
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE


# NÁVRH VR DÍLNY

DESIGN OF A VR WORKSHOP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS


AUTOR PRÁCE — Filip Veškrna
AUTHOR

VEDOUCÍ PRÁCE — Ing. Tomáš Hůlka
SUPERVISOR



BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

# Zadání bakalářské práce

| | |
|---|---|
| Ústav: | Ústav automatizace a informatiky |
| Student: | **Filip Veškrna** |
| Studijní program: | Strojírenství |
| Studijní obor: | Aplikovaná informatika a řízení |
| Vedoucí práce: | **Ing. Tomáš Hůlka** |
| Akademický rok: | 2020/21 |

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

## Návrh VR dílny

**Stručná charakteristika problematiky úkolu:**

Úkolem studenta bude nejprve stručné zhodnocení dostupných enginů pro VR. Ve vhodně zvoleném prostředí pak vytvoří model VR dílny, který bude doplněn o aktivní a interaktivní prvky a otestuje ho na reálném VR headsetu.

**Cíle bakalářské práce:**

Stručná rešerše problematiky.
Vytvoření VR modelu dílny s aktivními a interaktivními prvky.
Otestování funkčnosti na reálném zařízení.

**Seznam doporučené literatury:**

WANG, S., MAO, Z., ZENG, C., GONG, H., LI, S. and CHEN, B.: A new method of virtual reality based on Unity3D, 2010 18th International Conference on Geoinformatics, Beijing, 2010, pp. 1-5.

TRENHOLME, D., SMITH, S.P.: Computer game engines for developing first-person virtual environments, Virtual Reality (2008) 12: 181.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

| | |
|---|---|
| doc. Ing. Radomil Matoušek, Ph.D.<br>ředitel ústavu | doc. Ing. Jaroslav Katolický, Ph.D.<br>děkan fakulty |

# ABSTRAKT

Tato bakalářská práce pojednává o problematice vytváření programu pro zařízení virtuální reality. V teoretické části jsou rozebrána dostupná zobrazovací a pomocná zařízení. Dále se práce zabývá popisem dostupných enginů pro tvorbu VR aplikací. V praktické části je vytvořena aplikace pro virtuální realitu v programu Unity, reprezentující model domácí dílny. Aplikace obsahuje množství aktivních a interaktivních prvků za účelem vytvoření co nejpřesvědčivějšího dojmu z reálné dílny.



# ABSTRACT

This bachelor thesis deals with creating a program for virtual reality devices. The theoretical part analyzes available headsets and other accessories. Furthermore, the work deals with the description of available engines for creating VR applications. In the practical part of the thesis, an application for virtual reality is created in Unity, representing a simple workshop. The application contains a number of active and interactive elements in order to create the most convincing impression of a real workshop.



# KLÍČOVÁ SLOVA

Virtuální realita, Unity, návrh dílny, VR hardware, XR interaction toolkit

# KEYWORDS

Virtual reality, Unity, design of workshop, VR hardware, XR interaction toolkit

ÚSTAV AUTOMATIZACE A INFORMATIKY

2021

# BIBLIOGRAFICKÁ CITACE

VEŠKRNA, Filip. *Návrh VR dílny*. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/132854. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Tomáš Hůlka.

# PODĚKOVÁNÍ

Tímto děkuji panu Ing. Tomášovi Hůlkovi za odborné vedení práce.

# ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 22. 5. 2021 ……

Filip Veškrna

# OBSAH

# 1 ÚVOD

Virtuální realita, v minulosti vize budoucnosti, v dnešní době rychle se rozvíjející technologie, má velice široké spektrum využití, ať už se jedná o jednodušší využití, jako například zobrazování 360° videa nebo o složitější využití, jako například simulace pracovního prostředí. Tato technologie nám nabízí zažít aktivity, které jsou v realitě mnohdy těžko dostupné. Každý určitě někdy snil o tom, že by se rád podíval do vesmíru nebo o setkání s dinosaury. Díky simulaci, které jsme schopni dosáhnout pomocí technologie virtuální reality, je tohle vše možné.

Abychom mohli aplikaci pro virtuální realitu spustit a skutečně se do virtuální reality ponořit, potřebujeme hned několik zařízení, mezi které patří převážně zobrazovací zařízení a k nim příslušející ovládací zařízení. Tomuto tématu se věnuje první teoretická část této práce, kde jsou mimo zobrazovací zařízení také stručně shrnuta ostatní podpůrná zařízení, jako například haptické vesty nebo rukavice.

Ve druhé teoretické části jsou jednotlivě zhodnoceny dostupné enginy sloužící k vývoji simulačních aplikací pro virtuální realitu. Dostupných enginů je hned několik, každý má své výhody a nevýhody a každý je více vhodný na jiný typ aplikace. Enginy, využívající se pro vývoj aplikace ve virtuální realitě jsou zpravidla takové, které slouží k vývoji počítačových her. Mají tedy většinou již vestavěné systémy sloužící k simulaci fyzických vlastností objektů, které se poté dají snadno aplikovat na objekty v aplikaci, a právě to poté tvoří velice realistický zážitek.

Cílem praktické části této práce je vytvoření modelu dílny ve virtuální realitě. Toho je docíleno pomocí herního enginu Unity, který je díky své jednoduchosti, široké podpoře pro vývoj aplikací v XR a nepřebernému množství uživateli vytvořených a volně dostupných assetů skvělým nástrojem právě pro tento účel. Model je vytvořen pro VR headset Oculus Rift S, který nabízí 6 stupňů volnosti, kterých je v modelu plně využito. Model je doplněn o množství aktivních a interaktivních prvků s ohledem na zachování realističnosti tak, aby výsledný dojem na uživatele byl co nejblíže k dojmu z reálné dílny.

# 2 DOSTUPNÉ ZAŘÍZENÍ PRO VR

Virtuální realita, nezastavitelný trend dnešní doby má velké využití ve videoherním průmyslu a díky tomu láká nemalé množství uživatelů. Čím dál více společností vstupuje na trh s novými zařízeními a se snahou konkurovat ostatním výrobcům. To vytváří širokou nabídku dostupných zařízení. Mezi největší a nejpopulárnější výrobce patří například HTC, Oculus (Facebook), Sony, Microsoft a další.

