VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

EDITOR KOOPERATIVNÍHO PROSTŘEDÍ VE VR

EDITOR OF COOPERATIVE VR ENVIRONMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MICHAEL SCHNEIDER**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. TOMÁŠ MILET, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Schneider Michael, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačová grafika a interakce
Název: **Editor kooperativního prostředí ve VR**
Editor of Cooperative VR Environment
Kategorie: Počítačová grafika

Zadání:

1. Nastudujte si herní engine Unity a Unreal, nastudujte si techniky virtuální reality a nastudujte si editory 3D prostředí.
2. Navrhněte editor 3D prostředí pro virtuální realitu pro spolupracující uživatele. Zaměřte se na jednoduché a intuitivní ovládání a obecnost a rozšířitelnost editoru.
3. Implementujte navržený editor.
4. Vytvořte několik demonstračních úrovní.
5. Uživatelsky ověřte a zhodnoťte výsledky.
6. Vytvořte demonstrační video.

Literatura:

- Kevin Mack, Robert Ruud.: Unreal Engine 4 Virtual Reality Projects: Build immersive, real-world VR applications using UE4, C++, and Unreal Blueprints.Packt Publishing Ltd, 2019. ISBN-13: 978-1789132878.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2 a kostra aplikace.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Milet Tomáš, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Cílem této práce je vytvoření editoru v herním enginu Unity, ve kterém lze jednoduše sestavit svět pro virtuální realitu.

Práce seznamuje s aktuálními VR technologiemi a jejich omezeními. Také se zabývá návrhem a implementací VR aplikace v Unity, včetně potřebných pluginů. Součástí řešení je i VR multiplayer, který umožňuje sdílet vytvořený svět s ostatními hráči VR.

Pro funkčnost takovéto aplikace bylo potřeba vytvořit komplexní systémy pro správu prostoru, dále vytvořit implementaci serveru, který podporuje klienta ovládaného pomocí virtuální reality a systémy interakcí, aby svět okolo hráčů zůstal interaktivní.

Abstract

The goal of this work is to create an editor in the Unity game engine, in which you can easily set up a world for virtual reality.

The work introduces current VR technologies and their limitations. The thesis deals with the design and implementation of a VR application in Unity, including the necessary plugins. The solution also includes VR multiplayer, which allows you to create a world with other VR players.

For such an application to work, it was necessary to create complex space management systems, create a server implementation that supports a client controlled by virtual reality and interaction systems, so that the world around the players remained interactive.

Klíčová slova

Virtuální realita, Unity, editor, voxelový prostor, kooperace, herní engine, multiplayer

Keywords

Virtual reality, Unity, editor, voxel space, cooperation, game engine, multiplayer

Citace

SCHNEIDER, Michael. *Editor kooperativního prostředí ve VR*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet, Ph.D.

Editor kooperativního prostředí ve VR

Prohlášení

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

.....
Michael Schneider
18. května 2022

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce panu Ing. Tomáši Miletovi, Ph.D. za odborné rady, podnětné připomínky, nápady, čas, vstřícnost, trpělivost a ochotu, kterou mi během zpracování diplomové práce věnoval. Děkuji rovněž rodině za podporu během psaní diplomové práce.

Obsah

Kapitola 1

Úvod

Téma diplomové práce jsem si zvolil z toho důvodu, že jsem již bakalářskou práci psal na podobné téma. Bakalářskou práci s názvem „Inovativní herní demo v Unity“ [13] jsem rozšířil o herní editor. Díky tomu hráči dostali prostor využít svoji fantazii a vytvářet tak obsah i po vydaní. V dnešní době existují mnohá herní studia, která do svých her editory implementují. Takové hry si pak mnohem déle udrží aktivní komunitu, jelikož neustále přibývá nový obsah nezávisle na vývojáři.

Herní editor v této diplomové práci umožnuje tvořit hry ve virtuální realitě. Virtuální realita nabírá na popularitě. Většina lidí, kteří si ji vyzkouší alespoň v podobě brýlí, se ocitne ve světě, který dříve mohl existovat pouze v myšlenkách a fantaziích. Každý má ale svůj svět vymyšlený malinko jinak. Pro člověka, který neovládá programování, není v současné době moc způsobů, jak by si svůj svět mohl vytvořit.

Koncept editoru se stal základem pro tuto práci. Konkrétně je nachystán tak, aby si na počítači každý vytvořil svět a do něj mohl vstoupit za pomoci brýlí virtuální reality.

Tento editor umožní bez složitého programovaní uživatelům svůj svět vytvořit, pohybovat se v něm, a dokonce ho sdílet s přáteli v kooperativním režimu. Lze se tak v něm setkávat, vytvářet společné události a hrát si v něm.

Editor poslouží například v situaci, kdy lze ukázat virtuální realitu členům domácnosti s možností zasahovat do světa z pohledu počítače. Na počítači zůstává otevřená možnost editace během hraní v daném světě. Lze tak vytvářet četné scénáře, živě přidávat do prostředí různé hádanky, překážky a překvapení pro uživatele. Lze tak přímo ovlivňovat zážitek hrající osoby.

Díky tomuto by se editor dal i využít například ke zkoumání reakcí lidí na různé situace. Konkrétně si to lze představit na příkladu, kdy se uživatel snaží zbavit nějakého strachu. Lze si postupně dávkovat situace a strachu čelit vhodným tempem. Tento nástroj by mohl být tedy i potenciálně přínosný při zkoumání reakcí lidí na různé podněty.

Na trhu je dnes několik velkých společností, které vytvářejí velké statické světy s rozsáhlými místnostmi. Tyto místnosti fungují jako velké sociální platformy. Naopak vytvořená aplikace cílí na zážitek malé skupiny lidí ve velmi dynamicky se měnícím světě. Dalo by se to přirovnat k hraní dračího doupěte, ale ve virtuální realitě, kdy má vypravěč plnou kontrolu nad dějem okolo hráčů.

V aplikaci jsou využity pluginy, které se stanou na dlouhou dobu standardem. Je zde vysvětleno, jak tyto pluginy fungují a jak s nimi pracovat. Samotný hardware se také aktivně vyvíjí. Zmiňuji se v práci o jeho limitacích a co je potřeba dodržet, aby virtuální zážitek byl příjemnou událostí. Kinetóza je opravdový problém, kterému je potřeba se vyhnout.

K naprogramování tohoto editoru mě přivedlo samotné představování virtuální reality členům rodiny a kamarádům. Vždy, když si headset poprvé někdo nasadí, musí si nejprve zvyknout na prostředí, způsob pohybu a další věci. V práci také rozebírám problémy spojené s virtuální realitou, jako je Kinetóza. Virtuální realita se musí zpočátku dávkovat, a i když aplikace jsou hodnocené podle náročnosti, žádná neumožnovala tuto náročnost za běhu měnit. Proto jsem potřeboval aplikaci, kde bych řídil dění a systémy, které hráč může používat tak, aby vstup do světa VR byl co nejpohodlnější a pro každého v jeho vlastním tempu.

Z předešlého vývoje her a dalších aplikací mám bohaté zkušenosti s Unity, a to i s vývojem klasických virtuálních aplikací. Proto je celá práce napsaná v Unity. Snažím se srozumitelně popsat některé správné praktiky, které by se měly při vývoji pro virtuální realitu dodržovat. Je využito technologií, které jsou aktivně vyvíjeny během vytváření této práce.

Celá práce je rozdělena do pěti kapitol.

Ve 2. kapitole je představení editoru. Seznámení se základním rozdělením samotné aplikace a ukázky z aplikace.

Ve 3. kapitole je uvedeno, jaké technologie virtuální reality jsou dnes k dispozici, jak vznikly a jaké je omezení virtuální reality.

Ve 4. kapitole jsou popsány jednotlivé systémy a použití pluginů k vytvoření editoru.

V 5. kapitole jsou vysvětleny jednotlivé součásti aplikace.

Kapitola 6. obsahuje výběr zajímavých řešení, které bylo potřeba naimplementovat.

Poslední kapitola 7. shrnuje aktuální stav práce a nastiňuje cestu pro její rozšíření.

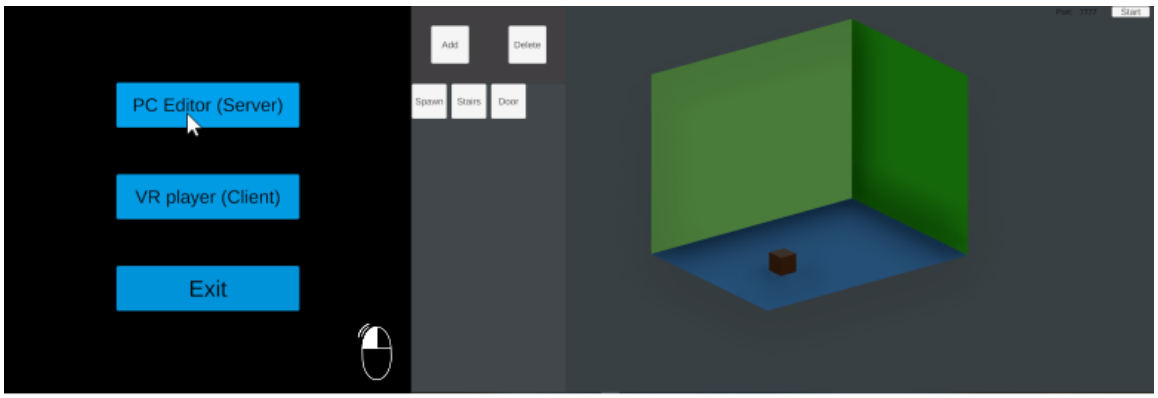
Kapitola 2

Představení produktu

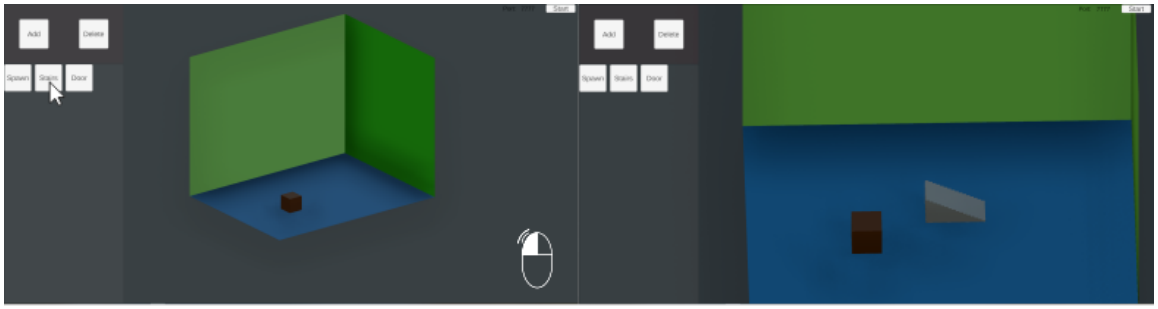
Aplikace umožňuje vytvoření svého vlastního světa. Tento svět lze ovládat přímo z editoru i při spuštění. Hráči se do něj připojují za použití virtuální reality. Při spuštění aplikace funguje v jednom ze dvou režimů, a to jako editor (server, 2D) nebo jako hráč (klient, VR). Na jednom počítači může běžet jak server, tak klient. Je potřeba aplikaci spustit dvakrát. V okně na popředí bude server, kde administrativní hráč ovládá místnost. Ve druhém okně na pozadí bude spuštěn klient, který je ovládán pomocí VR. Jelikož aplikace nezablokuje výhradní přístup k uživatelskému vstupu, lze spustit druhou instanci aplikace.

Aplikace přináší jednoduché vytvoření virtuálního světa, dynamický svět, který řídí jeho stvořitel a umožňuje postupně dávkovat náročnost virtuální reality. Pro nezkušené uživatele tak zmírňuje dopady kinetózy. Díky kooperaci si lze užít zábavu ve vytvořeném světě i s kamarády.

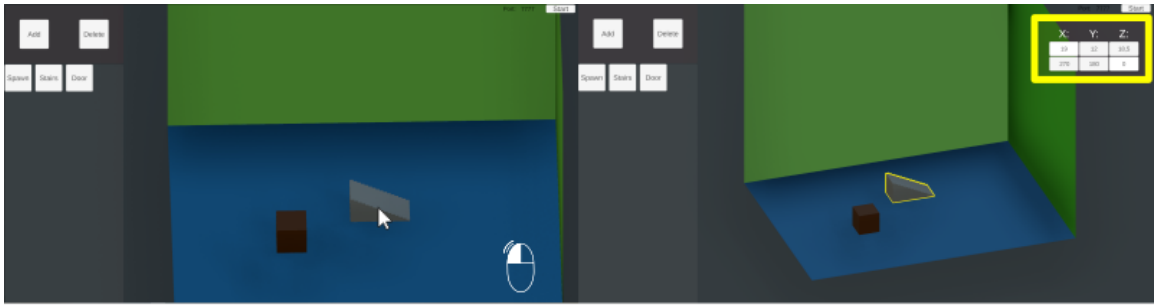
Aplikaci zvládne používat každý hráč. Samotný editor vychází z principu drag and place. Podobný editor se nachází například v sérii her Portal, ze které byla čerpána inspirace. Na obrázcích 2.1 až 2.7 se nacházejí příklady použití aplikace.



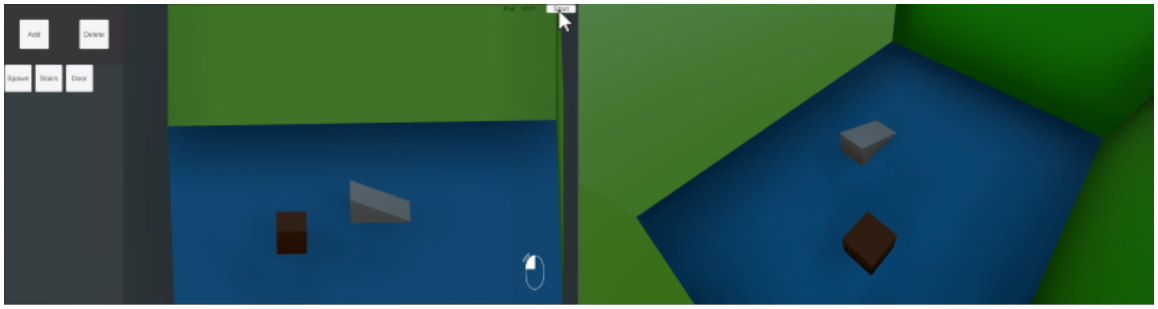
Obrázek 2.1: Po stisknutí první volby v menu se spustí editor, jedná se o režim administrativního hráče.



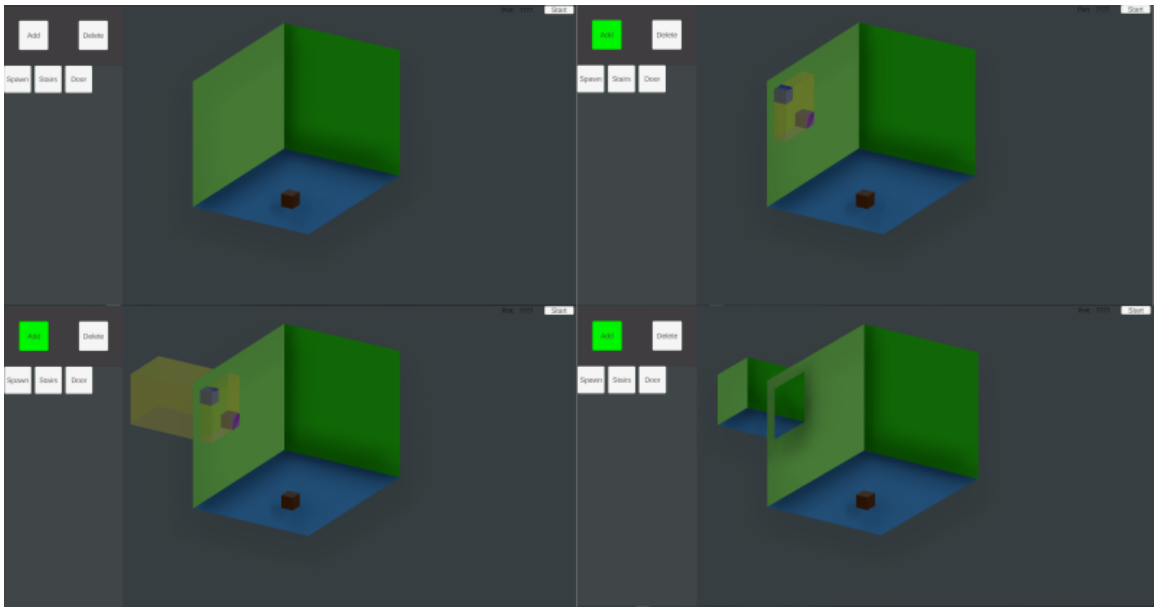
Obrázek 2.2: V levém menu se vybírá objekt, který se po kliknutí na něj objeví ve scéně.



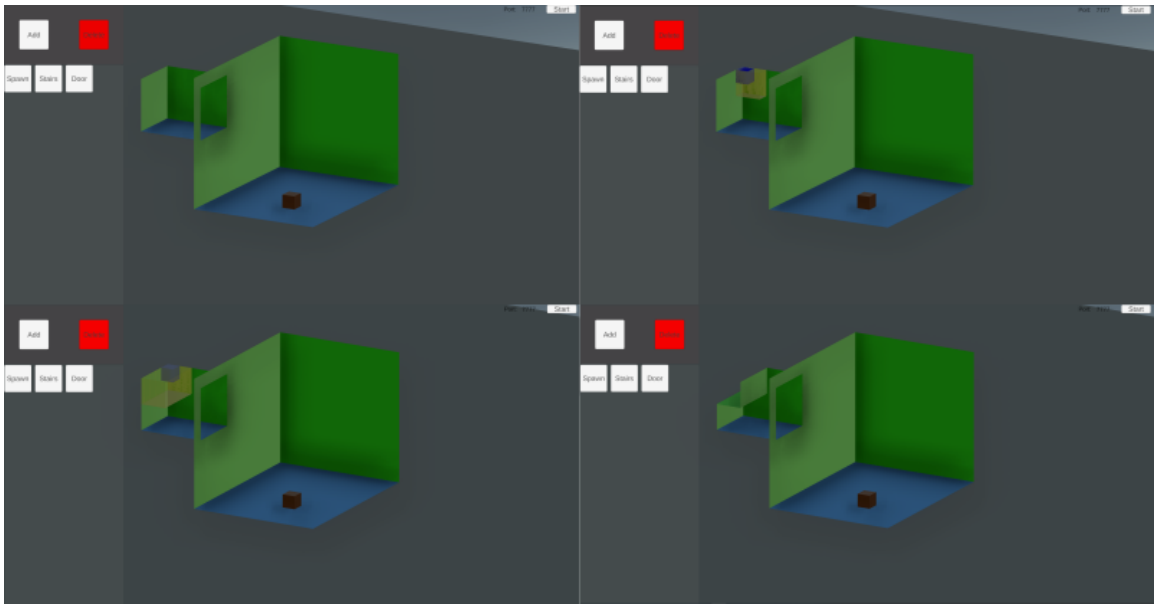
Obrázek 2.3: Po kliknutí na objekt ve scéně se objeví v inspektoru jeho hodnoty, které lze upravit.



