FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Návrh a implementace aplikace pro práci s historickými artefakty ve virtuální realitě
Student:	Bc. Petr Polívka
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 uživatelského rozhraní a nástrojů pro manipulaci s historickými artefakty a objekty ve virtuální realitě (VR) se zaměřením na uživatelskou skupinu historiků.

Cíle práce:
1. Analyzujte aktuální řešení uživatelského rozhraní ve VR aplikacích.
2. Navrhněte sady testovacích uživatelských rozhraní v závislosti na typu použití (menu, panel nástrojů, atd..)
3. Implementujte navržená testovací rozhraní.
4. Implementujte vybrané nástroje vhodné pro historiky (řez objektem, měření rozměrů, objem, transformace, lupa, změna textur, ...).
5. Proveďte a vyhodnoťte uživatelské testování.
6. Navrhněte a vytvořte balíček pro Unity3D.

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 práci s historickými artefakty ve virtuální realitě

Bc. Petr Polívka

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

Polívka, Petr. *Návrh a implementace aplikace pro práci s historickými artefakty ve virtuální realitě*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato diplomová práce se zabývá analýzou, návrhem a implementací základních prvků uživatelského rozhraní a nástrojů pro manipulaci a interakci s předměty ve virtuální realitě. Součástí analýzy je rozbor a výběr enginu pro implementaci návrhu a analýza uživatelského rozhraní a nástrojů pro manipulaci a interakci s objekty v aktuálních aplikacích pro virtuální realitu. Návrh a následná implementace zahrnuje vytvoření základních řešení pro UI (uživatelské rozhraní) ve virtuální realitě na základě výsledků analýzy. Implementace uživatelského rozhraní obsahuje i systém pro interakci s prvky tohoto UI a rozsáhlé možnosti přizpůsobení. Součástí je i návrh a implementace zmíněných nástrojů, včetně správy a možnosti výběru aktuálního nástroje, jejichž návrh je úzce propojen s implementací uživatelského rozhraní. Nástroje jsou navrženy na základě využití pro manipulaci s historickými artefakty v aplikacích určených pro historiky a zaměřují se na manipulaci s objekty, změnu jejich vlastností a získávání informací o těchto objektech. Celá implementace byla vytvářena za účelem následného využití vývojáři aplikací pro virtuální realitu, pro které je veškerý implementovaný obsah, včetně dokumentace a videonávodu k použití zdarma dostupný na cílové platformě. Celý projekt byl tedy vytvářen jako šablona, kterou je možné libovolně upravovat a skládat prvky UI bez vlivu na funkčnost celého systému.

Klíčová slova Virtuální realita, Unity, Uživatelské rozhraní, Manipulace a interakce s objekty

Abstract

This master's thesis is focused on analysis, design and implementation of basic user interface components and also on tools used for object manipulation and interaction in virtual reality. Part of the analysis section is studying and selecting proper engine, that will be used for this implementation. Also analysis of user interface and tools for object manipulation and interaction in current VR applications is present. The design and following implementation contains creating simple solutions for UI (user interface) in virtual reality based on analysis results. The implementation of user interface also contains a system for interaction with those UI elements and huge customization possibilities. Part of this is design and implementation of mentioned tools including management and selection of specific tool. This systems design is loose coupled with implementation of user interface. Tools are designed for manipulation with historical artefacts for usage in applications that are targeted for historians and these tools are focused on manipulation with objects, changing their attributes and getting information about these objects. Whole implementation was developed for future usage by other developers of virtual reality applications. They can use whole implementation with documentation and instructional video for free on target platform. Whole project was developed as a template, that can be edited and UI elements can be composed without impact on functionality of the whole system.

Keywords Virtual reality, Unity, User interface, Manipulation and interaction with objects

Obsah

Seznam obrázků

Úvod

V posledních letech se vývoj virtuální reality posunul o velký kus kupředu a díky moderním technologiím je již možné díky headsetům pro virtuální realitu navodit opravdový pocit přesunu do jiné reality. Na trhu lze nalézt desítky různých headsetů pro virtuální realitu a v tomto odvětví pracují hlavně velké technologické firmy jako Sony, HTC, nebo Microsoft. Jak lze vyčíst z článku [3] obsahujícího graf počtu připojených headsetů pro virtuální realitu k platformě Steam. V dnešní době se počet aktivně připojených headsetů blíží ke dvěma milionům, narozdíl od necelých 200 000 z roku 2016. Tato statistika zahrnuje pouze zařízení přojené k platformě Steam, která běží výhradně na osobních počítačích a neobsahuje tak údaje o headsetech pro konzolové uživatele. Přestože je téma virtuální reality zpracováváno již delší dobu, jak popisuje publikace [4] z roku 2003, až v poslední době jsou headsety pro virtuální realitu dostupnější a má proto smysl se tomuto tématu věnovat a nadále podporovat jeho rozvoj. Možnosti využití virtuální reality jsou obrovské a již dnes se kromě zábavního průmyslu využívá například ve zdravotnictví či v armádě. Díky virtuální realitě můžeme navštěvovat místa, kam bychom se sami dostat nemohli, ať už se jedná o reálné místo, nebo uměle vytvořený svět. Zároveň pomocí nástrojů ve VR můžeme pracovat s virtuálními modely předmětů tak, jak by to v realitě či v desktopových aplikacích nebylo možné. Otevírají se nám tak další nové možnosti využití. V každé aplikaci jak desktopové, tak pro virtuální realitu je potřebné mít dobré uživatelské rozhraní, abychom mohli s virtuálním světem a objekty, které se vněm nachází, snadno interagovat. V mé diplomové práci se zabývám právě tímto tématem. Cílem práce je výzkum, návrh a implementace možností prvků uživatelského rozhraní pro virtuální realitu. Od analýzy řešení v aktuálních aplikacích, po návrh a implemetaci jednotlivých prvků uživatelského rozhraní a nástrojů pro práci s předměty ve virtuální realitě. Nástroje budou vyvíjeny se zaměřením pro práci historiků s historickými artefakty. Výsledkem bude balíček s implementovanými funkcemi pro platformu Unity, který bude jednoduchý pro

použití a pomůže tak vývojářům při tvorbě vlastních aplikací pro VR či dalším, kteří se tímto tématem budou dále zabývat. Součástí balíčku bude připravené základní uživatelské rozhraní skládající se z několika možných zobrazení UI elementů a systém pro interakci s těmito elementy pomocí ovladačů, které jsou součásí hardwaru pro virtuální realitu. Dalším prvkem balíčku bude také připravený systém nástrojů obsahující všechny implementované nástroje včetně správy a možností přepínání mezi nimi. Součástí bude také dokumentace a návod pro nastavení a použití výsledného balíčku. Ten bude poté publikován a zdarma dostupný všem, kteří by ho chtěli ve své aplikaci použít.

Motivací k výběru tohoto tématu, byla možnost práce s novou a přelomovou technologií, kterou virtuální realita je. Díky výslednému balíčku funkcí práce pomůže začínajícím vývojářům v tomto odvětví a zpřístupní tak vývoj pro VR širší skupině lidí. Zároveň již mám předešlé zkušenosti s vývojem na platformách určených pro tvorbu 3D aplikací či her.

Kapitola 1

Cíl práce

Cílem práce je navrhnout a vytvořit základní a jednoduše použitelné prvky uživatelského rozhraní pro virtuální realitu. Součástí budou a jsou i nástroje pro manipulaci, transformaci a získávání informací o objektech ve VR. Nástroje budou zaměřené na využití historiky pro práci s historickými artefakty ve VR a na uživatele, kteří budou s objekty ve virtuální realitě manipulovat.

Práce zahrnuje analýzu aktuálních řešení UI v aplikacích a hrách pro virtuální realitu. Cílem analýzy je získání informací o různých možnostech řešení prvků UI jako je např. hlavní či herní menu, inventář, výběr nástrojů atd. Součástí analýzy budou i nástroje pro manipulaci a interakci s objekty ve VR, popsání jejich výhod či nevýhod a určení podmínek, za kterých je jejich použití nejvhodnější. Tyto informace budou dále použity ve fázi návrhu.

Cílem návrhové části je vybrat a navrhnout několik způsobů řešení uživatelského rozhraní ve VR na základě informací získaných z analýzy. Bude navrženo několik možností řešení umístění uživatelského rozhraní ve scéně a aplikace pro různé podmínky použití. Součástí návrhu budou i zmíněné nástroje pro práci s předměty, jejichž součástí budou nástroje pro manipulaci a úpravu vlastností objektů a pro získávání informací o daných objektech.

Implementační část vychází z návrhu. Během ní se vytvoří prvky UI, nástroje i samotné testovací prostředí, ve kterém se poté budou prvky testovat. Pro všechny prvky UI a nástroje bude vytvořena scéna, ve které se na připravených úkolech budou implementované funkcionality testovat.

Při testování uživatelského rozhraní budeme kromě chyb zkoumat jeho přehlednost, náročnost použití a situace, ve kterých bude použití řešení nejvhodnější. Testování UI bude probíhat formou sady úkolů v připraveném prostředí, následným vyhodnocením průběhu testu a zpětné vazby od testerů. Testy budou provádět uživatélé s širokou škálou různých technických znalostí. Nástroje budou také součástí testování a budeme sledovat, jak uživatelé nástroje používají a podle reakcí upravovat jejich použití. Pomocí spolupráce s Bc. Matějem Sedlákem, který pracuje na možnostech pohybu a manipulaci s objekty ve VR, se zaměřením na historické budovy, bude možné testovat

obě implementace dohromady, což usnadní testování a umožní řešit problém propojení všech potřebných funkcí pro virtuální realitu v jedné scéně.

Po vyhodnocení testování budou na základě výsledků nástroje a prvky UI upraveny do finální podoby. Výsledkem bude samostatný balíček implementovaných funkcí pro cílovou platformu. Ten bude připravený pro vývojáře, kteří budou tyto prvky moci použít ve svých aplikacích a dále s nimi pracovat. Výsledný balíček funkcionalit bude jednoduchý na použití a díky tomu přístupný i pro programátory začínající s vývojem aplikací ve VR.

Kapitola 2

Analýza

2.1 Analýza platformy

V této sekci se budu zabývat analýzou platformy použité při diplomové práci. Nejdříve volbou enginu a popisu jeho funkcí a dále VR headsetu, který bude využit při vývoji a na testování.

2.1.1 Engine

Enginů pro tvorbu aplikací či her pro virtuální realitu existuje více. Každý má své výhody či nevýhody a liší se i cenou. Pro účely diplomové práce jsme vybírali hlavně mezi Unity a Unreal Engine, které jsou ideální svojí dostupností i nabídkou funkcí a knihoven (např. SteamVR). Při analýze a výběru platformy jsem vycházel z informací a uživatelských zkušeností popsaných na webu [5] a zároveň i ze zkušeností vlastních.

- Unity – jednou z nejrozšířenějších platforem pro nezávislé vývojáře je Unity. Hlavní výhodou je dostupnost a díky velké komunitě a dobré dokumentaci jsou začátky při vývoji jednodušší. Zároveň existuje několik publikací o vývoji aplikací a her, jako například [6] a některé se zaměřují přímo na vývoj aplikací pro virtuální realitu [7]. Unity obsahuje Assetstore, ve kterém mohou uživatelé sdílet či prodávat své modely, scripty a funkcionality. Je tak možné se jednoduše dostat k objektům, které do své aplikace potřebujete. Další výhodou je vcelku jednoduché a přehledné použití, které zároveň obsahuje velmi komplexní možnosti práce. Unity dovoluje práci na multiplatformních aplikacích a je možné tak vyvíjet aplikaci například pro PC, Mac a Android zároveň. Unity nabízí více cenových plánů. Základní je zdarma a je možné ho využít pokud příjmy z vaší aplikace nepřesáhnou 100 000 dolarů během jednoho roku. Nabízí základní funkce, které jsou však dostatečné pro vývoj v menším týmu a tak ideální pro začínající vývojáře. Zároveň nabízí více možností pro studenty a podporuje tak vývoj této platformy v edukativních sférách.

V Unity je možné psát kód ve třech programovacích jazycích, přesněji v C#, JavaScriptu či Boo.

- Unreal Engine – dalším z často používaných enginů je Unreal Engine. Jedná se o dobře fungující engine, který nabízí spoustu možností práce s 3D prostředím. Grafické možnosti Unreal Enginu jsou úžasné, jak lze vidět například ve videu [8]. Jsou zaměřeny hlavně na graficky náročné scény. Unreal Engine obsahuje také marketplace, ve kterém je možné získat či kupovat výtvory jiných vývojářů. Práce v Unreal Enginu je složitější než v ostatních Enginech, hlavně díky své komplexnosti. Oproti Unity nemá tak Unreal Engine silnou komunitu a podporuje pouze programovací jazyk C++. V Unreal Engine je taktéž možné vyvíjet multiplatformní aplikace. Pro nekomerční použití je engine dostupný zcela zdarma a nabízí více funkcí pro spolupráci na projektu než Unity. Při publikování aplikací a her si Unreal Engine účtuje 5 % poplatků, pokud vaše tržby za poslední čtvrtletí přesáhly 3 000 dolarů. Hodí se tak spíše pro větší studia či početnější týmy.

- Godot – za zmínku stojí i zajímavý projekt se jménem Godot [9], který funguje jako opensource engine pro tvorbu her a je tak možné jednoduše vytvářet vlastní úpravy samotného enginu. Engine se zaměřuje na minimalistickou barevnou grafiku a jednoduchost při vývoji. Engine podporuje základní programovací jazyky jako je C# a C++, včetně vlastního jazyku s názvem GDScript vytvořeného výhradně pro tuto platformu. Godot obsahuje také vlastní knihovnu pro sdílení assetů od komunity a vývojářů, který však není tak rozsáhlý jako u ostatních enginů, avšak všechny dostupné assety jsou k dispozici zdarma. Podpora vývoje pro virtuální realitu zde probíhá pomocí OpenVR modulu, který je dostupný ve zmíněné knihovně s assety. Jelikož se jedná o opensource engine, je dostupný zcela zdarma i ke komerčnímu využití.

Pro účely diplomové práce jsme vybrali (s kolegou Bc. Matějem Sedlákem) vývoj v engine Unity, hlavně díky velké komunitě a dobré dokumentaci. Unity engine nabízí veškeré potřebné funkcionality pro implementaci a aktivní podporu ze strany vývojářů i komunity. Zároveň díky větší popularitě, hlavně pro nezávislé vývojáře, se zde najde lepší využití pro výsledný balíček funkcí a případnou návaznost na tuto diplomovou práci. S Unity máme také větší zkušenosti než s ostatními enginy a vývoj tak bude jednodušší.

2.1.2 SteamVR

SteamVR [10] od společnosti Valve je dostupný díky aplikaci Steam, díky které je možné nakupovat aplikace, hry a komunikovat s komunitou. SteamVR slouží k propojení aplikace a headsetu pro virtuální realitu. Obsahuje podporu pro mnoho VR headsetů (HTC Vive, Oculus Rift, Windows Mixed Reality, atd.)

a velmi tak usnadňuje vývoj samotných aplikací pro virtuální realitu. Pro vybraný engine Unity je dostupný SteamVR balíček na Assetstoru, který toto propojení umožňuje integrovat do vyvíjené aplikace či hry. Balíček obsahuje základní funkce potřebné pro pohyb hlavou či zobrazení ovladačů v aplikaci. Obsahuje i základní možnost pohybu pomocí teleportu a základní možnosti úchopu a pohybu s předměty. Z tohoto balíčku použiji základní funkcionality a prvky primárně pro pozdější testování UI a manipulace s předměty. SteamVR bohužel neobsahuje kvalitní dokumentaci a je tedy nutné čerpat z jiných zdrojů, buď v návodech na internetu či publikacích, jako je například [11].

2.1.3 VR Headset

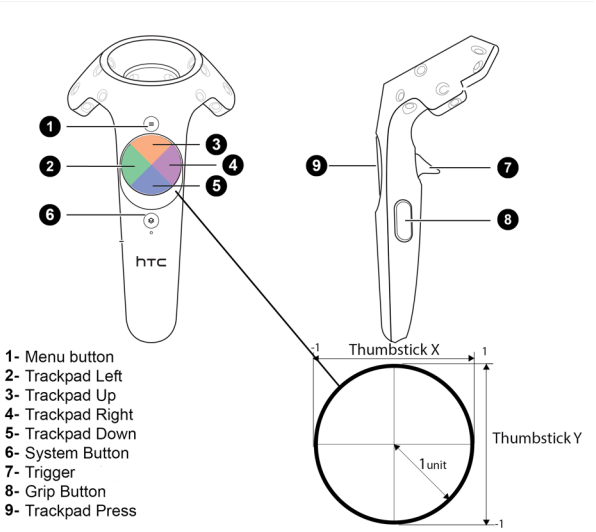
Headset pro virtuální realitu je samotný hardware zajišťující zobrazení a ovládání. VR headsetů je na trhu spousta a hlavními výrobci jsou velké firmy jako HTC, Microsoft, nebo Sony. Cena se díky technologii a konkurenci stále snižuje a virtuální realita se tak stává stále dostupnější.

Pro účely testování a vývoje UI a manipulace s předměty bude použita sada pro virtuální realitu s názvem HTC Vive [12], jelikož byl tento headset k zapůjčení na fakultě. Součástí sady je náhlavní souprava, která je s počítačem propojena pomocí dvojice kabelů (HDMI a USB) zajišťující přenos obrazu a ovládání. Náhlavní souprava zajišťuje zobrazení virtuální reality pomocí dvou displayů s rozlišením 1080 x 1200 pixelů a obnovovací frekvencí 90 Hz. To zajišťuje dostatečnou kvalitu obrazu a obnovovací frekvence, aby se dalo zařízení používat i delší dobu bez problémů. Ovládání zajišťují dva bezdrátové ovladače se čtyřmi typy ovládacích prvků. Snímání pohybu v prostoru zajišťují dvě snímací stanice, které jsou připevněny naproti sobě v rozích snímané oblasti. Tyto stanice emitují do snímané oblasti paprsky, které ovladače a náhlavní souprava snímají pomocí mnoha světelných senzorů a díky tomu je určena jejich pozice a rotace v prostoru, jak je posáno ve videu [13]. Všechny části headsetu jsou zachyceny na obrázku 2.1.

Obrázek 2.1: Součásti VR headsetu HTC Vive [1].

2.1.4 Ovladače HTC Vive

Ovladače headsetu HTC Vive obsahují několik tlačítek, na jejichž stisknutí a další akce lze reagovat ve vyvíjené aplikaci. Na přední straně je v horní části ovladače menu tlačítko, které se již podle názvu primárně využívá pro zobrazení menu či nastavení. Pod ním se nachází trackpad obsahující tlačítka na každé straně či ve středu trackpadu. Je možné i detekovat dotyk na trackpadu a pozici pomocí dvou souřadnicových hodnot. Jelikož ovladače pro HTC Vive nedisponují joystickem, na rozdíl od většiny ostatních VR sad, je tento chybějící ovládací prvek simulován právě pomocí trackpadu. Pod ním se nachází systémové tlačítko, které se používá pro otevření obecného VR menu mimo aplikaci, dovolující změnu aplikace či ukončení a nastavení VR. Z druhé strany ovladače se nachází tlačítko spouště a z obou stran ovladače je grip tlačítko, většinou používané pro uchopení předmětů. Přehled tlačítek je zobrazen na obrázku 2.2



Obrázek 2.2: Tlačítka na ovladači HTC Vive [2].