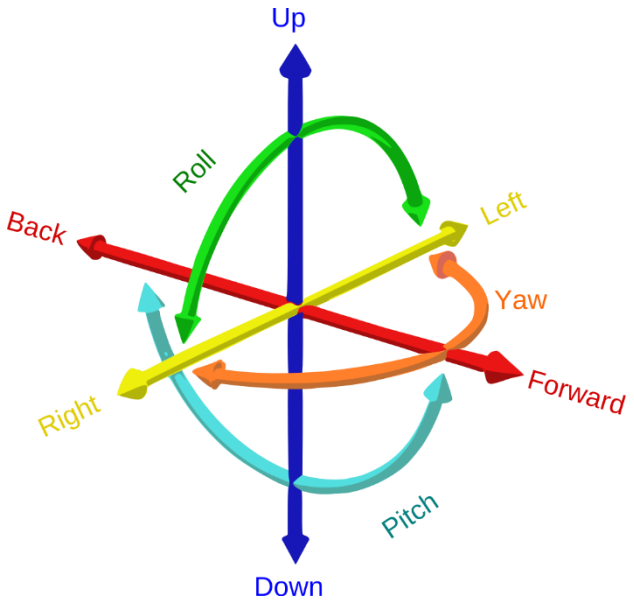
## 2.1 Trackovací technologie

Existují různé metody snímání pohybu [1,2], které svojí kombinací dovolují pohyb v prostoru.

Zařízení pro virtuální realitu se v tomhle ohledu dělí na dvě kategorie: zařízení se třemi stupni volnosti a zařízení se šesti stupni volnosti.

Tři stupně volnosti znamenají, že zařízení snímá pouze rotaci. V praxi to tedy vypadá tak, že po nasazení brýlí pro virtuální realitu se může uživatel pouze rozhlížet okolo sebe a jakmile se pohne ve fyzickém prostoru, tak ve virtuálním prostoru setrvá na stejném místě. Pro snímání rotace se využívá gyroskop. Tento princip využíval dnes již nedostupný headset Oculus Go a veškeré zařízení využívající mobilní telefon jako zobrazovací zařízení.

Zařízení se šesti stupni volnosti dovoluje kromě rotace kolem os také pohyb podél os. Tento typ pohybu poté dovoluje volný pohyb ve fyzickém prostoru, který se poté přenese do prostoru virtuálního.




Obr. 1: Stupně volnosti pohybu [1]

K tomu, aby byl umožněn pohyb ve všech osách je zapotřebí aby jak samotný headset tak i ovladače byly nějakým způsobem snímány. Moderní HMD (head mounted display) využívají jeden ze dvou druhů snímání.

Prvním z těchto dvou druhů je takzvaný outside-in. Ke snímání pohybu se využívají externí kamery umístěné na stacionárních pozicích mířících směrem na snímaný objekt. Ke snímaní se používají senzory snímající infračervené světlo.

Druhým způsobem snímaní pohybu je Inside-out. Oproti předchozímu způsobu jsou snímače umístěny přímo na zařízení a pomocí snímání okolního prostředí určují svoji polohu v prostoru. Tento systém snímání má jednu obrovskou výhodu a tou je že není potřeba mít předem přichystané pole, ve kterém se uživatel bude pohybovat. V ranné fázi vývoje této technologie bylo zapotřebí k přesnému snímání polohy mít umístěné na stěnách místnosti pomocné značky, ale díky vývoji technologie to v dnešní době již potřeba není a Inside-out tracking dosahuje stejných výsledků jako outside-out snímání. S touto technologií ale přichází i mnoho nedostatků, mezi které patří převážné nekompatibilita s některými pomocnými zařízeními a potřeba, aby prostor mezi snímačem a snímaným objektem nebyl nijak blokován. Tento problém nastává převážně jestliže uživatel umístí ovladače za hlavu, nebo blízko HMD.

## 2.2 HTC

Společnost HTC ve spolupráci se společnosti Valve vyvíjí několik řad produktů zaměřených na různé potřeby uživatelů [3].

HTC VIVE byl společně s Oculus Rift CV1 jedním z prvních komerčně dostupných VR headsetů. Kvůli nízkému rozlišení displeje (1080x1200 na oko) byl velice viditelný takzvaný „screen door effect“, který výrazně ovlivňoval celkový prožitek.


Obr. 2: HTC VIVE [3]

Nástupcem původního VIVE je zařízení s názvem VIVE Pro. Výrazné zlepšení zaznamenalo rozlišení, které činí 1440 x 1600 pixelů na jedno oko. Zařízení i přes svůj věk stále vyčnívá svým vysokým zorným polem 110° a velice komfortní konstrukcí.


Obr. 3: HTC VIVE Pro [3]

Nejnovějším komerčně dostupným zařízením společnosti HTV VIVE je VIVE Cosmos. Zařízení zaznamenalo vylepšení v několika směrech. Rozlišení displejů bylo zvýšeno na 1440 x 1700 pixelů a typ displeje byl z původního AMOLED displeje změněn na LCD panel, který nabízí ostřejší obraz a lepší podání barev.

Základní verze této řady nabízí inside-out tracking, kdežto vyšší verze s označením Cosmos Elite nabízí původní snímaní pomocí externích senzorů, které dosahuje přesnějšího snímaní pohybu.

Obr. 4: HTC VIVE Cosmos Elite [3]

## 2.3 Valve Index

Index, VR headset společnosti Valve, se zaměřuje převážně na hraní PC VR her a tomu odpovídají specifikace displejů [4]. Rozlišení je stejné jako u HTC Vive Pro (1440 x 1600 na jedno oko), zorné pole dosahuje 140° a variabilní obnovovací frekvence displeje může být až 144 Hz.


Obr. 5: Valve Index [4]

Ovladače Index Knuckles se od ovladačů ostatních výrobců liší unikátní funkcí snímaní polohy jednotlivých prstů. Ovladač je k ruce uchycený pomocí pásku. To dovoluje uvolnit všechny prsty bez toho, aniž by uživatel ovladač upustil na zem. V neposlední řadě ovladač disponuje senzory snímající sílu stisku a tak ovladač rozpozná rozdíl mezi lehkým a těsným stiskem.

Valve Index nabízí beze sporu nejlepší PC VR zážitek. Kromě vysoké ceny, která činí okolo 1100€, má zařízení také vysoké systémové požadavky. To zapříčiňuje nedostupnost pro mnohé potenciální uživatele.

## 2.4 Oculus

Oculus, značka vlastněná firmou Facebook Technologies, LCC vyvinula řadu komerčních produktů, mezi které patří jak VR využívající výkon počítače tak i autonomní VR [5].

Mezi autonomní zařízení patří již vyřazený headset Oculus Go a řada produktů s názvem Oculus Quest, kdy nejnovější model má označení Quest 2. Jelikož je headset autonomní, nepotřebuje k funkčnosti žádné externí zařízení. Po připojení headsetu k počítači pomocí USB či Wi-Fi, headset nabízí stejný zážitek jako ostatní zařízení pro virtuální realitu.