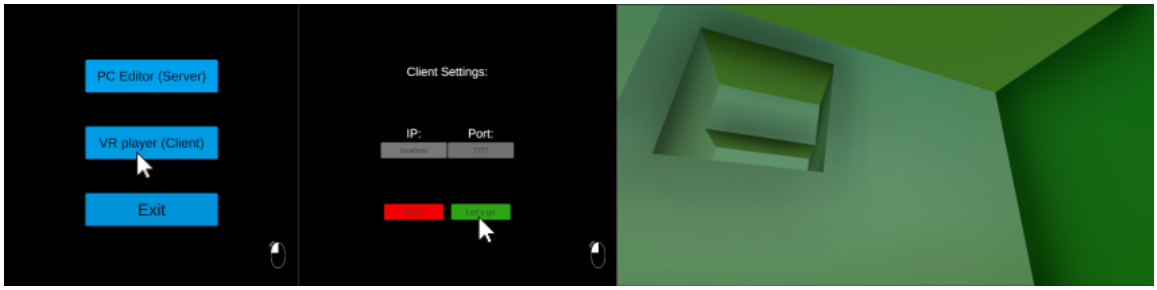
Obrázek 2.4: Po zadání portu v pravém horním rohu lze spustit server. Po puštění serveru se editor přepíná do režimu administrativního hráče.



Obrázek 2.5: Funkce Add umožňuje třífázově vybrat oblast, do které se místnost rozšíří.



Obrázek 2.6: Mazání voxelů místnosti. Při mazání se opět třífázově vybere oblast, kde se místnost ruší.



Obrázek 2.7: Připojení jako klient. Po připojení se inicializuje VR. A hráč je vytvořen v herní místnosti.

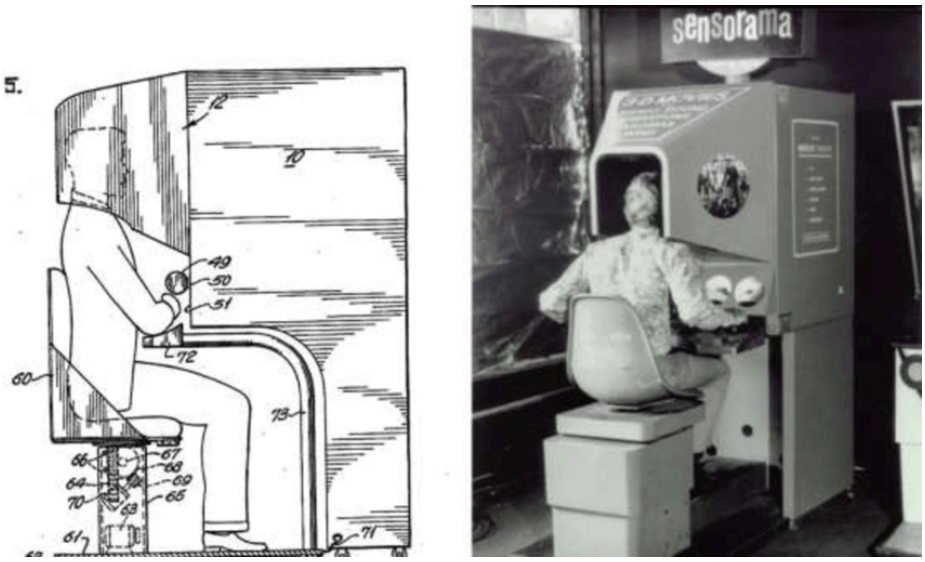
Kapitola 3

Technologie VR

Technologie VR, neboli virtuální realita, má za cíl přenést hráče do virtualního prostředí a v tomto prostředí odstínit hráče zcela od světa reálného. Pod pojmem Virtuální realita si nelze představit pouze brýle, v literatuře často označované jako HMD - head mounted display. Ale také ovladače, rukavice, vesty, threadmilly a spoustu dalších zařízení. Všechna tato zařízení společně umocňují vjemy z virtualního prostředí, ve kterém se hráč nachází.

3.1 Historie

První experimenty s virtualní realitou se datují do 50.let 20. století. První náčrt se objevuje roku 1955, a to v dokumentu The Cinema of the Future[7], jehož autorem byl Morton Heilig. Morton Heilig byl slavný producent, spisovatel a kameraman v Hollywoodu. Díky zkušenostem v oboru filmu se mu podařilo sestavit první funkční prototyp virtualní reality, nazývaný Sensorama. Sensorama byla stavěna od roku 1957 a v roce 1962 byla patentována.



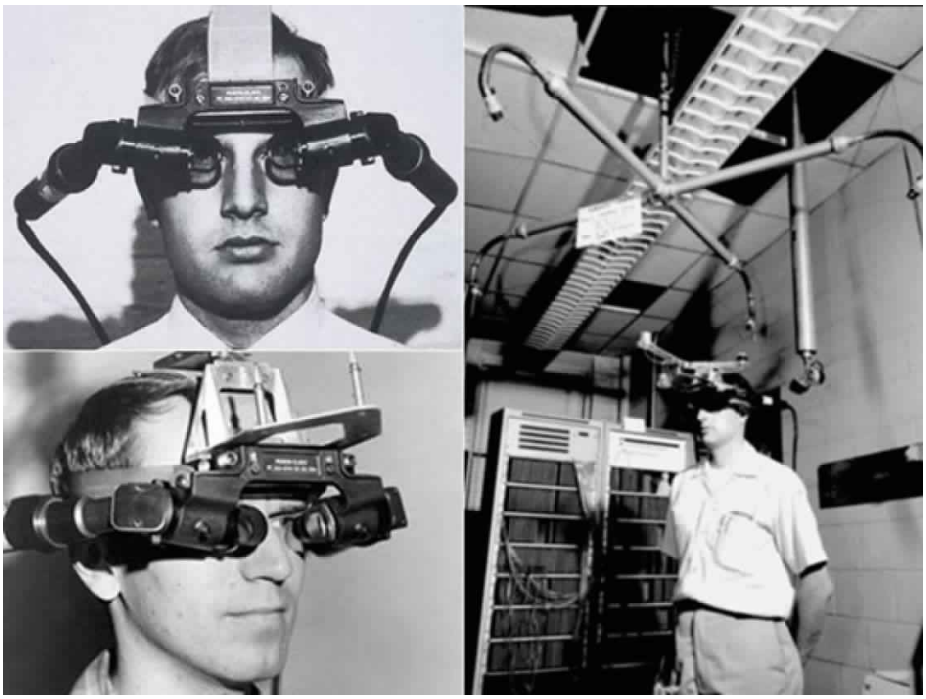
Obrázek 3.1: Senzorama. Zdroj: [17].

Senzorama byla vybavena 3D displayem, stereo reproduktory, generátorem vůní a pohyblivou židlí. Jeden ze snímků, který se na Senzoramě přehrával, byl například průjezd

New Yorkem. Vibrace židle v kombinaci s větráky simulovaly sezení na motorce. K tomu se generovala vůně pizzy a výfukových plynů z okolo projíždějcího autobusu.

Zvažovalo se několik použití. Buď jako arkáda nebo jako showroom pro automobilky. Senzorama však byla příliš složitá a vše nakonec skončilo neúspěchem u investorů.

V roce 1968 vzniklo první zobrazovací zařízení, které se montovalo přímo na hlavu (HMD - head mounted display). Jmenovalo se Damoklův meč a sestrojil ho Ivan Sutherland a jeho studenti. [3]



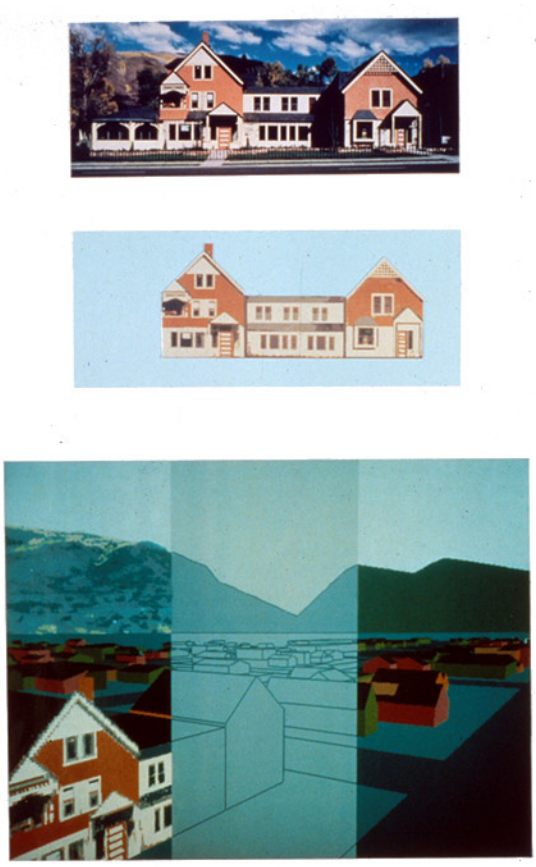
Obrázek 3.2: Damoklův meč. Zdroj: [3].

Zařízení bylo velice těžké, proto muselo viset ze stropu. Samotné zobrazování bylo pak velmi jednoduché. Zařizení umožnovalo zobrazovat pouze wireframy mistností.

Do roku 1990 pak byla virtualní realita převážně používána v letectví, armádě, automobilovém prumyslu a medicíně. Také vznikla první virtualní mapa a to města Aspen. NASA v roce 1985 představila system Ames, který je jeden z prvních, který se bliží dnešní podobě virtualní reality. [16] V roce 1988 vzniklo první SDK pro virtualní realitu. Byl to projekt firmy Autodesk a nesl název WorldToolkit. Byl velmi popularním napříč celým odvětvím.

Od roku 1990 se virtualní realita začíná dostávat do komerční sféry. Vznikaly první headsety určené pro běžné uživatele. Společnost Sega vydala svůj headset Sega VR kompatibilní s jejich konzolí. Headset již používal LCD obrazovku, stereo sluchátka a trackovací systém. [5]

Významnou firmou v tomto období byla také Virtuality. Dodávala první zařízení do arakdových heren. Byly k dospozici dva modely. V prvním se používal threadmil v kombinaci s VR. Druhý měl podobu formule s volantem, ve které se simulovaly zavodní hry. Zařízení však stála 60 000 USD, respektive 73 000 USD v závislosti na konfiguraci. I přes tuto cenu se virtualní realita poměrně rychle rozšířila.



Obrázek 3.3: Virtuální mapa Aspenu. Zdroj: [3].

Nintendo představilo v roce 1995 Virtual Boy konzoli, která měla způsobit převrat mezi konzolemi. Avšak díky množství technických problémů a průměrného zážitku se z konzole stal nejhůře prodávaný produkt firmy Nintendo. Prodalo se 770 000 kusů.

V roce 1999 vznikla velmi uznávaná hra Second Life. Hra je unikatní v tom, že v ní funguje reálná ekonomika a hráči tak dokáží ve hře vydělávat opravdové peníze. Hra funguje dodnes a komunita se nadále rozrůstá.

3.2 Současnost

V současnosti se velké firmy předhánějí, kdo nabídne lepší headset. Existují dva základní systémy sledování pohybu. První, kdy je pohyb headsetu sledován kamerami přímo z headsetu. Headset si vytváří po místnosti referenční body. Druhý spoléhá na stanice rozmístěné po prostoru, které fungují jako pevné body.

3.2.1 Oculus

V současnosti je velký zájem o to, aby se z VR stal uživatesky dostupný mainstream. Procenta headsetů u uživatelů stabilně rostou. Vloni se přesáhla hranice 2 % uživatelů používajících aktivně VR v Steam hardware survey. Na tomto úspěchu má převažující podíl firma Oculus. Oculus byl založen v roce 2010. Zakladatel Palmer Luckey v 17 letech vy-

Obrázek 3.4: The Virtual Interface Environment Workstation, NASA. Zdroj: [16]

Obrázek 3.5: Sega VR. Zdroj: [5]

pracoval první Oculus Rift v garáži.[10] Vzal si příklad z velkých firem, které se snažily virtualní realitu prosadit jako hlavní herní médium. Palmer ve volném čase opravoval staré konzole, kde se setkával s prvními pokusy o VR headsety. Vypozoroval chyby společností a s aktualním hardwarem jich bylo možno spoustu odstranit. To byla hlavní záminka pro vytvoření vlastního headsetu. Aby bylo možné založit firmu, která bude headsety vyrábět a prodávat, bylo nutné vybrat 250 tisíc dolarů. Palmer založil Kickstarter kampaň. Doufal, že jeho projekt zaujme alespoň 100 lidí. Během 24 hodin bylo vybráno 670 tisíc dolarů od 2 750 lidí. Do tří dnů byla pokořena hranice milionu dolarů. [11]

Následovalo několik Development verzí a v roce 2014 Facebook koupil Oculus. Cena byla kolem 3 miliard dolarů. Od odkoupení Oculusu Facebookem nastalo několik změn. Facebook se snažil donutit uživatele Oculusu používat Facebook k přihlášení k headsetu a tak je s headsetem propojit. Díky odporu komunity se naštěstí od tohoto kroku ustoupilo a vydrželo to pouze krátkou dobu. Facebook se posléze přejmenoval na Meta a snaží se jméno Oculus tímto názvem zcela nahradit. Aktuálně se headsety prodávájí už pouze pod názvem Meta. Aktuálně jediný hedset aktivně prodávaný je Oculus Quest 2. Očekává se příchod nové generace headsetu. [12]

3.2.2 Pimax

Firma byla založena v roce 2015. Založila ji skupina VR nadšenců v Shanghai. V roce 2016 vydala svůj první headset Pimax 4K. Tento headset se stal velmi populární a snažil se řešit

Obrázek 3.6: Quest 2. Zdroj: [12]

takzvaný screen-door efekt. Následující headset Pimax 8K byl zameřen na periferní vidění a stal se tak headsetem s největší šířkou zorného pole. Díky tomuto headsetu se Pimax stal světově populární a to i přes vyšší cenu. [8]

3.2.3 Varjo

Finský výrobce headsetů byl založen v roce 2016 skupinou bývalých vedoucích manažerů Nokie a Microsoftu. Headsety obsahují nejlepší, aktualně existující, technologie a lze říci, že tato firma je na špičce celého oboru. S tím se však váže cena, která je pro bě. K tomu se platí každoroční předplatné, a to 1495 euro, aby vůbec headset fungoval. Tento headset používají velmi známé firmy, například: Volvo, Kia, Boeing, Lockheed Martin a několik dalších. Primárně se používá v showroomech pro představení produktů firem. [19]

3.2.4 HTC

Společnost HTC vydala v roce 2016 svůj první headset Vive. Celým názvem označován jako HTC Vive SteamVR. Spolupracovala na něm Společnost Valve. Headset začínal na 1 200 dolarech. V roce 2018 vyšel Vive Pro, který obsahoval několik vylepšení, včetně trackovaní očí. Oba tyto headsety potřebovaly mít v místnosti instalované stanice pojmenované Lighthouse. Vůči těmto bodům se pak trackovala pozice. Od této technologie upustili s posledním headsetem Cosmos(2019), kdy začali také trackovat pomocí kamer po vzoru Oculusu.

3.2.5 Valve

Společnost Valve byla založena v 90 letech. Založili jí Game Newell a Mike Harrington bývalí kolegové a zaměstnanci Microsoftu. Svoje úspory použili na založení firmy Valve v roce 1996. Herní studio vyvýjelo FPS Half-life. Hra byla vydána v roce 1998 a byl to pro firmu velký úspěch. Hra bodovala ve všech ohledech. Platforma Steam byla představena na

Obrázek 3.7: Pimax Vision 8K X. Zdroj: [15]

GDC 2002 a spuštěna byla ještě ten rok. Ze Steamu se postupem času stala nějvětší herní platforma. Díky výdělkům mohla Valve začít experimentovat i s výrobou hardwaru. Valve prozatím vydalo pouze jediný headset a to ještě ve spolupráci s firmou HTC. Jde o headset Valve Index. Index je nejdoporučovanější headset pro náročného hráče. [6]

3.3 Budoucnost

VR velmi rychle vylepšuje. Zvedá se rozlišení headsetů, snímkovací frekvence nebo například se zvětšuje zorné pole. Dalším vývojem prochází periferie, kdy se začínají používat vesty s haptickou odezvou. Co se týče aplikací ustálili se dva projekty, které se zmiňují velmi často.

3.3.1 VRChat

VRChat je novodobá sociální síť, která se zaměřuje na propojení lidí pomocí VR. Síť podporuje plně trackované avatary, a to od pohybu prstem na noze až po mrknutí oka. Samozřejmě pouze tehdy, má-li uživatel příslušný hardware pro trackování. Sociální síť je primárně o interakci mezi jejími uživateli a VRChat nabízí každému vytvořit si svůj unikátní prostor a sdílet ho s ostatními. Jako seznamovací aktivity mohou posloužit hry, ale také společné promítaní filmů a další aktivity. V tomto virtuálním světě se také odehravájí každý týden eventy a celý VRChat je vyvýjen okolo potřeb komunity. Tato sociální síť má nejblíže se stát opravdovým, komunitou řízeným, virtuálním světem. VRChat byl vytvořen v Unity a každý si může v Unity vytvořit vlastní místnost a nahrát ji do světa VRChatu. [9]

Obrázek 3.8: Varjo XR-3. Zdroj: [19]

Obrázek 3.9: Valve Index. Zdroj: Stránky společnosti Valve [4]

3.3.2 Metaverse

V úspěchu VRChatu se zhlédla síť Facebook, nově Meta. Ta se rozhodla investovat přes 300 milionů dolarů do Metaverse. Metaverse má být virtuální prostor, idealně stejný jak ve filmu Ready player One. Vsází na NFT a reálnou ekonomiku. V podstatě se Facebook snaží vybudovat kompletní virtualní svět, který bude vlastnit. To se prozatím lidem velice nelíbí a VRChat roste na popularitě. Metaverse jestě nevyšel.

3.4 Rizika a Limitace

VR je stále vyvíjející se technologie a svojí hlavní éru má jestě před sebou. V dnešní době je obrovské množství projektů, které se virtualní realitu snaží posunout dál. Byly také již zaznamenány úspěchy v podobě standardizace některých parametrů VR. Díky tomu se vývoj samotných technologií velice urychluje.

3.4.1 Trackovaní

Aby hráč pociťoval imersivní zážitek ve VR, je potřeba trackovat jeho pohyby. Při trackovaní pohybu hlavy se již dneska používají kamery přímo na headsetu. Tyto kamery zmapují prostor okolo uživatele a dle referenčních bodů dokáží přepočítávat uživatelovu polohu.

V headsetech začínají být také běžně používané pokročilé technologie, jako trackovaní očí. Nejvíce rošířený headset s trackovaném očí je Vive Pro, který v sobě má trackovaní očí od firmy Tobii. Je to další důležitý krok, jelikož s touto informací lze lokálně zvyšovat kvalitu obrazu tak, aby to bylo pro lidské oko přirozenější a také lépe upravovat zaostřování objektů ve scéně.

Pro sledování rukou se zatím nejčastěji používají ovladače. Každý výrobce má svoji vizi, jak by ovladače měly vypadat.

3.4.2 Kinetóza

VR se stává postupně čím dál dostupnější. Původně touto dobou mělo být VR hlavní herní platformou. Všechny hry měly VR podporovat a téměř každý AAA titul by měl minimálně pro VR podporu. Úplně původní plán byl, že se vytvoří pluginy pro starší hry, které umožní hrát jakoukoli hru ve VR. V podstatě by jen byly prostředníky pro samotné ovládání ve hře. To se však ukázalo jako zcela nereálné z několika důvodů. Tím hlavním je kinetóza (Motion sickness), neboli nemoc z pohybu. Vzniká, když vjem vestibulárního systému neodpovídá zrakovému vjemu. Ve VR to znamená pohyb herní postavy, aniž by se hráč hýbal v realitě. Problém je, že při vývoji se s tímto problémem musí počítat. Existuje mnoho technik, jak se s kinetózou ve hrách vypořádat.