2.2 Desktop aplikace vs VR

Hlavní rozdíl při návrhu uživatelského rozhraní pro VR je pohled hráče, který se téměř pořád mění. Kdyby byly prvky UI pevně připnuté do pohledu uživatele, brání mu ve výhledu a výběr položek, který se provádí pomocí ovladače, by tak nebyl možný. Prvky uživatelského rozhraní tak musí být ukotveny ve scéně a musí být modelovány jako objekty a ne pouze jako obrázky. To je na jednu stranu nevýhoda, ale zároveň to dovoluje zobrazování UI prvků ve 3D. Například lze zobrazit více vrstev UI za sebou, čímž se výsledné uživatelské

rozhraní zpřehlední. Kvůli potřebě vložení prvků UI do scény je potřeba řešit nejen velikost a pozici, ale i vzdálenost od uživatele a rotaci, jak je dobře popsáno v přednášce [14]. Ve VR neexistují žádné zažité standardy pozice prvků uživatelského rozhraní a jejich návrh tak vychází hlavně ze zkušeností, analýzy konkurenčních aplikací a následného testování. Existují však články, které toto téma zpracovávají a nabízejí tipy pro zlepšení pocitu z uživatelského rozhraní ve VR jako například článek [15]. Při testování bude možné využívat metody používáné při tvorbě desktopové aplikace jako například usability testování (testování použitelnosti). Při návrhu UI pro virtuální realitu se nemusí UI přizpůsobovat rozlišení Headsetu. To pouze definuje kvalitu zobrazení, ale velikost světa zůstává pro každý headset stejná. Není tak nutné vytvářet více návrhů pro různé druhy headsetů. Při manipulaci s předměty nám VR technologie přináší spoustu nových možností oproti desktopovým aplikacím. Kromě možnosti zobrazení objektů v plném stereoskopickém 3D je možné při manipulaci pracovat se dvěma ovladači a používat tak obě ruce najednou.

2.3 Analýza Aplikací VR

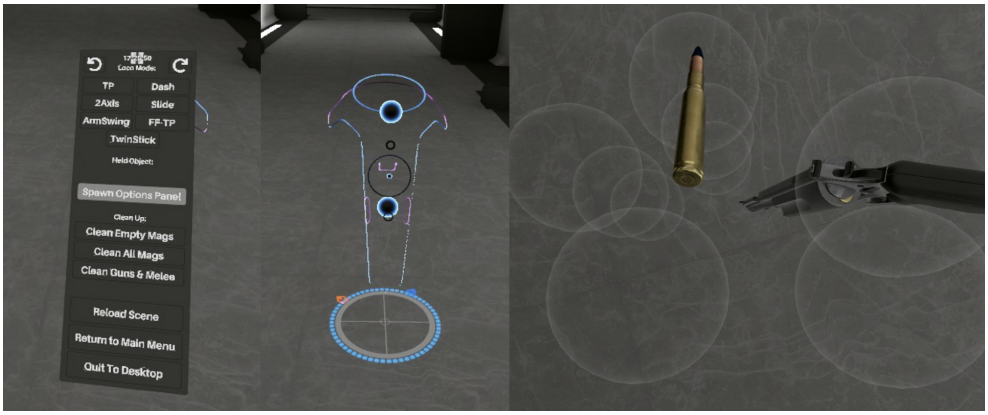
V této sekci se zaměřuji na analýzu vybraných aplikací a her pro virtuální realitu. Z poměrně velké škály her a aplikací, které jsou dostupné pro virtuální realitu na PC, jsem vybral ty, které obsahují pokročilé a zajímavé prvky uživatelského rozhraní a manipulace s předměty.

2.3.1 Hot Dogs, Horseshoes and Hand Grenades

Tato hra [16] je jeden z nejlépe zpracovaných simulátorů používání střelných zbraní pro virtuální realitu se zaměřením na realistické používání zbraní. Ve hře se nachází spousta herních módů. Je možné si zde různé zbraně zkoušet ve virtuálních střelnicích nebo přejít do akčních módů, kde se postavíte počítačem řízeným nepřátelům.

- V této hře je inventář řešený pomocí slotů připevněných na místě okolo pasu hráče. Při pohybu po scéně toto řešení nevadí výhledu a při potřebě interakce s inventářem se stačí podívat dolů a inventář je dostupný. Nevýhodou je malé množství pozic pro uložení objektů. Při přidání více pozic se inventář stává nepřehledným a kromě špatné orientace se zároveň často stává, že se z inventáře vezme jiný předmět než požadujeme. Řešení je vhodné pro malý počet položek, které je potřeba mít rychle dostupné. Tento inventář je zachycen v pravé části obrázku 2.3.

- Herní menu je zajímavě řešeno pomocí boxu s tlačítky připevněného k zadní straně libovolného ovladače. Při normálním používaní není viditelné a zobrazí se pouze pokud ovladač otočíte zády k vám, jak lze vidět v levé části obrázku 2.3. Dobře se v menu orientuje a díky ukotvení



Obrázek 2.3: Hot Dogs – menu, mapa, inventář.

k ruce je možné při špatné čitelnosti textu jen ruku jednoduše přiblížit. Výběr položek se provádí pomocí paprsku jdoucího z druhého ovladače. Při otevření rozšířeného menu se před ovladačem objeví velký box, opět ukotvený k ovladači, na kterém je možné provádět komplexnější nastavení. Toto řešení dovoluje přehledně zpracovat menší herní menu a díky ukotvení k ovladači se v něm lze dobře a jednoduše orientovat.

- Hlavní menu je zde řešeno pomocí samostatné scény, ve které se nachází box s nastavením (stejný jako v rozšířeném herním menu) a krátký návod na ovládání a pohyb. Následně se uživatel vlastní kontrolou pohybu přesune do druhé části scény, kde si vybírá požadovanou herní scénu. Potřeba přesunu může dobře sloužit k testování pohybu popřípadě k výběru nejvhodnějšího. Výběr herní scény se provádí pomocí paprsku jdoucího z ovladače a velkých boxů umístěných po obvodu scény menu. Výběr je potřeba ještě potvrdit pomocí malého displaye umístěného uprostřed scény. Hlavní menu se pomocí samostatné scény dobře odděluje od herních světů a výběr požadovaného módu tak muže být zpracován komplexněji a přehledněji. Zároveň však pro změnu herního módu je potřeba se vrátit do úvodní scény, což zpomaluje plynulost aplikace. Toto řešení je vhodné pokud herní světy měníte pouze zřídka.

- Zajímavým prvkem UI je i minimapa s důležitými body v okolí. Tato mapa se zobrazuje pod levým ovladačem při jeho zvednutí, střední část obrázku 2.3 a v aplikaci je velmi užitečná. Dle mého názoru se jedná o výbornou implementaci pro minimapu.

- Hlavními předměty, se kterými se v této hře interaguje, jsou zbraně. Není však možné měnit jejich vlastnosti (velikost, barvu, atd.). Manipulace probíhá pomocí kliknutí na tlačítko pro uchopení předmětu. To způsobí jeho ukotvení k ovladači na definovanou relativní pozici vůči ovladači.

Tento způsob je vhodný z důvodu použití stejného tlačítka pro uchopení zbraně i pro samotné střílení. Při manipulalci s menšími předměty, např. náboji, je pořeba tlačítko stále držet a při uvolnění tlačítka je objekt puštěn na zem. Je možné pomocí druhého ovladače interagovat s různými částmi zbraně jako je zásobník, pojistka nebo závěr.

2.3.2 Neos VR

Aplikace s názvem Neos VR [17] je virtuální multiplayerový svět, ve kterém hráči tvoří své vlastní světy pomocí umisťování nabízených i vlastních objektů. Je možné navštěvovat světy jiných hráčů a interagovat s nimi. Aplikace obsahuje podobné funkcionality pro manipulaci a interakci s objekty jako ty, které budou součástí této diplomové práce a mohou tak sloužit jako dobrý zdroj pro návrh.

- V této aplikaci se hlavní menu a herní menu spojuje do jednoho pomocí zaobleného panelu 3D boxů ve spodní části výhledu. Díky velikosti a umístění však menu často překáží ve výhledu po scéně, lze vidět ve spodní části obrázku 2.4. Výběr se provádí pomocí paprsku z ovladače a v případě rozšířeného výběru se zobrazí dodatečný seznam možností nad vybraným boxem. Popisky položek v menu nejsou orientovány směrem k hráči a ztěžují tak čitelnost. Spojení herního a hlavního menu je vhodné, pokud chceme často měnit herní světy jako právě v případě této aplikace.

- Inventář je implementován pomocí obrazovky ukotvené ve světě, se kterou je možné i ve scéně manipulovat jako s normálním objektem. Položky či složky jsou představovány pomocí ikon nebo boxů s názvy. Velikost a přizpůsobení textu v boxu jsou velmi nepřehledné, lze pozorovat v levé části obrázku 2.4 a působí tak špatným dojmem. Výběr se provádí opět pomocí paprsku a dvojitým kliknutím na tlačítko na ovladači. Při prvním kliknutí se zobrazí dodatečné informace o složce či objektu v inventáři. Při výběru objektu z inventáře se objekt objeví na vaší pozici, tudíž přímo ve vás. To způsobuje, že musíte ustoupit a poté můžete s předmětem začít manipulovat. Základ implementace je vhodný pro větší počet položek v inventáři, kvůli zpracování je však inventář velmi nepřehledný.

- Výběr nástrojů se provádí pomocí otočného menu kolem ovladače, které se zobrazí po stisknutí příslušného tlačítka. Ve výběru je možné se orientovat pomocí otáčení ruky či pomocí joysticku. Výběr je však špatně zpracovaný a je těžké se orientovat v tom, jakou možnost máte právě vybranou a často je pro výběr potřeba rukou otočit do nepohodlného úhlu. Střed otočného menu není ve středu ovladače, ale mimo něj, což způsobuje, že výběr pomocí otáčení ovladače prakticky nejde využít

a používá se tak pouze výběr pomocí joysticku. Potenciál otočného menu tak ztrácí smysl. Zobrazená nápověda při prvním výběru překrývá text možností a orientaci to tak ještě ztěžuje, jak můžeme vidět v pravé části obrázku 2.4. Menu se používá hlavně pokud zrovna držíte v ruce nějaký objekt. Řešení je originálním nápadem pro rychlý výběr z několika položek. Kvůli neideální implementaci však pozbývá smyslu.

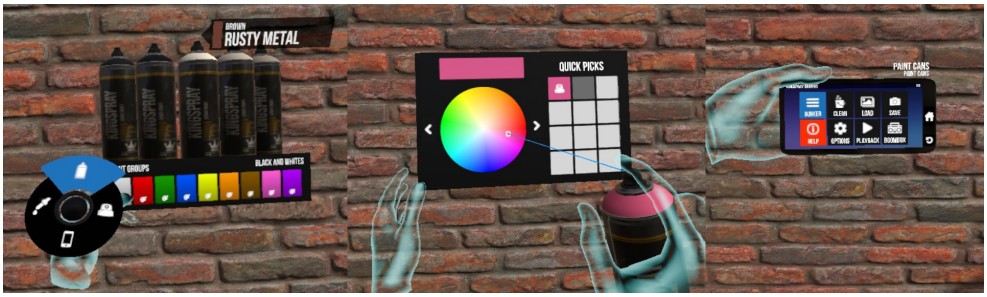
- S objekty lze pohybovat pomocí tlačítka pro uchycení. Objekt lze uchopit přímo do ruky, pokud je na dosah, nebo pomocí paprsku pro předměty, které jsou v dálce. Při uchopení je potřeba tlačítko stále držet. Pokud držíte předmět, je možné ho oddalovat a přibližovat k sobě pomocí joysticku na ovladači směrem nahoru či dolů. Pomocí joysticku směrem do stran je možné měnit rotaci objektu po jedné ose. Po jiných osách je možné s objektem rotovat pouze pomocí uchopení a následného otáčení samotným ovladačem, nebo pomocí speciálních pokročilých nástrojů.

Obrázek 2.4: Neos VR – inventář, otočné menu, hlavní menu.

2.3.3 Kingspray Graffiti VR

Tento výtvarný nástroj [18] pro virtuální realitu slouží k simulaci vytváření graffiti a obrazů pomocí sprejů na zdi v simulovaných prostředích. Práce a finální výtvory působí velmi reálně a jedná se tak o velmi dobře zpracovanou aplikaci pro VR. Aplikace nabízí velké množství barev a typů trysek na spreje a zajímavé prvky uživatelského rozhraní.

- V této aplikaci je hlavním prvkem UI výběr nástrojů, konkrétně sprejů, který je zobrazený v pravé části obrázku 2.5. Celkové ovádání UI je asymetrické (každý ovladač nabízí jiné funkce) a díky tomu, nabízí přehledně spoustu možností. Výběr je rozdělen do čtyř sekcí vybraných pomocí čtyř směrů joysticku levého ovladače. Po výběru sekce se zobrazí rozšířený výběr vedle ovladače. Většina UI je zpracována pomocí 3D objektů, což dobře prohlubuje pocit z virtuálního prostředí. Výběr se provádí pomocí paprsku jdoucího z druhého ovladače. Směr paprsku je odlišný a nemíří rovně ve směru ovladače (směr kam ukazuje palec), ale kolmo k ovladači (směr ukazováčku) což u objektu, který je blízko uživatele usnadňuje výběr, jelikož není nutné s rukou otáčet do nepřirozených úhlů.
- Důležitým aspektem aplikace je výběr barvy. Základní výběr se skládá z výběru jedné ze základních devíti barev a následného výběru z několika odstínů. Pro základní výběr je to dostačující a přehledná možnost výběru. Aplikace nabízí i výběr pomocí klasického kola barev, střední část obrázku 2.5 a následného uložení vybrané barvy. Dle mého názoru je ve VR kolo barev nejlepší a nejjednodušší možností pro výběr požadované vlastní barvy.
- Herní menu, obsahující nastavení a jednoduché nástroje, je reprezentováno formou jednoduché obrazovky představující mobilní telefon. Jedná se o jednoduchou implementaci UI, kde díky dobrému zasazení do modelu telefonu se stává velmi intuitivní pro ovládání, jak lze vidět v pravé části obrázku 2.5.



Obrázek 2.5: Kingspray Graffiti – menu, výběr barvy, nastavení.

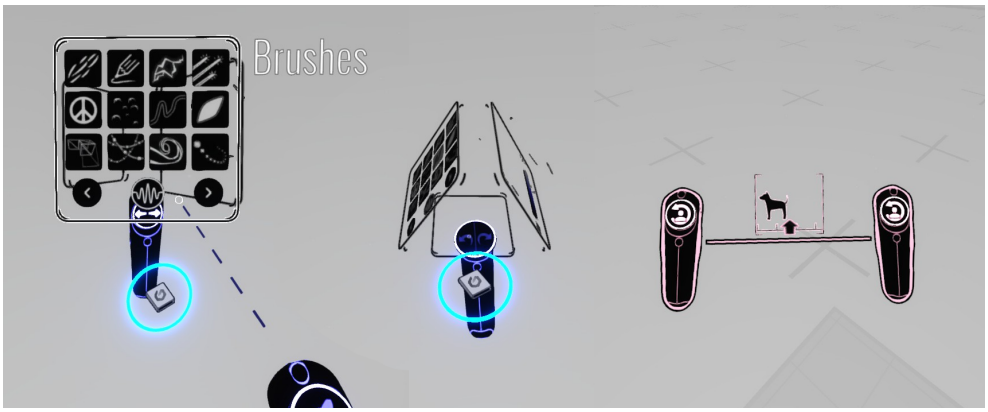
2.3.4 Tilt Brush

Tilt Brush [19] nástroj pro virtuální realitu dovolující malování ve 3D. Aplikace má originální nápad a dají se v ní tvořit opravdu zajímavé výtvory.

- Ovládání v této aplikaci je opět asymetrické. Na levém ovladači se nachází tři panely dokola kolem ovladače viditelné na levé části obrázku

2.6. První panel slouží pro výběr barvy, opět pomocí kola barev. Druhý obsahuje nastavení a výběr nástrojů (guma, teleport, atd.). Na posledním panelu se nachází výběr stylu kreslení. Otáčení panelu se provádí pomocí joysticku na ovladači. UI dobře kombinuje prvky 2D a 3D rozhraní. Je velmi přehledné a dobře se ovládá. Jedná se o zajímavou implementaci herního menu a ovládacích prvků.

- Originálním prvkem ovládání je nástroj pro změnu měřítka scény. Tento nástroj se aktivuje držením grip tlačítka na obou ovladačích. Po aktivaci se zobrazí čára spojující oba ovladače, na které se uprostřed zobrazuje aktuální hodnota měřítka, jak můžeme vidět v pravé části obrázku 2.6. Hodnota se poté nastavuje pomocí oddalování či přibližování ovladačů k sobě a jedná se tak o zajímavě řešené nastavování hodnoty.



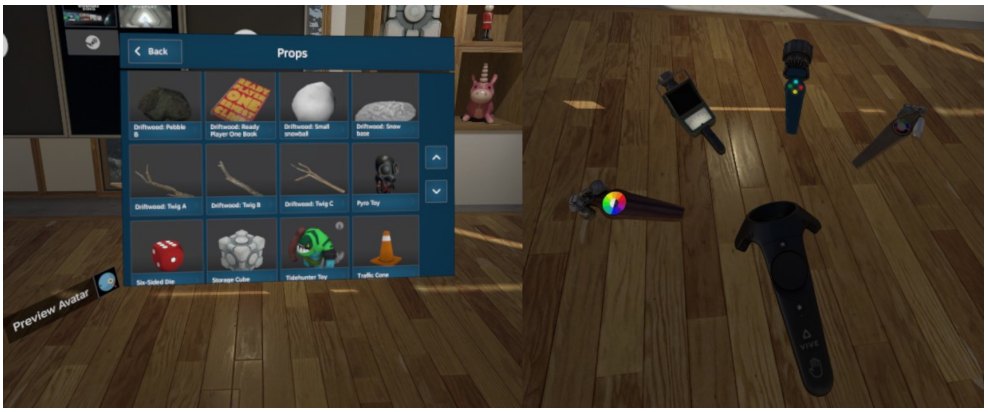
Obrázek 2.6: Tilt Brush – menu a nástroj na nastavení měřítka.

2.3.5 SteamVR Home

Poslední analyzovanou aplikací, která obsahuje zajímavé prvky UI je výchozí aplikace při používání SteamVR [10]. Tato aplikace slouží primárně jako lobby pro výběr další hry či aplikace, komunikace s přáteli, či procházení dostupných VR aplikací skrze Steam. I tak je možné v této aplikaci interagovat s prostředím a obsahuje některé zajímavé prvky UI a manipulace s předměty.

- Inventář, který se zobrazí po stisknutí menu tlačítka na ovladači, je součástí panelu obsahujícího různé informace a nastavení. Systém inventáře je podobný jako v aplikaci Neos VR, avšak je graficky lépe zpracovaný a přehlednější, jak lze vidět v levé části obrázku 2.7. Název objektu v inventáři je dobře čitelný a nekombinuje různě velká písma u různých objektů. Po výběru požadovaného objektu se objekt objeví před uživatelem a je možné s ním jednoduše manipulovat.

- Součástí aplikace jsou i užitečné nástroje vytvořené pro tvorbu prostředí formou umisťování objektů po scéně. Výběr nástrojů probíhá pomocí držení grip tlačítka a přesunutí ovladače na jeden z nástrojů, jak lze vidět v pravé části obrázku 2.7. Výběr obsahuje nástroje pro změnu barvy objektů, mazání či klonování. Je také možné objektům zapínat či vypínat gravitaci, což umožňuje objekty ukotvit v prostoru, nebo naopak s nimi pracovat jako v reálném světě.
- Manipulace s předměty je v této aplikaci řešena pomocí uchopování předmětů do ruky a následnými pohyby ovladači. Je tak možné měnit rotaci a pozici předmětů. Změna velikosti se provádí zajímavým způsobem, který vyžaduje použití obou ovladačů. Pomocí obou ovladačů uchopíme stejný předmět a jejich oddalováním či přibližováním měníme velikost předmětu.



Obrázek 2.7: SteamVR Home – inventář a výběr nástrojů.