Obr. 6: Oculus GO [5]

Druhým, aktivně prodávaným zařízením značky Oculus je zařízení nesoucí název Oculus Rift S. Tento headset vydaný roku 2019 je nástupcem modelu Oculus Rift CV1 a oproti svému předchůdci nabízí inside-out tracking a lepší rozlišení displeje.

Obr. 7: Oculus Rift S [5]

Oba aktuálně prodávané modely společnosti Oculus využívají Inside-out snímaní pohybu a díky své nízké ceně jsou jedním z nejvíce populárních komerčních headsetů.

Headset Oculus Rift S byl vybrán jako primární testovací zařízení této práce. Oproti headsetům společnosti HTC Vive nabízí lepší parametry a kvalitu snímání pohybu. V době psaní práce mělo API OpenVR SDK potíže s kompatibilitou mezi SDK a XR interaction toolkitem, proto nebyl headset Valve Index shledán vhodnou volbou.

## 2.5 Ostatní periferie

### 2.5.1 Přídavné snímače polohy

Využití přídavných snímačů polohy se dělí na dvě kategorie.

První kategorií je takzvané „full body tracking“, kdy se přidáním snímačů umístěných na chodidlech uživatele docílí přesnějšího snímání polohy celého těla. Pro zpřesnění snímaní je zapotřebí přidat třetí snímač umístěný na pase uživatele. Společně se snímáním polohy headsetu, ovladačů a přídavných snímačů je poté docíleno prezentace těla uživatele ve virtuálním prostoru.

Druhou kategorii využití je přenesení fyzických objektů do virtuálního prostoru. Umístěním snímače na fyzický objekt lze sledovat polohu objektu v reálném čase a tu poté reprezentovat v prostoru virtuálním určitým objektem. Příkladem použití může být umístění snímače na sportování nástroj, například tenisovou raketu, a v příslušném softwaru tento nástroj využít jako ovladač. Dalším využitím může být umístění snímače na kameru. Poloha této kamery se poté v reálném čase přenese do softwaru, kde příslušnou polohu zaujme kamera virtuální. Díky tomu je možné naráz snímat obraz v reálném světě s příslušným obrazem ve virtuálním prostředí.

Nejrozšířenějším zařízením této kategorie je VIVE Tracker [6], který je kompatibilní s velkým množstvím VR brýlí využívající outside-in snímání polohy, tedy externí snímače polohy.

Cena jednoho snímače se pohybuje v rozmezí od 2500 do 3000 Kč.

Obr. 8: VIVE Tracker [6]

### 2.5.2 Haptické rukavice a oblek

Haptické zařízení je takové zařízení, které pomocí vibrací a aplikování sil simuluje dotyk na část těla. Využívají se jako přídavné zařízení umocňující požitek uživatele ze simulace.

Haptický oblek má strategicky umístěné aktory a snímače na různých částech těla, pomocí kterých dává uživateli zpětnou vazbu na akce uskutečněné v simulaci. V kombinaci se snímači které sbírají zdravotní data z jednotlivých částí těla je toto zařízení ideálním nástrojem pro trénování sportovců popřípadě pro jiné profese, kde by tato technologie mohla být přínosem. [7]

Obr. 9: TeslaSuit [7]

Chybějící zpětná vazba při uchycení objektů ve virtuální realitě ubírá na realističnosti simulace. Tento nedostatek řeší haptické rukavice, které pomocí mechanismů mohou zamezit pohybu prstů po překročení určité meze, nebo naopak mohou s prsty hýbat a tím simulovat dění v simulaci. Pomocí velmi jemně rozmístěných haptických aktorů je možné simulovat realistické uchopení objektů s různými druhy povrchu. Samotné rukavice poté zastupují pozici ovladačů, kdy pomocí snímačů polohy pro různé části ruky dodávají mnohem přesnější snímaní polohy než klasické ovladače dodávané k VR brýlím. [8]


Obr. 10: Haptické rukavice haptx [8]

# 3 DOSTUPNÉ ENGINY

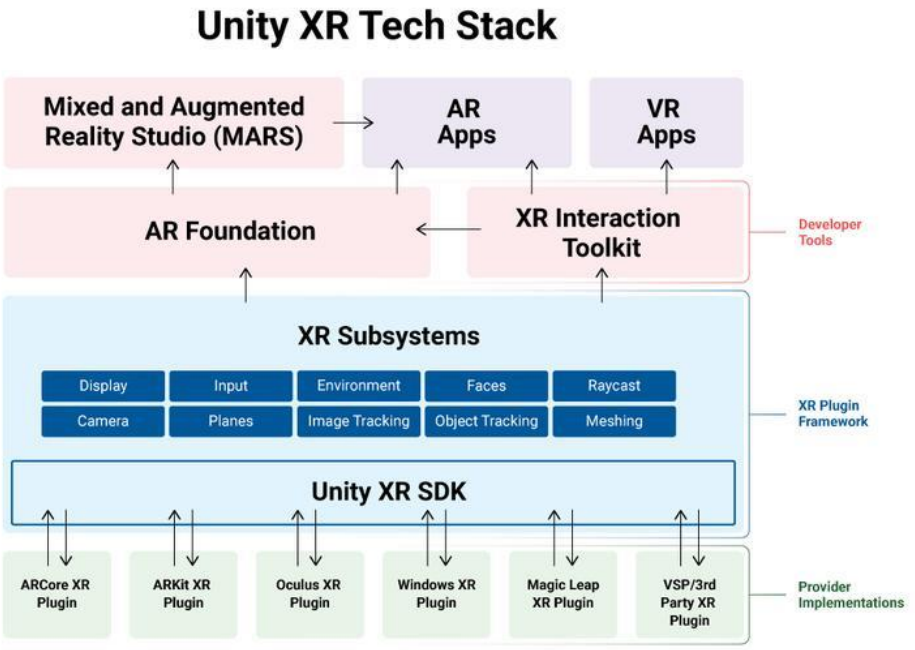
Engine, nezbytná součást pro vývoj každé VR aplikace, je software obsahující potřebné nástroje a API's pro vývoj a optimalizaci videoher. Pro vývoj běžných počítačových her existuje nepřeberné množství dostupných enginů, ale málokterý z nich nabízí dostatečnou podporu pro VR. Mezi nejpodporovanější a nejrozšířenější patří Unity a Unreal Engine, které jsou v tomto oboru běžným standardem [9].

Pro tuto konkrétní práci byl zvolen vývojový engine Unity, který oproti Unreal Engine nabízí o něco pokročilejší, uživatelsky přívětivější prostředí pro vývoj VR aplikace. Programovací jazyk využívaný Unity je C#, Unreal Engine využívá jazyka C++ nebo vizuálního skriptování.