Starší hry, bouhužel, s tímto nikdy nepočítaly a tak kompatibilní nejspíše nikdy nebudou. Na straně hardwaru se snaží výrobci tento problém vyřešit, ale v plně pohlcující Virtualní realitě hodně záleží na samotném softwaru. Co ale výrobci mohou ovlivnit, je vzdálenost očí od sebe, takzvané IPD. Některé headsety mají manualní posuvník, kdy si lze vzdálenost mezi čočkami přizpůsobit. Dokonce dražší profesionální headsety mají IPD elektronicky nastavitelné pomocé trackovaní očí(automatická kalibrace po nasazení, elektronicky ovladané). Kinetóza postihuje ve VR častěji ženy a děti do 12 let.

Co kinetóza vyvolává: studený pot, nevolnost, únavu, migrénu, podrážděnost, velká tvorba slin a zvracení, či lapání po dechu.

Léčba není většinou potřeba. Příznaky odezní do dvou dnů. Léčbu urychlují antihistaminika, stejné léky určené k potlačení alergií. Při zvracení je pak potřeba doplňovat tekutiny, aby nedošlo k dehydrataci.

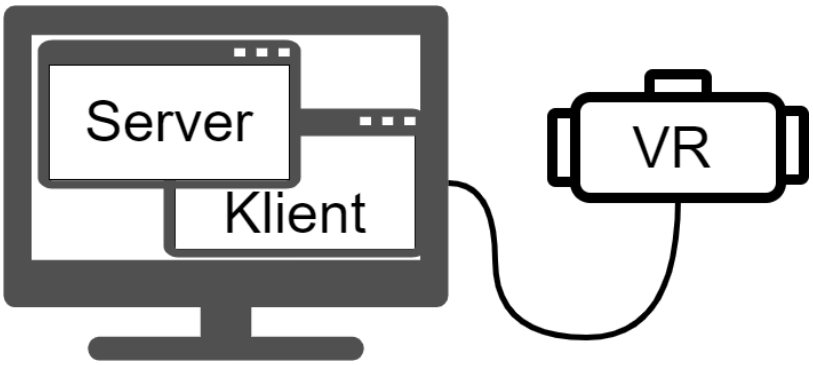
Kapitola 4

Návrh

4.1 Struktura Aplikace

Při navrhovaní aplikace je potřeba zohlednit její běžné užití. V tomto případě šlo o specifické použití. Uživatel přijde k PC a zapne aplikaci. V aplikaci vytvoří uroveň. Spustí uroveň jako server a stává se vnějším pozorovatelem. Tento pozorovatel má možnost zasahovat do scény, vytvářet objekty, či věci nastavovat. Zároveň je v tomto okamžiku připojen virtualní headset. Ten musí umožnit, aby se uživatel po nasazení headsetu dostal do hry a byl schopen s úrovní interagovat. Problém nastává ve vstupech a vykreslování aplikace. Do aplikace jdou dva nezavislé vstupy z VR a z klavesnice. V úrovni jsou dva nezávislí hráči. To znamená dvě kamery, které souběžně vykreslují a každá odesílá do správného zařízení. Tento přístup byl velice problémový, jelikož jakmile hráč se nachází ve VR, vykreslování na monitor je omezené a snižuje se framerate na minimum. Toto nesplňovalo požadavky na funkčnost aplikace. Bylo proto zvoleno jiné řešení. Řešení spočívá v souběžném běhu aplikace dvakrát jedna na pozadí jako klient, která pouze spravuje VR hráče. Druhá na popředí s hlavním oknem v systému a tato je server se superuživatele. Zároveň si neklade výhradní užívání vstupu. Tímto je schopna aplikace v pozadí komunikovat s VR headsetem.

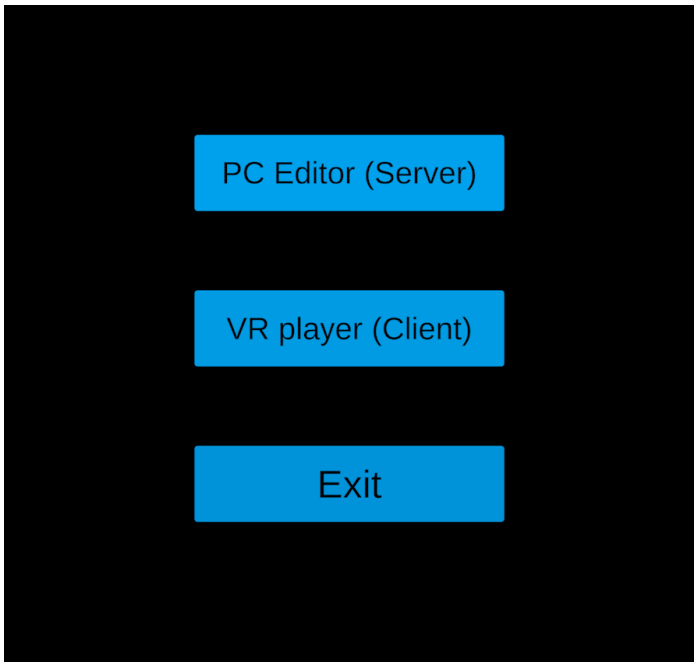
Aby byla podporována tato funkčnost aplikace, je rozdělena na část editoru (serveru) a část VR-hráče (klienta). Obě části jsou pak přístupné z menu.



Obrázek 4.1: Schéma použití aplikace.

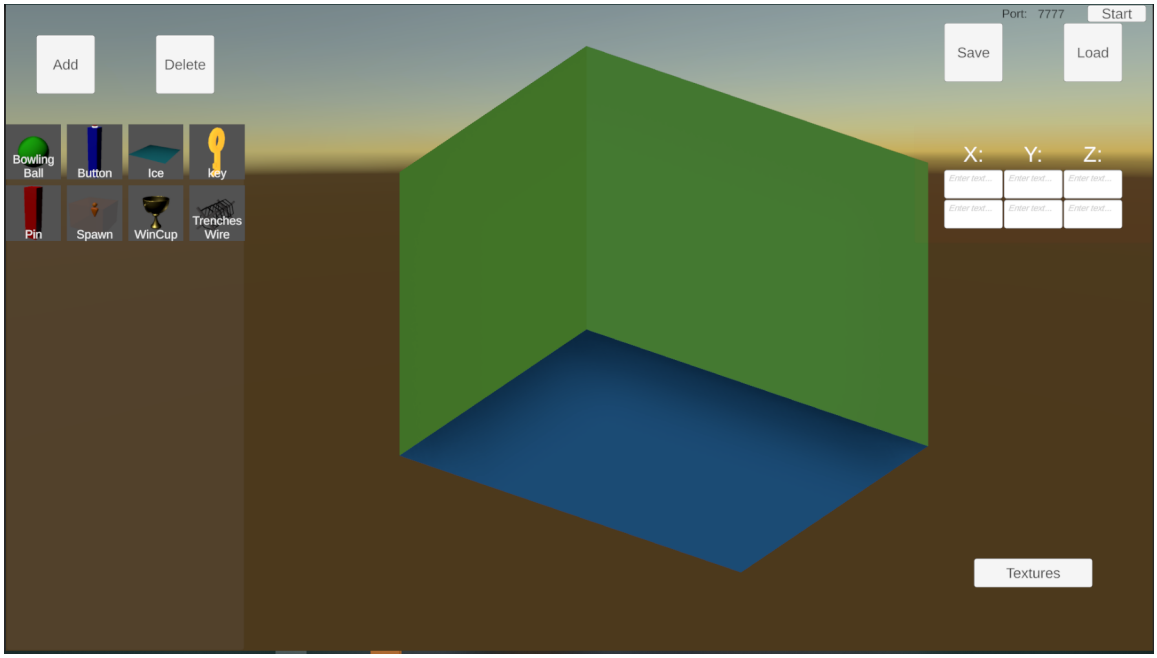
4.1.1 Menu

Výchozí obrazovka po spuštění aplikace je menu. Základní funkce menu je rozcestník pro spuštění příslušných částí aplikace. Během fáze menu se na pozadí inicializují komponenty pro síťové připojení. Nutností je možnost aplikaci zmenšit do malého okna kvůli souběžnému běhu Editoru a Klienta ve dvou aplikacích vedle sebe.



Obrázek 4.2: Ukázka menu.

4.1.2 Editor



Obrázek 4.3: Editor

Editor je hlavní část programu. Po spuštění vygeneruje prázdnou místnost. Lze načíst již vyrobené úrovně nebo vytvořit zcela novou unikátní.

Prostor editoru je složen z voxelů. Voxel si lze představit jako krychli o hraně jednoho metru. Z takových to krychlí je pak složen veškerý editovatelný prostor a hráč určuje, zda se jedná o voxel vnější nebo vnitřní. Velikost prostoru v editoru je ve výchozím nastavení na 32 x 32 x32 voxelů. Nejvíce inspirace v návrhu editoru pochází ze hry Portal 2. Také se v něm uplatňují principy z bakalářské práce. [13]

Editor poskytuje prostor, který je editovatelný a lze do něj vkládat objekty. Editor umí objekty vyhledávat v předem nastavených složkách. Pokud se dodrží předem daná struktura objektů editor je schopen si je sám načíst. Načtené objekty se pak dají vkládat do voxelové místnosti. Díky tomu, že editor si sám objekty načítá, lze vytvořit plugin do Unity, který by jen automaticky dogenerovával potřebné komponenty a zaručoval adresářovou strukturu a umožnil by tak přidávat objekty přímo uživateli.

U objektů je pak možno nastavit požadované parametry v závislosti typu objektu. Objekty zároveň nejsou pevně vázány na voxelovou mřížku. Při práci s objekty v editoru jsou v deaktivovaném stavu a spouštějí se až při spuštění serveru. Jednotlivé voxely lze v editoru obarvovat a budoucnu by nebylo těžké tento systém adaptovat na použití textur (prozatím jen barvy)

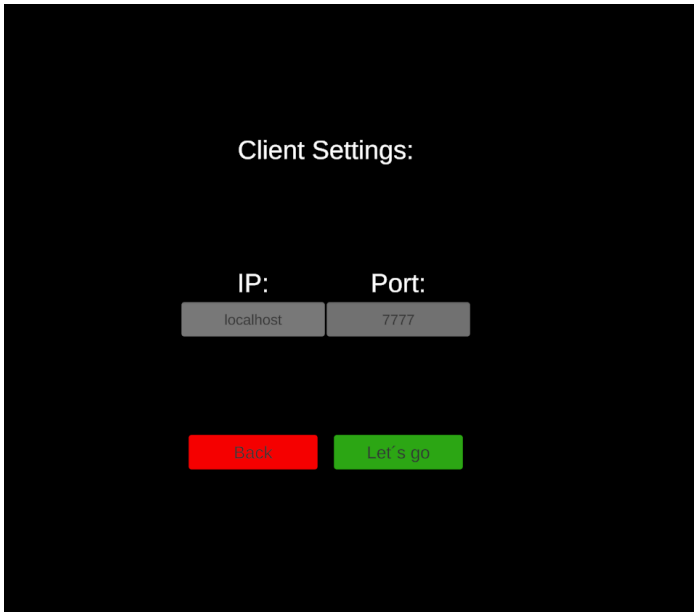
Po vytvoření úrovně lze místnost uložit a rovnou spustit. Při tomto spuštění běží server na lokálním počítači na specifikovaném portu. Po spuštění se Editor přepíná na administrativního uživatele. Tento uživatel má pak možnost ovládat úroveň.

4.1.3 Klient

Klient se spouští v momentě, kdy se z menu přejde do konfigurace připojení. Nastavuje se cílová IP adresa a cílový port. Pokud se hráč připojuje na stejném počítači, výchozí adresa je nastavena jako localhost.

Po úspěšném navázání komunikace se serverem nastává inicializace VR. Aplikace je hybridní, neboli samotné připojování probíhá na obrazovce za pomocí klávesnice. Teprve po připojení nastává fáze přechodu do VR. Po připojení je již klient v plném režimu VR, dokud neztratí spojení se serverem. Při ukončení nebo výpadku serveru, klient přechází zpět do 2D módu a vrací se do základního menu.

Průběh samotné hry pak řídí administrativní hráč, který určuje co je v dané úrovni cílem.



Obrázek 4.4: Nastavení připojení. Zdroj: vlastní dílo.

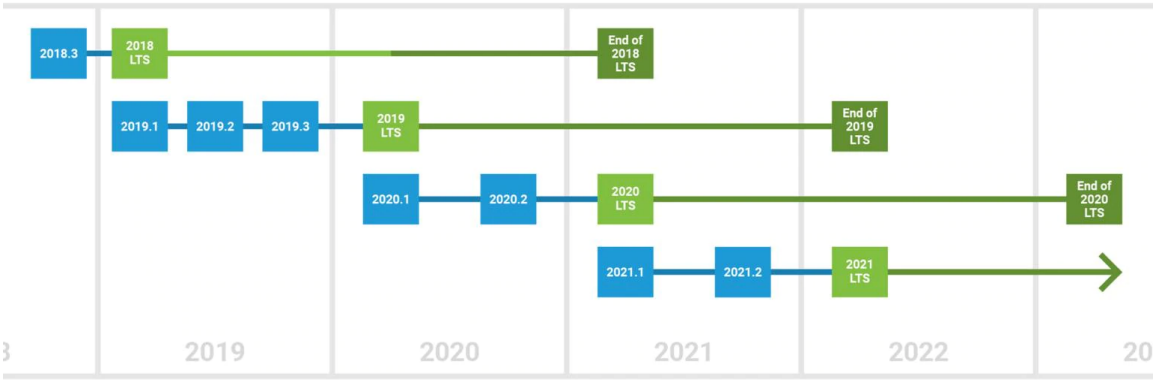
4.2 Výběr enginu

V počátku bylo potřeba zvolit herní engine, ve kterém bude projekt napsán. Na trhu existuje velké množství herních enginů. Každý má trochu jiné přednosti, ale na základě zkušeností se pro tuto práci hodil Unity engine nebo Unreal engine. V počátcích této práce vyšel Unreal engine 5 v první veřejné alfě. Ačkoli bylo lákavé použít nejmodernější technologie, bylo však třeba si uvědomit, že ve VR jde primárně o výkon. Poslední trendy ve VR hardwaru dokonce směřují primárně na standalone headsety. To znamená, že než fotorealistická grafika, je mnohem užitečnější vytvořit nadčasovou stylizaci. Tato stylizace musí být optimalizovaná pro vykreslování na mobilních zařízeních. Pro Unity jsem se rozhodl právě proto, že mám dlouhodobé zkušenosti optimalizování her pro Unity i zkušenosti s VR pluginy v tomto enginu. Také je Unity považováno jako lepší varianta právě pro již zmíněné mobilní aplikace. Tudíž nový Unreal by do celkového projektu nepřinesl žádnou zásadní inovaci, proto se Unity pro tento projekt hodí více.

4.3 Verze

Unity je unikátní v tom, že v počátku vývoje má vývojář k dispozici hned několik verzí a každá je určena k jinému stylu vývoje. Vetšinou jde o verzi LTS, Tech stream, Afla a Beta. Verze LTS je dlouhodobě udržovaná verze. Je udržována po dobu dvou let a jsou pro ni vydávány updaty zaměřené pouze na stabilitu. Tato verze je pro aplikace, které jsou určeny k vydání pro veřejnost. Zvláštností je číslování, jelikož LTS verze pro daný rok je vždy označená rokem minulým. Jako příklad pro rok 2021 byla doporučována verze LTS 2020. LTS verze je také doporučena všem vývojářům začínajícím pracovat s tímto enginem. Má vždy odzkoušené funkce a největší kompatibilitu. Podporuje také kompletní zpětnou kompatibilitu s projekty z předchozích verzí. Každý rok pak vycházejí dvě verze Unity zvané Tech release. Dříve tyto verze vycházely tři, ale protože si komunita stěžovala na stabilitu těchto verzí, Unity se z toho důvodu rozhodlo věnovat každé verzi více času a vycházejí tak pouze dvě ročně. Z druhého Tech releasu se pak stává LTS verze pro další rok. Tech release je verze Unity, do které jsou přidanné nové funkce, avšak za cenu stability. Tyto verze se doporučují používat pouze tehdy, je-li nová funkce vyžadována ve tvořené aplikaci. Popřípadě zda-li je potřeba podpora novějších pluginů. Alfa a Beta verze jsou pak vždy testovací a obsahují nejnovější vývoj Enginu. Jsou to pouze testovací verze a nedoporučuje se v nich aplikace tvořit. Jsou určené k dřívějšímu vyzkoušení nástrojů a poskytnutí zpětné vazby. V těchto verzích se objevují nejnovější funkce, ale občas jsou odebrány a nedostávají se tak až do tech releasu. Záleží na komunitě a stabilitě nových funkcí.

Pro tuto diplomovou práci byl zvolen Tech release 2021.1. Důvod byl kompatibilita pluginů, jelikož mnoho z používaných technologií je vydáno pouze jako pre-release. Později byl engine aktualizován na Tech release 2021.2. Avšak primárně kvůli stabilitě bylo finální vydání exportováno z Unity 2021.3.2f1, což je nejnovější LTS verze pro rok 2022.



Obrázek 4.5: Plán vydání Unity verzí. Zdroj: [18]

4.4 Pluginy

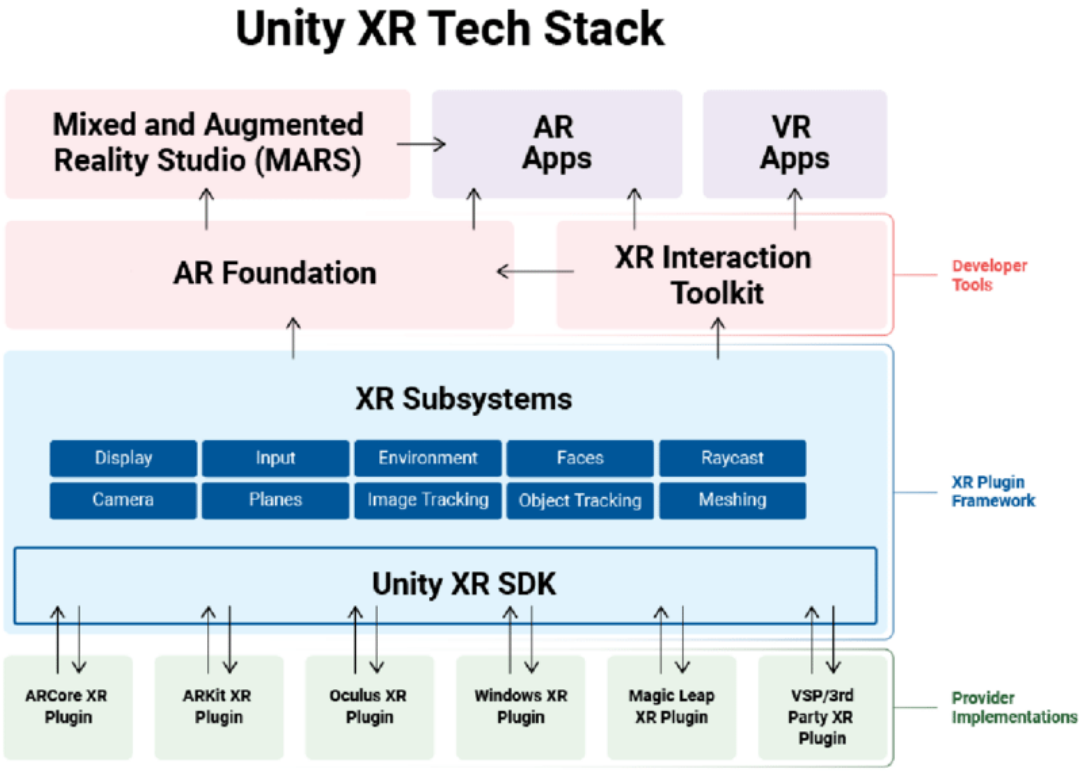
4.4.1 XR Interaction Toolkit

Tento XR Interaction Toolkit slouží k usnadnění vývoje VR aplikací v Unity. XR je pojmenován, protože lze využít, jak pro VR, tak i pro vývoj AR aplikací. Plugin je možno stáhnout v několika variantách. Pro vývoj virtualní reality stačí základní funkcionalita. Pro

vývoj rozšířené reality AR je nutno stáhnout jak základní balík, tak i jeho nástavbu. Pro tento projekt postačí podpora virtualní reality, nicméně použití tohoto pluginu umožňuje v budoucnu implementovat i AR verzi aplikace.