Kapitola 3

Návrh

3.1 Návrh uživatelského rozhraní

Hlavními parametry při tvorbě uživatelského rozhraní je jednoduchá použitelnost pro uživatele, kteří budou uživatelské rozhraní ve svých aplikacích používat a zároveň dostatečná možnost úpravy. Uživatelské rozhraní se bude skládat z containerů, které budou představovat obrazovky vložené do herního světa a samotných UI elementů, ze kterých si uživatel vytvoří rozhraní, které potřebuje. Veškeré elementy budou vycházet ze základních elementů používaných při tvorbě klasického UI v Unity, a proto bude pro vývojáře jednoduché tyto prvky používat i při implementaci UI do virtuální reality. Součástí kompletního uživatelského prostředí bude několik containerů, základní spektrum elementů, včetně pole pro psaní textu, jehož součástí je i klávesnice a systém pro interakci s těmito elementy.

3.2 Interakce s UI

O interakci s prvky UI se bude starat samostatná třída, která bude řešit nastavení paprsku a komunikaci se samostatnými UI elementy. Interakce bude moci být prováděna pomocí jakékoliv akce na ovladači, která vytváří hodnoty typu boolean (např. tlačítko) s možností přidání dvojitého stisknutí. Dle analýzy vychází jako nejvhodnější a nejpoužívanější tlačítko spouště, které bude nastaveno jako výchozí akce. Je možné si také upravit vzhled a maximální dosah paprsku, který bude využíván pro interakci. Paprsek bude vykreslován pomocí dvoubodové čáry vytvořené pomocí komponenty *LineRenderer*, jejíž body a zásahy objektů budou počítány pomocí třídy *Raycast*. Uživatel si bude moci vzhled samotých paprsků jednoduše upravit, ale základní nastavení bude pracovat s paprsky různých barev pro každou ruku, aby byl zdroj těchto paprsků jednoduše identifikovatelný. Paprsek bude možné zapnout či vypnout a zároveň nastavovat jeho směr. Tyto vlastnosti bude možné měnit za běhu a jejich hodnoty budou záviset na právě zobrazeném containeru.

3.3 Containery

Součástí návrhu je několik typů containerů, které mohou být jednoduše použity a upravovány uživatelem. Veškeré úpravy vzhledu a velikosti obrazovek obsažených v containerech nebudou mít vliv na funkčnost a strukturu daného containeru.

3.3.1 Zobrazení containerů

Zobrazování containerů bude řešeno pomocí samostatné třídy, která zobrazování bude spravovat. Každý container bude po spuštění aplikace schován a bude ho možné zobrazit pomocí zmáčknutí příslušného tlačítka. Bude možné si akci pro zobrazení nastavit na jakoukoliv akci ovladače, která vytváří hodnotu boolean (např. tlačítko). Bude možné si určit typ reakce na tuto akci, neboli jestli se má menu otevřít již při zmáčknutí či až při uvolnění tlačítka. Bude možné buď použít jednu akci, která při prvním stisknutí container zobrazí a při druhém ho opět schová, nebo rozdělit zobrazení a schování na 2 odlišné akce, čímž je možné nastavit zobrazení na zmáčknutí tlačítka a schování na jeho uvolnění.

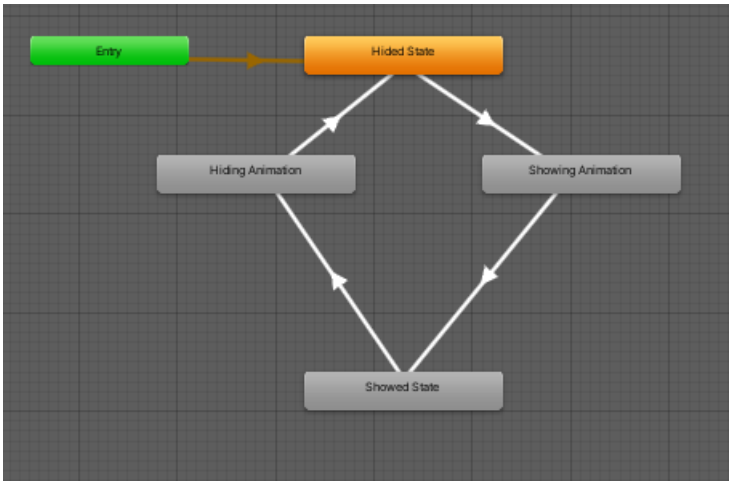
Při zobrazení containeru bude zobrazen paprsek pro interakci s UI elementy v daném containeru. Bude možné nastavit na jaké ruce má být paprsek zobrazen, neboli jestli se bude s UI interagovat pomocí pravé, levé či oběma rukama. Bude možné si také zvolit směr tohoto paprsku ze dvou variant. Paprsek směrem dopředu se hodí pro interakci s většími containery zobrazenými před uživatelem či containery napevno umístěnými v prostoru. Pro UI zobrazené na těle uživatele, například na ruce, je lepší paprsek, který bude směřovat kolmo k ovladači, jelikož UI elementy budou blízko uživatele a uživatel tak bude moci potvrzovat výběr pohodlněji. Po zavření containeru bude stav paprsků uveden do původního stavu před otevřením, tedy pokud byly paprsky již zobrazené, nedojde k jejich vypnutí.

3.3.2 Animace

Životní cyklus UI containerů bude řízen pomocí komponenty *Animator*. Ta při spuštění aplikace převede objekt do schovaného stavu a pomocí metod pro zobrazení a schování objektu bude přecházet mezi těmito stavy pomocí vybrané animace. Základní animací bude zmenšování a zvětšování mezi normální velikostí objektu a nulovou velikostí. To zajistí plynulý přechod mezi zobrazeným a schovaným containerem.

3.3.3 Malé herní menu

Prvním navženým containerem je otočné menu zobrazené okolo ovladače. Bude možné vybrat počet obrazovek, které chceme použít a podle jejich počtu se menu přizpůsobí do objektu obklopujícího celý ovladač. Maximálním počet

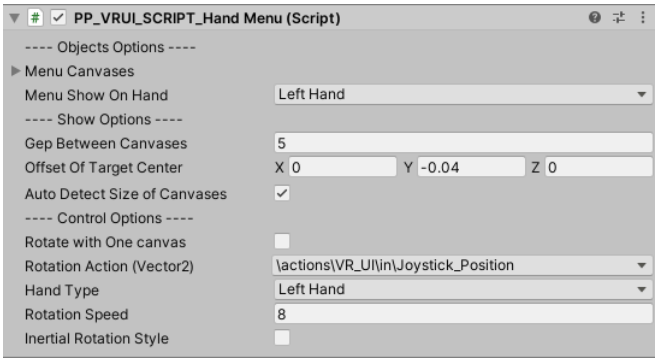


Obrázek 3.1: Přehled životního cyklu UI containerů.

obrazovek je 5, jelikož při použití většího počtu se menu stává nepřehledné a ztrací tak na efektivnosti. Menu se zároveň bude přizpůsobovat velikosti obrazovek, které si může uživatel upravit podle sebe a změnit tak velikost celého menu. Je možné velikost obrazovek detekovat automaticky, nebo zadat ručně, například pokud obrazovka bude obsahovat 3D model, který bude představovat okraje obrazovky a nebude tak možné přesně detekovat výslednou velikost obrazovky. Bude možné si také nastavit mezeru mezi obrazovkami a přizpůsobit si tak vzhled menu své potřebě. Další možností úpravy je nastavení odsazení středu menu. Pokud bude uživatel používat vlastní model pro ovladače (např. rukavice) bude potřeba menu vycentrovat, aby nezasahovalo do tohoto modelu a otáčení působilo co nejlépe. Otáčení obrazovek se bude provádět pomocí jakékoliv akce ovladače, která vytváří hodnoty typu *Vector2* (bod ve 2D prostoru), ale základní použití bude pomocí joysticku na ovladači a menu tak bude rotovat kolem ovladače, aby byly zpřístupněny i obrazovky umístěné na zadní straně ovladače bez potřeby otáčet ovladač o 180 stupňů. Bude možné použít dva druhy rotování celým menu. Pomocí první možnosti se bude menu otáčet při pohybu joysticku na ovladači a podle vzdálenosti od středu joysticku bude vypočítána rychlost a směr otáčení. Druhou možností bude otáčení pomocí pohybu prstem po trackpadu, kde rychlost otáčení bude záviset na rychlosti pohybu po trackpadu. Požadovanou možnost otáčení bude možné vybrat při implementaci menu do projektu. Bude možné si i přizpůsobit rychlost otáčení.

Editace menu probíhá v herní scéně, kde je menu ukotveno. Práce s ním tak probíhají stejně jako s klasickým UI a pro uživatele jsou tak úpravy jednoduché. Po spuštění aplikace se v závislosti na počtu a velikosti obrazovek vypočítají pozice a rotace jednotlivých obrazovek ve 3D světě a vše se tak inicializuje do výsledné podoby.

Tento druh herního menu může dobře sloužit v aplikacích, kde chceme v herním menu vybírat položky, aniž by se scéna pozastavovala či zakrývala. Nevytrhává nás tedy ze zážitku a slouží jako aktivní a rychlý prvek UI. Součástí tohoto containeru může být i malý inventář, který tak bude rychle dostupný.



Obrázek 3.2: Navržené veřejné proměnné pro malé herní menu.

3.3.4 Velké herní menu

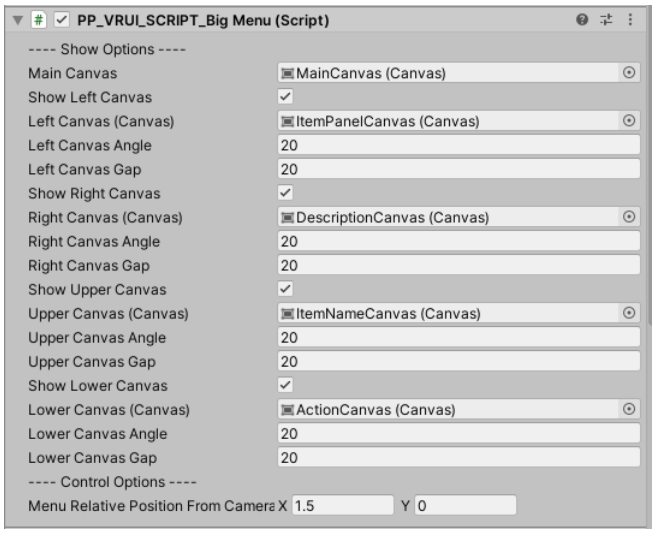
Druhým návrhem herního menu je složení několika obrazovek, které se po stisknutí příslušného tlačítka, nebo vyvoláním pomocí jiné akce zobrazí před uživatelem ve směru, kterým se uživatel právě dívá v předem definované vzdálenosti, kterou si uživatel může nastavit. Obrazovka bude ukotvená ve scéně a nebude se tak pohybovat s uživatelem.

Menu se bude skládat z pěti oddělených obrazovek. Hlavní obrazovka bude zobrazena přímo před uživatelem a z každé ze čtyř stran bude připojena další přidaná obrazovka. Bude možné libovolně měnit velkost obrazovek a přizpůsobovat si tak velikost menu vlastním potřebám. U přídavných obrazovek bude možné si nastavit úhel náklonu ve vztahu k hlavní obrazovce, aby obklopily uživateli výhled a byly tak lépe čitelné. Podle nastaveného náklonu a velikosti obrazovek bude potřeba vypočítat přesnou pozici obrazovky, aby k hlavní obrazovce přesně přilnuly svojí hranou ve všech osách 3D prostoru. Bude možné si u každé z vedlejších obrazovek nastavit mezeru a přizpůsobit si tak strukturu obrazovek ještě lépe.

Editace menu probíhá v herní scéně, kde je menu ukotveno. Práce s ním tak probíhají stejně jako s klasickým UI a pro uživatele jsou tak úpravy jednoduché. Po spuštění aplikace se v závislosti na velikosti obrazovek a nastavených úhlech náklonu vypočítají pozice a rotace jednotlivých obrazovek ve 3D světě a vše se tak inicializuje do výsledné podoby.

Tento návrh herního menu je vhodný pro rozsáhlejší nastavení, jelikož z části zakrývá výhled po scéně, ale zároveň může nabízet větší prostor a kom-

plexnější možnosti. Vhodný je například jako hlavní menu, které slouží pro výběr scény a obsahující větší objem informací.



Obrázek 3.3: Navržené veřejné proměnné pro velké herní menu.

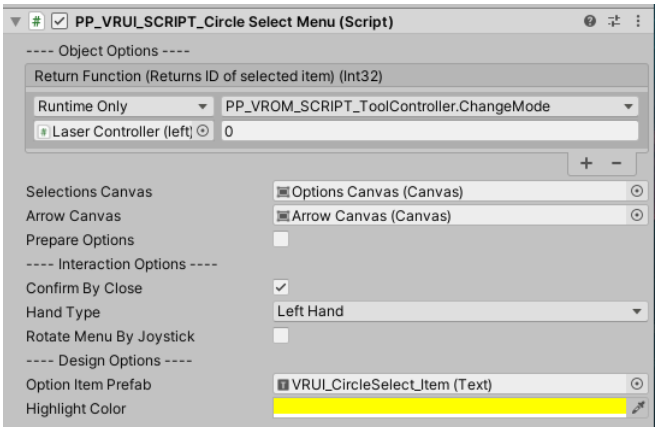
3.3.5 Kruhové Menu

Toto menu slouží pro rychlý výběr z několika položek. Položky se zobrazí do kruhu kolem ovladače uživatele a pomocí otáčení ovladače a akce, která vrací jako hodnotu *Vector2* (např. joystick) je možné vybrat požadovanou položku. Menu bude složeno ze dvou obrazovek. První bude obsahovat položky, ze kterých si uživatel vybírá a druhá bude obsahovat šipku, určující právě zvolenou možnost. Tyto obrazovky se budou překrývat a pomocí otáčení ovladače se bude pohybovat pouze s obrazovkou obsahující šipku určující zvolenou možnost. Pomocí zvolené akce bude možné otáčet i obrazovkou obsahující položky výběru, což usnadní uživateli oriantaci v menu bez potřeby otáčení ruky o více než 90 stupňů na každou stranu. Položky v menu budou vždy rovnoběžně s podlahou a díky tomu při otáčení budou stále dobře čitelné. Po potvrzení výběru bude podle vzájemné rotace obou obrazovek vypočítána zvolena možnost.

Jednotlivé možnosti obsahují název a ikonu definující danou možnost. Jednotlivé položky se budou vytvářet při startu aplikace z připraveného objektu (prefabu), který je možné upravit a změnit tak vzhled samotných položek. Podle velikosti daného objektu představujícího položku ve výběru a počtu položek bude vypočítána pozice na obrazovce tak, aby při pohybu s obrazovkou do sebe názvy položek nezasahovaly a byly tak dobře čitelné ve všech polohách. Kvůli dobré čitelnosti a přehlednosti a zároveň omezené velikosti menu bude maximální počet položek nastaven na 8. Položky v menu bude možné

připravit předem, nebo předat jako parametr až při otevírání samostatného menu. To bude umožňovat použití jednoho menu pro výběr z různých skupin možností bez pořeby pro každou skupinu přidávat nové menu.

Po výběru dané položky bude vyvolána předem definovaná událost či události s hodnotou představující ID vybrané možnosti. Zobrazení menu bude probíhat pomocí metod pro otevření a zavření menu. Povrzení výběru zvolené položky bude probíhat buď při uzavření menu, nebo samostatnou Boolean akcí ovladače. Toto menu je ideální pro rychlý výběr z menšího počtu položek, jelikož je přehledné a rychle se v něm orientuje. Toto menu bude použito pro výběr nástrojů pro manupilaci a interakci s předměty v další části této diplomové práce.



Obrázek 3.4: Navržené veřejné proměnné pro kruhové menu.

3.3.6 Inventář

Inventář bude vytvořen pro použití ve více možných containerech a bude tak obsahovat dvě výsledné podoby. o správu inventáře se bude starat samostatná třída, která bude nezávislá na použití daného containeru. Součástí inventáře bude více UI prvků, které bude možné poskládat do containerů dle libosti a uživatel si tak bude moci upravit vzhled inventáře dle sebe.

Základními elementy budou:

- Název a popis – textová pole, která budou představovat název a popis právě vybraného objektu.
- Náhled – samostatný prvek UI, který bude zobrazovat náhled právě vybraného objektu. Náhled bude zobrazen pomocí obrázku, který bude rotovat v prostoru. Pokud objekt v inventáři bude obsahovat i 3D model, bude místo obrázku zobrazen tento model.

- Seznam objektů – seznam objektů bude reprezentován pomocí mřížky předem připravených UI elementů, jejichž vzhled si bude moci uživatel upravit podle sebe, stejně jako ostatní elementy.

- Orientační tlačítka – pokud bude počet objektů v inventáři větší, než se vejde najednou do seznamu objektů, bude rozdělen do několika stránek, mezi kterými bude možné si pomocí tlačítek přepínat.

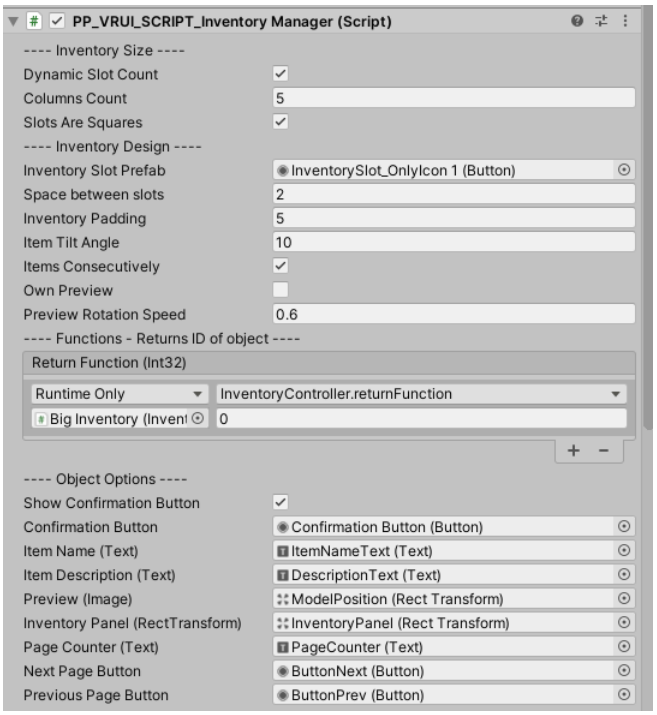
- Potvrzovací tlačítko – potvrzení výběru předmětu z inventáře bude probíhat pomocí potvrzovacího tlačítka nebo vyvoláním dvojitého kliknutí akce pro interakci s UI na vybraný předmět ze seznamu.

Velikost a počet položek v inventáři bude možné nastavit pomocí několika proměnných. Základním prvkem bude objekt představující jeden slot v inventáři, neboli předem připravený UI element obsahující rámeček a element pro zobrazení náhledu předmětu, jehož vzhled si uživatel může upravit. Je možné inventáři nastavit padding, kvůli rámečku kolem celého inventáře a velikost prostoru mezi jednotlivými sloty v inventáři. Velikost těchto prvků se poté vypočítá podle zadaných parametrů. Bude možné si nastavit, jestli budou jednotlivé sloty v inventáři čtvercové. V tom případě se bude nastavovat pouze počet sloupců v inventáři a počet řádků se vypočítá podle velikosti elementu, do kterého se sloty vkládají. Pokud uživatel nebude chtít čtvercové sloty v inventáři, jejich velikost se vypočítá podle zadáného počtu sloupců a řádků tak, aby přesně seděly podle velikosti elementu, do kterého se sloty vkládají. Položky v inventáři budou moci být nakloněny pomocí parametru udávající úhel náklonu, pro vytvoření lepšího 3D efektu inventáře.