## 3.1 Unity

Unity je výborným nástrojem pro tvoření vysoce kvalitního softwaru pro virtuální realitu. Díky dobře vypadající a optimalizované technologii vykreslování High Definition Render Pipeline unity nabízí možnost vytvoření realisticky působícího prostředí s nižším dopadem na výkon než ostatní konkurenční enginy [10, 11].

Čím se unity výrazně liší od ostatních enginů je stavba architektury pro XR aplikace. Unity spojuje XR pluginy jednotlivých výrobců do unifikovaného Unity XR SDK, který společně s XR Interaction Toolkitem nabízí velice jednoduchou implementaci XR prvků. Výhodou toho unifikovaného přístupu je možnost vytvářet multiplatformové aplikace bez nutnosti vytváření specifických nastavení pro každý headset.




Obr. 11: Architektura Unity XR

Samotné vytváření VR aplikací se poté výrazně neliší od vytváření aplikací pro běžné použití a proto je velice jednoduché do aplikací vytvořených v novějších verzích Unity (2019 +) implementovat možnost zobrazení a ovládání ve virtuální realitě [12].

Využívání Unity je zcela zdarma i pro komerční užití do momentu, kdy firma nebo jednotlivec na vytvořeném softwaru nemají zisk větší jak 100.000 $ za posledních 12 kalendářních měsíců. Poté je každý člen nucen využít placenou variantu. Cena placené varianty se liší na základě výnosu ze softwaru a dostupných funkcí. [13]

## 3.2 Unreal engine

Unreal Engine je oproti Unity vhodný spíše na aplikace, kde je hlavním parametrem vzhledová stránka programu. Renderovací technologie využívaná jak na klasické programy tak na programy VR je vysoce zaměřená na realističnost, to ale přináší své nedostatky mezi které patří převážně hardwarové nároky výsledného programu a rozsáhlejší grafická optimalizace [14].

Unreal Engine využívá ke spojení XR pluginů jednotlivých výrobců standard OpenXR, jehož je stejná firma vlastnící Unreal Engine základajícím členem.

Unreal Engine je stejně jako Unity zcela zdarma pro nekomerční využití. Jestliže je software vytvořený v tomto Enginu komerčně dostupný a přesáhne celkového výdělku minimálně jeden milion dolarů, je vydavatel povinen platit 5% svého výdělku ze softwaru společnosti Epic Games vlastnící Unreal Engine.

# 4 VLASTNÍ ŘEŠENÍ

K realizaci praktické části práce bylo využito vývojového herního enginu Unity. Tento herní engine byl zvolen hned z několika důvodů, mezi které patří hlavně široká komunitní podpora, přehledně zpracovaná dokumentace a programování v jazyce C#. Konkurenční Unreal Engine sice nabízí kromě programování v jazyce C++ i programování ve vizuálním programovacím systému nazývaným "Blueprints", který je sice přívětivější pro začínající vývojáře, ale díky složitější povaze systému využívajícího se pro ovládání ve virtuální realitě tento systém není vhodnou volbou.

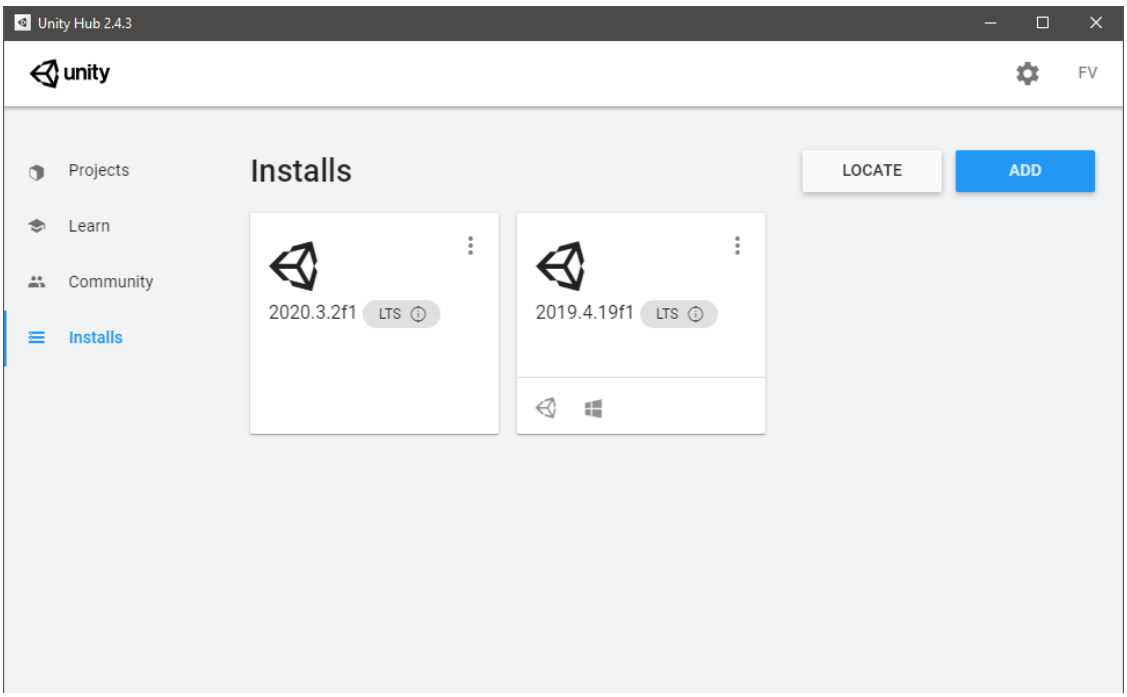
Aplikace byla vyvíjena primárně pro zařízení Oculus Rift S, při zakládání projektu bylo ale zvoleno takových nástrojů, aby se zachovala možnost jednoduchého předělání aplikace na jiné zařízení pro virtuální realitu.

## 4.1 Založení projektu

Po stažení aplikace Unity Hub bylo potřeba přidat příslušnou verzi enginu. V tomto případě se jedná o verzi 2019.4.(LTS). Tato verze byla vybrána především kvůli tomu, že se jedná v době vytváření projektu o nejnovější LTS verzi.