V práci byla použita verze plginu 1.0, která vyšla 10.09.2021. Avšak pořád se jedná o prerelease verzi pluginu a samotný plugin jestě ve stabilni verzi nevyšel. Ke konci roku 2021 pak vyšla verze 2.0, pořád ve verzi prerelease, která změnila základní stavební kompunentu XRRig. Práce je zatím napsaná na starší implementaci tohoto pluginu, nicmeně do zveřejnění práce bude naimplementováno co nejvíce věcí kompatibilních s verzí 2.0, aby bylo možno na tuto verzi přejít.

XR Interaction toolkit se skláda z mnoha komponent a objektů, které slouží jako základní stavební kameny pro vývoj aplikace. Funguje jako nástavba nad SDK jednotlivých výrobců hardwaru a postará se tak o kompatibilitu aplikace napříč různými platformami. Zároveň plugin spravuje veškeré uživatelské vstupy, ale také zpětnou vazbu v podobě vibrací ovladačů, vest a podobných zařízení. Taktéž nabízí řešení pro UI s kterým lze interagovat přímo ve VR.



Obrázek 4.6: XR Tech Stack. Zdroj:[14]

4.4.2 OpenXR

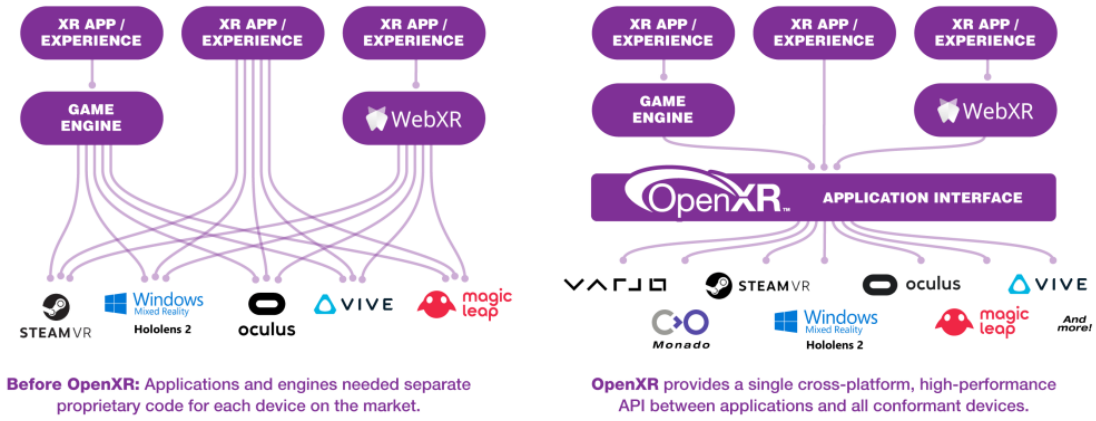
V roce 2000 založily firmy Intel, ATI, 3Dlabs, Evans & Sutherland, SGI a Sun Microsystem skupinu The Khronos Group. Založena byla jako konsorcium pro vydávání otevřených standardů. Tyto standardy velmi urychlují vývoj nových technologií a umožňují vývojářům

užívat sjednocené API. Firmy mohou přes zástupce hlasovat pro schvalování jednotlivých změn standardů v nových revizích.

Až do roku 2017 si pro VR vyvíjel každý vlastní rozhraní. Pokud vývojář chtěl vydat aplikaci, která by podporovala headsety od více výrobců, musel pro každý vytvořit separátní podporu. Toto nebylo ideální řešení, i když každý engine si přišel se svojí adaptací na tato jednotlivá rozhraní. Byl to jeden z hlavních důvodů, proč vznikl XR interaction toolkit. Khronos Group koncorsium se rozhodlo jednotlivá rozhraní sjednotit a vytvořit otevřený standard. Standard se nazývá OpenXR 1.0 a byl zveřejněn 18. března 2019. První z enginu, který jej plně podporoval byl Unreal engine. V Unity podpora tohoto standardu byla v preview implementaci až ke konci roku 2020. Nyní je plně podporován a pokud se vývojář rozhodne pracovat s VR, doporučuji zvolit stabilní verzi enginu, nejlépe LTS verzi. V Tech stream verzích Unity dochází k problémům s kompatibilitou ovladačů. Alfu či Betu jakékoli verze Unity společně s jakýmkoli VR se v podstatě použít nedá.

V současnosti OpenXR uznává mnoho společností a stal se tak nejvýznamnějším standardem pro VR.

Obrázek 4.7: Firmy podporující standard OpenXR 1.0. Zdroj: [2].



Obrázek 4.8: XR před a po Standardizaci. Zdroj: [2].

4.5 Síťová komunikace

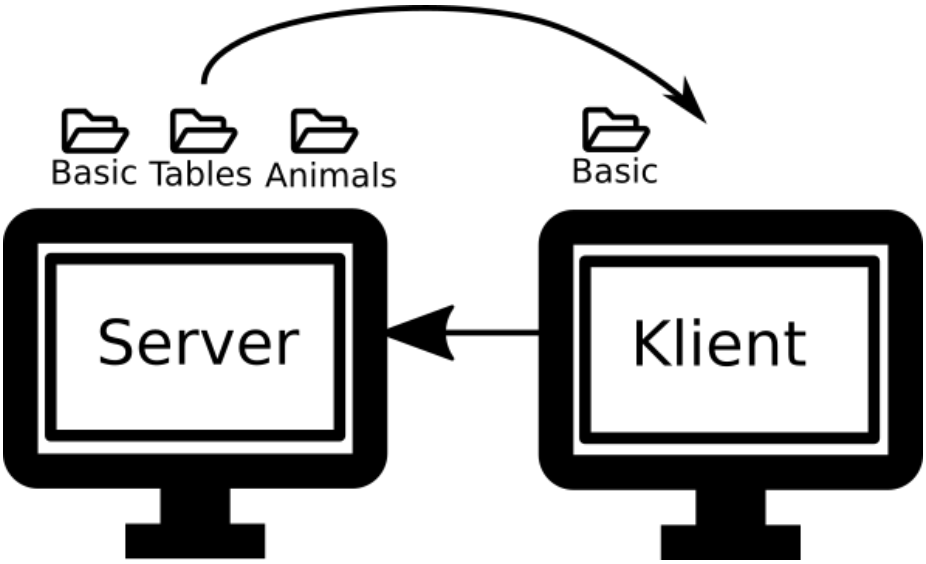
Komunikace po síti je pro tuto aplikaci stěžejní, jelikož aplikace může vystupovat jako server i jako klient. V budoucnosti se počítá s rozšířením i jako dedikovaný server. To by zajistilo možnost připojení daleko většího počtu hráčů, podobně jako aplikace VRChat. Samotné Unity neobsahuje příliš kvalitní rozhraní pro komunikaci na síti. Nastala situace, kdy Unity původní síťovou vrstvu označilo jako zastaralou, vývoj nové se pozdržel a vydaní stabilní verze se očekává koncem roku 2022. Proto pro tento projekt byla použita open source nástavba Mirror. Jedná se o plugin do Unity, který umožňuje v rámci jednoho kódu mít jak serverovou, tak klientskou část. Tím značně zjednodušuje a zpřehledňuje implementaci dané aplikace. Byl napsán jako rozšíření původní síťové vrstvy Unity. Zajímavostí je, že implementace nové síťové vrstvy do Unity se inspiruje principy z pluginu Mirror. V samotné aplikaci je potřeba využít mnoho z předpřipravených komponent z tohoto pluginu, aby se docílilo požadované funkčnosti. [1]

4.6 Tenký klient

Existují dva typy klientů. Tlustý klient je ten výchozí pro Unity a většinu her. Klient má k dispozici kompletní assety ze hry (modely, materiály, textury). Takový klient je zpravidla poměrně velký. Novodobé aplikace a zvlášť VR hry, kde jsou zapotřebí propracovanější modely mají v řádu 10 GB. Jako příklad lze uvést Pavlov VR, které má na disku přibližně 58GB.

Tenký klient na druhou stranu obsahuje pouze klíčové assety a zbytek si stáhne před spuštěním požadované úrovně. Výhodou je malá velikost klienta. Nevýhodou je čekání na stažení assetů a předem neznámá velikost aplikace. Tenký klient se hodí v situacích, kdy serverová aplikace může mít tisíce assetů, ale v úrovni použitý jen malý zlomek z nich.

V této práci se používá prozatím tlustý klient, avšak tenkému jsem věnoval velice mnoho času a je funkční za předpokladu, že se nachystají asset balíky dopředu v editoru. V budoucnu by bylo potřeba pro celou funkčnost systému implemntovat rozšíření do Unity, aby umožnilo jednoduché generování těchto assetů, ke kterému se používají vnitřní funkce samotného Unity. Demonstrace tenkého klienta muže být předvedena zatím pouze z editoru Unity. Této technologii jsem věnoval hodně času a v budoucnu ji plánuji velmi rozšířit. Samotné Unity nic takového nepodporuje.



Obrázek 4.9: Ilustrace tenkého klienta.

Kapitola 5

Realizace

5.1 Založení projektu

Pro správu verzi Unity a projektů se už výhradně používá Unity Hub. Jednotlivé verze si lze stáhnout zvlášť ale Unity hub poskytuje mnoho funkcí nad správou jednotlivých instalací. Mezi funkce například patří jednoduché doinstalování součástí pro vývoj na jednotlivé platformy, nebo také instalace Visual Studia včetně potřebných nástrojů pro integraci s Unity. Unity Hub prošel v poslední době velkým vývojem a stalo se z něj primární prostření pro správu verzí unity , správu projektů a v neposlední řadě také informační centrum s velkým množstvím návodů a odkazů na komunitní projekty. Další velkou výhodou poskytuje mnoho různých šablon pro nový projekt. Kdysi bylo Unity omezené jen na základní výběr 3D, 2D a 3D s přídavky. Poslední možnost znamenala projekt s ukázkovou úrovní. Později se rozdělili vykreslovací řetězce takže přibily dvě další možnosti (URP a HDRP). Nyní už nelze vyjmenovat základní šablony, jelikož Unity se rozhodlo ze šablon udělá univerzální systém. Šablony začala tvořit ve velkém komunita a lze si tak stáhnout přesně šablony zaměřené pro konkrétní styly her.

Dokonce změna ohledně šablon se dotkla Package manageru. Package manager je správce použitých pluginů během vývoje. Nově je tak strukturovaný podle stylu jednotlivých aplikací/herních stylů. Obsahuje doporučené kombinace pluginů pro daný styl. Jde tak o snahu zpřehlednit začínajícím vývojářům pluginy, jejich správu a to, že síla Unity editor je právě v použitých pluginech.

5.2 Vykreslovací řetězec

Unity od svého počátku obsahovalo pouze jeden vykreslovací řetězec. Tento grafický řetězec však zastaral a primárně kvůli optimalizaci byl nahrazen Univerzálním vykreslovacím řetězcem (URP). Rozdělení řetězce mělo více důvodů. Jeden z nich bylo to, že Unity mělo horší grafické zpracování než ostatní enginy. Komunita tak naléhala na vytvoření propracovanějšího grafického řetězce. Proto vznikl High Definition Render Pipeline (HDRP). Tento řetězec je primárně určen k zobrazování realistické grafiky.

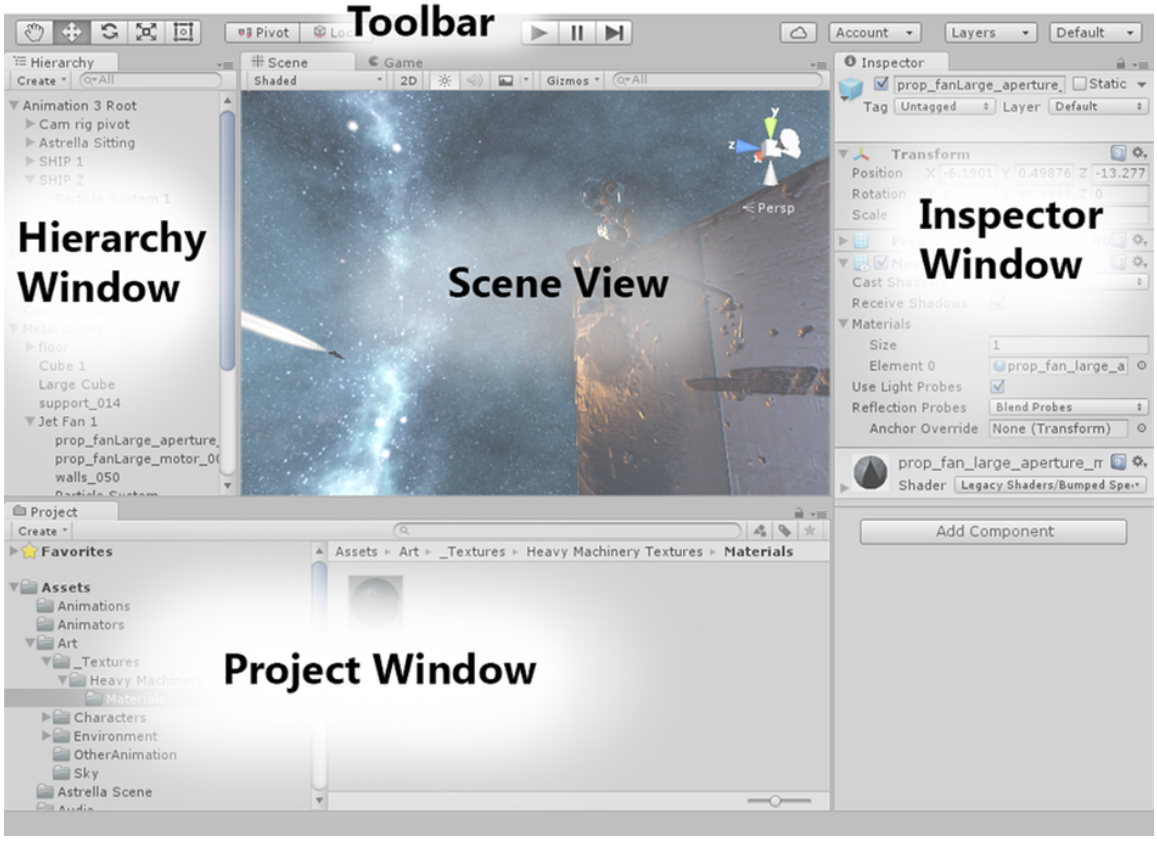
Pro tuto práci byl podstatný výkon a přenositelnost, proto byl zvolen univerzální vykreslovací řetězec a hra bude stylizována, aby se dosáhlo ještě další úspory výpočetního výkonu.

Obrázek 5.1: Vlevo univerzální vykreslovací řetězec (URP), vpravo realistická HD grafika (HDRP). Zdroj: [20]

5.3 Práce v Unity

Po vybraní šablony a vykreslovacího řetězce se otevře samotný Editor Unity. Tato sekce slouží jako shrnutí základních principů v enginu.

Takto vypadá přibližně základní layout enginu.



Obrázek 5.2: Unity UI Zdroj: [20]

Scene view je hlavní okno pro editaci scény. Scéna bývá zpravidla jedena úroveň ve hře. V této záložce tak probíhá pozicování všech objektů ve scéně.

Hierarchy window – okno hierarchie. Každá scéna je kořenový objekt. Do této scény lze vkládat objekty různého typu. Mezi nejběžnější patří objekty, které tvoří viditelnou část scény. Ovšem scéna jako taková obsahuje zpravidla i plátno (Canvas) pro zobrazení UI nebo také prázdné objekty na kterých se vykonávají skripty. Objekty lze volně do sebe zanořovat. Vždy musí mít ale svého rodiče. Nejvyšším rodičem je právě scéna.

Hra nebo aplikace se skládá pak z posloupnosti scén které se přepínají mezi sebou. Dobrá praxe je mít řízené přepínaní scén herním správcem (Game manager). Při přepnutí ze scény na scénu zaniknout všechny objekty v aktuální, a naopak se načtou všechny objekty ze scény nové. Pokud je však potřeba zachovat mezi scénami nějaká data je potřeba vytvořit perzistentní objekt. Obecně to bývají právě správci úrovní, ale mohou to být i jiné objekty. Tyto perzistentní objekty musí být naprogramovány jako jedináček (singleton) pro případ návratu do původní scény. Návrhový vzor jedináček (singleton) zaručuje že, pokud objekt ve scéně již existuje, nový se nevytvoří.

Projekt window – průzkumník obsahuje pak adresářovou hierarchii projektu odkud se do scény nahrávají všechny assety. Asset je v podstatě jakýkoli soubor který je součástí projektu. Zpravidla jde především o materiály, modely, skripty, textury, zvuky ale i o scény jako takové. Duležitý pojem je také prefab. Jedná se o objekt, který již obsahuje všechny potřebné komponenty ze scény a zpravidla se používá jako šablona pro spawnované objekty ve scéně.

Inspector window – Inspektor, správce vlastností objektu. Každý objekt ve scéně se skládá z komponent. Základní komponentou každého objektu je Transform. Transform by se dalo přeložit jako komponenta transformace, obsahuje pozici, rotaci a velikost objektu. Jenže kromě toho lze na tuto komponentu nahlížet jako na samotný objekt, jelikož obsahuje přímé odkazy na potomky objektu včetně referencí na všechny další komponenty daného objetu. Bez této komponenty nemůže objekt existovat ve scéně. Mezi běžné komponenty pak patří renderer – který se stará o vykreslení meshe, mesh filter, který reprezentuje samotný tvar – mesh objektu. Mezi komponenty například patří komponenty simulující fyziku jako Rigid body, Collider, Joint atd. Do komponent patří veškeré skripty i komponenty přidané jednotlivými pluginy.

Toolbar – Ovládání spuštění aplikace přímo v enginu, popřípadě stopnutí při profilovaní či debugu.

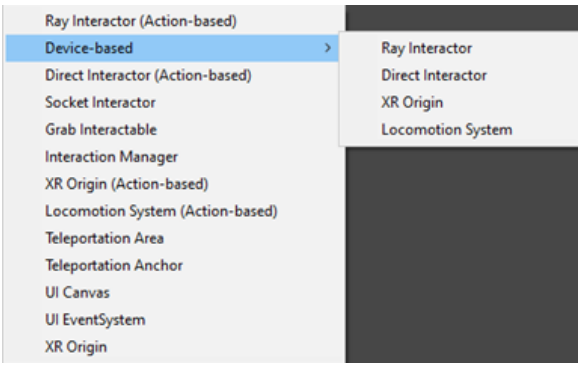
5.4 XR Interaction Toolkit

XR Interaction toolkit je plugin zprovoznění VR v aplikaci. Obsahuje všechny potřebné součásti. Prvním krokem při nastavování VR je nastavit toolkitu jaký plugin výrobce VR se bude využívat. XR interaction toolkit je jen nástavba nad implementacemi jednotlivých SDK výrobců headsetů. V případě této diplomové práce jsem volil OpenXR. Důvody volby OpenXR byly převážně kvůli kompatibilitě napříč headsety. Také je potřeba vypnout automatickou inicializaci VR při startu aplikace, jelikož se jedná o hybridní aplikaci. Poté, co je toolkit takto nastaven, veškerou VR interakci provádíme skrz komponenty toolkitu. Některé klíčové funkce jsou již sdružené v podobě komponentů na nachystaných objektech. Tyto defaultní objekty tak stačí pouze vložit do scény. Po vložení takovéhoto objektu do scény se vygeneruje Interaction Manager, který slouží jako prostředník pro komunikaci mezi komponenty. Těchto správců může být ve scéně více, minimálně vždy alespoň jeden. Komponenty si defaultně nastavují toho správce, který naleznou ve scéně jako první. Pokud se jim skriptem nenastaví jiný.