S inventářem bude možné pracovat dynamicky nebo staticky. Pro dynamickou práci budou jednotlivé sloty inventáře vytvářeny a ničeny při přidávání čí mazání objektů z inventáře. Při smazání objektu mimo poslední pozici budou objekty v inventáři zatříděny a nebudou tak vznikat prázdná místa. Při statické práci s inventářem se při spuštění aplikace vytvoří definovaný počet míst v inventáři a při přidávání objektů je potřeba udávat, na jakou pozici se předmět umisťuje. Tento inventář bude vytvořen ve dvou verzích v závislosti na použitém containeru. Prvním z nich bude malý inventář, který bude součástí malého herního menu a pro zobrazení podrobností o vybraném přemětu bude využívat další obrazovku připevněnou k inventáři. Celá obrazovka s podrobnějšími informacemi se zobrazí až po vybrání předmětu, aby nepřekážela při použití jiné obrazovky z malého herního menu. Toto řešení se hodí pro menší inventář s rychlým přístupem, který uživatele nevytrhne ze scény, jelikož mu nezakrývá výhled.

Druhé použití je připojení k velkému hernímu menu, které má rozmístěné elementy inventáře po všech pěti obrazovkách tohoto menu. Toto řešení se hodí pro obsáhlejší popis objektů a přehlednější výběr z většího počtu položek. Kvůli své velikosti však zakrývá výhled po scéně a je tak nutné ho používat

v situacích, kdy má uživatel dostatek času na výběr položek z inventáře. Objekt v inventáři bude definován pomocí vlastní třídy, která bude obsahovat veškeré informace o objektu včetně náhledového obrázku a popřípadě i modelu. Po potvrzení výběru bude vyvolána předem definovaná událost s parametrem ID daného předmětu a poté bude možné podle ID získat z inventáře veškeré informace o daném objektu.

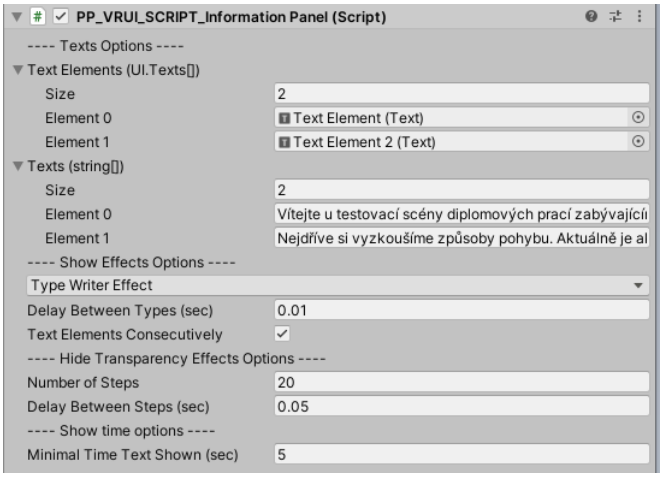


Obrázek 3.5: Navržené veřejné proměnné pro inventář.

3.3.7 Informační panel

Informační panel představuje obrazovku umístěnou v samotné scéně, která slouží pro informační účely. Informační panel bude obsahovat textové elementy, ve kterých se informace zobrazují. Texty, které se mají v elementech zobrazit, bude možné předávat pomocí metod, nebo předem nastavit před spuštěním aplikace. Informační panel bude nabízet dva typy efektů pro zobrazení textů. Prvním z nich je zobrazení pomocí průhlednosti, kdy text plynule přechází do viditelného stavu. U tohoto efektu bude možné nastavovat rychlost a plynulost přechodu mezi neviditelným a viditelným stavem. Druhým efektem bude psaný text, který postupně přidává části textu po jednom písmenu a simuluje tak psaní textu na klávesnici. U tohoto efektu bude také možné si nastavit rychlost mezi zobrazením jednotlivých písmen. U obou efektů si bude moci uživatel určit jestli se budou zobrazovat všechny textové elementy najed-

nou, nebo budou zobrazovány postupně. Při předání nového textu k zobrazení přejde starý text plynule do neviditelného stavu a zobrazí se text nový. Bude možné nastavit minimální dobu zobrazení textu, aby pokud uživatel předá více textů najednou, stihl čtenář přečíst všechny stávající texty, než se zobrazí další.



Obrázek 3.6: Navržené veřejné proměnné pro informační panel.

3.4 UI Elementy

Elementy, které se budou vkládat do navržených containerů, budou vycházet z klasických UI elementů implementovaných v Unity a bude se s nimi moci pracovat stejně. Budou pouze rozšířeny o další funkcionality dovolující interakci s nimi pomocí paprsků a zajišťující jejich správné zobrazování v prostoru. Veškeré UI elementy je možné upravovat jak z hlediska velikosti, tak z hlediska vzhledu bez vlivu na jejich funkci, či správné zobrazení.

3.4.1 Obrázek

Element představující obrázek bude totožný se základním UI elementem. Obsahuje jen malou úpravu zajištující správné zobrazení v prostoru, aby nedocházelo k překrývaní se vzhledem containeru.

3.4.2 Tlačítko

Tlačítko bude velmi podobné tlačítku pro základní UI, které nabízí Unity. Navíc bude obsahovat pouze vlastní kód, který bude zajišťovat správné zobrazení v prostoru a možnost interakce pomocí paprsku. Tlačítko také bude mít možnost přidání události, která se vyvolá po dvojitém stisknutí. Bude

obsahovat taky možnost zvýraznění tlačítka pomocí předem definované barvy, která se používá například pro zvýraznění právě vybráné položky v inventáři.

3.4.3 Přídavné menu

Jedná se opět o element vycházející ze základního UI tlačítka. Po stisknutí zobrazí další panel, ve kterém se mohou nacházet další UI elementy. Bude možné nastavit, kde se toto přídavné menu zobrazí v závislosti na pozici tlačítka. Je možné ho využít pro rozbalovací nabídku, nebo pro rozšíření možnosti výběru v menu. Po kliknutí mimo zobrazenou rozšířenou nabídku se menu opět schová.

3.4.4 Nápověda

Nápověda slouží k zobrazení dodatečné informace po stisknutí tlačítka, které nápovědu představuje. Po stisknutí se na předem definované pozici, která je základně nastavena na pravou stranu od tlačítka, zobrazí nový textový element, který obsahuje dodatečné informace, které uživatel předem u tlačítka definoval. Bude se vytvářet nová instance objektu představujícího nápovědu, aby uživatel mohl upravovat vzhled všech elementů obsahujících napovědu najednou.

3.4.5 Psaní textu

Tento element představuje políčko pro zadání textu uživatelem. Po kliknutí na tlačítko se zobrazí klávesnice, pomocí které se zadá hodnota do políčka. Bude možné nastavit barvu psaného textu a po potvrzení napsaného textu bude vyvolána událost či události s parametrem obsahujícím vložený text.

3.4.6 Klávesnice

Klávesnice se bude skládat ze tří částí, každá na samostatné obrazovce. Bude obsahovat numerickou část klávesnice, která bude moci být vyvolána samostatně, pokud uživatel bude chtít zadávat pouze numerické hodnoty. Další částí bude klasická část klávesnice obsahující písmena a znaky a bude moci být vyvolávána společně s numerickou, pokud uživatel bude chtít vkládat všechny možnosti vstupu. Poslední částí bude obrazovka, na které bude zobrazován právě napsaný text, aby uživatel přesně viděl, co právě napsal. Po kliknutí na element pro psaní textu se klávesnice zobrazí před uživatelem v předem definované relativní pozici. Bude možné si ale nastavit i přesnou pozici a rotaci, na které se má klávesnice v prostoru objevit. Toto nastavení však bude v závislosti na daném elementu pro psaní textu, aby si uživatel mohl nastavit pozici pro každý element zvlášť. Vzhled klávesnice může uživatel měnit dle libosti. Klávesnice bude obsahovat tlačítka rozdělená do dvou skupin. První skupina jsou znaky, které se při spuštění aplikace namapují na daná tlačítka podle názvu (ID). Mapování bude probíhat pomocí předem nastavených symbolů,

které budou rozděleny do dvou vrstev, jedné pro písmena a druhé pro znaky, mezi kterými je možné přepínat pomocí speciálního tlačitka. Druhou skupinou jsou právě tyto speciální tlačítka, která mají předem danou funkci (shift, enter, backspace, atd.). Uživatel si tak může nejen měnit vzhled a uspořádání tlačítek, ale i jednoduše přidávat vlastní tlačítka reprezentující nový znak či s vlastní specifickou funkci.

3.5 Návrh manipulace s předměty

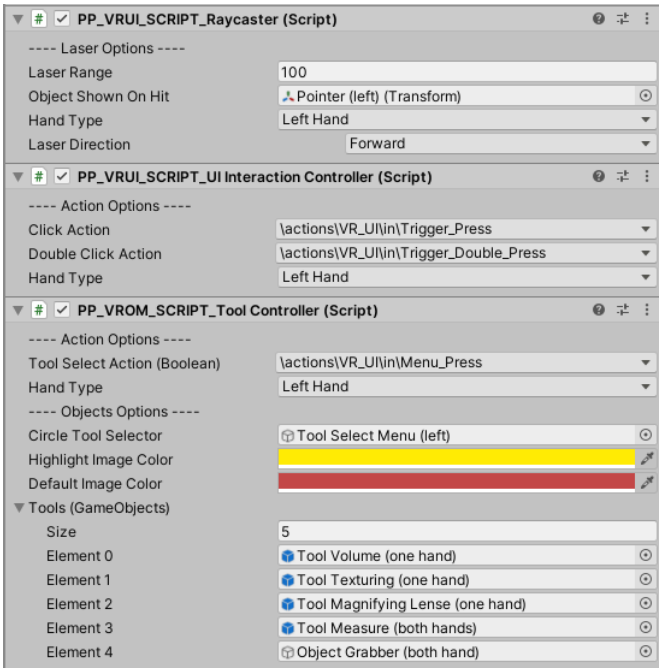
Další částí diplomové práce jsou nástroje pro manipulaci a interakci s předměty ve virtuální realitě. Tyto nástroje budou úzce propojené s dříve navrženým UI a budou využívat některé jeho prvky. Pro návrh bylo vybráno několik nástrojů, využitelných při práci s předměty v VR, specificky pro práci s historickými artefakty. Kromě nástrojů pro transformaci bude navrženo i několik nástrojů pro zkoumání vlastností objektů, jako je například objem či velikost.

3.5.1 Správa nástrojů

O správu nástrojů se bude starat samostatná třída připojená ke stejnému objektu jako třída spravující paprsky a interakci s UI elementy. Tato třída bude obsahovat samostatnou instanci pro každý ovladač a bude spravovat nástroje připojené k danému ovladači. Výběr daného nástroje bude probíhat pomocí kruhového menu popsaného v návrhu UI. U každého nástroje si bude moci uživatel zvolit vlastní ikonu i název. Po výběru požadovaného nástroje bude deaktivován předešlý nástroj a vybraný se připraví k použití. Aktuální nástroj bude zobrazen pomocí ikony nad ovladačem, která při použití změní barvu, aby bylo viditělné, kdy nástroj interaguje s objektem. Některé nástroje budou obsahovat implementaci pro jednu ruku, a pokud bude uživatel chtít nástroj používat na obou ovladačích, bude potřeba vytvořit dvě instance, kdy každá bude připojena k danému ovladači. Nástroj pro měření vzdálenosti bude potřebovat propojení obou ovladačů a bude tak nutné mít nástroj připojen k oběma ovladačům najednou.

3.5.2 Transformace objektů

Nástroj pro transformaci objektů bude představovat sdružení tří nástrojů, které budou použity pro transformaci základních parametrů objektů, jimiž jsou pozice, velikost a rotace. Objekt, se kterým bude možné interagovat, bude označen speciálním tagem a při interakci se tag změní, aby nebylo možné manipulovat s jedním předmětem oběma ovladači najednou. Zahájení interakce se bude provádět pomocí namíření paprsku na daný objekt a držení předem definovaného tlačítka. Z analýzy a definice tlačítek bude nejlepší volbou grip tlačítko, které je proto definováno jako výchozí akce. Po uvolnění tlačítka je



Obrázek 3.7: Navržené veřejné proměnné pro interakci s UI a správy nástrojů.

interakce ukončena. Po začátku interakce probíhá transformace na základě vybraného nástroje.

- Změna pozice – po uchopení předmětu bude možné pomocí tohoto nástroje pohybovat s objektem po scéně. Pohybem ovladače se bude objekt pohybovat společně s ovladačem a bude tak zůstávat ve stejné pozici relativně k ovladači. Pro pohyb objektem blíže či dále od ovladače bude možné použít libovolnou akci ovladače, která vrací hodnotu typu *Vector2*. Hlavní využití ale bude pomocí joysticku či trackpadu a nabídne dva typy použití. První možností bude neustálý pohyb objektu, jehož rychlost a směr se bude počítat na základě vzdálenosti dotyku od středu trackpadu, či pozice joysticku. Druhou možností je přesun objektu pomocí pohybu prstu po trackpadu, kde budou rychlost a směr počítany na základě směru a rychlosti pohybu po trackpadu. Bude možné i nastavit výchozí rychlost přibližovaní a oddalovaní.

- Změna velikosti – po uchopení objektu bude pomocí tohoto nástroje možné měnit jeho velikost. První možností bude pomocí pohybu ovladače směrem nahoru a dolů od bodu, ve kterém se ovladač nacházel při zahájení interakce s předmětem. Tato možnost je vhodnější pro přesnější, ale menší změny velikosti. Druhou možností bude úprava velikosti pomocí stejné akce jako v případě změny pozice, tedy ve výchozím případě

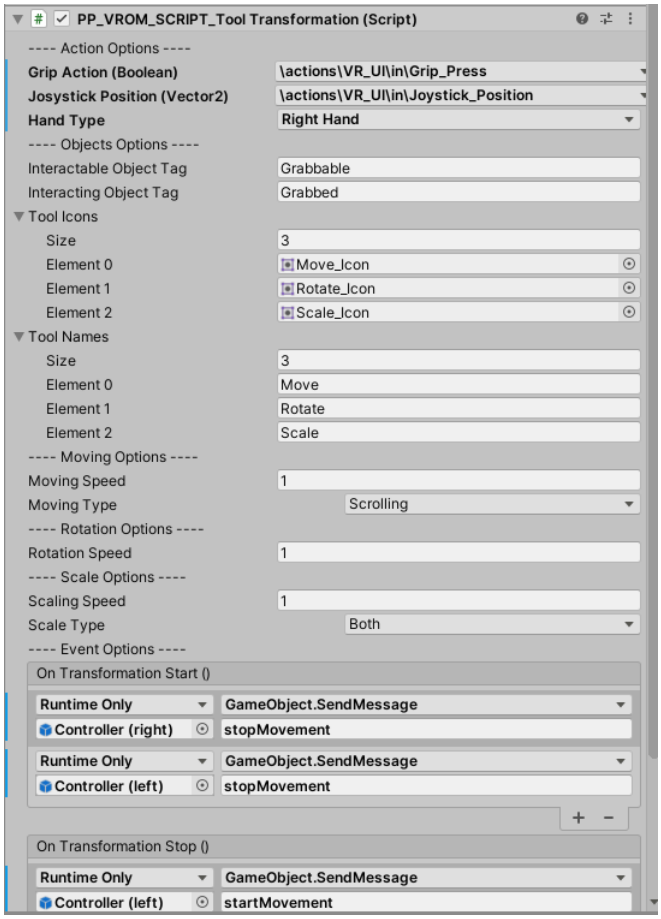
pomocí joysticku či trackpadu. Zvětšování a zmenšovaní objektu bude tedy probíhat pomocí pohybu prstu po trackpadu, nebo pohybem joysticku. Bude možné si vybrat mezi jednou z těchto metod, nebo kombinovat obě dvě. Bude možné si také změnit výchozí rychlost škálování velikosti objektu.

- Změna rotace – pomocí tohoto nástroje bude možné po uchopení předmětu měnit jeho rotaci. Rotace se bude měnit pomocí pohybu ovladačem v jednom ze čtyř směrů. Pomocí pohybu ovladačem vpravo či vlevo se objekt bude otáčet kolem globální osy Y. Pomocí pohybu ovladačem nahoru a dolů se bude uchopený předmět otáčet po ose Z definované směrem pohledu uživatele. Tento způsob dovoluje otočit předmět do všech pozic. Bude však kvůli malému dosahu rukou nutné provést transformaci několikrát. Proto bude ještě přidaná možnost otáčení objektu pomocí joysticku či trackpadu po ose X v závislosti na směru pohledu uživatele. To by mělo usnadnit rotaci s objektem a zrychlit přesun objektu do požadované pozice. Nástroj bude obsahovat parametr na určení výchozí rychlosti rotace.

Nástroj pro transformaci bude určen pouze pro jednu ruku, a pokud bude chtít uživatel používat tyto nástroje na obou ovladačích, bude potřeba vytvořit dvě instance objektu. Třída bude obsahovat definovatelné události, které budou volány při začátku a ukončení transformace a dovolují tak reagovat na tyto situace. Například pokud bude aktivovaný pohyb pomocí joysticku, bude možné tento pohyb při manipulaci s předmětem vypnout, aby nedošlo ke kolizi využití tlačítek.

3.5.3 Uchopování předmětů

Tento nástroj slouží pro možnost uchopení předmětu do ruky. Nástroj bude v jedné instanci zpřístupňovat funkcionalitu oběma ovladačům. Uchopení předmětu bude probíhat pomocí přiblížení ovladače k danému objektu a držení definovaného tlačítka. Výchozím tlačítkem, na základě analýzy, bude jako v předešlém případě grip tlačítko. Po uchopení předmětu se objekt ukotví k ovladači a bude se pohybovat spolu s ním. Po uvolnění tlačítka bude uvolněn i objekt. Uchopování předmětu bude nabízet dvě možnosti výběru předmětu k uchopení. V obou případech musí být objekt opatřen speciální tagem pro definici objektu, kterým může být manipulováno. První možností bude uchopení pomocí kolize, kterou zajišťuje komponenta *Collider*. Pokud *Collider* ovladače vytvoří kolizi s nějakým předmětem, je možné ho poté uchopit pomocí grip tlačítka. Druhou možností je uchopení předmětu na základě vzdálenosti. Po stisknutí se tedy uchopí nejbližší předmět opatřený speciálním tagem. Bude možné si v tomto případě nastavit maximální vzdálenost, na kterou jde předměty uchopit. Výhodou tohoto typu je, že můžeme uchopit předměty i na dálku, ale interakce tak působí méně reálněji.



Obrázek 3.8: Navržené veřejné proměnné pro nástroj pro transformaci.

Bude také možné opatřit vybraný objekt speciální komponentou obsahujícím rozšířené možnosti pro uchopení předmětu. Bude možné nastavit:

- Bod uchopení, na kterém se objekt uchopí, pokud dojde ke stisknutí uchopovacího tlačítka. Například pokud chceme uchopit hrnek, můžeme nastavit, že ho vždy uchopíme za ucho. Tento bod bude možné nastavit pomocí relativní pozice od pivotu objektu, buď v globálních či lokálních hodnotách.

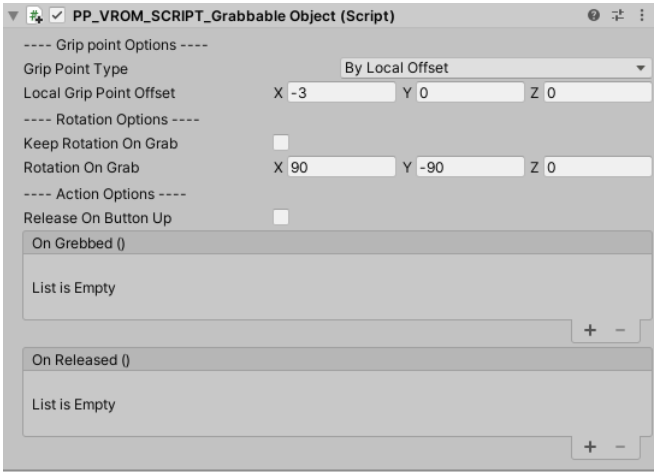
- Dalším nastavitelným parametrem bude rotace při uchopení. Například pokud chceme uchopit meč, chceme, aby vždy čepel směřovala dopředu od ovladače. Tuto rotaci bude možné nastavit pomocí eulerových úhlů.

- Dále bude možné nastavit, jestli se má objekt uvolnit při puštění uchopovacího tlačítka, či až při jeho opětovném stisknutí. Pokud bude uživatel

chtít držet nějaký objekt delší dobu, nebude tak muset držet tlačítko stále zmáčknuté.