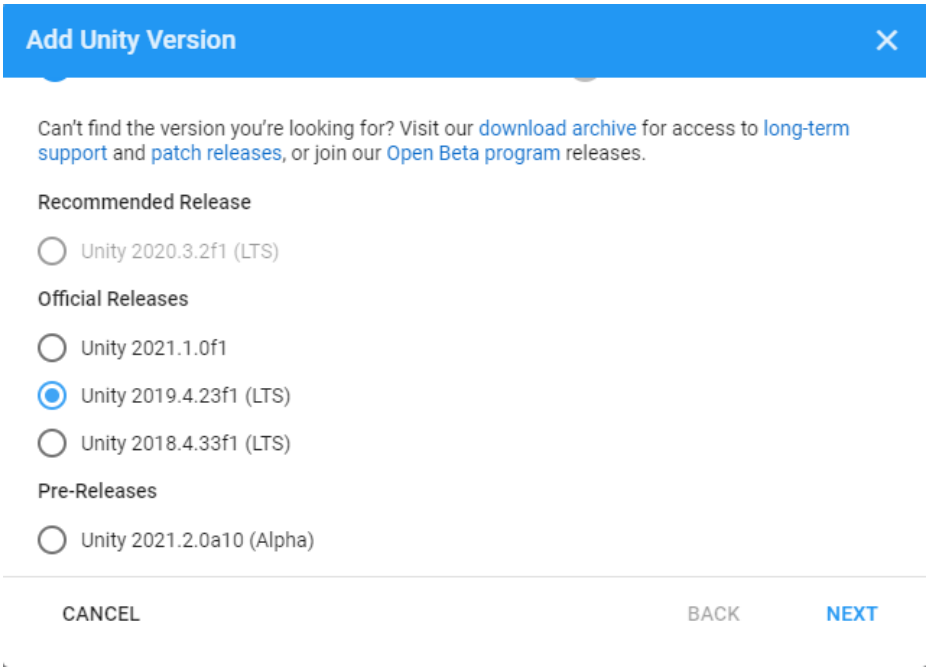
LTS, neboli Long Time Support, je taková verze programu, která nabízí stabilní základnu pro vytváření aplikací s garancí pravidelných updatů (první rok od spuštění se jedná o dvoutýdenní cyklus a rok od spuštění se poté jedná o měsíční cyklus).

Instalace nové verze probíhá v kartě *Installs*, do které se uživatel dostane po stisknutí stejnojmenného tlačítka v levé části Unity Hubu. Přidání nové verze bylo docíleno stisknutím modrého tlačítka *ADD* v pravé horní části karty.



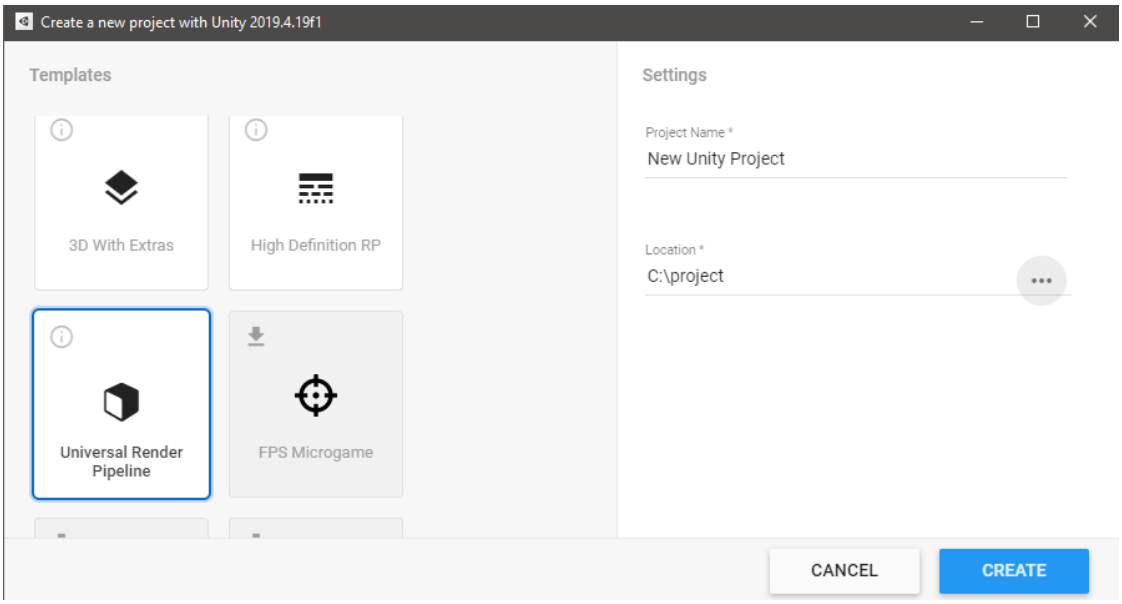

Obr. 12: Unity Hub

Po stisknutí tlačítka je uživatel vyzván k výběru požadované verze, jak již bylo řečeno, pro tento konkrétní program se jedná o verzi 2019.4 (LTS). Po vybrání verze a zmáčknutí tlačítka *Next* se objeví možnost nainstalování doplňkových modulů. Pro tuto konkrétní aplikaci není nutné instalovat žádné doplňkové moduly, ale pro zjednodušení práce byl nainstalován modul *Microsoft Visual Studio Community 2019* a *Documentation*.