Základním stavebním kamenem každé VR místnosti je přednastavený VR objekt (rig), který reprezentuje hráče a obsahuje již několik klíčových komponent pro propojení s VR. V nové verzi došlo k přejmenování objektu Rig na Origin, došlo i k přejmenování stejnojmenných komponent toolkitu.

Mezi klíčové komponenty Originu patří například: typ komponenty Interactor, jako základní se používá XR Ray Interactor. Ten používá ray casting (proložení scény paprskem) a pokud protnutý předmět má příslušnou masku a interaktivní komponentu, hráč s ním může interagovat dle implementovaného chovaní (například objekt uchopit). Aby bylo možné přenášet prostorovou pozici ovladače od uživatele, musí být na objektu komponenta XR Controler. Ten umožňuje namapovat akce jako změna pozice, rotace, aktivace stisknutím tlačítka, interakce s UI a další.

Action-based vs. Device-based chování komponent. V počátku vývoje XR Interaction toolkitu se pracovalo s přímými hodnotami zařízení. V XR Interaction toolkitu se tak odkazovalo na konkrétní ovladače a různé hodnoty. Ne všechny ovladače měly například tlačítka jako ostatní výrobci a byla nutnost mít několik alternativních způsobů ovládání. To změnil příchod standartu OpenXR a přechod na nový input systém přímo v Unity.



Obrázek 5.3: V XR Interaction toolkitu lze stále zvolit objekty obsahující starší komponenty.

5.5 Input systém

New Input systém je v Unity se v unity postupně stává standardem. Nyní si lze stále vybrat, zda se v projektu bude používat starší input systém nebo již nový. Nový Input systém není tak nový, jak se zdá postupně se na něj přechází již od roku 2017. Hlavní výhodou nového systému je přenositelnost na různé platformy. Zároveň nový systém velice jednoduše podporuje různé ovládací zařízení, a to i simultánně. Lze tak například použít klávesnici ve spojení s gamepad ovladačem či joystickem.

Nový input systém se tak jako ostatní pluginy přidává přes package manager a po přidání Unity vyžaduje restart a nastavení který input systém se má využívat, jestli starý, nový nebo oba zároveň. I když vše směřuje k výhradnímu použití nového input systému, starý se hodí na rychlé prototypování. Zůstává tak nadále komunitou hojně využíván.

Nový systém je založen na eventech, není potřeba trvale ověřovat v updatu, zda byl proveden nějaký vstup. Samozřejmě jsou situace, kdy se toto chování hodí. Existuje poměrně jednoduchý způsob, jak k novému input systému přistupovat jako ve starém systému. Stačí se manuálně odkázat na konkrétní připojené zařízení. Každý uživatelský vstup v novém

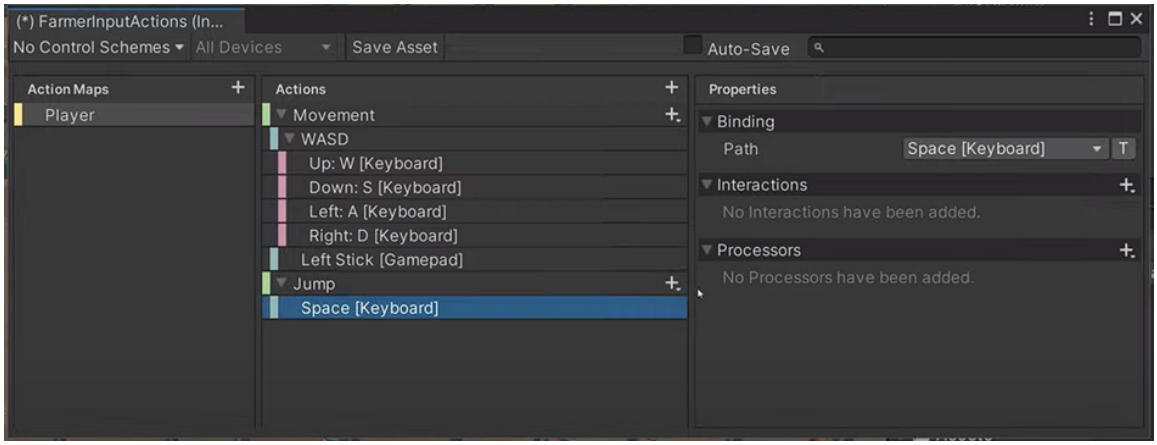
systému vyvolá tři události. Počátek události, událost vykonána a konec události. Na každý z těchto eventů lze reagovat.

Velkou výhodou nového systému je Input Debug, Zvlášť pro VR, které obsahuje mnohdy různé ovladače je potřeba vidět, zda se ovladač chová dle očekávání a odhalování chyb spojené se vstupem je tak mnohem jednodušší.

Pro fungování vstupu je potřeba vytvořit nový Input Actions Asset. Pro editaci Input assety má Unity speciální rozhraní. V něm lze vytvořit jednotlivé input mapy. Mapy si lze představit jako různé druhy ovládání ve hře například jinak se ovládá hráčova postava během chůze a jinak když nasedne do auta/letadla. V této práci se například takto přepíná mezi vstupy VR a ovládání menu pomocí myši.

Každá mapa má pak libovolný počet akcí. Tyto akce si lze představit velmi obecně jako například chůze, skok, použij. Nejde zatím o přímé propojení s konkrétním tlačítkem jen o rozdělení na jednotlivé akce, které hráč může vykonávat.

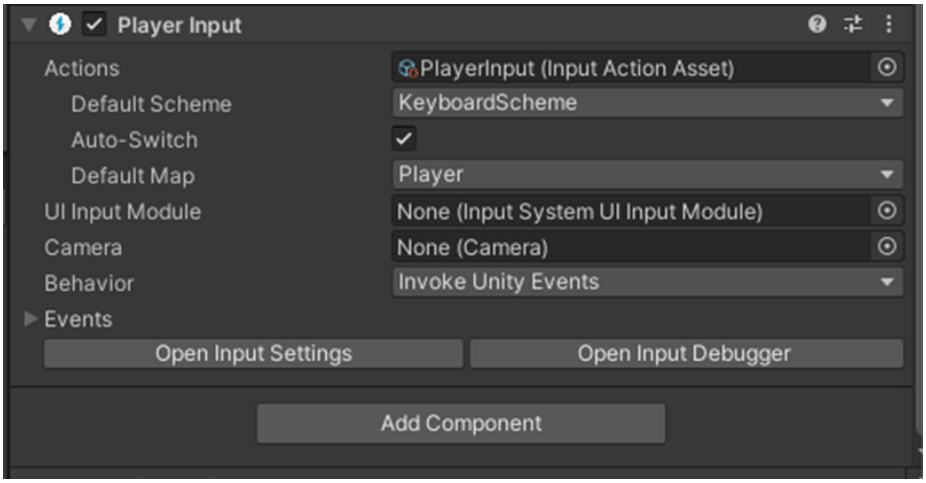
Pro každou akci pak se vytváří binding a to už je skutečné přiřazení tlačítka pro spuštění dané akce. Binding pro každou akci nemusí být pouze jeden, ale libovolné množství, a to včetně takzvaných kompozitních typů, které například simulují pomocí kláves 2d ovladač. Nejčastějším kompozitem bývá ovládání hráče pomocí kláves WSAD.



Obrázek 5.4: Editor Input Action assetu, nastavení ovládání.

Pro nový input systém existuje defaultní Input Action asset. Tento asset má v sobě už přichystané nejčastější eventy.

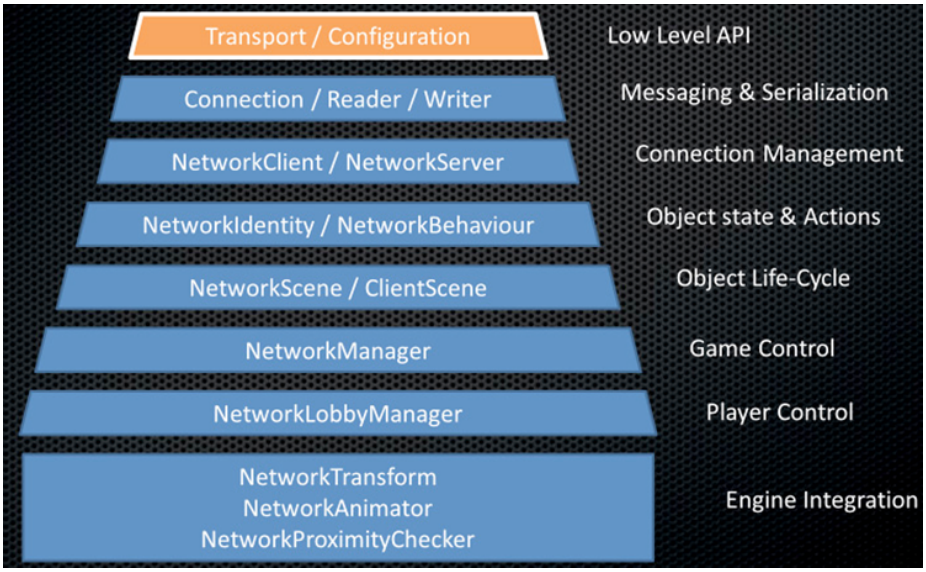
Aby celý systém fungoval, je potřeba objektu, který na vstup bude reagovat, přiřadit komponentu Player Input a v této komponentě přiřadit Input Action asset. V této komponentě pak lze na vytvořené akce reagovat voláním Unity eventů a voláním konkrétních funkcí.



Obrázek 5.5: Komponenta Player Input

5.6 Mirror

Mirror je plugin do Unity. Jedná se o systém, který umožňuje vytvořit multiplayerovou hru. Je postaven nad transportní vrstvou. Mirror je server autoritativní systém. To však neznamená, že musí vždy existovat dedikovaný server. V základním režimu je jeden klient zároveň serverem. Tento klient se nazývá host. V základu Mirror poskytuje posílání zpráv, serializaci, management síťových objektů a synchronizaci stavů.

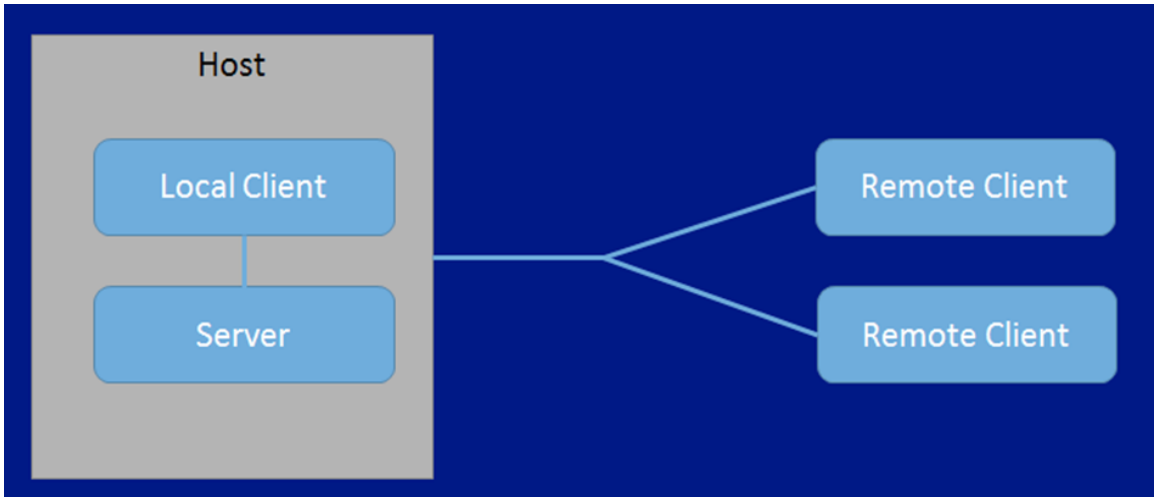


Obrázek 5.6: Komponenta Player Input

Jak již bylo zmíněno, multiplayerová hra se dělí na server a klienty. K serveru se připojují klienti a server zodpovídá za konzistentnost dat na klientech. Jde například o synchronizaci skóre a polohy objektů. Server obecně může být dedikovaný nebo host. Dedikovaný server běží zpravidla bez UI a často na fyzických serverech s veřejnou IP, aby se k němu bylo možno

připojit odkudkoli. K nasazení dedikovaných serverů se často používají velké cloudové služby jako AWS (Amazon Web Services).

Druhou variantou je mít server typu host. Host je server běžící na jednom z klientů, zpravidla to bývá ten klient, v kterém se vytvoří herní scéna. Host se chová jako server a klient zároveň.



Obrázek 5.7: Komponenta Player Input

Mirror rozlišuje několik typů objektů tím základním je objekt Player. Je to hráč, který vzniká připojením k serveru. Každý klient má svého lokálního hráče, kterého může ovládat. Klient nemá pak oprávnění jakkoli pozměnit ostatní objekty. Musí požádat server a ten zařídí příslušnou akci, pokud je validní. Existuje však nad tímto koncept autority. Server může předat správu nějakého objektu klientovi. V podstatě ale trvale musí být jasné, který klient je tvůrce pravdy vzhledem ke stavu objektu.

Tato vlastnost Mirroru vedla v diplomové práci k vytvoření vlastního správce autority. Jde o problematiku, kdy VR hráč zvedne objekt a musí být schopen objekt přemístit bez zpoždění z komunikace se serverem. Naopak když objekt upustí, musí vrátit objekt serveru, aby jej mohl chytit případně jiný hráč. Avšak s tím se dále váže problém, že musejí být serveru synchronizovány všechny síly, které na objekt působí (mnohdy VR hráč totiž objekt hodí).

5.7 Herní scény

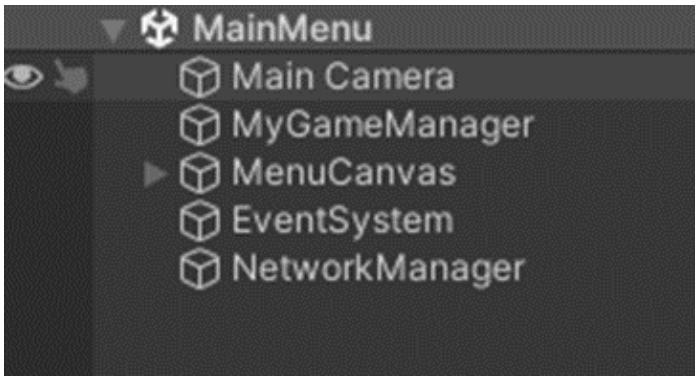
V podstatě celá aplikace se odehrává jen ve dvou scénách. V projektu původně byli tři Menu, Server a Klient. Ale Mirror funguje na tom že scéna pro server a klienta je ta stejná. Tedy místnost pro klienta byla nadbytečná a mezi klientem a serverem se rozlišuje dle způsobu načtení scény Server. Celá aplikace začíná v Menu.

5.7.1 Scéna MainMenu

UI prvky se v unity vykreslují pomocí plátna Canvas. Canvas je nastaven, aby upravoval svoje rozlišení dle referenční velikosti obrazovky 1920x1080. Toto nastavení zaručuje nezdeformované UI objekty. Jednotlivé obrazovky jsou pouze panely přes celé referenční plátno. Tyto panely se navzájem přepínají tak, že je vždy viditelný pouze jeden. Takto funguje

zobrazení samotných obrazovek v rámci menu, avšak napojení na herní scény je trochu složitější.

Menu musí být vždy modulární. Během vývoje samotné aplikace se menu velice často mění. Pokud by menu bylo namapované na konkrétní scény, bylo by potřeba při každé změně struktury přepsat. V aplikaci je také často potřeba scény přepínat a opět se objevuje problém, zda mapovat tyto přepínání staticky. U větších projektů se vyplatí založit objekt správce hry (Game Manager).



Obrázek 5.8: Objekty obsažené ve scéně MainMenu

O přepnutí na další serverovou se postará správce hry anebo pokud se připojuje hráč jako klient k serveru tak samotný network manager. Pokud se přepíná scéna přes správce hry dojde ještě k založení serverové kamery pro serverovou místnost. Serverová místnost kameru v samotné scéně neobsahuje, protože kamera je pokaždé jiná v závislosti, jak se do scény přepíná. Network manager vytvoří pro klienta rig, který již kameru obsahuje a Game Manager vytvoří zase orbitální kameru pro editování místnosti.

5.7.2 Scéna Server

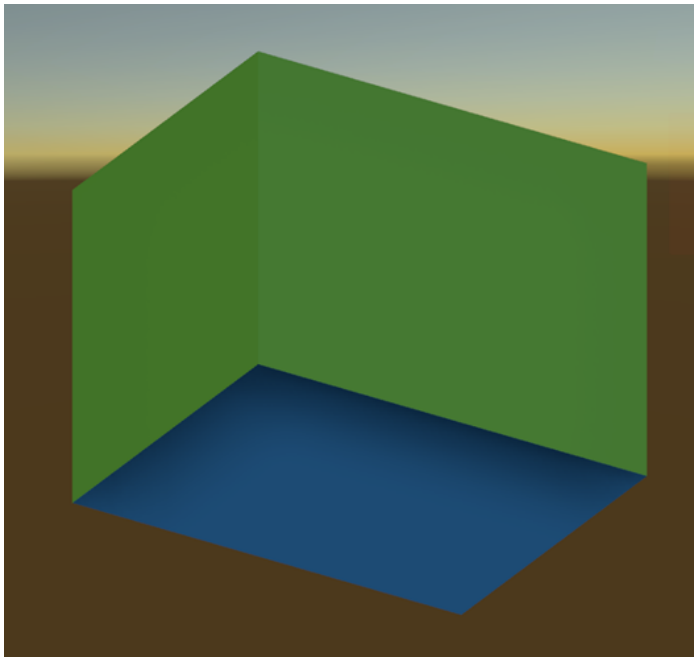
Obrázek 5.9: Objekty obsažené ve scéně Server

Jak již bylo zmíněno chování scény editoru závistí na způsobu, jakým se místnost načte. Při načtení jako server se z Menu přesunou do místnosti objekty GameManager, NetworkManager a vytvoří se onjekty OrbitCamera a Select plane. Po vstoupení do místnosti se začne

generovat výchozí prostor. Toto generování obstarává skript RoomEditor na objektu SpaceManager.