- Komponenta bude obsahovat možnost definice vlastních událostí, které se vyvolají při uchopení či uvolnění objektu.

Pokud bude objekt podporovat fyziku a bude využívat gravitaci, bude jeho gravitace při uchopení vypnuta a po uvolnění opět zapnuta, aby nedošlo k narušení správného chování objektu při manipulaci. Zároveň po uvolnění bude potřebné objektu předat sílu a směr pohybu ovladače, aby bylo možné objekty házet.



Obrázek 3.9: Navržené veřejné proměnné pro uchopitelný předmět.

3.5.4 Měření objemu

Tento nástroj bude využíván k měření objemu objektu. Pro změření objemu objektu bude potřeba namířit paprsek na daný objekt a pomocí zvoleného tlačítka provést měření. Výpočet objemu se bude provádět na základě modelu daného objektu. Výsledek měření bude vypsán do textového pole umístěného před ovladačem.

3.5.5 Lupa

Tento nástroj poskytne lupu pro přibližování pohledu na objekty. Po zvolení tohoto nástroje se před ovladačem objeví model lupy obsahující zobrazovací plochu, na kterou bude promítán pohled kamery před lupou. Pomocí definované akce, ve výchozím stavu pomocí pohybu prstu po trackpadu, bude možné přibližovat a oddalovat pohled kamery. Přiblížení se bude měnit pomocí změny zorného pole a bude obsahovat i zdroj světla, aby byl objekt, který chceme pozorovat, dobře viditelný. Bude možné nastavit výchozí rychlost přibližování.

3.5.6 Měření vzdálenosti

Měření vzdálenosti bude nástroj, který bude využívat oba ovladače najednou. Pokud tedy na jednom z ovladačů přepneme na tento nástroj, přepne se automaticky na tento nástoroj i na druhém ovladači. Tento nástroj bude měřit vzdálenost mezi konci paprsků jdoucích z ovladačů. Výsledek měření bude zobrazen do textového pole před každým z ovladačů. Zároveň bude výsledek zobrazen v textovém poli umístěném mezi oběma měřenými body, jelikož toto místo může při měření vzdálenosti uživatel nejlépe sledovat. Bude možné si přizpůsobit přesnou pozici tohoto textového pole. Nástroj pro měření vzdálenosti bude obsahovat i stabilizaci, jelikož při míření ovladačem není možné udržet ovladač bez hnutí stále ve stejné pozici. Výsledek bude tedy stabilizován a počítán pomocí průměrné hodnoty z několika po sobě jdoucích měření.

3.5.7 Změna textury

Posledním z navržených nástrojů je nástroj pro změnu textury objektů. Po namíření na objekt, kterému chceme změnit texturu a stisknutí požadovaného tlačítka se zobrazí UI panel vycházející z podoby velkého inventáře, popsaného v návrhu uživatelského rozhraní. V tomto panelu bude inventář, ve kterém bude seznam dostupných textur pro daný objekt. Po výběru požadované textury se textura změní, aby uživatel mohl vidět náhled změny na daném objektu. Pomocí potvrzovacího tlačítka nebo dvojitého kliknutí na vybranou texturu se změna provede a nastaví vybranou texturu cílovému objektu. Pokud je změna zrušena, buď pomocí křížku v pravém horním rohu panelu, nebo změnou nástoje, bude objektu vrácena původní textura.

Dostupné textury bude možné nastavit obecně, neboli pro všechny objekty společně, nebo bude možné vybraným objektům přidat vlastní komponentu, která bude umožňovat definici dostupných textur pro daný objekt. Texturovatelný objekt musí být také opatřen příslušným tagem.

Kapitola 4

Realizace

Implementace návrhu probíhala v prostředí Unity ve verzi *2019.3.05f*. Díky dobré zpětné kompatibilitě prostředí by neměl být problém aplikaci spustit na novějších verzích Unity. Veškeré kódy byly psány v jazyce C#, kvůli předešlým zkušenostem a dobré rozšířenosti na vývojové platformě. Pro editaci kódu bylo využito Visual Studio [20] hlavně díky možnosti propojení s Unity a s tím spojeným usnadněním programování. Toto propojení umožnuje jednoduchou synchronizaci se samotným projektem a zpřístupňuje nápovědu pro knihovny, které Unity poskytuje.

4.1 Přijímání akcí z ovladačů

K přijímání akcí z ovladačů, například stisknutí tlačítka, pozice joysticku, nebo pohyb ovladače, je důležité využití balíčku SteamVR, které tuto komunikaci zpřístupňuje. Díky SteamVR je možné si definovat vlastní akce, které představují interakci s ovladačem. Každá akce má jednu z definovaných návratových hodnot. Pokud se jedná o tlačítko a tedy návratovou hodnotu typu boolean, je možné této akci přidat posluchače (metodu), která se vyvolá při stisknutí či uvolnění tlačítka. Problém nastává při změně scény v aplikaci, kdy objekty, které přijímaly zprávy o akcích jsou zničeny, ale stále jsou definovány jako posluchači dané akce. Proto je důležité při zničení objektu zrušit příjem zpráv z těcho akcí, aby nedocházelo k chybám při běhu programu. Tímto způsobem je implementováno přijímání informací o stisknutých tlačitkách na ovladači. Při získávání informací o akcích, které nejsou typu boolean, ale jedná se o vektory, nabízí SteamVR metody pro získání aktuální i minulé hodnoty a díky těmto metodám je možné v reálném čase detekovat například pohyb ovladače či pozici na joysticku.

4.2 Implementace uživatelského rozhraní

Jelikož výsledek práce bude hlavně sloužit k dalšímu využití vývojáři VR aplikací v Unity, budou všechny třídy opatřeny předponou PP_VRUI, aby nedošlo ke kolizi názvů tříd při využití v jiných projektech.

4.2.1 Struktura projektu

Celá implementace je oddělena od objektů *SteamVR*, aby uživatel nemusel zasahovat do připravených objektů a aby bylo možné vytvořit co nejlépe připravený prefab s většinou již připravených referencí mezi objekty. Hlavním objektem je *UIController*, který obsahuje stejnojmennou třídu, která definuje reference na objekty SteamVR, například ovladače, nebo kameru a reference na tyto objekty distribuuje ostatním třídám UI, které je vyžadují. Tato třída běží jen v jedné instanci a obsahuje tak návrhový vzor singleton. Zároveň tato třída referencuje i některé prvky UI, například klávesnici, nebo nápovědu, které jiné UI elementy mohou vyžadovat. Další třídou připojenou k tomuto objektu jsou akce pro zobrazení UI containerů. Tato třída se stará o správné otevření těchto containerů a pro každý nový UI container je potřeba přidat novou instanci, která dovoluje přizpusobení akcí pro zobrazení UI. Tato třída reaguje na vybrané akce vyvolané ovladačem a pomocí definovaného rozhraní posílá zprávu danému containeru o změně stavu. Přechod mezi zobrazeným a schovaným stavem si řeší třída sama a pomocí komponenty *Animator* definuje aktuální stav a přechodové efekty. Pro všechny další objekty bude v hierarchii aplikace tento objekt rodičem a díky tomu bude možné vytvořit prefab, který bude obsahovat celou strukturu objektů a bude tak pro jiné vývojáře jednoduché použít celou strukturu ve své aplikaci.

4.2.2 Interakce s UI

Při implementaci interakce s UI bylo nejdůležitější vyřešit problém komunikace mezi ovladačem a samotnými UI elementy. O tuto komunikaci se stará třída *UIInteractionController*, která je připojena ke speciálnímu objektu představujícímu správu operací týkajících se ovladačů, ale odděleného od samotných objektů ovladačů ze SteamVR. Interakce probíhá pomocí paprsků, které spravuje třída *Raycaster*, využívající objekt typu *Raycast* vykreslovaný pomocí komponenty *LineRenderer*. Tato třída při každém snímku aplikace vysílá paprsek a detekuje zasažený objekt. Pokud byl nějaký objekt zasažen, informuje pomocí broadcast zprávy všechny třídy připojené k tomuto objektu, včetně třídy *UIInteractionController*. Pokud byl zasažený objekt opatřený speciálním tagem, který definuje, že se jedná o element uživatelskéhho rozhraní, uchovává si tento objekt pro potenciální interakci. Pomocí definovaného rozhraní předává zprávy danému UI elementu o stavu interakce, tedy jestli je

element označen či o stisknutí interakčního tlačítka. Každý element pak tyto zprávy zpracovává sám a mění své chování na základě předané zprávy.

4.2.3 Editor

Pro každou implementovanou třídu byl vytvořen vlastní editor, který reprezentuje zobrazení nastavitelných proměnných v inspektoru Unity, tedy na místě, kde je bude uživatel používající tuto knihovnu nastavovat. Vytvoření vlastního editoru dovoluje přidání popisků a upozornění a pro lepší přehlednost. Zároveň je díky editoru možné vytvořit proměnnou, která bude v inspektoru zobrazena jako rozbalovací seznam, který je ideální pro výběr z více než dvou možností, jelikož se takové proměnné často ve třídách nacházejí. U těchto proměnných však existuje problém, který jejich použití znesnadňuje. Pokud je do aplikace vložen připravený objekt (prefab) obsahující třídu s proměnnou vytvořenou v editoru jako rozbalovací seznam, je po spuštění aplikace hodnota této proměnné nastavena na původní hodnotu prefabu. Lze však tomuto problému předejít jednoduchým odemčením prefabu v hierarchii aplikace, čímž se z daného objektu stane nový objekt a ne jen instance daného prefabu.

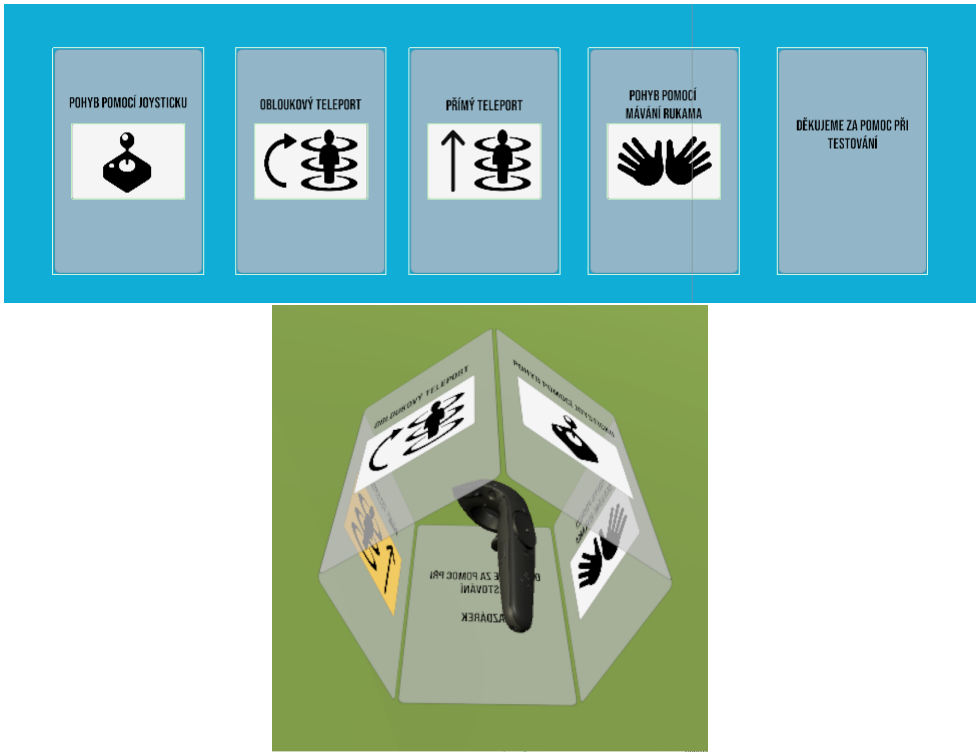
4.2.4 Implementace obrazovek ve 3D světě

Obrazovky jsou implementovány pomocí třídy *Canvas* s parametrem *RenderMode* rovným hodnotě *WordSpace*. Díky tomu je možné *Canvas* umístit do prostoru ve scéně a stále s ním pracovat jako s normálním prvkem UI. Ostrost UI elementů se nastavuje pomocí rozlišení, které se řídí poměrem velikostí komponenty *Canvas* v globálním měřítku parametru komponenty *Transform.Scale* a velikosti nastavené pomocí komponenty *rectTransform*, která udává počet pixelů dané obrazovky. Pokud chceme tedy dvakrát větší rozlišení obrazovky, můžeme její velikost v globální proměnné snížit na polovinu a rozlišení zdvojnásobit. Poté bude mít obrazovka ve scéně stejnou velikost, ale bude obsahovat 2x více pixelů a bude tedy ostřejší a detailnější.

4.2.5 Malé herní menu

Implementace malého herního menu byla složitá z hlediska výpočtu pozice a rotace obrazovek zobrazených kolem ovladače. Menu se musí přizpůsobit velikosti a počtu obrazovek a na základě těchto parametrů vypočítat přesnou pozici obrazovek relativně k ovladači. Pro získání těchto pozic a rotací byla vytvořena statická třída *HandMenuConstants*, která má definované výpočty pro všechny dostupné počty obrazovek a vrací tak výslednou pozici a rotaci obrazovek na základě jejich velikosti. Složitější výpočet pozice probíhá při použití všech pěti obrazovek. Výpočet vychází ze základních vlastností pozice bodů v soustavě souřadnic při konstrukci pravidelného pětiúhelníku popsaného na webu [21]. Výpočet zahrnuje i možnost odsazení od středu ovladače

a možnost přidání mezery mezi obrazovkami. Pokud je mezera nastavena na hodnotu nula, obrazovky poté přesně doléhají jedna na druhou a obklopují tak kompletně celý ovladač. Otáčení menu je implementováno pomocí akce vracející hodnotu *Vector2*, tedy primárně pozice joysticku či trackpadu. Pomocí posunu prstu po trackpadu, nebo pohybem joysticku je počítána rychlost otáčení a menu se tak otáčí kolem středu a dovoluje zpřístupnit obrazovky na zadní straně ovladače.

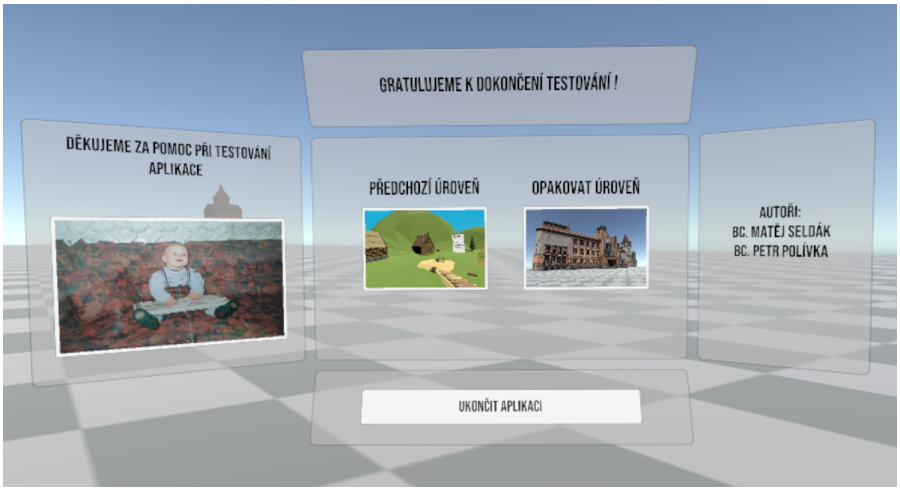


Obrázek 4.1: Malé herní menu s pěti obrazovkami.

4.2.6 Velké herní menu

Při implementaci velkého herního menu byl nejsložitější výpočet pozice přídavných obrazovek přiložených okolo hlavní obrazovky. Kvůli možnosti variabilní velikosti všech obrazovek, ale hlavně kvůli možnosti nastavení vlastního úhlu náklonu bylo složité vypočítat přesnou pozici obrazovek. Jelikož pivot obrazovek je definován na střed, bylo nutné vytvořit vzorec pro výpočet pozice těchto obrazovek na základě jejich velikosti a úhlu náklonu. Jelikož tento výpočet je používaný více prvky UI, například náhled textu pro klávesnici, byl oddělen od této třídy a připojen do statické třídy *Constants*, která spravuje obecné konstanty a výpočty, které využívá více prvků UI. Výpočet také počítá

s možností přidání mezery mezi tyto obrazovky. Třída spravující tento container provede při startu aplikace výpočet pozic obrazovek pomocí příslušné metody, umístí obrazovky na správné pozice a pomocí animace přejde do schovaného stavu. Jelikož jsou samotné obrazovky potomky hlavního objektu zpravujícího chování tohoto containeru, je možné jim při inicializaci nastavit relativní pozici k tomuto objektu, rovnou definované hodnotě zobrazovací pozice od uživatele. Při zobrazení containeru pak stačí nastavit pozici objektu na pozici kamery a pouze na základě směru pohledu uživatele nastavit rotaci celého objektu. To způsobí, že menu bude vždy ve stejné vzdálenosti od uživatele a bude vždy směřovat přímo na něj.

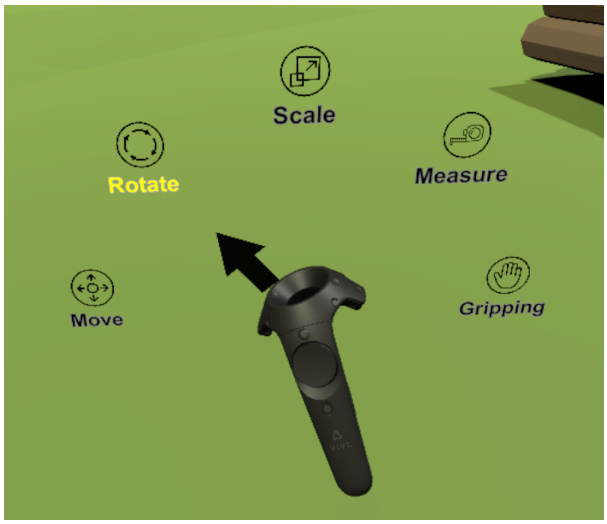


Obrázek 4.2: Velké herní menu se všemi dostupnými obrazovkami.

4.2.7 Kruhové menu

Kruhové menu bylo implementováno pomocí dvou překrývajících se obrazovek, které svou relativní pozicí určují právě vybranou možnost. Na první obrazovce jsou při spuštění instancovány položky výběru na základě připraveného prefabu obsahující textové pole a obrázek a definující tak vzhled položky. Na druhé obrazovce je pouze ikona šipky vizuálně určující právě zvolenou možnost. Pro přehlednější zobrazení vybrané možnosti je textové pole s aktuální možností zvýrazněno. O veškeré výpočty se stará samostatná třída *CircleMenuConstants*, která obsahuje dvě výpočetní metody. První metoda se používá při inicializaci a na základě velikosti elementů představujících jednu položku v menu vypočítá pozici daného elementu, aby při rotaci obrazovky obsahující položky menu nedocházelo k jejich překrývání. Jelikož maximální počet položek je roven osmi, vychází výpočet ze základních vlastností osmiúhelníku. Podle definice konstrukce bodů osmiúhelníku z [22] je na základě velikosti elementu a jeho pořadí vypočtena jeho pozice. Druhá metoda na základě

rozdílu rotace obou obrazovek vrací aktuální zvolenou možnost. Výpočet probíhá na základě rozdílu úhlů na ose Z a předem definovaného rozsahu pro každou možnost. Pro lepší čitelnost textu se položky v menu uchovávají ve vodorovné pozici vzhledem k podlaze a tak i při otáčení obrazovkou obsahující položky výběru, zůstávají texty dobře čitelné. Při otáčení obou obrazovek je nastavena hranice pro maximální otočení na základě počtu položek v menu. Toto opatření zabraňuje možnosti otočit obrazovky mimo vyznačené možnosti a zároveň otáčet položky stále dokola, pokud jich je méně než osm.