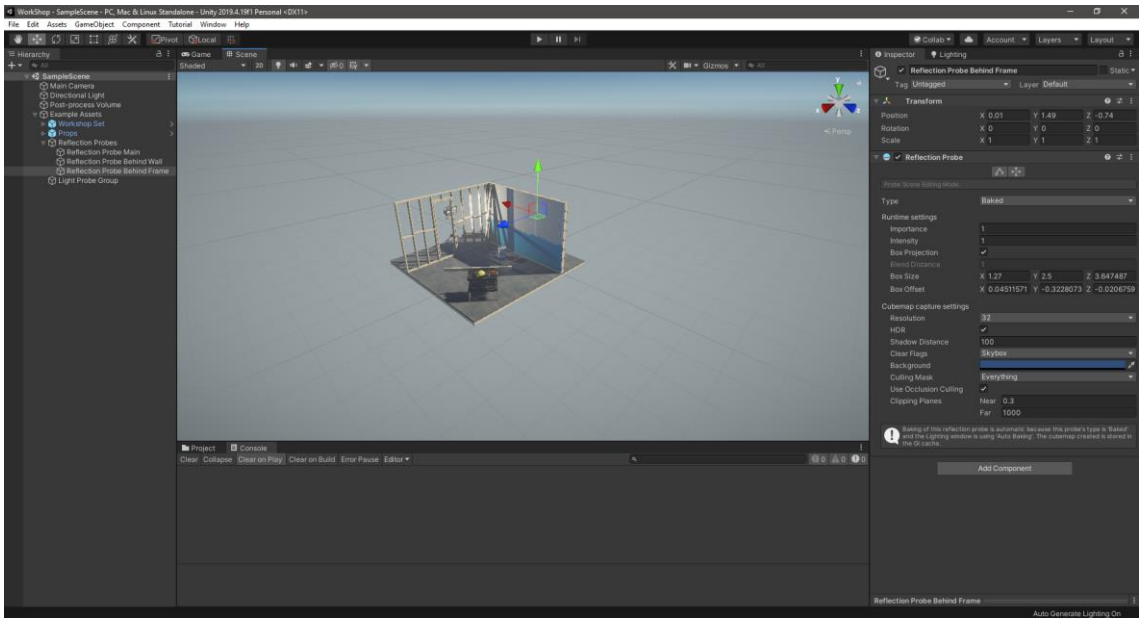

Obr. 13: Instalace nové verze Unity

Vytvoření projektu poté probíhá v záložce *Projects*. Po stisknutí modrého tlačítka *NEW* se objeví možnost výběru požadované šablony. Pro tuto aplikaci byla vybrána *Universal Render Pipeline*, která nabízí vyváženou úroveň grafické kvality a výkonu.




Obr. 14: Založení nového projektu

### 4.1.1 Základní popis UI



Obr. 15: Základní rozložení uživatelského rozhraní

UI, neboli user intefrace se skládá hned z několika důležitých oken, které je možné libovolně přeskupovat podle preferencí uživatele. Okna je možné přeskupovat manuálně, nebo využít předvolby, které se nachází v levém horním rohu v záložce *Window/Layouts*.

V okně hirearchie, které se nachází v levé části (viz obr. 15), jsou zobrazeny všechny objekty vložené do scény. Jestliže má objekt po své levé části šedý trojuhelník, znamená to, že se objekt skládá z více podobjektů, takzvaných child objektů. Ikonka krychle nacházející se vedle názvu objektů může mít dvě podoby. Šedá nevyplněná znamená, že je objekt jedinečný pro danou scénu a není zvlášť uložen jako prefab soubor. Jestliže uživatel tento objekt samostatně uloží nebo importuje objekt z jiného zdroje, ikonka bude mít modrou vyplněnou barvu.

Ve spodní části uživatelského prostředí se poté nachází hned dvě okna mezi kterými se dá volně přepínat. V okně *Console* se vypisují chybové hlášení, popřípadě textový výstup scriptů. Ve druhém okně *Project* se nachází veškeré soubory aktuálně otevřeného projektu.

Pravá část uživatelského prostředí poté zobrazuje vlastnosti. Při vybrání libovolného objektu se zobrazí komponenty, které tento objekt obsahuje. V této části se také nachází vlastnosti projektu jako například nastavení osvětlení, nebo uživatelské předvolby.

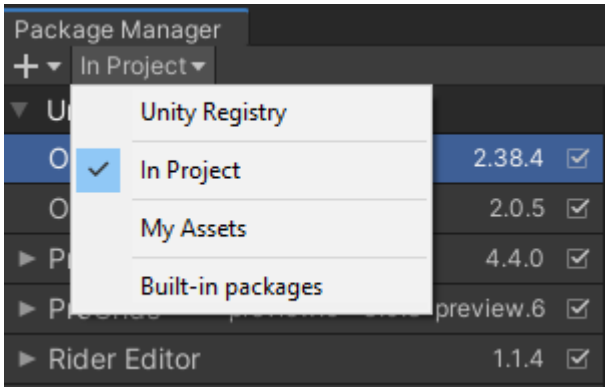
V nejdůležitější, prostřední části se nachází pohled na scénu. Po stisknutí tlačítka pro spuštění se poté prostřední část přepne na zobrazení herního okna.

Jestliže uživatel otevře jiná podpůrná okna v záložce *Window*, okno se otevře jako samostatná záložka, kterou je možné vložit do jakékoliv části UI jako novou kartu.

## 4.2 Potřebné package

Package, neboli balíčky, jsou soubory funkcí, které mají za účel rozšiřovat základní funkcionalitu enginu. Pro určitý typ aplikace, jako například aplikace pro virtuální realitu, je implementace určitých balíčků nezbytností.

Tyto balíčky se dají nelézt v *Package Manageru*, nacházejícím se v kolonce *Window*. Balíčky jsou poté rozděleny do čtyřech kategorií, které lze vidět na následujícím obrázku.




Obr. 16: Výběr kategorie v Package Manager

V kategorii *Unity Registry* jsou obsaženy všechny dostupné balíčky. V základním nastavení se zobrazují pouze ověřené balíčky. K zobrazení balíčků, které jsou v preview nebo beta verzi, je potřeba v záložce *Advanced*, nacházející se v pravé horní části vedle pole pro vyhledávání, vybrat možnost *Show preview packages*.

Kategorie *In Project* poté obsahuje všechny balíčky, které byly importovány do aktuálního projektu.

Kategorie *My Assets* slouží k importování uživatelského obsahu, který je dostupný z *Unity AssetStore* [15]. (viz. Kapitola 4.3).

Poslední kategorie *Built-in packages* obsahuje veškeré balíčky, které jsou obsaženy v základní verzi Unity při prvním spuštění projektu.

### 4.2.1 XR Interaction Toolkit

XR Interaction Toolkit je balíček, obsahující předem vytvořený Framework [16], který umožní lehčí implementaci VR funkcí. Balíček obsahuje veliké množství funkcí, jako například VR camera rig (viz kapitola 4.4), nebo funkce zajišťující interakce mezi uživatelským ovladačem a objekty nacházejícími se ve scéně.

### 4.2.2 Oculus Desktop

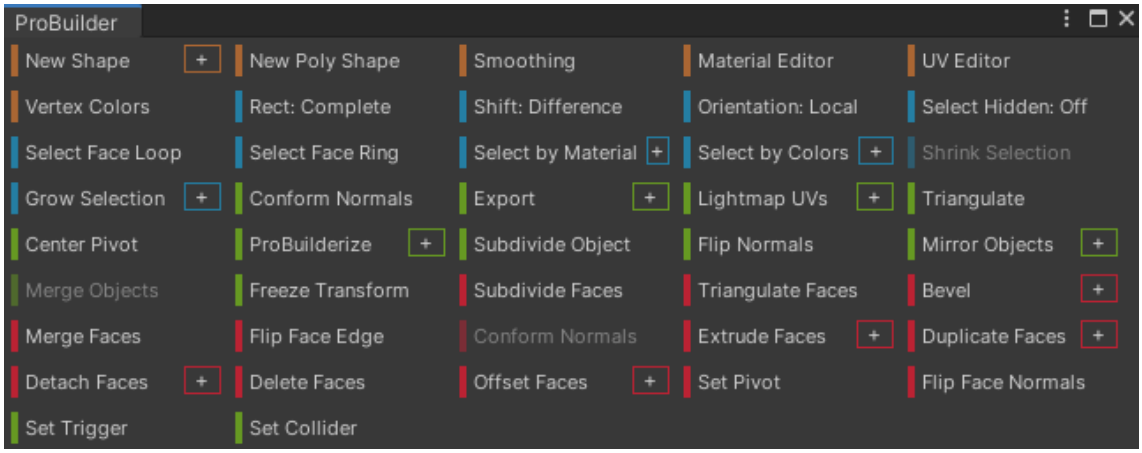
Tento balíček obsahuje nezbytné komponenty potřebné k užívání *Oculus Virtual Reality SDK* ve vyvíjené aplikaci [17]. Bez toho balíčku by nebylo možné aplikaci spustit na zařízení pro virtuální realitu. Tento balíček je specifický pro zařízení firmy Oculus,