5.8 Prvky Editoru

5.8.1 Prostor



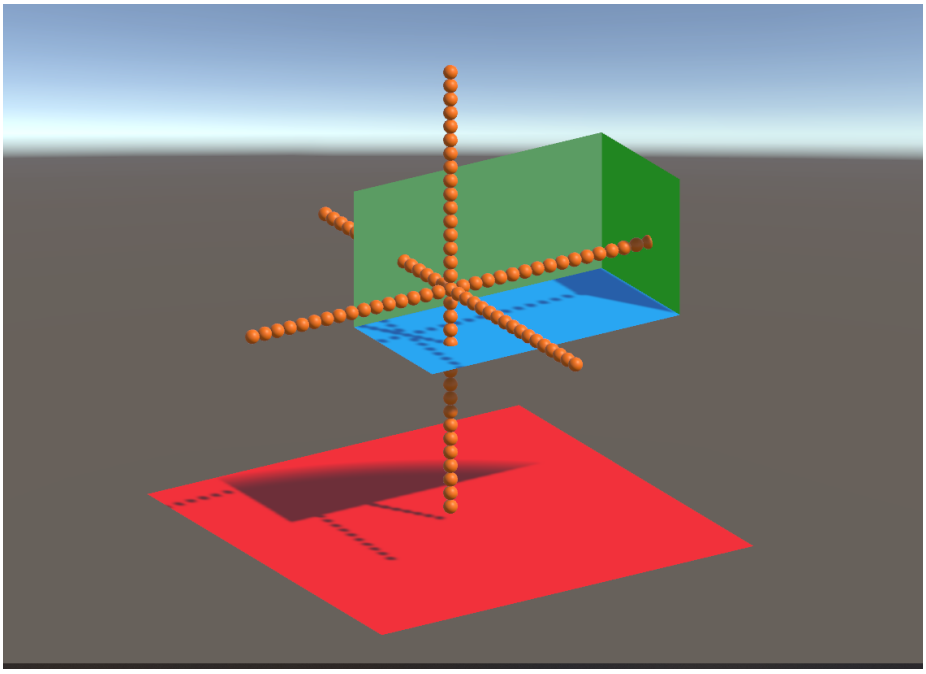
Obrázek 5.10: Výchozí vygenerovaná místnost

Vytvořená místnost má rozměry 13mx10mx10m a je určená jako základ pro libovolné uzpůsobení prostoru. Celý editor je založen na voxelovém prostoru. Voxel (základní jednotka) je kostka o délce strany 1 m. Tento prostor je ovládaný správcem SpaceManagerem, u kterého lze nastavit velikost generovaného prostoru v násobcích 2. Ideální velikost prostoru pro VR je do velikosti 64 m. Větší velikost už není vhodná primárně kvůli VR. Ve VR jsou načítací časy jednotlivých místnosti nejvíce rušivým elementem. Pro srovnání: celá hra Portal 2 od společnosti Valve neobsahuje testovací místnosti větší než 25m x 25m x 25m.

SpaceManager vygeneruje pro každou rovinu v každé ose jejího správce (SideManager). Tento správce je pak zodpovědný za samotné generování zdí ve scéně. Každý správce (SideManager) kontroluje a sravuje pouze svoji rovinu a je volán ze správce prostoru.

5.8.2 Optimalizace

Prostor takto generovaný je již značně optimalizovaný oproti standartnímu přístupu, kdy každý voxel je reprezentován objektem. Ve scéně by pak při velikosti hrany 64 došlo k vytvoření 260 000 objektů. Tento přístup je možný za předpokladu využití lokálních optimalizací a provolávání jen voxelu v dané aktivní oblasti. Avšak počáteční generování takového prostoru je značně problematické a náročné na výkon počítače a způsobuje dlouhý zásek při spuštění.

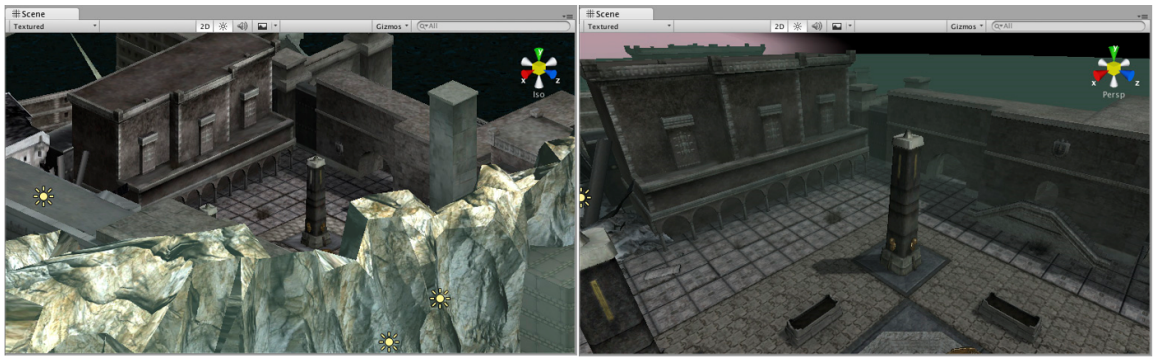


Obrázek 5.11: Oranžově vygenerovaní jednotliví spravci rovin. Zdroj: vlastní dílo.

Metoda použita v tomto projektu hierarchie správců nad rovinami je mnohem více škálovatelná a při zvětšování prostoru se počty objektů správců zvyšují lineárně. Díky tomu je možné spravovat i mnohem větší prostory.

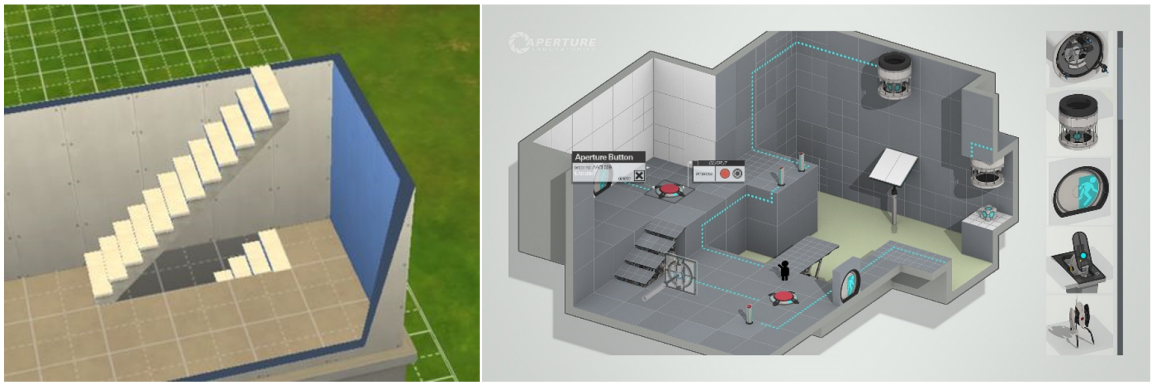
5.8.3 Pohled do místnosti

Při editaci místnosti je ve scéně použita orbitální kamera. Tato kamera zároveň podporuje zoom. Jedná se o kameru ortografickou. Toto řešení bylo zvoleno na základě předchozích zkušeností z bakalářské práce.[13] Díky ortografické projekci lze lépe pozicovat předměty a zároveň zvyšuje přehled o rozmístění prvků po úrovni.



Obrázek 5.12: Vlevo ortografické zobrazení, vpravo perspektivní. Zdroj:

Bylo potřeba vytvořit systém skrývaní stěn, aby bylo možno editovat vnitřky místností. Řešení editací vnitřních prostorů existuje několik. Konkrétně jsem se zde inspiroval hrami jako The Sims nebo opět editor místností v Portálu.

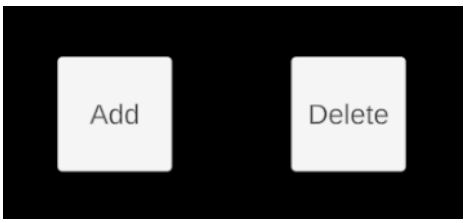


Obrázek 5.13: The Sims a Portal 2, náhled do vnitžního prostoru Zdroj: The Sims Portal 2

Zvolen byl přístup skrývání odvrácených stěn v kombinaci s ortografickou projekcí kamery. To znamená, že ke kameře jsou trvale přivrácené tři stěny a 3 jsou skryté. Při pohybu se pak počítá úhel kamery vůči souřadnicovým osám. Dle toho je vyhodnoceno, která strana má být vidět a tato informace se zasílá správci prostoru. Ten pak informaci rozesílá jednotlivým správcům rovin. Ti pak podle typu stěny vyhodnotí, zda se má skrýt a zároveň pokud ji skryjí, tak stále uchovávají odkaz na opětovnou aktivaci.

5.8.4 Úprava místnosti

Výchozí místnost lze upravovat pomocí nástrojů pro přidání nebo naopak vymazání vnitřních voxelů.



Obrázek 5.14: UI funkcí pro úpravu místnosti.

Po stisknutí příslušné tlačítko změní barvu a lze vybrat oblast pro úpravu místnosti. Výběr probíhá tak, že se stiskne levé tlačítko myši a stále při držení se vybere základna přidávaného obdélníku. Po uvolnění tlačítka se na příslušnou stranu vybere jeho výška. Po opětovném stisku se výběr potvrdí a přidají se potřebné voxely. Je to stejné přidávání jako například v 3D modelovacích programech (Blender, SketchUp).

5.8.5 Ukladací systém

Obrázek 5.15: Ukládání úrovně

Aplikace dovoluje vytvořit libovolnou místnost. Je nutné tato data umět serializovat. Editor generuje velké množství dat v podobě Voxelů, objektů a jejich parametrů. Tento ukládací systém používá formát JSON. Neboť je potřeba uložit složitější objekty, byl použit pokročilejší serializer, než je standardně v Unity. Jedná se o JSON .NET For Unity. I tento pokročilejší serializer měl s komplexností dat problémy a bylo potřeba data před serializací upravit. A samozřejmě po načtení struktury znovu manuálně složit.

Při ukládání stačí zadat jméno úrovně a uloží se k datům aplikace do složky Saves. Z této složky jsou pak načítány všechny úrovně. Při načítání jsou zobrazeny všechny soubory z této složky a uživatel zvolí, který je potřeba načíst. Při samotném spuštění místnosti na server se místnost uloží pod názvem LocalLevel a při zpětném vrácení do editoru se načte. Je to z obnovení výchozích hodnot na všech objektech.

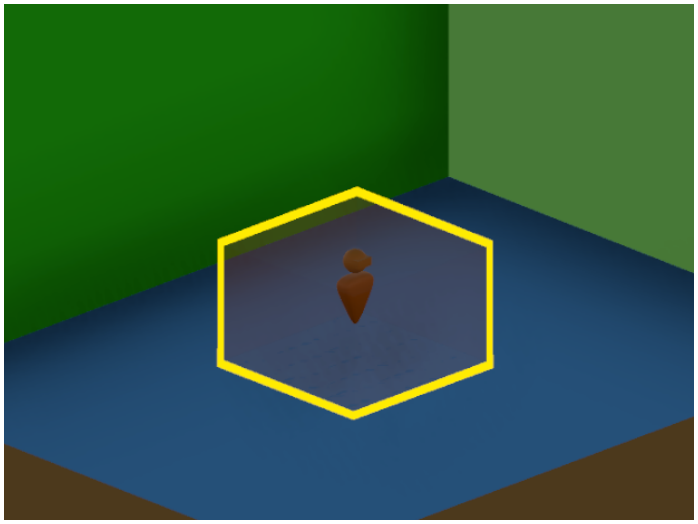
Ukládací systém je prozatím jediný způsob, jak změnit úroveň, ale je potřeba to vždy udělat manuálně. V budoucnu by měl přibýt manager úrovní, kde by se jednotlivé místnosti daly skládat za sebe a docílit tak několikaúrovňových scénářů.

5.8.6 Vložení objetku

Při startu editoru se prohladájí složky v Resources/Prefabs. Objekty, které mají příslušnou ikonu se pak považují jako objekty nachystané na spawnování. Tyto objekty se zaregistrují u správce spawnovaní, který má databázi dostupných objektů. Když je pak v editoru kliknuto na ikonu objektu, správce přes svoji databázi vytvoří požadovaný objekt. Každý spawnovatelný objekt obsahuje skript, který má uloženo, jaké komponenty se musí pro editor deaktivovat. Správce spawnování tak ihned po vytvoření takto deaktivuje objekt. Objekt bude znovu aktivován až během spuštění serveru.

Obrázek 5.16: Objekty vložitelné do místnosti

Objekty se umísťují po voxelové mřížce. Úmyslně není kontrolována kolize, aby bylo možné objekty skládat přes sebe. Po umístění objektu je objekt zaregistrován mezi aktivní objekty ve scéně. Zároveň je zaregistrována i jeho kategorie. Tyto kategorie umožňují tenkému klientovi posílat balíky prefabů. Pokud je kliknuto na přidaný objekt, objeví se jeho hodnoty v inspektoru a lze ho libovolně pozicovat. Tentokrát už není pozicování v mřížce, ale je zcela libovolné. Zároveň je vybranému objektu zvýrazněn obrys. Druhým kliknutím na objekt se výběr zruší.

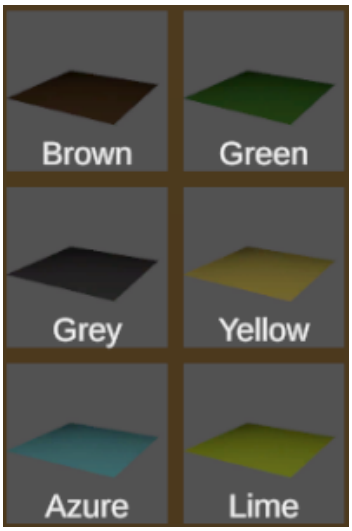


Obrázek 5.17: Vybraný objekt pro editaci

5.8.7 Textury

Snahou práce je vytvoření univerzálního editoru, proto je potřeba mít možnost ovlivnit vizuální podobu samotných místností. V diplomové práci je z toho důvodu vytvořen správce textur. Tento správce textur dovoluje obarvit stěny vybranou barvou. Údaje o vizuální podobě prostředí se ukládají zcela zvlášť. Důvodem takového rozhodnutí je, aby se systém pro dekorace mohl postupně rozrůstat nezávisle na zbytku aplikace. Nyní lze v podstatě jen stěny obarvit, neboť barvu lze uložit jako proměnou, oproti textuře nebo materiálu. Ty se ukládají odkazem. File manager, který je součástí tenkého klienta, by musel všechny přenést před začátkem úrovně. Toto chování lze doimplementovat, avšak samotný tenký klient vyžaduje ještě mnoho času na odladění a materiály by tento proces zpomalovaly. Takto se ve scéně vytváří hashovaná tabulka s potřebnými údaji během obarvovaní. Před

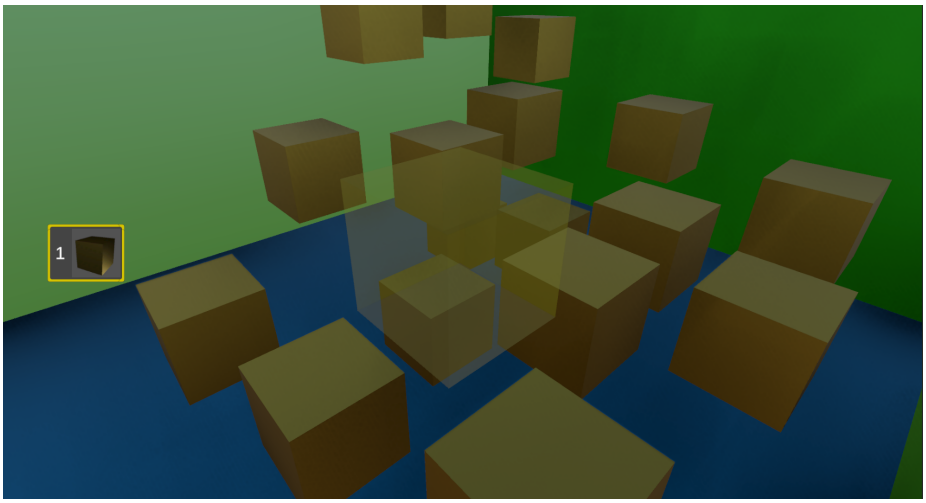
spuštěním se nahraje do struktury pouze pro čtení, která je pak synchronizována napříč klienty. Každá zeď si tak po vytvoření v tabulce zjišťuje, zda ji byl přiřazen materiál(barva).



Obrázek 5.18: Možnosti barev na jednotlivé zdi

5.8.8 Administrativní hráč

Když je místnost hotová, lze nastavit port a spustit ji na serveru. Dojde k aktivaci síťových objektů a server postupně spawnuje všechny přidané objekty v editoru. Takto se pak posílá zpráva všem klientům o existenci objektů. Zároveň Server je zpočátku vlastník všech objektů, které nejsou hráč. Orbitální kamera se deaktivuje a místo ní se načte administrativní hráč. Jedná se o hráče ovládaného na klávesnici a je správcem úrovně. Mezi jeho mechaniky patří upravovat vlastnosti již vložených překážek v místnosti, ale také zároveň může vytvářet objekty nové. Mezi jednotlivými objekty, které může spawnovat, přepíná pomocí čísel. V podstatě lze říci, že je to průvodce pro připojené VR hráče.



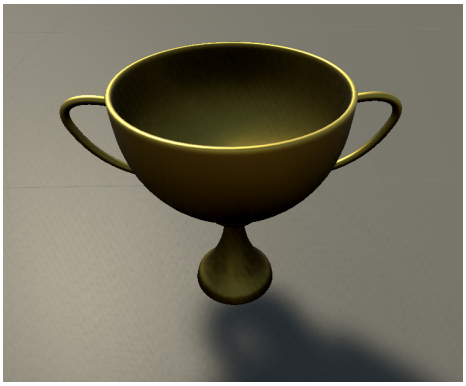
Obrázek 5.19: Administrativní hráč ve scéně

5.9 Herní prvky

V této sekci je rozepsán příklad herních prvků umístitelných v herním editoru. Editor počítá s velkým množstvím podobných objektů. Každý objekt pak může mít svoje vlastní herní mechaniky.

5.9.1 Vítězný pohár

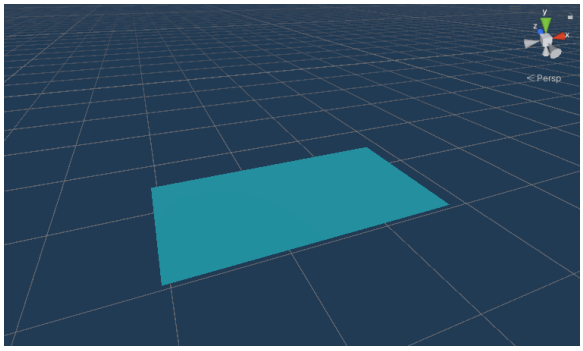
Pohár je cíl úrovně. Pokud se hráč dostane k poháru a chytne ho jako první, vyhrál místnost. Hráč je i s pohárem přesunut do speciální výherní místnosti. Pokud není pohárů více, žádný další VR hráč již místnost nevyhraje.



Obrázek 5.20: Výherní pohár

5.9.2 Objekt led

Jedná se o překážku pro VR hráče. Obvykle bývá umístěna nad propastí, či jinou nebezpečnou zónou. Pohybuje se zprava do leva a neumožnuje VR hráči se na ni teleportovat. Hráč je tak nucen načasovat průchod tak, aby stihl přeběhnout tuto plošinu, než mu podklouzne pod nohama. Pokud je plošina dostatečně pomalá, lze její pohyb kompenzovat pohybem po míst-nosti. Její rychlost ovládá hlavní hráč.

