleportovat. U pohybu pomocí joysticku a máchání rukama by šla přidat možnost pro skok, aby měl uživatel možnost se dostat i na vyvýšená místa. To by ale znamenalo přidání gravitace pro hráče, což by se muselo řádně otestovat a doladit, aby nedocházelo k chybám, protože pokud se v Unity přidá objektu komponenta pro gravitaci a fyziku, tak například při nárazu se tento objekt může otočit, což pro hráče nechceme.

U způsobů manipulace s objekty by bylo zajímavé přidat možnost manipulace pomocí fyzických nástrojů, jako je implementováno v rámci aplikace SteamVR Home. K tomu by bylo potřeba vytvořit k těmto nástrojům 3D modely a navrhnout a implementovat způsob pro braní těchto nástrojů do ruky.

Závěr

Dle zadání byla provedena analýza stávajících aplikací pro virtuální realitu, návrh a implementace řešení pohybu a manipulace s předměty ve virtuální realitě. Implementované funkcionality byly otestovány a upraveny, aby byly uživatelsky co nejpřívětivější. Funkcionality byly vyvíjeny pro účely publikování balíčku na obchod Asset Store, kde si ho mohou vývojáři pro platformu Unity zdarma stáhnout a použít ve své aplikaci.

Součástí výsledného balíčku jsou čtyři způsoby pohybu a dva způsoby manipulace s předměty ve virtuální realitě. První způsob pohybu pracuje na principu teleportu, kde uživatel namíří na místo, kam se chce přesunout a pomocí tlačítka na ovladači se na požadované místo přemístí. Další alternativa je také na principu teleportu, ale tentokrát se cílové místo určuje pomocí obloukové křivky. Kromě způsobu pohybu pomocí teleportu byl implementován i pohyb pomocí joysticku. Poslední způsob pohybu funguje na principu máchání rukama, jako při běhu. Tento způsob pohybu mě osobně přišel nejzajímavější, ovšem při testování se ukázalo, že nejoblíbenější způsob pohybu je teleport.

Při manipulaci s objekty je možné použít jeden ze dvou způsobů. První způsob používá k manipulaci joystick, pomocí kterého můžeme požadovaný objekt přesouvat, měnit jeho rotaci po vertikální ose a nebo ho zvětšovat či zmenšovat. Druhý způsob manipulace probíhá pomocí pohybu ovladači. Manipulace byla navržena a implementována pro využití při práci s historickými budovami, proto je pro rotaci zvolena pouze jedna osa a manipulace probíhá na dálku pomocí paprsků.

Výsledný balíček byl umístěn na Unity Asset Store, kde je dostupný pro ostatní vývojáře. Součástí tohoto balíčku je i dokumentace k použití implementovaných komponent. Zároveň je i k dispozici video, které ukazuje nastavení a použití těchto komponent.

Literatura

[1] HTC Vive. [online]. Dostupné z: https://edigital.cz/virtualni-realita/htc-vive-p630029

[2] Wexelblat, A.: *Virtual reality: applications and explorations*. Academic Press, 2014.

[3] Games, E.: Unreal Engine. [online]. Dostupné z: https://www.unrealengine.com/en-US/

[4] Technologies, U.: Unity. [online]. Dostupné z: https://unity.com

[5] Crytek: CryEngine. [online]. Dostupné z: https://www.cryengine.com

[6] Unity vs Unreal Engine: which game engine is for you? *Creative Bloq*, [cit. 2020-07-21]. Dostupné z: https://www.creativebloq.com/advice/unity-vs-unreal-engine-which-game-engine-is-for-you

[7] SteamVR on Steam. [online], [cit. 2020-05-06]. Dostupné z: https://store.steampowered.com/app/250820/SteamVR/

[8] Buttfield-Addison, P.; Manning, J.; Nugent, T.: *Unity Game Development Cookbook: Essentials for Every Game*. O'Reilly Media, 2019.

[9] Murray, J. W.: *Building virtual reality with Unity and Steam VR*. CRC Press, 2017.

[10] RUST: Hot Dogs, Horseshoes & Hand Grenades. [online], 2016. Dostupné z: https://store.steampowered.com/app/450540/Hot_Dogs_Horseshoes__Hand_Grenades/

[11] Solirax: Neos VR. [online], 2018. Dostupné z: https://store.steampowered.com/app/740250/Neos_VR/

[12] Teleport Curves with the Gear VR Controller. *Gabor Szauer*, listopad 2017, [cit. 2020-6-20]. Dostupné z: https://developer.oculus.com/blog/teleport-curves-with-the-gear-vr-controller/

[13] Columbia University research on VR sickness prevention, dynamic FOV modification showing promise. *Tom's Hardware [online]*, leden 2015, [cit. 2020-6-20]. Dostupné z: https://www.tomshardware.com/news/columbia-university-vr-sickness-research,32093.html

[14] Karaouni, H.: GearTeleporter. [online], 2018. Dostupné z: https://github.com/FusedVR/GearTeleporter

[15] with Andrew, V.: Radial Menu for SteamVR 2.0. [online], 2019. Dostupné z: https://www.youtube.com/watch?v=C7Lo3VoMHn0

[16] RPG Poly Pack - Lite — 3D Landscapes — Unity Asset Store. [online], červenec 2019, [cit. 2020-07-08]. Dostupné z: https://assetstore.unity.com/packages/3d/environments/landscapes/rpg-poly-pack-lite-148410

[17] Movement and object manipulation setup tutorial. *YouTube. Copyright © 2020 Google LLC [online]*, [cit. 2020-07-21]. Dostupné z: https://youtu.be/Gk2EgC6x89I

Příloha A

Seznam použitých zkratek

VR Virtuální realita

UI User interface

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `build` ..... spustitelná forma implementace (testovací aplikace)
- `src`
  - `impl` ..... zdrojové kódy implementace (Unity project)
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF
- `preview`
  - `screenshots` ..... snímky obrazovky z výsledné implementace
  - `video` ..... ukázkové video z výsledné implementace