jestliže je aplikace vyvíjena pro jiný typ zařízení, je potřeba využít jiný balíček zajišťující stejnou funkci.

### 4.2.3 ProBuilder a ProGrids

ProBuilder a ProGrids jsou balíčky rozšiřující základní možnost vytváření a editace objektů. Základní editor není vhodný pro vytváření složitějších objektů a postrádá velice důležité funkce, a právě ProBuilder řeší tento nedostatek.

Kromě možnosti vytvořit předem dané tvary objektů nabízí nepřeberné množství funkcí sloužící k práci s 3D objekty. Tento balíček se samozřejmě nevyrovná profesionálním programům zaměřujícími se na tuto problematiku, ale pro základní účely jako je vytvoření místností či objektů, které nepotřebují detailní modelování slouží dostatečně.




Obr. 17: Menu rozšiřujícího balíčku ProBuilder

ProGrids je balíček umožňující vytvoření mřížek, které lze poté využít k zarovnání objektů na předem dané pozice. Tento balíček je ideálním doplňkem k balíčku ProBuilder.

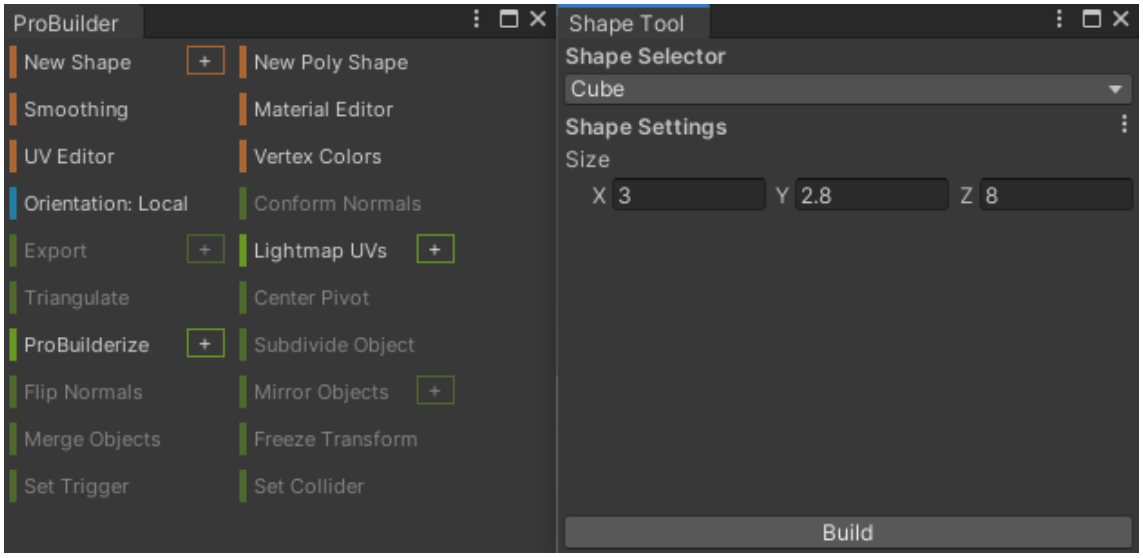
Podrobný popis všech funkcí se nachází v dokumentaci balíčků [18,19].

## 4.3 Vytváření scény

Prvním krokem vytvoření scény je vytvoření základního prostředí, ve kterém se bude uživatel pohybovat. Podoba tohoto základního prostředí závisí převážně na účelu aplikace a může mít nejrůznější podobu. V případě této práce se jedná o uzavřenou místnost, která byla vytvořena pomocí nástroje ProBuilder.

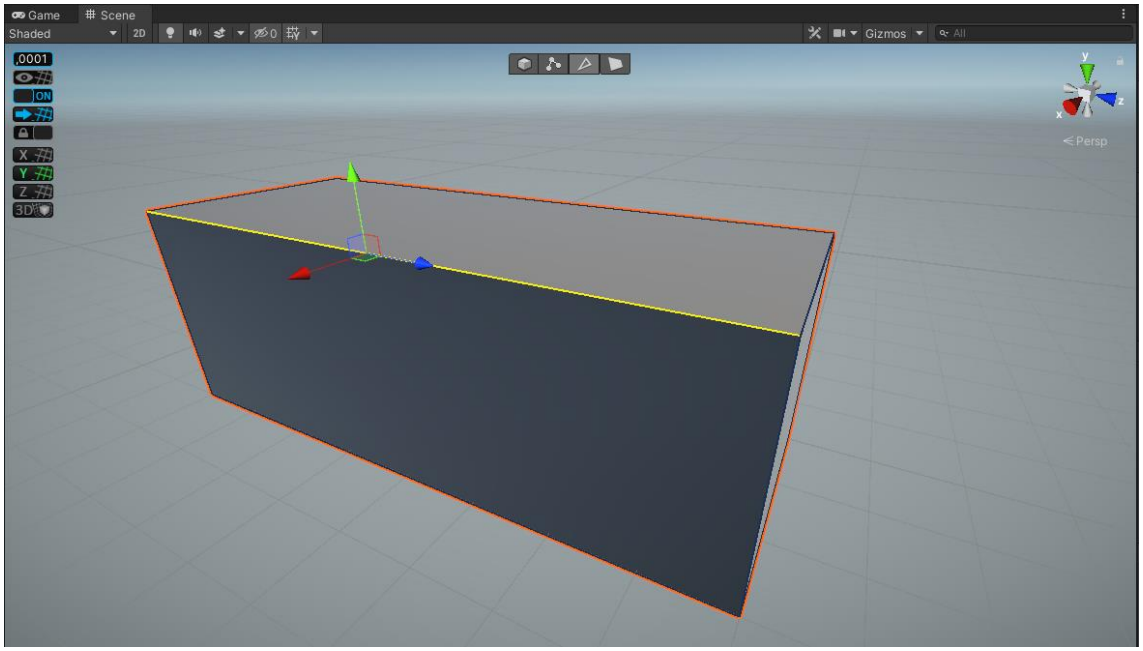
Po stisknutí tlačítka *New Shape* v okně ProBuilderu se nabídne možnost zvolení tvaru objektu a nastavení velikostí. Poměr jednotek je 1:1, jedna jednotka tedy znamená 1 metr. Pro vytvoření prostředí působící na uživatele co nejvíce realisticky je vhodné

použít takové rozměry, které se co nejvíce podobají reálné předloze. V tomto případě byl zvolen tvar kostky o šířce 3 metry, výšce 2,8 metrů a délce 8 metrů.