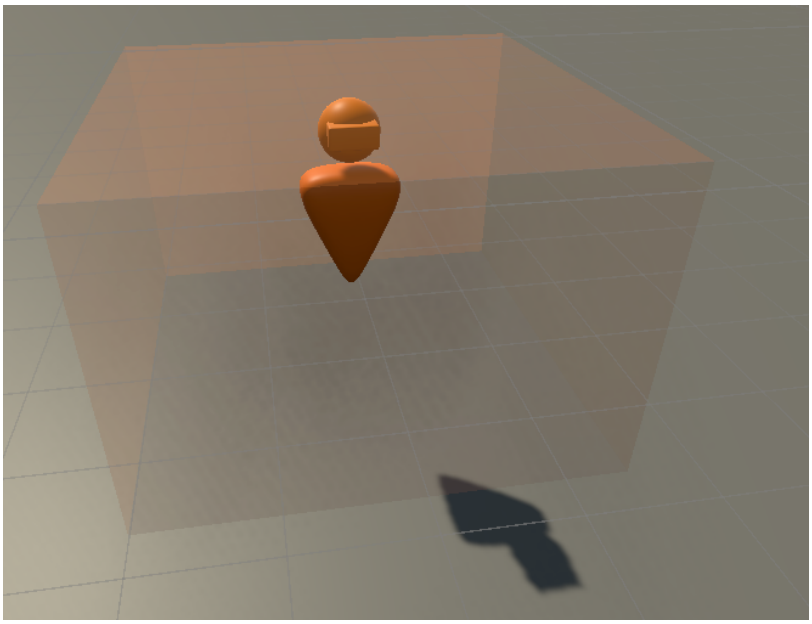


Obrázek 5.21: Objekt ledu

5.9.3 Spawn

Objekt, který ukazuje oblast, kde se budou spawnovat VR hráči. Oranžový box ukazuje přibližný rozptyl od středové polohy. Místnosti na VR obvykle nemívají více jak 3m. Pokud

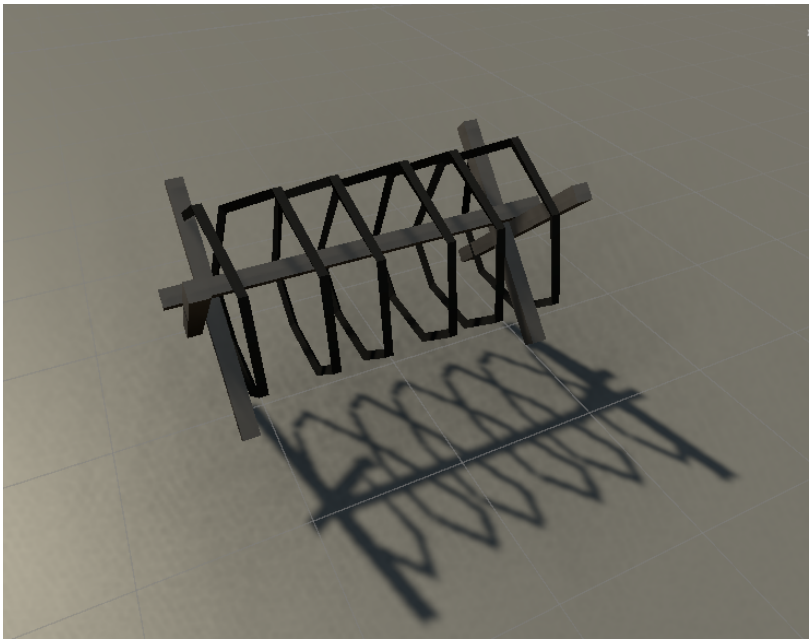
se jich do místnosti umístí více startovací poloha se vybere náhodně. Objekt je ve spuštěné místnosti neviditelný. Poloha jednotlivých spawnů se automaticky registruje v Network Managerovi hned po spuštění serveru.



Obrázek 5.22: Objekt Spawn

5.9.4 Drát

Jedná se o smrtelnou překážku pro VR hráče. Jakmile přijde VR hráč do kontaktu s drátem, je vrácen na spawn. Model meshe je z asset balíku Low Poly Ultimate Pack od polyperferct, na který vlastním licenci.



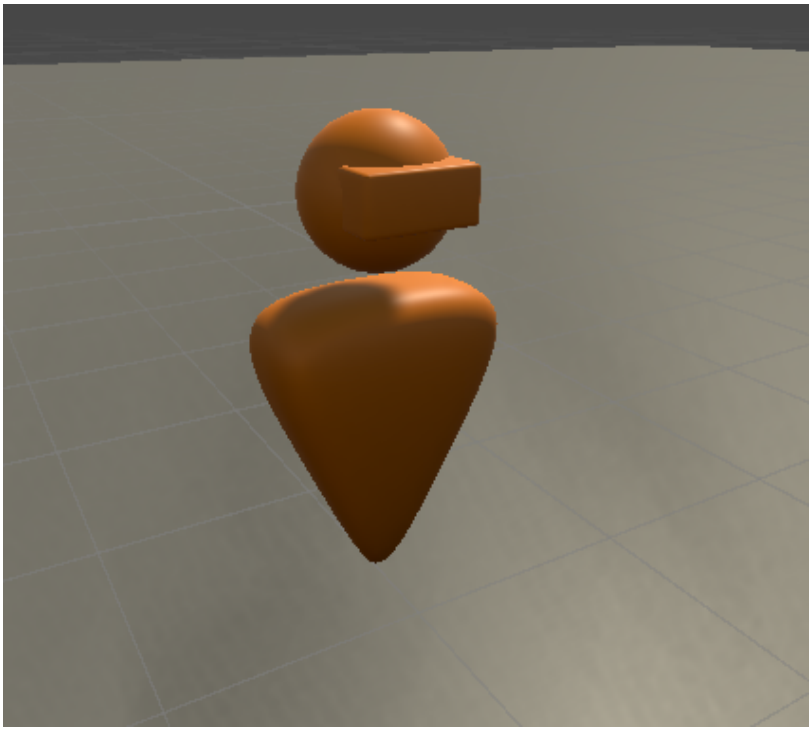
Obrázek 5.23: Ostnatý drát, překážka

5.10 Klient

Klient před svým zapnutím potřebuje IP adresu a port. K tomu slouží potřebné UI v Menu. Pokud se klient úspěšně připojí k serveru, přepne místnost na scénu server a načítá potřebná data.

Synchronizace místnosti se děje krátce po připojení zároveň s inicializací VR. Je přenesena pouze bitová mapa místnosti. Každý Klient si proto mapu generuje sám a tím šetří množství přenesených dat. Všechny objekty vložené administrativním uživatelem musí mít klient jako lokální prefaby. Kontroluje se, zda klient má všechny prefaby k dispozici (tlustý klient) a pokud ano, pokračuje se do hry. Jinak se pokusí potřebné prefaby stáhnout od serveru (tenký klient). Toto je více popsáno v kapitole o tenkém klientu.

Klient je ve scéně pak reprezentován VR hráčem a jeho úkolem je získat zlatý pohár. Obecně ale lze říct že cíl úrovně určuje administrativní hráč, který provozuje server.



Obrázek 5.24: VR hráč

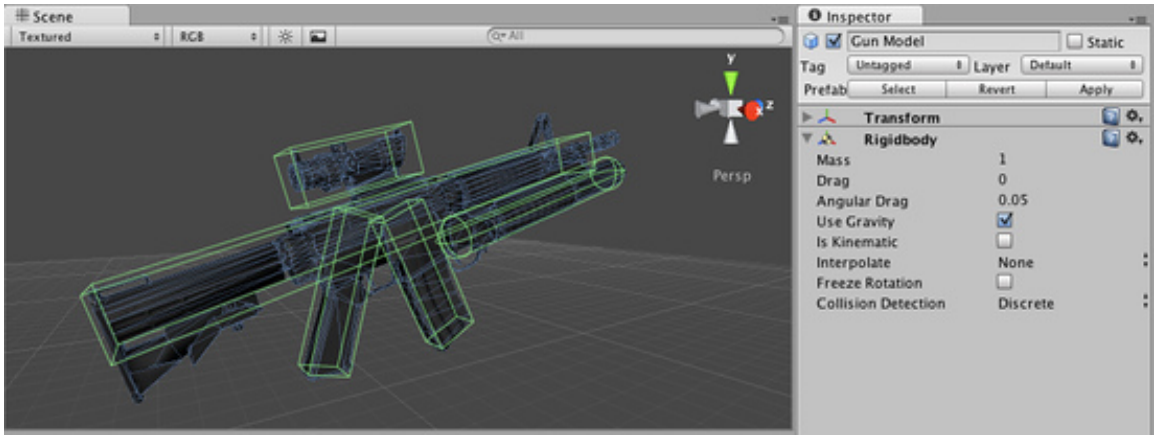
Pokud dojde k odpojení serveru Network manager, přepne scénu na menu a deinicializuje VR.

Kapitola 6

Implementace

6.1 Engine Unity

Jak již bylo zmíněno, scéna je složena z hierarchie objektů. Ty obsahují komponenty, kdy speciální typ komponenty je skript. Skripty obecně dědí z třídy MonoBehaviour. Výjímečně dědí z jiných tříd například síťové skripty dědí z NetworkBehaviour. Obecně engine zajišťuje volání klíčových funkcí při startu jako Start(), Awake(), OnEnable(), ale také funkce Update(), FixedUpdate() volané periodicky. Dále taky engine reaguje na fyzikální události. Do těch můžeme zařadit funkce jako OnCollisionEnter(), OnTriggerStay(), OnCollisionExit() a řadu dalších. Ty jsou spojeny s komponentami koliderů nebo rigidbody.



Obrázek 6.1: Kolidery, Rigidbody Zdroj: Unity

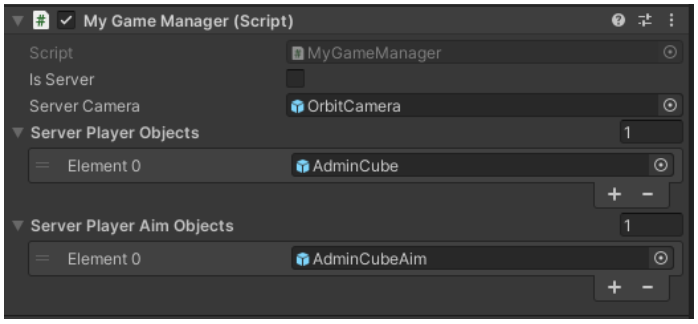
V těchto funkcích se tedy nejčastěji reaguje a skriptuje požadované chování objektu. Unity má poměrně propracovaný manuál, kde je všechno vysvětleno i s příklady. Také je důležité poznamenat možnost nastavení fyzikálního enginu. Právě pro VR mnohdy je potřeba častější přepočet anebo vypnutí kolizí mezi některými vrstvami objektů.

6.2 Správce hry

Tento objekt obsahuje stejnojmenný skript. V tomto skriptu se nachází funkce, které obstarávají přepínaní scén. Obsahuje také mnoho užitečných funkcí, které obstarávají běžný chod aplikace, včetně nastavení uživatele. Příkladem může být ukončení aplikace v libo-

volné scéně nebo preferovaný režim obrazovky. Objekt GameManager existuje nezávisle na scéně, ve které byl poprvé vytvořen. V případě této aplikace je to právě scéna Menu. Je napsán jako singleton. To znamená, že pokud se přepne aplikace zpět do menu z jakékoli herní místnosti, objekt zůstává a nový se již nevytváří. Kdekoliv v aplikaci se pak jakýkoli prvek může odkázat na GameManagera pomocí tagu a volá příslušnou funkci např. přepnutí scény. Takto lze nejen v menu, ale kdekoliv v aplikaci libovolně měnit a upravovat UI prvky, aniž by na sobě obsahovaly funkcionalitu. Zároveň GameManager umožňuje odposlouchávat přímo uživatelův vstup a reagovat tak na různé klávesové zkratky. Příkladem tohoto je rychlé vypnutí pomocí klávesy Esc.

Tento přístup také umožňuje jednoduší debugování aplikace. Jelikož je GameManager v každé scéně, lze si přes něj kontrolovat a psát výpisy ze všech objektů ve scéně.



Obrázek 6.2: Správce hry

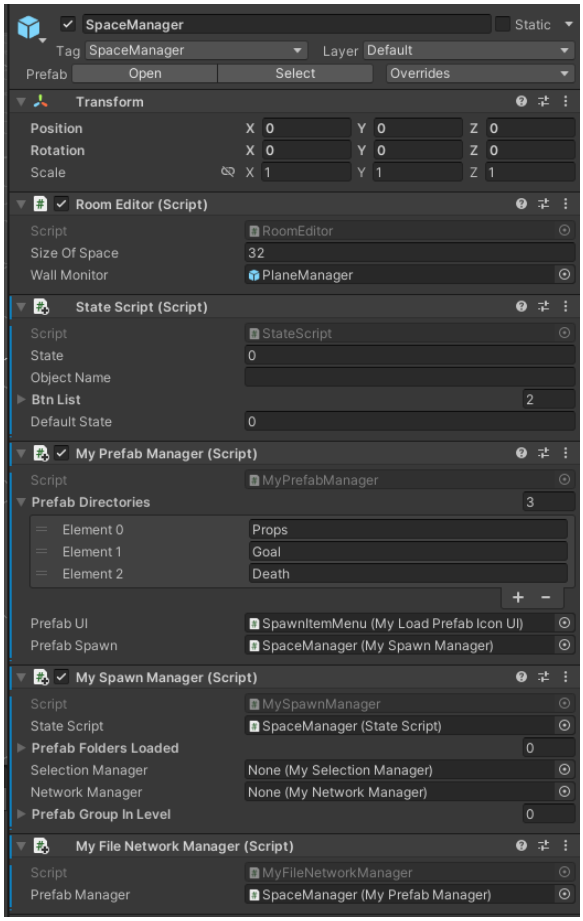
Načítání scén v Unity přešlo ze starého způsobu načítaní scén k novějšímu takzvanému Scene managementu. Většina věcí funguje za pomoci delegáta, který se ve funkci OnEnable() přihlásí k potřebným událostem, na které bude potřeba reagovat. Důležitou událostí zpravidla bývá SceneManager.sceneLoaded. Tato událost se spouští, pokud se zcela načte scéna. Tohoto se převážně využívá u síťových skriptů, které při spuštění počítají, že místnost je již zcela načtena a mohou tak začít propojovat jednotlivé skripty mezi objekty.

6.3 Správce prostoru

Samotný SpaceManager má v sobě uloženo 3D pole prostoru. Toto pole nabývá hodnot 0 pro voxely vnější a hodnoty 1 pro voxely vnitřní. SpaceManager pak obsahuje funkci, která na základě polohy voxelu spočítá, kteří manažeři rovin budou kontaktování a posílá jim informaci, zda v prostoru bude existovat zeď. Tuto informaci má díky svému poli a kontroluje vždy nejbližší okolí přidávaného voxelu. Společně s pozicí odesílá manažerům dané roviny také směr plochy. Je to z důvodu potřeby rozlišovat v rámci roviny orientace jednotlivých stěn. V jedné rovině totiž může být podlaha jedné místnosti a strop druhé. Pokud by se pak aplikovaly například skripty pro pohyb do dané roviny, tak by to mohlo zapříčinit možnost chození po stropě v druhé místnosti.

6.4 Správce jedné roviny

OneSideManager spravuje jednu rovinu. Při vytvoření si ukládá, na které ose leží a generuje 2D pole pro uložení pozic zdí v rámci roviny. Tento manažer má v sobě přiřazené prefaby již konkrétních stěn. Aby stěnu vygeneroval, musí ho kontaktovat spávce prostotu a předat mu



Obrázek 6.3: Objekt správce prostoru. Skládá se z několika důležitých komponent, včetně State Scriptu, který určuje v jakém stavu je editor.

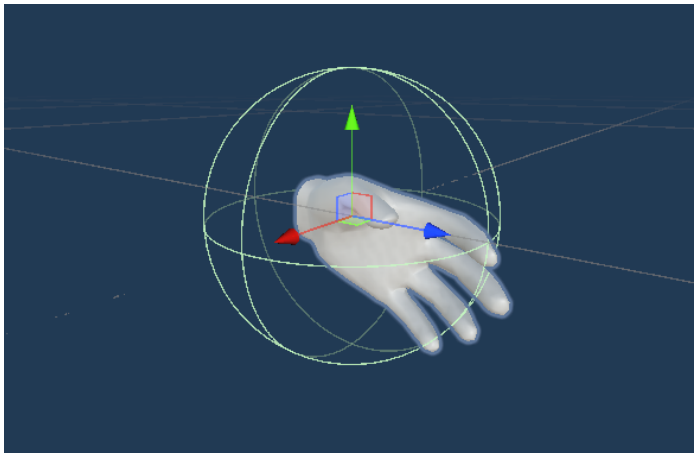
souřadnice a směr stěny. Tuto informaci si uloží správce do své tabulky a vygeneruje příslušný prefab. Tady probíhá několik kontorol, zda pozice již nejsou obsazeny, či při mazání, zda zeď jestě existuje, jelikož mohla být vymazána například optimalizací. Vytvářená zeď je pojmenována podle jejího směru. Toto rozlišení je funkčním předpokladem pro systém výběru voxelu.

Obrázek 6.4: Správce jedné strany s přiřazenými prefaby podlahy a stěny.

6.5 XR Interakce

Jak již bylo zmíněno, objekty, s kterými bude hráč interagovat, musejí na sobě mít kompenentu typu Interactable. Příkladem této komponenty je XR Grab Interactable. Objekty, které tuto komponentu obsahují, může hráč uchopit. Komponenta umožňuje nastavit několik režimů úchopu. Například režim, kdy držený objekt je pouze reprezentován meshem a není pro něj počítána fyzika v rámci místnosti. Opakem je pak režim, kdy objekt zůstává plně fyzikálně simulován, avšak toto vede na problémy se zpožděním, jelikož fyzika objektu se přepočítává nezávisle na snímkovací frekvenci. To vede k neplynulému přesouvání objektu. Také lze objektu přiřadit pomocný objekt, který reprezentuje kotvící pozici s rotací. Objekt, který takto uchopíme, pak vždy vezmeme za tento objekt. Samozřejmě lze volit z mnoha dalších nastavení. Vše je pak popsáno v dokumentaci k jednotlivým komponentám.

V každé scéně musí být nejméně jeden XR Interaction Manager. Komponenta XR Grab se u něj zaregistruje. Aby se předmět dal zvednout, hráčova ruka musí obsahovat XR Direct Interactor. V tomto případě se povoluje předmět zvednout pouze přímím kontaktem. Interactor vyžaduje přiřadit Collider, což je kolizní oblast kolem ruky.



Obrázek 6.5: Zobrazený Collider kolem ruky.

Druhou variantou by byl XR Ray Interactor. Ten pak dokáže předmět zvednout paprskem a dokonce ho přitáhnout do ruky. XR Direct Interactor se rovněž zaregistruje do XR Interaction Manageru. XR Interaction Manager pak kontroluje, zda dochází ke kolizi těchto objektů a zda dojde k aktivaci uchopení. Po uchopení objektu se dále volají funkce Authority manageru. Ten převede síťovou autoritu na hráče, který objekt zvedl a po puštění objektu vrací autoritu zpět serveru.

6.6 Implementace rukou

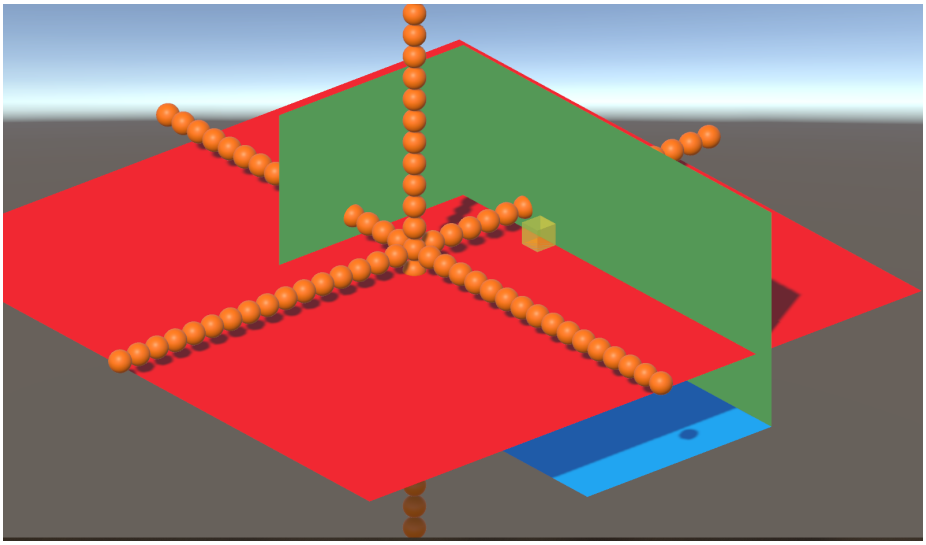
Model rukou pochází přímo z volně dostupných souborů od firmy Oculus. Byla potřeba vytvořit animace sevření ruky pro tyto modely. Na animaci se poté uplatnila maska tak, aby se vždy spouštěla jen příslušná část animace. Byl napsán skript, který má za úlohu krokovat animaci dle hodnot příslušných tlačítek na ovladači. Podpora je zatím pouze pro obecné ovladače, které nerozlišují jednotlivé prsty. Avšak v případě potřeby není problém animaci uzpůsobit a rozšířit o potřebné masky pro jednotlivé prsty.