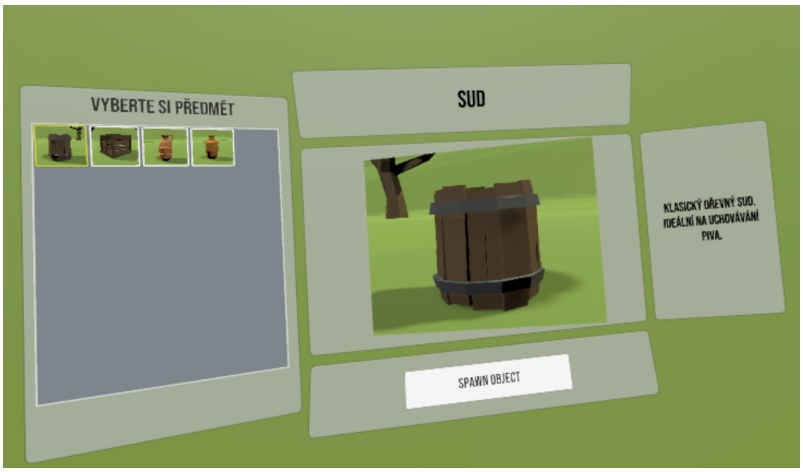


Obrázek 4.3: Kruhové menu.

4.2.8 Inventář

O správu inventáře se stará třída s názvem *InventoryManager*. Tato třída je nezávislá na implementaci UI a pouze vyžaduje reference na potřebné UI elementy. Je tedy možné inventář implementovat v libovolném UI containeru a proto z implementace vychází dvě verze inventáře. První je připojena jako obrazovka k malému hernímu menu a druhá je připojena k velkému hernímu menu. Obě možnosti fungují totožně pouze mají rozdílné uspořádání UI elementů v prostoru. Při inicializaci třídy *InventoryManager* dochází k instancování jednotlivých položek inventáře, které jsou prázdné, neboli neobsahují žádný objekt. Na základě parametrů zadaného uživatelem se výpočítá velikost a pozice jednotlivých položek v inventáři. Pokud uživatel vyžaduje čtvercové tvary položek v inventáři, je mu po výpočtu velikostí a pozic, vypsána ideální výška panelu inventáře, která odpovídá zadaným parametrům. Pokud je počet položek v inventáři větší než kolik se jich vejde do panelu inventáře, je inventář rozdělen na stránky, mezi kterými je možné pomocí tlačítek přepínat. Po vybrání předmětu z inventáře se v definovaných UI elementech zobrazí

podrobnější informace o objektu včetně náhledu. Tento náhled byl navržen formou rotujícího obrázku pro objekty v inventáři, které neobsahují 3D model a formou rotujícího 3D modelu pro objekty, které ho obsahují. Při implementaci se bohužel nepovedlo možnost rotujícího 3D modelu vytvořit kvůli problémům s velikostí modelů. Každý model má jinou velikost a ani pomocí výpočtu hranic objektu přes komponenty *Mesh* ani *MeshCollider* se mi nepovedlo získat dostatek informací potřebných pro výpočet zmenšení či zvětšení modelu na požadovanou velikost. Náhled tedy zůstává pouze formou obrázku i pro objekty obsahující 3D model. Tato třída obsahuje základní metody pro práci s inventářem. V aplikaci definujeme novou třídu *InventoryItemInfo* představující objekt v inventáři. Tato třída obsahuje název a popis objektu, obrázek představující náhled objektu a je možné přidat i model daného objektu. Pomocí základní operací je možné tyto objekty do inventáře přidávat, mazat a přesouvat jejich pozice. Inventář obsahuje i frontu, která uchovává vložené objekty pro případ, že inicializace třídy proběhne později než uživatel začne vkládat objekty do inventáře. Tato fronta je po inicializaci zpracována a objekty jsou vloženy do inventáře. Třída definuje vlastní událost, která je vyvolána při potvrzení výběru předmětu z inventáře a předává jako parametr ID daného předmětu. Podle ID je poté možné od inventáře získat informace o daném předmětu pomocí příslušné metody.

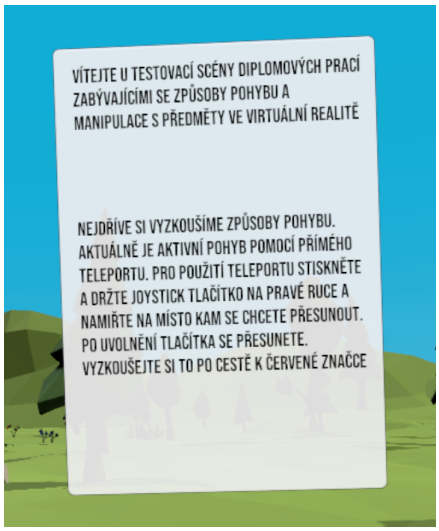


Obrázek 4.4: Velký inventář.

4.2.9 Informační panel

Informační panel obsahuje jednu obrazovku, do které lze vkládat libovolný počet textových elementů, na kterých se po spuštění či vyvolání příslušné metody zobrazují informační texty. Hlavní částí implementace bylo vytvoření efektů pro zobrazení textů. Tyto efekty jsou spravovány metodami běžícími na

vlastním vlákně procesu, jelikož běží souběžně s herním světem. Prvním efektem je plynulý přechod z neviditelného stavu do viditelného. Ten je definován pomocí počtu kroků přechodu mezi transparencí textu rovnou nule a jedné. Druhým parametrem je čas mezi jednotlivými kroky. Tímto je možné regulovat jak plynulost daného efektu tak i dobu trvání. Druhým efektem je postupné přidávání písmen daného textu za účelem simulování psaného textu na psacím stroji či klávesnici. Textovému elementu je při každém kroku přidáván jeden symbol z výsledného textu a je možné definovat dobu mezi přidáním dalšího symbolu. U těchto efektů je možné nastavit, jestli se mají dané textové elementy zobrazovat postupně, nebo všechny najednou. Informačnímu panelu je možné předat pomocí definovaného rozhraní nové texty k zobrazení. Třída spravující chování informačního panelu obsahuje frontu pro ukládání budoucích zpráv a tak je možné předat panelu více textů najednou. Ty jsou postupně zpracovávány a podle vybraného efektu zobrazovány na informačním panelu. Lze nastavit i minimální dobu zobrazeného textu, neboli čas, který budou vybrané texty viditelné než se zobrazí další texty z fronty. Tato možnost dovoluje odeslat více textů najdenou bez toho, aby uživatel měl problém stihnout přečíst všechny informace zobrazené na panelu.



Obrázek 4.5: Informační panel.

4.2.10 Klávesnice

Implementace klávesnice vychází z návrhu a nabízí tak možnosti úpravy navržené v předešlé části. Po spuštění aplikace dojde k namapování symbolů základní vrstvy klávesnice a dojde k jejímu schování pomocí animace. Zároveň je vypočtena pozice pro políčka zobrazujícího právě napsaný text, který se nachází v definovaném úhlu oproti klávesnici. Výpočet probíhá na základě ve-

likosti klávesnice a samotného políčka a na základě definovaného úhlu náklonu. Zobrazení klávesnice probíhá pomocí definovaného rozhraní a je možné i zobrazit pouze numerickou klávesnici jak s náhledovým políčkem tak bez něj. Základní zobrazení obsahuje celou klávesnici včetně náhledového políčka, jak lze vidět na obrázku 4.6

Obrázek 4.6: UI klávesnice.

4.3 Implementace manipulace s předměty

Kvůli kolizi názvů s jinými třídami, které by u vývojářů, kteří budou tuto knihovnu používat, mohly vzniknout, mají všechny třídy implementující manipulaci s předměty předponu PP_VROM. Třída spravující nástroje i všechny třídy jednotlivých nástrojů jsou odděleny od objektů balíčku SteamVR kvůli přehlednosti a jednoduchému použití v jiných projektech, díky možnosti vytvoření prefabu, který se poté pouze vloží do projektu. Implementace nástrojů pro manipulaci a interakci s předměty úzce souvisí s implementací uživatelského rozhraní. Využívá spoustu jejich tříd a funkcí, a proto je pro použití potřeba použít část implementace UI. Objekty, se kterými je možné manipulovat či interagovat musí být označeny tagem *Grabbable*. Při interakci se tento tag dočasně změní na *Grabbed*, aby nedocházelo ke kolizi interakcí kdybychom chtěli interagovat s jedním objektem oběma ovladači najednou. Názvy těchto tagů lze změnit při konfiguraci nástrojů.

4.3.1 Správa nástrojů

Správu nástrojů používaných pro manipulaci a interakci s předměty zařizuje třída s názvem *ToolController*. Tato třída je připojena ke stejnému objektu, který obsahuje třídu pro správu interakce s UI a třídu *Raycast* pro vytváření a vykreslování paprsku. Od této třídy přijímá informace o zasažených objektech, které poté předává aktivnímu nástroji, který s objektem interaguje. Třída spravuje nástroje pouze pro jednu ruku a je tedy pořeba vytvořit instanci pro každý ovladač zvášť. To také dovoluje připojení různých nástrojů pro každou

ruku a vytvořit tak specifické asymetrické ovládání. Třída obsahuje seznam objektů obsahujících třídy představující dané nástroje. Při startu aplikace si od těchto objektů vyžádá informace o nástrojích včetně ikony názvu a počtu obsažených druhů nástrojů. Tyto informace uchovává a předává nástrojům informace o změnách stavů. Po inicializaci všech nástrojů předá informace připojenému kruhovému menu, které je využíváno pro výběr nástrojů. Pokud je zvolený nový nástroj, třída upozorní aktuální nástroj pomocí definovaného rozhraní o ukončení jeho aktivity a poté aktivuje nový vybraný nástroj. Právě aktivnímu nástroji poté předává informace o aktuálně zasaženém předmětu pomocí paprsku a tyto informace si poté nástroj zpracovává sám. Třída obsahuje i referneci na jednoduché UI obsahující obrázek zobrazující právě aktivní nástroj. Při použití nástroje je správce informován o začátku interakce a změní barvu obrázku pro lepší upozornění uživatele. Zároveň obsahuje jedenoduché UI obsahující textové pole umístěné před ovladačem a díky tomu umožňuje nástrojům vypisovat textové informace, například výsledky měření.

4.3.2 Transformace objektů

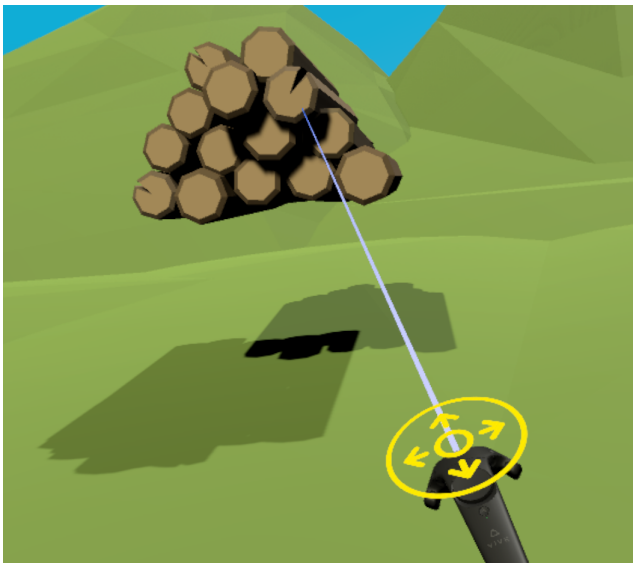
Transformace objektů je sdružení tří nástrojů se stejnou implementací ovládání a pouze s rozdílným vlivem na objekt. Po aktivaci nástroje přijímá nástroj informace o právě zasaženém objektu. Pokud je objekt zasažen a vyvolá se akce pro interakci, objekt se označí a podle zvoleného typu nástroje jsou měněny jeho vlastnosti.

Pokud je aktuální nástroj změna pozice, vytvoří se nový objekt s komponentou *Transform* určující pozici objektu. Tento objekt je vytvořen jako potomek ovladače a při pohybu ovladače tak zůstává ve stejné relativní pozici. Při každém snímku je poté pozice manipulovaného objektu upravena na základě pozice objektu s komponentou *Transform* a manipulovaný objekt se tak pohybuje spolu s ovladačem ve stejné relativní pozici. Jednodušším řešením by bylo přesunutí objektu jako potomka ovladače a ten by se poté pohyboval sám s ovladačem bez potřeby dalšího zásahu do změny pozice. To by však mohlo narušit hierachii, ve které se tento objekt nachází a mohlo by tak dojít ke ztrátě funkčnosti související s umístěním tohoto objektu v projektové hierarchii. Při ukončení interakce je dočasný objekt zničen a manipulovaný objekt nadále nenásleduje pozici ovladače. Pro přiblížení a oddálení je použita akce typu *Vector2*, v základní verzi se jedná o pohyb prstu po trackpadu ovladače. Tento pohyb způsobuje přiblížení či oddálení objektu po směru, který ho spojuje s ovladačem. Při přiblížení je nastavena minimální hranice vzdálenosti a není tak možné objekt při neustálém přibližování začít oddalovat na opačnou stranu ovladače.

Druhým nástrojem je změna velikosti. Tento nástroj mění lokální velikost objektu ve všech osách najednou, aby nedocházelo ke změně poměru stran objektu. Při začátku interakce si nástroj uloží pozici ovladače a na základě pohybu směrem nahoru a dolů mění velikost zvoleného objektu. Aktuální

velikost objektu je vypočítána na základě aktuální vzdálenosti ovladače od jeho původní pozice a kvůli konečnému dosahu ruky tak není možné objekt zvětšovat či zmenšovat donekonečna. Kvůli tomu je zde přidána možnost úpravy velikosti i na základě pozice prstu na trackpadu, či jiné akci definované pro všechny tři typy nástroje. Pomocí trackpadu je tak možné objekt zvětšovat donekonečna. Pro zmenšování je vytvořena hranice velikosti větší než nula a objektu tak nelze nastavit nulovou ani zápornou velikost. Rychlost škálování velikosti je konstantní a nemění se tak v závislosti na aktuální velikosti objektu.

Posledním z nástrojů pro transformaci objektů je nástroj sloužící pro změnu rotace. Tento nástroj upravuje rotaci předmětu pomocí otáčení po jedné ze tří os. Tyto osy se však určují na základě pozice a rotace kamery a ne samotného objektu. To slouží pro lepší a přirozenější možnosti otáčení. Výpočet se opět provádí na základě změny pozice ovladače. Při horizontálním pohybu ovladače se objekt otáčí kolem osy Y a při vertikálním pohybu se objekt otáčí po ose Z. Tyto dvě osy jsou kolmé na vektor propojující kameru a daný objekt. Pomocí josysticku či trackpadu lze pohybovat po třetí ose, která náleží právě vektoru propojující kameru a vybraný objekt. Pomocí tohoto otáčení je možné otočit objekt do jakékoliv polohy.



Obrázek 4.7: Nástroj pro manipulaci.

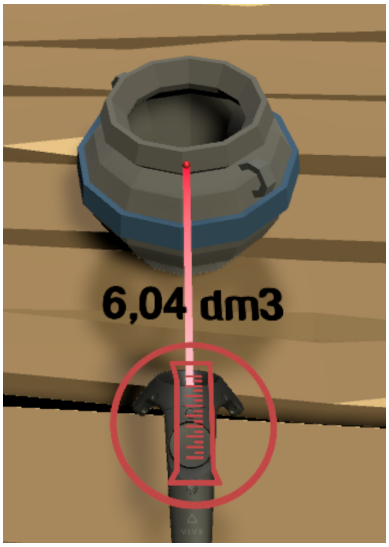
4.3.3 Uchopování předmětů

Nástroj pro uchopování předmětů je druhou z implementovaných možností manipulace s předměty. Uchopení předmětu je implementováno obdobně jako nástroj pro transformaci pozice, je však nutné být ovladačem u objektu blíže.

Při aktivaci tohoto nástroje se přeruší vysílání paprsku jelikož není k interakci používán. Po spuštění akce definované pro intrakci s předmětem je vytvořen dočasný objekt s komponentou *Transform* na pozici definované pomocí nastavení bodu pro uchopení. Tento bod a další parametry se dají nastavit pomocí přidání třídy *GrabbableObject* danému objektu. Pokud objekt neobsahuje tuto třídu je pozice dočasného objektu nastavena na pozici objektu a ten tak zůstává v relativní pozici k ovladači jako ve chvíli, kdy byl uchopen. Pomocí třídy *GrabbableObject* je možné třídě nastavit bod pro uchopení a při uchopení předmětu je tak vypočítána pozice pro dočasný objekt, aby uchopený objekt kolidoval s ovladačem právě v tomto bodě. Pomocí této třídy lze objektu nastavit i rotaci při uchopení. Ta je základně ponechávána a při uchopení se nezmění. Pokud je však pomocí této třídy nastavena, objekt se při uchopení otočí do požadované pozice. Objekty k uchopení je možné vybírat dvěma způsoby. První z nich je pomocí kolize komponent *Collider*. To vyžaduje, aby ovladač měl tuto komponentu a identifikoval tyto kolize. Při inicializaci třídy představující nástroj pro uchopování předmětů je proto ovladačům komponenta *Collider* přidána a s ní i třída s názvem *CollisionChecker*, která kolize zaznamenává. Není možné sledovat kolize jiného objektu než který vlastní danou třídu a proto je potřeba třídu vložit přímo ovladači. Tato třída zaznamenává všechny objekty, se kterými ovladač koliduje a při zahájení akce pro uchopení předává tento seznam třídě, která spravuje tento nástroj. Ta poté ze seznamu vybere nejbližší objekt, se kterým zahájí interakci. Druhou možností je pouze výběr nejbližšího objektu. Zde je důležité nastavit maximální vzdálenost pro uchopení předmětu. Při akci pro uchopení je poté nalezen nejbližší uchopitelný předmět a pokud je blíže než je maximální vzdálenost pro uchopení, zahájí se interakce. Pokud objekt podporuje fyziku a používá gravitaci, je mu gravitace vypnuta pomocí úpravy parametru v komponentě *Rigidbody*. Při uvolnění předmětu je gravitace objektu opět zapnuta a zároveň je mu předána rychlost a směr pohybu ovladače, aby objekt při upuštění pokračoval směrem pohybu ovladače. Tyto informace lze získat pomocí knihovny SteamVR.

4.3.4 Měření objemu

Tento nástroj po spuštění akce pro interakci změří objem daného objektu. Výpočet probíhá pomocí samostatné třídy s názvem *VolumeCalculator*, které je předána reference na měřený objekt. Výpočet probíhá na samostatném vlákně jelikož u složitějších objektů může výpočet trvat déle a došlo by tak k zamrznutí aplikace. Výpočet probíhá na základě vrcholů obsažených v komponentě *Mesh* definující model objektu. Při tvorbě algoritmu pro výpočet objemu jsem vycházel z postupu popsaného na webu [23]. Kvůli přehlednosti a omezenému počtu číslic v textovém poli, ve kterém se výsledek zobrazuje, je výsledek převáděn mezi jednotkami podle nejvyšší nenulové cifry.



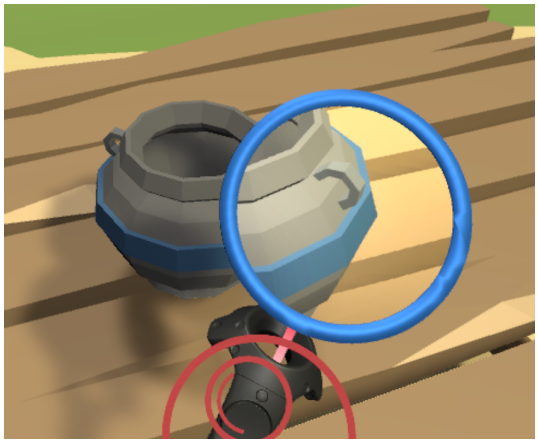
Obrázek 4.8: Nástroj pro měření objemu.