6.7 Úprava místnosti

SelectionManager je správce výběru voxelů ve scéně. Samotný výběr se skládá ze 3 částí. Prvním kliknutí a táhnutí, poté výběr výšky a klinutím potvrzení. Toto je standardní aditivní přidávání objektů v mnoha editorech, například Blender(interaktivní tělesa). V tomto editoru se však přidávají voxely vnitřního prostoru. Výběr funguje jako stavový automat. Uchovává stav a při dokončení části přepne na další.

První kliknutí

Výběr začíná stisknutím levého tlačítka myši v oblasti stěny editoru. SelectionManager tento počátek kliknuti odchytí a vyšle do prostoru paprsek Ray. Tento paprsek vyvolává událost takzvanou hit, pokud paprsek protne kolider jakéhokoli tělesa ve scéně. Pro výběr pouze voxelů bylo zapotřebí vytvořit novou fyzikální vrstvu, ve které se nacházejí právě pouze kolidery stěn. Tato vrstva je pojmenována Room. Pokud je zasažena stěna, skript zjistí její jméno a pozici. Díky tomu lze dopočítat souřadnici voxelu, který byl zasažen "vybrán", i když samotný voxel ve skutečnosti je reprezentován pouze v paměti. Toto je poměrně výrazná optimalizace a umožňuje pracovat s velkými prostory. Následně se vytvoří objekt SelectCube, který označuje vybraný voxel. Dále se nastaví takzvaná SelectPlane na výšku vybraného pixelu. Tato rovina slouží v následujícím kroku pro raycastovaní souřadnic. Stav se změní z 0 (nevybráno) na stav 1 (vybrán první voxel).



Obrázek 6.6: Žlutě SelectCube označující vybraný voxel, červeně pak SelectPlane označující výběrovou rovinu pro druhý krok výběru. Tato rovina je v editoru neviditelná.

Tah

Po prvním kliknutí, lepe řečeno pouze stisknutím tlačítka myši, nastává fáze držení tlačítka. Během této fáze hráč vybírá druhou souřadnici v ose X a Z, které se dogeneruje Selection Cube. Ve scéně je každý snímek vysílán paprsek z pozice myši. Ten detekuje pouze souřadnici SelectPlane, je to udělané pomocí fyzických vrstev enginu Unity. Rovina je zařazena do vrstvy Select. Při vytváření paprsku se pak aplikuje bitová maska, která určuje detekovatelné vrstvy. Po zjištění souřadnice a zaokrouhlení do mřížky voxelu, je potřeba spočítat

střed. Konkrétně střed mezi počátečním voxelem výběru a konečným voxelem výběru. Tento střed slouží jako výchozí souřadnice pro selection cube. Střed se vypočítá jako S+((K-S) /2), kde S je počáteční voxel a K je vybraný koncový. Samotná vzdálenost K-S se pak použije jako lokální velikost strany Selection Cube pro každou osu zvlášť v rovině X a Z. Takto upravená selection cube pak reprezentuje druhý krok výběru. Po uvolnění tlačítka myši se změní stav z 1 (první voxel) na 2 (výběr výšky).

Výběr výšky

Ve středu Selection Cube se vygenerují dvě roviny, dle kterých se bude určovat výška. Metoda funguje obdobně jako v druhém kroku, pouze se dogenerovává Selection Cube po ose y. Po stisknutí tlačítka myši (event Button Down) se výběr potvrdí, aby nebylo možné si výběr ještě rozhodit.

6.8 Vytvoření objektů pro editor

Toto je možné zatím jen v Unity do budoucna je možné udělat plugin na přidáváni objeků. Zatím pouze manuálně. Přiblížím postup vytvoření nového prefabu: Prefab se přemístí do složky pro danou kategorii, vygeneruje se mu ikona do složky Icons, což je podsložka v každé kategorii. Složku přiřadíme do správce prostoru, který při startu projde všechny prefaby, které jsou pak k dispozici pro položení v editoru. Přidá se skript MyEditorPrefab a do něj je potřeba přiřadit rigidbody objektu. Do tohoto skriptu se nastaví i všechny komponenty, které mají být vypnuté, dokud je objekt v editoru. Mezi takové například patří pohyb objektu a další podobné skripty. Každý objekt je pak aktivován po zapnutí úrovně. Pak již stačí nakonfigurovat, aby střed objektu byl uprostřed metrové kostky (obvykle offset 0.5f) tak, aby objekt byl zarovnán s podlahou. Rovněž na sobě musí mít komponentu Network Identity a Network Transform, aby se zobrazil přes internet klientům.

6.9 Tenký klient

Diplomová práce obsahuje prvotní implementaci tenkého klienta. Tenký klient pro Unity je stále vyvíjen a v diplomové práci je implementován jako ukázka technologie, ale bude vyžadovat ještě mnoho času, než se dostane do finální podoby.

Spawn Manager si během pokládaní objektů utváří seznam složek ve složce Resources, ze kterého se objekty pokládají. Složka Resources je jako jediná v Unity speciální tím, že z ní lze načíst prefaby za běhu aplikace.

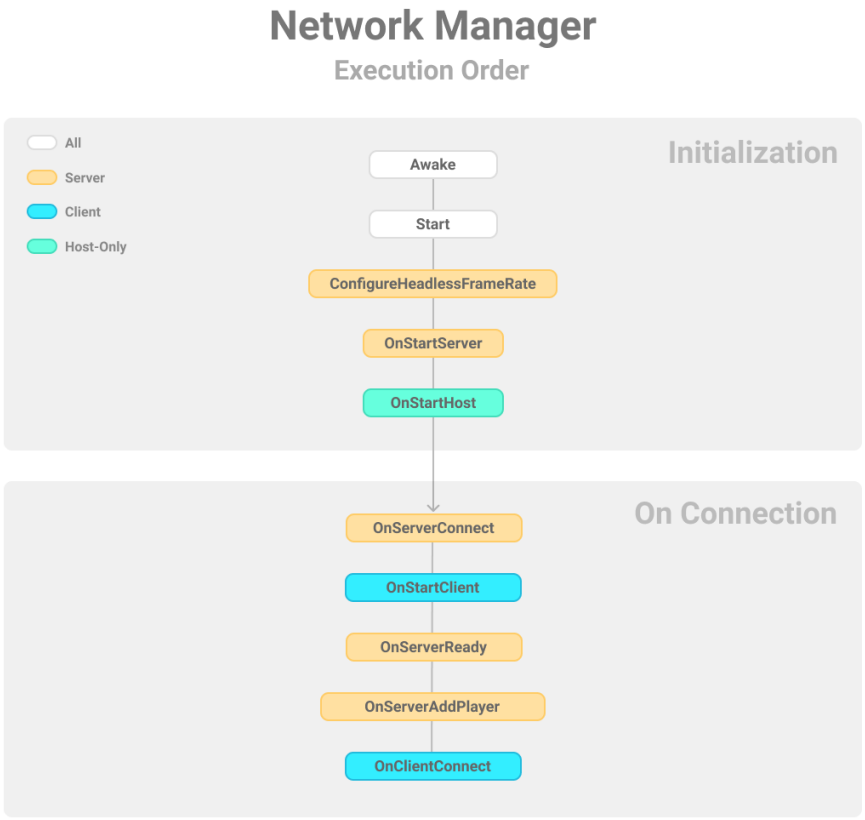
Při startu serveru se tento seznam pošle všem klientům. Klienti ověří zda jsou potřebné prefaby pro místnost k dispozici. Pokud ano pokračuje se standartně do hry a objekty se zaregistrují standardně do Network Manageru odkud jsou spawnovány.

Pokud ne je požádán File Network Manager o stáhnutí příslušných prefabů. Na serveru se musí nacházet předem předkompilované balíky assetů, které se následně pošlou klientovi, ten si je rozbalí nahraje do příslušných složek v Resources a pokračuje se standartně načtením místnosti. K zaslání asset balíků se používá serializace na bitový stream, který je odesílán pomocí Mirroru.

Toto řešení není zcela soběstačné, jelikož server potřebuje mít ke své funkčnosti předkompilované balíky assetů. Kompilaci lze zautomatizovat pluginem. Tento plugin plánuji vyvíjet v budoucnu a je svým rozsahem zcela mimo časové možnosti této diplomové práce.

Proto současnou implementaci považuji spíše za důkaz funkčnosti technologie a nereprezentuje finální podobu této technologie.

6.10 Mirror a spuštění místnosti



Obrázek 6.7: Schéma funkcí které Mirror volá při připojovaní a na které je možno reagovat. Zdroj: Mirror

Pokud je Network manager inicializovaný ale samotný server ještě vypnutý, každý objekt obsahující síťové komponenty je ve scéně deaktivován. Toto představuje problém pro editor, který nemůže tyto objekty zobrazit ani editovat. Je tak potřeba objekty registrovat samotnému Network manageru až při samotném spuštění serveru. Zároveň je potřeba myslet na to, že každý objekt, který má být sdílen přes síť, musí mít komponentu Network Identity. Na tuto identitu jsou v Mirroru vázány všechny ostatní komponenty, mezi které patří Network Transform, potažmo Network Transform Child. Mirror vyžaduje komponenty součástí kořenového objektu, proto mnoho skriptů má variantu child pro cílový objekt. A neplatí že se hierarchie automaticky celá synchronizuje napříč klienty. Tranformace nekořenových

objektů se například používá u samotného VR hráče kde tělo samotné je reprezentováno kořenovým objektem, ale ruce jsou pod ním. Pro funkci celé aplikace bylo nutno napsat upraveného síťového správce, který dědí z originálního síťového správce důležité funkce spouštění a výrazně je rozšiřuje. Těsně před spawnováním objektů si síťový správce vyžádá všechny prefaby po správci objektů. Teprve tehdy se nastavuje Ready flag, který označuje připravenost pro zprávu typu charakter. Na serveru se dle obsahu této správy vytvoří buď administrativní uživatel nebo VR uživatel. V tento moment se spouští místnost.

V této diplomové práci se velmi využívá fáze Mirroru před ready statusem. Toto umožňuje upravovat assety a nastavovat reference jednotlivých správců. V klasickém použití Mirroru, je stav sítě do ready stavu uveden ihned po připojení k serveru.

6.11 Systém uložení

Pro vytvořeni funkčního ukládacího systému bylo potřeba využít pokročilejší Json parser, než nabízí samotné Unity. Jedná se o JSON .NET For Unity. Tento parser dovoluje serializovat hodnoty typu pair <key value>. I tento serializer však nezvládal všechny typy. Hashovací tabulky s typem Vector3 bylo potřeba vždy rozložit na seznam dvojic a po načtení tabulku opětovně složit. Obecně pro uložení se serializuje třída UserData, která obsahuje trojrozměrné pole voxelového prostoru, databázi objektů a hashovanou tabulku barev. Databáze objektů obsahuje jména objektů včetně upravených pozic, rotací a velikosti. (V případě povolení dalších parametrů není problém přidat do třídy objektu další proměnné). Tato třída se serializuje a zapíše do souboru do složky Saves. Při opětovném načítání místnosti se ve scéně vymažou všechny správci rovin. S tím se automaticky smažou i jednotlivé stěny prostoru. Do Space Manageru se pak uloží nové pole voxelů z načteného souboru a vygenerují se správci rovin dle rozměrů prostoru. Poté se projde pole voxelů a dogenerují se stěny. Každá stěna si pak nastaví svoji barvu dle hashované tabulky barev.

6.12 UI

Obecně lze říci, že každé UI v aplikaci má svého správce. Obvykle to bývá skript se jménem končícím příponou UI. Správci se starají o správné přepínání UI napříč scénami a často si musí dynamicky hledat reference na objekty v kterých spouštějí funkcionalitu. Nejčastěji se tak děje pomocí vyhledávání tagu nebo předem nastavených statických referencí. Pokud se v UI zadává uživatelský vstup skript obsahuje řadu kontrol na korektnost vstupu. Příkladem může být pojmenovávání ukládaných úrovní. Toto pole je omezené pouze na alfanumerické znaky o maximální délce 20 znaků. Příkladem klasického pojmenovávání může být Level1, Theatre, Camping, Street. Dynamicky generované UI například u prefabů pak během vytváření obdrží odkaz na funkci, kterou bude spouštět a mnohdy svoji unikátní proměnou, se kterou bude funkci volat. Takto lze rozlišit jaké UI tlačítko vylo stisknuto. Příkladem paramterů jsou jména objektů pro spawnování nebo jména načítaných místností.

Kapitola 7

Závěr

Cílem práce bylo vytvořit kooperativní editor ve VR. Tento editor nepřímo navazuje na moji bakalářskou práci. [13] Zároveň tato diplomová práce obsahuje technologie na hranici možností samotného editoru Unity. Diplomová práce obsahuje primárně velké množství připravených systémů, které umožní uživatelům vytvářet vlastní obsah. Mezi systémy patří například editovatelný voxelový prostor editoru, systém načítání objektů z disku, správce ukládání, správce textur, chování jako hybridní aplikace, Server – klient prostředí, asymetrický multiplayer, správa autorit objektů. Podařilo se vytvořit funkční kostru aplikace, která spoléhá na obsah od uživatelů. Vzhledem k omezenému přístupu k potřebnému hardwaru bylo testování poměrně složité a aplikace není zcela doladěna. Mezi možná rozšíření patří implementace pokročilejšího materiálového systému spojené se spolehlivým zasíláním assetů mezi serverem a klientem. Také implementace pluginu tenkého klienta. V plánu je i vytvořit velké databáze herních objektů.

Po bakalářské práci jsem zkoušel věci změnit a přejít na engine Unreal, avšak po několika projektech jsem se vrátil zpět k Unity. Ani jeden engine není lepší, než ten druhý a vesměs kvalitu hry v dnešní době více ovlivní myšlenky a mechaniky než samotný engine. V Unity jsem se dostal v pokročilých mechanikách až tak daleko, že jsem začal narážet na samotná omezení enginu. Ale ani ta mě nezastavila a začal jsem s vývojem svých řešení (například implementace tenkého klienta). V této práci jsem pak shrnul veškeré zkušenosti z Unity a pokusil se vytvořit velmi komplexní projekt. Neodhadl jsem však časovou náročnost, ale to nevadí, jelikož diplomovou prací práce nekončí. Tak jak jsem v bakalářské práci uvedl, že budu pokračovat, tak jsem vytvořil v diplomové práci další vývojový stupeň editoru. V rozvíjení editoru plánuji pokračovat a v budoucnu jeho třetí, čtvrtou verzi zveřejnit herní komunitě.

Literatura

[1] *Mirror Networking*. Dostupné z: https://mirror-networking.gitbook.io/docs/.

[2] *OpenXR - high-performance access to AR and VR -collectively known as XR- platforms and devices*. Dec 2016. [Online; navštíveno 15.1.2022]. Dostupné z: https://www.khronos.org/openxr/.

[3] *Virtuální Realita - Historie a současnost*. Nov 2021. [Online; navštíveno 15.1.2022]. Dostupné z: https://vreducation.cz/virtualni-realita-historie-a-soucasnost/.

[4] Corporation, V. *Valve index*. [Online; navštíveno 15.1.2022]. Dostupné z: https://store.steampowered.com/valveindex.

[5] D'Anastasio, C. *How video game historians resurrected Sega's lost VR headset*. Conde Nast, Nov 2020. [Online; navštíveno 15.1.2022]. Dostupné z: https://www.wired.com/story/sega-vr-headset-video-game-preservation/.

[6] Dunn, J. *Full steam ahead: The history of valve*. GamesRadar+, Oct 2013. [Online; navštíveno 15.1.2022]. Dostupné z: https://www.gamesradar.com/history-of-valve/.

[7] Heilig, M. L. EL Cine del Futuro: The Cinema of the Future. *Presence: Teleoperators and Virtual Environments*. Srpen 1992, sv. 1, č. 3, s. 279–294. DOI: 10.1162/pres.1992.1.3.279. Dostupné z: https://doi.org/10.1162/pres.1992.1.3.279.

[8] Hestad, B. a Mason, A. *A bit of history behind Pimax VR*. Apr 2021. [Online; navštíveno 15.1.2022]. Dostupné z: https://tvovermind.com/a-bit-of-history-behind-pimax-vr/.

[9] Inc., V. [Online; navštíveno 15.1.2022]. Dostupné z: https://hello.vrchat.com/.

[10] Kumparak, G. *A brief history of Oculus*. TechCrunch, Mar 2014. [Online; navštíveno 15.1.2022]. Dostupné z: https://techcrunch.com/2014/03/26/a-brief-history-of-oculus/.

[11] Kumparak, G. *A brief history of oculus*. TechCrunch, Mar 2014. [Online; navštíveno 15.1.2022]. Dostupné z: https://techcrunch.com/2014/03/26/a-brief-history-of-oculus/.

[12] Meta. *VR headsets, Games amp; Equipment*. [Online; navštíveno 15.1.2022]. Dostupné z: https://www.oculus.com/.

[13] Michael Schneider. *Inovativní herní demo v Unity*. 2019. [Online; navštíveno 15.1.2022]. Dostupné z: https://www.fit.vut.cz/study/thesis-file/21873/21873.pdf.

[14] Neill, J. *EEE521 Final Year Project - Exploring the potential of Virtual Reality in the Delivery of Personalised Exercise*. Říjen 2020. DOI: 10.13140/RG.2.2.26723.68642. [Online; navštíveno 15.1.2022].

[15] Pimax Technologies, L. *Pimax Vision 8K X*. 2022. [Online; navštíveno 15.1.2022]. Dostupné z: https://pimax.com/.

[16] Rosson, L. *The Virtual Interface Environment Workstation (view), 1990*. NASA, Apr 2014. [Online; navštíveno 15.1.2022]. Dostupné z: https://www.nasa.gov/ames/spinoff/new_continent_of_ideas/.

[17] Spence, C., Obrist, M., Velasco, C. a Ranasinghe, N. Digitizing the chemical senses: Possibilities pitfalls. *International Journal of Human-Computer Studies*. 2017, sv. 107, s. 62–74. DOI: https://doi.org/10.1016/j.ijhcs.2017.06.003. ISSN 1071-5819. Multisensory Human-Computer Interaction. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1071581917300885.

[18] Technologies, U. *2019 LTS*. [Online; navštíveno 15.1.2022]. Dostupné z: https://unity.com/releases/2019-lts.

[19] Technologies, V. *Varjo is a VR/XR company changing computing for good.* Nov 2021. [Online; navštíveno 15.1.2022]. Dostupné z: https://varjo.com/company/.

[20] Unity Technologies. *Unity User Manual*. 2019. [Online; navštíveno 9.10.2021]. Dostupné z: https://docs.unity3d.com/Manual/.