4.3.5 Lupa

Pro tento nástroj byl vytvořen vlastní 3D model lupy, do kterého je vložen kruhový objekt představující čočku. Tento objekt má definovaný materiál, který reflektuje pohled z kamery, která je umístěna těsně před tímto objektem. Zároveň je součásti lupy i kruhové světlo namířené směrem pohledu kamery. Díky tomuto světlu vzniká lepší viditelnost objektů, které chceme tímto nástrojem sledovat. Pomocí definované *Vector2* akce, v základním případě pomocí trackpadu je možné měnit zorné pole, což způsobuje přibližovaní a oddalování pohledu kamery, bez problémů způsobených pohybem kamery po scéně. Hodnoty zorného pole mají definované hranice, aby se uživatel neztratil ve velikosti přiblížení.

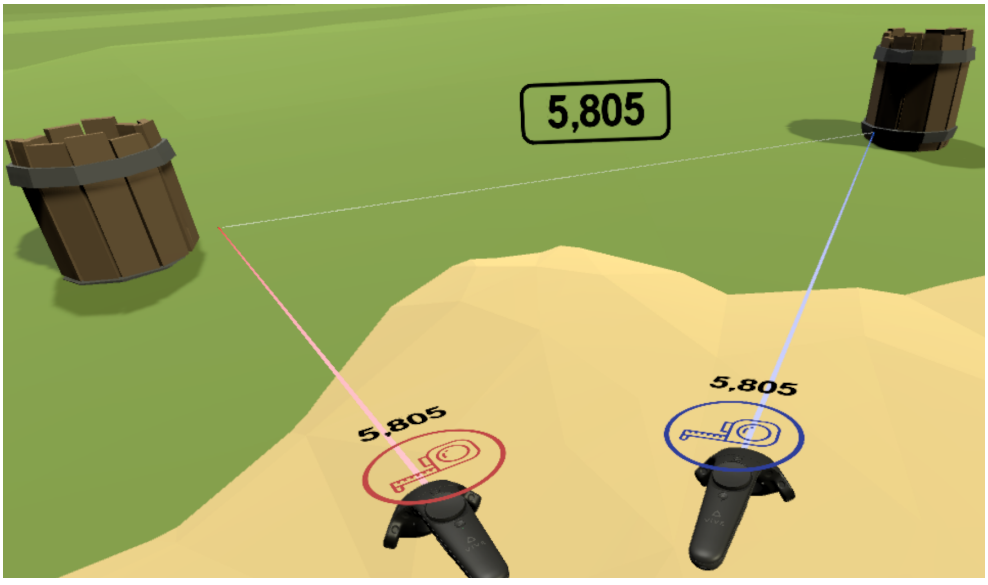
4.3.6 Měření vzdálenosti

Pro použití tohoto nástroje je potřeba využít obou ovladačů a proto při výběru tohoto nástroje na jedné z rukou, je nástroj aktivován i na druhé ruce. Pokud je poté nástroj na jedné ruce změněn jeho funkce je deaktivována, ale na druhé ruce zůstává označen stále jako vybraný nástroj. Nástroj využívá pozice objektů pojmenovaných *Pointer* nacházejích se na pozici střetu paprsků jdoucích z ovladačů a objektů, které zasáhly. Pomocí pozic těchto objektů vypočítává nástroj vzdálenost mezi těmito objekty. Tuto vzdálenost poté zobrazuje v textových polích před ovladači spravovanými třídou *ToolController* popsané výše. Zároveň přidává i vlastní jednoduchý UI prvek obsahující další textové pole. Pozice tohoto textového pole se udržuje veprostřed mezi měřenými body. Podle vzdálenosti od kamery se také mění jeho velikost, aby



Obrázek 4.9: Lupa.

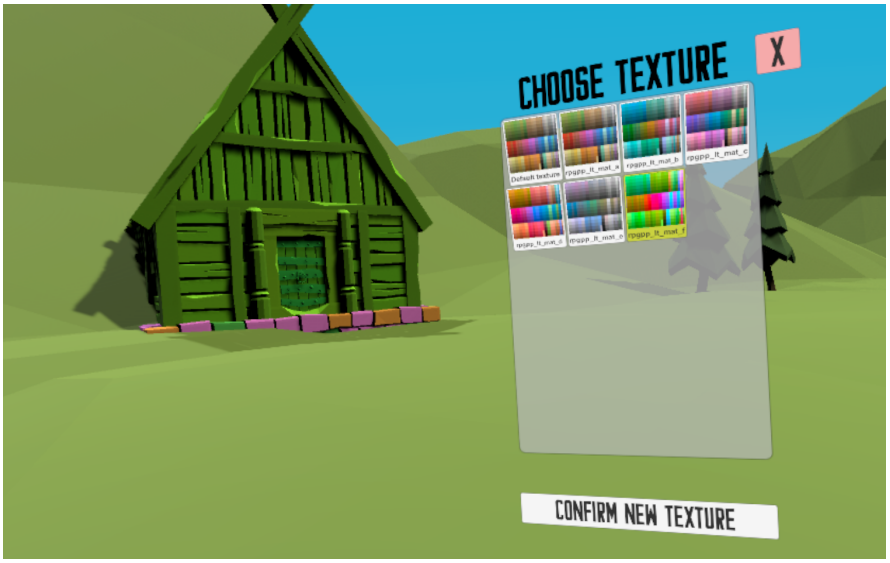
pro uživate působil stále stejně velký a byla tak hodnota pořád dobře čitelná. Mezi měřenými body se zároveň nachází čára tvořená pomocí komponenty *LineRenderer*, která vizuálně zobrazuje měřenou vzdálenost. Jelikož je prakticky nemožné udržet ovladače bez hnutí stále na stejné pozici, je výsledek měření stabilizován. Stabilizace probíhá pomocí výpočtu průměru z definovaného, po sobě jdoucího počtu měření. Stabilizace probíhá pouze pokud se uživatel snáží s ovladači nehýbat a jejich pozice se tak mění minimálně.



Obrázek 4.10: Nástroj pro měření vzdálenosti.

4.3.7 Změna textury

Při implementaci tohoto nástroje byla využita implementace velkého inventáře vytvořeného při implemetaci UI. Tento inventář byl upraven pouze po vizuální stránce a bylo přidáno tlačítko pro uzavření tohoto UI, jelikož není vyvoláváno obecnou akcí a uživateli tak umožňuje jednoduše UI element zavřít. Po namíření paprsku na objekt a zahájení akce pro interakci se před uživatelem na nastevené relativní pozici objeví tento inventář obsahující dostupné textury. Uživatel si poté pomocí interakce s tímto inventářem zvolí požadovanou texturu, která se po potvrzení objektu nastaví. Nastavení textury samotné však není možné, jelikož změna textury v materiálu by ovlivnila všechny objekty obsahující tento materiál. Je tedy nezbytné pro každou z dostupných textur vytvořit nový materiál a ten následně objektu nastavit. Třída představující nástroj pro změnu textury obsahuje metodu pro vytoření obrázku ve formě *Sprite* z dodaného materiálu. Tento obrázek se poté používá jako náhledový obrázek při výběru materálu v inventáři. Pokud materiál neobsahuje texturu, ale jen barvu, je vytvořen obrázek reprezentující tuto barvu. Seznam dostupných materiálů je možné vytvořit globálně pro všechny objekty pomocí seznamu materiálů definovaného v třídě reprezentující texturovací nástroj. Je také možné texturovatelnému objektu přidat třídu s názvem *TexturableObject*, která dovoluje nastavit seznam dostupných materiálů pro každý objekt zvlášť. Lze tyto přístupy kombinovat a pokud chceme změnit texturu objektu, který třídu *TexturableObject* neobsahuje je nabídka tvořena z globálně definovaných materiálů a pokud třídu obsahuje, je nabídka tvořena z materiálů definovaných explicitně pro daný objekt.



Obrázek 4.11: Nástroj pro změnu textury objektů.

4.4 Propojení implementace UI a nástrojů

Implementace uživatelského rozhraní a nástrojů pro manipulaci a interakci s předměty jsou úzce propojeny a budou se tak ve výsledném balíčku knihovny nacházet společně. Jelikož nástroje používají prvky implementovaného UI, nebude možné použít nástroje bez využití UI. Je však možné využívat UI prvky samostatně bez rozhraní pro nástroje. Nástroje je možné úplně deaktivovat pomocí příslušného rozhraní třídy *ToolController*. Toto rozhraní je využíváno při otevření libovolného UI containeru a nástroje jsou tak deaktivovány po dobu interakce s daným UI, aby jejich rozhraní nazasahovalo do interakce s daným containerem. Nástroje i interakce s UI se provádí pomocí stejných paprsků, o které se stará třída *Raycaster* a informace o zasažených objektech odesílá oběma třídám.

Kapitola 5

Testování

Pro účely testování byla vytvořena scéna obsahující důležité části implementovaných prvků UI a veškeré implementované nástroje. Testování probíhalo na headsetu HTC Vive, pro které byla aplikace primárně vyvíjena. Testovací aplikace běžela na počítači s grafickou kartou *NVIDIA GeForce GTX 1060 3GB*, procesorem *AMD FX(tm) 6100 Six Core* s frekvencí jader 3.3GHz a s 16GB RAM. Tato sestava zajistila dostatečný výkon k plynulému testování implementace ve virtuální realitě.

5.1 Testované prostředí

Testovací scéna byla vytvářena ve spolupráci s Bc. Matějem Sedlákem, jehož diplomová práce se zabývá možnostmi pohybu a manipulace s budovami ve virtuální realitě. Testovací aplikace obsahuje propojení implementovaných funkcí obou diplomových prací. Pro jednodušší testování a propojení obou implementovaných částí byla pro obě diplomové práce vytvořena společná testovací aplikace, kterou je tak možné testovat individuálně a výsledky testování mezi sebou sdílet. Testovací scéna funguje formou jednoduchých úkolů, které postupně testují všechny implementované funkce. Před každým úkolem je uživateli zobrazen informační panel, který obsahuje informace o zadaném úkolu a návod pro použití právě testované funkcionality. Pro vytovření prostředí a objektů pro testování byl využit volně dostupný balíček [24] obsahující modely středověkých objektů a budov. Testovací aplikace se skládá ze tří částí:

- První částí testovací aplikace je testování možností pohybu. Tato část uživatele seznámí se všemi dostupnými způsoby pohybu. Jelikož je v další části testování potřeba využít jeden s možných způsobů pohybu, je potřeba uživatele s možnostmi a použitím pohybových funkcí seznámit nejdříve. Pro přepínání mezi těmito druhy pohybu je využito implementované UI ve formě malého herního menu kolem ovladače. Při této části

je tedy také testováno použití tohoto menu a základní forma interakce s UI.

- Po otestování všech způsobů pohybu je testovanému uživateli otevřena možnost používání nástrojů. Postupně jsou jednoduchými úkoly uživateli představeny všechny nástroje a vysvětleny způsoby jejich používání. Nástoje, které netransformují objekty, ale měří vlastnosti objektů, jsou testovány pomocí zadání naměřené hodnoty do textového pole pomocí klávesnice. Díky tomu je součástí testování nástrojů i testování interakce s klávesnicí a zadávání textu. Poslední částí testování nástrojů je použití velkého inventáře, ze kterého má uživatel za úkol vybrat objekt a z inventáře ho umístit do scény. Po splnění všech úkolů je uživateli zobrazeno prostředí obsahující několik objektů, se kterými lze interagovat a uživatel si tak může nástroje vyzkoušet komplexněji a nalézt chyby, nedostatky a možnosti pro vylepšení.

- Testování manipulace s budovami – Tato část testovací aplikace se nachází v samostatné scéně,do které se po dokončení předešlé části uživatel přesune pomocí výběru tlačítka další úrovně obsaženého v implementovaném velkém herním menu. Tím je testován i poslední z hlavních implementovaných UI containerů. V poslední části jsou testovány způsoby manipulace s budovami, které jsou součástí diplomové práce Bc. Matěje Sedláka.

5.2 Průběh testování

Pro testování bylo vybráno 10 testerů s rozdílnými technickými znalostmi i předchozími zkušenostmi s virtuální realitou, pro co nejširší náhled na možnosti úpravy a vylepšení implementace. Jelikož veškeré informace potřebné k dokončení všech úkolů, ze kterých se skládá testovací aplikace, jsou popsány přímo v této aplikaci, snažlili jsme se do testování co nejméně zasahovat. Největší problémy způsobovalo v celku komplexní ovládání, které v průběhu testu uživatelé zapomínali a bylo jim nutné připomíat, jaká tlačítka vyvolávají jakou akci. Pokud byl testovaný uživatel zmatený, snažili jsme se ho neovlivňovat při zjišťování správného postupu a do testu zasahovali pouze v případě výskytu implementační chyby. Nejdříve jsme nechali uživatele seznámit s ovládáním UI a nástrojů pro manipulaci a požádali je o komentování právě prováděné činnosti. Při fázi testování dovolující komplexní vyzkoušení všech dotupných nástrojů, možností pohybů a UI elementů jsme se dotyčných tázali na problémy spojené s jejich používáním či návrhy na zlepšení. V průběhu testu jsme všechny informace zaznamenávali pro budoucí zpracování. Testování implementovaných nástrojů, které jsou zaměřené na práci s historickými artefakty, mělo probíhat samotnými historiky, kteři by nástroje mohli využívat.

Kvůli nastalé situaci ovlivněné rozšířením korornaviru se však tato spolupráce neuskutečnila.

5.3 Výsledky testování

Výsledkem testování jsou nalezené chyby, problémy a také návrhy pro vylepšení implementovaných částí UI a nástrojů. Tyto poznatky zjištěné během testování jsem rozdělil do tří kategorií na základě priority a smysluplnosti zabývání se jejich řešením.

5.3.1 Chyby implementace

Při testování bylo nalezeno několik chyb, které v různé míře zasahují do používání aplikace.

- Náhled klávesnice – při zadávání naměřených hodnot v části testování, kde uživatel pracoval s nástroji, se několika testujícím stalo, že při zadávání hodnoty na klávesnici, se v náhledu zobrazeném nad klávesnicí zobrazil jiný text než v právě editovaném políčku. Tato chyba vznikla při pokusu o smazání právě napsaného nesprávného symbolu. Chybu nejspíše způsobuje špatná komunikace mezi tlačítky klávesnice a náhledovým okénkem. Tato chyba neměla velký dopad na testování a nekteří testující si chyby ani nevšimli.

- Zmizení nástrojů – tato chyba měla velký vliv na průběh testování a pokud se chyba objevila, nebylo tak možné testování řádně dokončit a byl tak nutný zásah do procesu testování. Chyba nastala při otevírání několika UI containerů najednou, přesněji při otevření inventáře a následného otevření malého herního menu. Při následném uzavření těchto containerů se testujícímu nezpřistupnila možnost používání nástrojů a nebylo tak možné splnit další úkoly potřebné k dokončení testu. Tato chyba je nejspíše způsobena špatnou orchestrací otevírání a zavírání UI containerů, které spravuje třída *ToolController*.

- Nástroj pro měření vzdálenosti – poslední nalezenou chybou bylo nechtěné aktivování nástroje pro měření vzdálenosti ve chvíli, kdy uživatel tento nástroj deaktivoval na jedné z rukou. K aktivaci došlo při otevření a následném zavření libovolného UI containeru. Chyba je způsobena neověřováním dostupnosti nástroje při pokusu o jeho aktivaci. Tato chyba neměla přímý vliv na testování a jednalo se tak spíše o znepříjemění a zmatení testovaného uživatele.

5.3.2 Problémy s ovládáním

Dalšími problémy nalezenými při testovaní jsou problémy s ovládáním. Tyto problémy mohou být subjektivní, ale s některými prvky ovládání měli problém téměř všichni uživatelé, a proto má smysl přizpůsobit tyto aspekty jejich očekávání.

- Výběr nástrojů – zobrazení tohoto výběru, implementovaného pomocí kruhového menu, probíhá pomocí držení menu tlačítka a potvrzení výběru pomocí jeho uvolnění. Většina testujících však předpokládala zobrazení tohoto menu při prvním dokončeném kliknutí na tlačítko a potvrzení opětovným zmáčknutím tlačítka. Díku těmto předpokladům často docházelo k vybrání nechtěného nástroje a tím i ke zmatení testujícího. Výběr nástrojů je tedy potřeba upravit a rozdělit jeho zobrazení a potvrzení výběru na dvě akce.

- Dvojité kliknutí – kvůli propojení projektu s projektem Bc. Matěje Sedláka, bylo potřeba sjednotit ovládání a zajistit, aby nedocházelo ke kolizi při ovládání pomocí stejného tlačítka na ovladači. Z tohoto důvodu bylo zobrazení inventáře nastaveno na dvojité kliknutí na tlačítko spouště. Aplikace však dvojité stisknutí špatně detekovala a často místo dvojitého kliknutí detekovala dvě oddělená kliknutí, která tak vyvolala jinou akci. Tento problém je způsoben špatnou detekcí dvojitého kliknutí, o které se stará sama knihovna SteamVR. Knihovna však dovoluje nastavení parametrů pro detekci akcí a pomocí těchto parametrů by tak mohlo být možné tento problém vyřešit. Pokud však bude problém přetrvávat, bude nutné vytvořit vlastní systém pro detekci dvojitého kliknutí a z ovladačů detekovat pouze normální kliknutí.

- Viditelnosti UI zkrze objekty – při používání UI containerů ve volném prostoru není problém s jejich zobrazením ani interakcí. Pokud se však nachází objekt na místě, na kterém se má UI zobrazit, není UI dobře viditelné a objekty tak zasahují do výhledu a dokonce znemožňují iterakci s elementy uživatelského rozhraní. Uživatel tento problém musí řešit přesunem na místo, kde objekty nebudou do UI elementů zasahovat, což může být při použití v malých prostorách velmi obtížné. Vhodným řešením by tedy bylo zajistit, aby prvky UI mohly být viditelné skrze objekty. Dále je nutné při aktivaci těchto UI elementů zajistit, aby komunikace pomocí třídy *Raycast* probíhala jen s UI elementy a paprsek tak procházel ostatními objekty. Také by bylo vhodné při zobrazení velkých UI elementů, které zasahují přímo do scény, znemožnit uživateli pohyb a tím zabránit chybám při interakci se zobrazeným UI.

- Viditelnost výsledků měření – při zobrazení výsledku měření, jak mezi měřenými objekty tak v textovém poli u ovladačů, se text stává na

černém pozadí špatně čitelný. Bylo by vhodné tento problém řešit přidáním pozadí či ohraničení zobrazovanému textu.

5.3.3 Návrhy na vylepšení

Nejedná se přímo o problémy, ale o podnětné návrhy pro vylepšení či zjednodušení používání nástrojů a UI. Tyto návrhy mohou být velmi subjektivní, ale některé z nich jsou zajímavé a určitě by stálo za to je implementovat.

- Lupa – při používání lupy neměli testující velké problémy, ale přišli s několika návrhy pro vylepšení a zpřehlednění jeho používání. Prvním návrhem byla stabilizace pohledu lupy. Při velkém přiblížení se kvůli nemožnosti držení ovladače ve stabilní pozici pohled kamery, představující lupu, velmi se klepe a lehce tak ztěžuje sledování objektů. Bylo by tedy dobré implementovat stabilizaci kamery představující lupu a zpříjemnit tak uživatelům její použití. Druhým návrhem bylo zobrazení aktuálního stavu přiblížení lupy. Uživatel by poté vždy jasně věděl jak moc má pohled přiblížený a zároveň jak moc ještě přibližovat může. Toto zobrazení by bylo ideální implementovat pomocí grafické lišty vedle objektu lupy.
- Měření vzdálenosti – při používání nástroje pro měření vzdálenosti se uživatelům zobrazuje pouze číslo představující vzdálenost. Někteří uživatelé se však dotazovali na jednotku měření. Bylo by tedy vhodné k měřenému číslu přidat i aktuální jednotku měření.

5.4 Úprava implementace

Problémy nalezené při testování, definované jako chyby implementace, bylo potřeba bezpodmínečně opravit. Jejich oprava proběhla bez větších problémů a byla provedena mezi jednotlivými testy, aby se ověřila správnost řešení problému. Úprava akcí pro výběr nástrojů proběhla jednoduchým rozdělením otevření a zavření menu na dvě oddělené akce. Dvojité kliknutí bylo upraveno pomocí parametrů nastavitelných v knihovně SteamVR a po úpravě již testující neměli problémy s otevřením inventáře pomocí dvojitého kliknutí na spoušť ovladače. Zároveň při řešení těchto problémů byl lehce upraven systém pro správu aktivních UI containerů a bylo zakázáno otevírat více těchto UI containerů najednou. Toto řešení zamezuje nechtěným otevřením UI containerů při interakci a zároveň řeší problém s rozhodováním o směru paprsku s ovladačem při otevření dvou UI s odlišným přístupem ovladání.

5.4.1 Viditelnost UI skrze objekty

Dalším řešeným problémem byla viditelnost UI skrze jiné objekty a s ním spojená iterakce. Pro viditelnost UI elementů skrze objekty byl vytvořen *Shader*,

který vycházel z popisu na webu [25] a z něho vytvořený materiál byl nastaven všem UI elementům, které mají být vždy viditelné. Dále bylo potřeba vyřešit interakci skrze dané objekty. Ve třídě *Raycaster*, která se stará o vytváření paprsků a detekci objektů byla vytvořena možnost změny typu paprsku, který interaguje pouze s objekty patřící do vrstvy UI a ignoruje všechny ostatní objekty. Při otevření UI containeru či zobrazení inventáře pro výběr textury je interakce přepnuta na tento typ a je tak možné s UI interagovat skrze jiné objekty.

5.4.2 Lupa

Podle návrhů pro vylepšení nástroje lupy byla vedle modelu samotné lupy přidána lišta zobrazující aktuální stav přiblížení, jak lze vidět na obrázku 5.1. Z důvodu přehlednosti byl také při aktivaci nástroje vypnut paprsek jdoucí z ovladače. Problém stabilizace byl vyřešen pomocí přidání akce pro aktivaci stabilizace. Po zapnutí stabilizace se kamera zobrazující obraz na model lupy ustálí na definovaném místě. Při ustálení kamery je možné stále přibližovat i oddalovat. Problém nastal, pokud uživatel přesunul ovladač s modelem lupy do výhledu kamery, na které se tak objevil nekonečný cyklus pohledů kamery. Tento problém byl vyřešen pomocí přidání objektů lupy do vlastní vrstvy objektů a kameře byla nastavena ignorace objektů z této vrstvy. Do budoucna by bylo možné vyrvořit vlastní stabilizační algoritmus, který by sám detekoval začátek stabilizace a upravoval míru stabilizace na základě velikosti přiblížení.

Obrázek 5.1: Upravená implementace lupy.

5.5 Možnosti dalšího vývoje

Při dalším vývoji UI pro virtuální realitu by bylo možné vytvořit další containery, do kterých by se mohly vkládat UI elementy. Bylo by možné vytvořit containery, které by byly umístěné na částech těla uživatele, například na předloktí či u pasu. Tyto UI containery by bylo možné zobrazovat nejen pomocí stisknutí tlačítka, ale pomocí detekce pohledu na dané místo. Dále by bylo možné přidat pokročilejší UI elementy, jako je například posuvný seznam. Bylo by možné vytvořit několik předdefinovaných motivů UI prvků a vývojář, který by chtěl implementované UI použít, by si mohl vybrat z těchto motivů a nemusel by design UI prvků řešit sám. Další možností rozšíření aplikace by mohlo být přidání nástrojů pro změnu dalších parametrů objektů, jako je například reakce na gravitaci. Taky by bylo možné přidat další speciální nástroje jako je řez objektem či měření malování ve 3D prostoru.

5.6 Vytvoření a zveřejnění výsledného balíčku

Po dokončení úpravy implementace bylo potřeba projekt připravit pro zveřejnění na *Unity Assetstore*. Cílem vytvořeného balíčku je především jednoduché použití vývojáři a proto byla většina funkcionalit zabalena do připravených objektů (prefabů), obsahujících potřebné reference. Uživatel si tak tyto připravené objekty pouze přesune do svého projektu a nastaví potřebné proměnné, jako jsou akce ovladače, jelikož tyto akce si každý vývojář sám definuje v knihovně SteamVR. Byl vytovřen jeden objekt představující veškeré dostupné nástroje a UI elementy a také druhý, který obsahuje pouze implementaci UI pro případ, že by vývojář chtěl využít pouze úživatelské rozhraní bez nástrojů. Zároveň byl vytvořen prefab pro každý nástroj, UI container a UI element, aby si vývojář mohl vše přizpůsobit tak jak potřebuje. Součástí balíčku je také testovací scéna, která představuje veškeré dostupné nástroje a prvky UI v připravené formě, aby si vývojář mohl nástroje a UI vyzkoušet. Poslední součástí balíčku je dokumentace obsahující veškeré důležité informace potřebné k použití implementovaných UI elementů a nástrojů. Součástí této dokumentace je i krátké video [26] představující součásti balíčku a návod pro použití, jelikož spousta vývojářů preferuje krátké video před pročítáním dokumentace. Tento výsledný balíček byl zveřejněn zdarma na *Unity Assetstore* (náhled lze vidět na obrázku 5.2), kde si ho bude moci stáhnout a použít jakýkoliv vývojář na platformě Unity. Postup zveřejnění výsledného balíčku byl komplikovanější, než jsem očekával a zahrnoval několik kroků, které musely být provedeny.

- Pro sdílení balíčků, bylo potřeba vytvořit nový publisher účet, který však bylo možné vytvořit formou povýšení běžného již registrovaného účtu na portálu Unity.

- Upload samotných assetů probíhá pomocí staženého nástroje pro Unity,

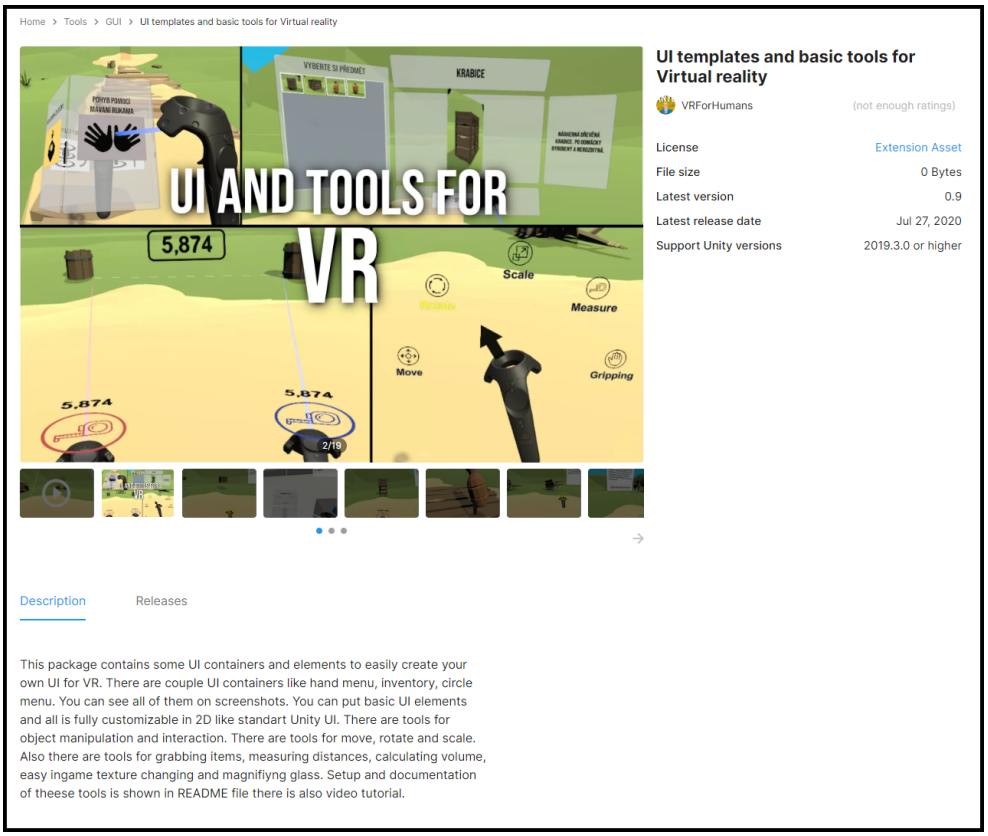
který dovoluje nahrávat soubory na Assetstore. Pomocí propojení Unity s registrovaným publisher účtem bylo možné vytvořit nový prázdný balíček a k němu pomocí staženého nástroje připojit veškerý obsah, který bude balíček využívat. Před samotným uploadem bylo ještě potřebné provést kontrolu obsahu daného balíčku. Tato kontrola se provádí automaticky a sleduje, jestli se v připravených souborech nenachází poškozené soubory, či soubory s neplatnou příponou. Pro nahrání, bylo například nutné veškeré obrázky ve formátu JPG převést do formátu PNG. Součástí kontroly je také zjištění, jestli balíček obsahuje dokumentaci, která může být ve formátu PDF, nebo RTF.

- Na webovém portálu bylo poté potřeba nastavit veškeré údaje o daném balíčku. Kromě názvu a popisu obsahu, bylo potřeba dodat i snímky z implementovaných funkcí, jejichž součástí mohlo být i video. Pro lepší prezentaci tak bylo natočeno ukázkové video, které obsahovalo záznam obrazovky při průchodu testovací aplikace, ve které byly přehledně představeny veškeré nástroje pro manipulaci a interakci s objekty, stejně jako hlavní prvky uživatelského rozhraní. Pro účely správného zobrazení v samotném obchodu bylo potřeba ještě vytvořit náhledové obrázky v předem daných velikostech, které byly vytvořeny z vybraných snímků pořízených při testování.

- Poslední částí před samotnou publikací bylo popsání informací o dostupnosti formou podporovaných verzí Unity, což v mém případě byly veškeré verze z roku 2019, na kterých byla aplikace vyvíjena, nebo novější díky dobré zpětné kompatibilitě. Součástí informací byl i seznam závislostí na jiných balíčcích nabízených na Assetstore, což v mém případě byl pouze SteamVR. Před potvrzením publikace bylo potřeba potvrdit pomocí zaškrtávacího políčka, že na všechny assety obsažené v balíčku máme právo pro sdílení a neporušujeme tak licenční podmínky.

- Po potvrzení publikování jsme byli informováni, že každý nový balíček musí projít ruční kontrolou ze strany správy samotného Unity. Jelikož počet nových publikovaných balíčku každý den přesahuje stovku a při kontrole se zaměřují na kvalitní přístup, tak potvrzení publikování může trvat více než několik dní.

Jelikož samotnému potvrzení publikování či vyjádření o stavu kontroly ze strany Unity nedošlo dříve než byl termín odevzdání diplomové práce, není možné přiložit k diplomové práci referenci na samotný balíček na Assetstore. Po dokončení publikace však díky názvu zobrazenému na obrázku 5.2 bude jednoduché si na stránkách Assetstore pro platformu Unity balíček dohledat a zároveň veškeré součásti balíčku budou obsaženy v přiloženém CD.



Obrázek 5.2: Náhled vytvořeného balíčku pro Unity Assetstore.

Závěr

V této práci byly navrženy a implementovány základní prvky uživatelského rozhraní pro virtuální realitu a nástroje pro manipulaci a interakci s objekty ve virtuální realitě. Návrh a implementace vycházely z analýzy aktuálních aplikací a dostupných materiálů věnujících se tomuto tématu. Součástí implementace je sada možností umístění obrazovek do 3D prostředí, do kterých je možné vkládat připravené UI elementy a vytvořit tak funkční a přehledné uživatelské rozhraní pro virtuální realitu. S implementovaným UI se pracuje obdobně jako s klasickým UI na platformě Unity a proto je použití implementace velmi jednoduché. Implementace zahrnuje i interakci s tímto UI pomocí ovladačů, které jsou součástí sady pro virtuální realitu a možnosti zadávání textu pomocí klávesnice. Uživatelské rozhraní je možné plně upravovat jak po stránce rozložení a umístění, tak po stránce vzhledu. Druhou implementovanou částí jsou nástroje pro manipulaci a interakci s předměty se zaměřením na práci s historickými artefakty. Implementace zahrnuje správu těchto nástrojů s možností přepínání mezi nástroji pomocí vytvořeného otočného menu. Mezi dostupné nástroje pro změnu vlastností objektů patří základní možnosti transformace pozice, velikosti a rotace objektů a nástroj pro změnu textury. Pro měření vlastností objektů byl vytvořen nástroj pro měření objemu, nástroj pro měření vzdálenosti a lupa s možností úpravy přiblížení. Součástí implementovaných nástrojů je i možnost uchopení objektů do ruky a jejich manipulace pomocí této interakce. Veškeré prvky UI a nástroje byly testovány pomocí připravené testovací scény formou krátkých úkolů. Po vyhodnocení výsledků testování byla implementace na základě těchto výsledků upravena do výsledné podoby. Součástí úpravy byla oprava nalezených chyb, upřesnění základní konfigurace ovládání a přidání nových funkcionalit na základě návrhů od testovaných uživatelů.

Celá aplikace byla vyvíjena pro účely dalšího použití pro širokou veřejnost pomocí publikování implementovaných částí na obchod s komponenty pro platformu Unity, kde si jakýkoliv vývojář na tuto platformu může zdarma implementaci stáhnout a použít ji ve své aplikaci. Z implementované apli-

kace byl vytvořen balíček, který obsahuje veškerou implementaci s příslušnou dokumentací a videonávodem pro nastavení a použití. Tento balíček bude po dokončení schvalovacího procesu veřejně dostupný na stránkách obchodu s komponenty na platformě Unity.

Aplikaci by bylo možné rozšířit o další containery a o pokročilejší UI elementy, jako jsou například posuvné seznamy. Bylo by také možné vytvořit několik motivů pro všechny UI elementy a usnadnit tak vývojáři, který bude implementaci používat ve své aplikaci, úpravu vzhledu uživatelkého rozhraní. Bylo by možné vytvořit další nástroje pro interakci či transformaci objektů, například pro změnu fyziky objektu či kreslení ve 3D prostoru. Součástí zadání byl i návrh na implementaci nástroje pro řez objektem, ale po pozdější konzultaci s vedoucím práce byl tento nástroj z návrhu vynechán. Při dalším rozšiřování aplikace by bylo možné tento nástroj přidat.

Literatura

[1] Oculus Rift a HTC Vive v jedné místnosti a na plný plyn, virtuální realita umí nadchnout. *tech.ihned.cz - Hospodářské noviny [online]*, [cit. 2020-05-06]. Dostupné z: https://tech.ihned.cz/pocitace/c1-65138990-virtualni-realita-v-brne-oculus-rift-a-htc-vive-nadchly-a-skoro-rozplakaly-cenou

[2] Steam VR Template - Unreal Engine Forums . *Forum - Unreal Engine Forums. [online]*, [cit. 2020-07-15]. Dostupné z: https://forums.unrealengine.com/development-discussion/vr-ar-development/78620-steam-vr-template?106609-Steam-VR-Template=

[3] VR Headsets on Steam Reach 1.7 Million - AR Insider. *AR Insider [online]*, [cit. 2020-07-19]. Dostupné z: https://arinsider.co/2020/04/13/vr-headsets-on-steam-reach-1-7-million/

[4] Burdea, G. C.; Coiffet, P.: *Virtual reality technology*. John Wiley & Sons, 2003.

[5] What are the best game engines for Virtual Reality development? *Slant.co - TRUSTWORTHY PRODUCT RANKINGS FOR ALL YOUR SHOPPING NEEDS [online]*, [cit. 2020-06-06]. Dostupné z: https://www.slant.co/topics/2202/~best-game-engines-for-virtual-reality-development

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

[7] Glover, J.; Linowes, J.: *Complete Virtual Reality and Augmented Reality Development with Unity: Leverage the power of Unity and become a pro at creating mixed reality applications*. Packt Publishing Ltd, 2019.

[8] 10 Graphically INSANE Levels Created In UNREAL ENGINE 4 - YouTube. *YouTube. Copyright © 2020 Google LLC [online]*, [cit. 2020-07-21]. Dostupné z: https://www.youtube.com/watch?v=QVbRO-6MFV4

[9] Godot Engine - Free and open source 2D and 3D game engine. *Godot Engine [online]*, [cit. 2020-07-21]. Dostupné z: https://godotengine.org/

[10] SteamVR on Steam. *Steam. Copyright © 2020 Valve Corporation. [online]*, [cit. 2020-05-06]. Dostupné z: https://store.steampowered.com/app/250820/SteamVR/

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

[12] VIVE™ — Buy VIVE Hardware. *VIVE Copyright © 2011 [online]*, [cit. 2020-05-06]. Dostupné z: https://www.vive.com/sea/product/vive/

[13] SteamVR's Lighthouse for Virtual Reality and Beyond - YouTube. *YouTube. Copyright © 2020 Google LLC [online]*, [cit. 2020-07-15]. Dostupné z: https://www.youtube.com/watch?time_continue=33&v=xrsUMEbLtOs

[14] Designing Screen Interfaces for VR (Google I/O '17) - YouTube. *YouTube. Copyright © 2020 Google LLC [online]*, [cit. 2020-07-15]. Dostupné z: https://www.youtube.com/watch?v=ES9jArHRFHQ

[15] 8 Ways to Create a Better UX in Virtual Reality — Learning Solutions Magazine. *Learning Solutions Home [online]*, [cit. 2020-07-15]. Dostupné z: https://learningsolutionsmag.com/articles/8-ways-to-create-a-better-ux-in-virtual-reality

[16] Hot Dogs, Horseshoes Hand Grenades. *Steam. Copyright © 2020 Valve Corporation. [online]*, [cit. 2020-07-15]. Dostupné z: https://store.steampowered.com/app/450540/Hot_Dogs_Horseshoes__Hand_Grenades/

[17] Neos VR. *Steam. Copyright © 2020 Valve Corporation. [online]*, [cit. 2020-07-15]. Dostupné z: https://store.steampowered.com/app/740250/Neos_VR/

[18] Kingspray Graffiti VR. *Steam. Copyright © 2020 Valve Corporation. [online]*, [cit. 2020-07-15]. Dostupné z: https://store.steampowered.com/app/471660/Kingspray_Graffiti_VR/

[19] Tilt Brush by Google. *Steam. Copyright © 2020 Valve Corporation. [online]*, [cit. 2020-07-15]. Dostupné z: https://store.steampowered.com/app/327140/Tilt_Brush/

[20] Visual Studio. *Visual Studio IDE, Code Editor, Azure DevOps, App Center. [online]*, [cit. 2020-07-15]. Dostupné z: https://visualstudio.microsoft.com/cs/

[21] Petiúhelník – Wikipedie. *Wikipedie [online]*, [cit. 2020-07-08]. Dostupné z: https://cs.wikipedia.org/wiki/Petiuhelnik

[22] Osmiúhelník – Wikipedie. *Wikipedie [online]*, [cit. 2020-07-08]. Dostupné z: https://cs.wikipedia.org/wiki/Osmiuhelnik

[23] How would one calculate a 3d Mesh volume in Unity? *Unity Answers [online]*, říjen 2018, [cit. 2020-07-08]. Dostupné z: https://answers.unity.com/questions/52664/how-would-one-calculate-a-3d-mesh-volume-in-unity.html

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

[25] How to make World Space UI always visible in Unity? *Unity Answers [online]*, [cit. 2020-07-08]. Dostupné z: https://stackoverflow.com/questions/36223736/how-to-make-world-space-ui-always-visible-in-unity

[26] UI and Tools for VR setup tutorial - YouTube. *YouTube. Copyright © 2020 Google LLC [online]*, [cit. 2020-07-21]. Dostupné z: https://www.youtube.com/watch?v=FeisxYio3Zg&feature=youtu.be

Příloha **A**

Seznam použitých zkratek

UI User interface

VR Virtual reality

Příloha **B**

Obsah přiloženého DVD

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