Obr. 18: Vytvoření nového tvaru v balíčku ProBuilder

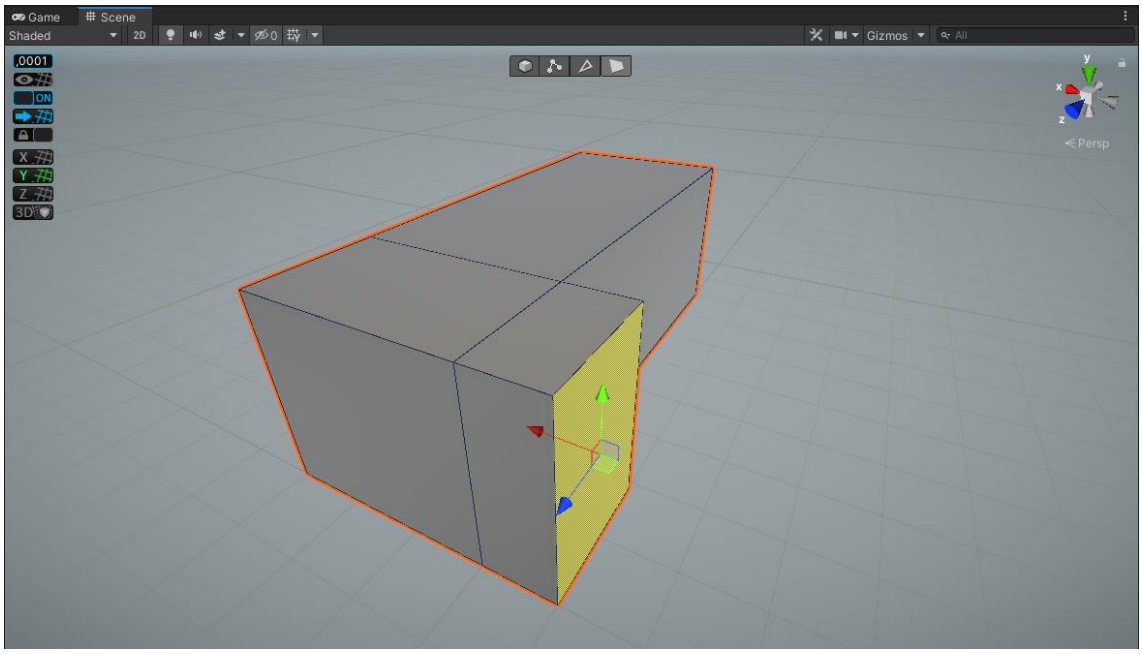
Pro změnu tvaru místnosti, v tomto případě vytvoření výklenku, je prve nutné rozdělení jedné strany na více ploch. Toho je docíleno změnou výběru na výběr hran v horní části okna a zvolení jedné z hran tělesa.




Obr. 19: Výběr hran tělesa

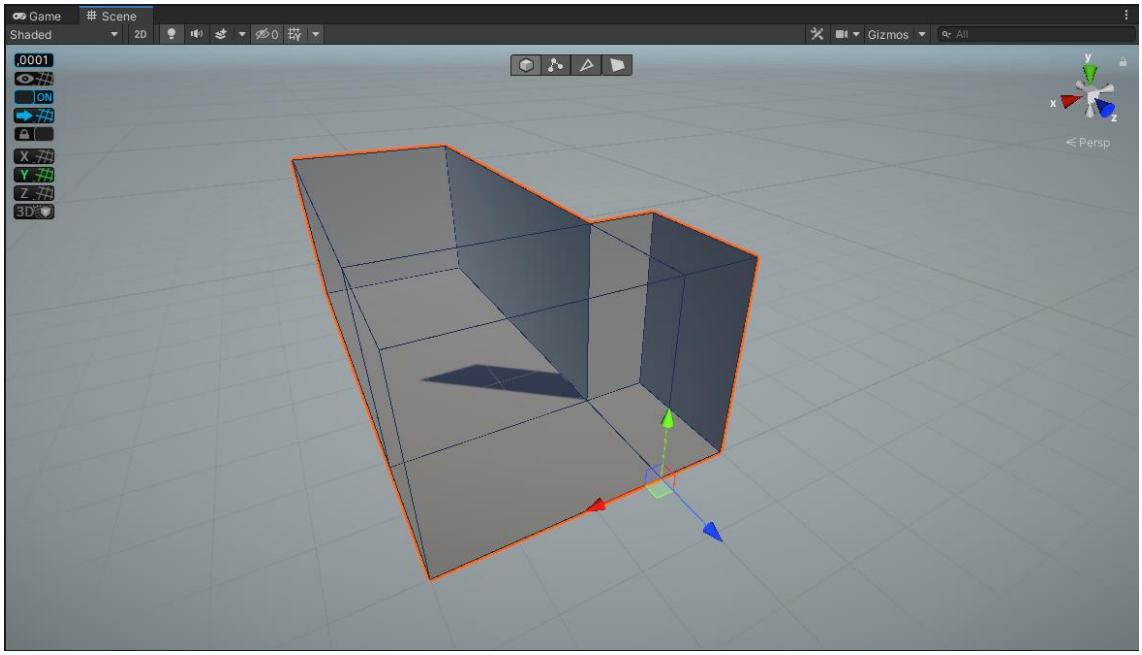
Po stisku tlačítko *Subdivide Edges* v okně ProBuilder se na obvodu celého objektu vytvoří hrana, kterou je možné libovolně přemístit po celém objektu.

Pro vysunutí vytvořené plochy je zapotřebí změnit typ výběru na výběr plochy, vybrat příslušnou plochu a v okně ProBuilder vybrat volbu *Extrude Faces*, která nám umožní vybranou plochu vysunout o určitou hodnotu.




Obr. 20: Vysunutí plochy tělesa

Posledním krokem je obrátit normály tělesa a změnit krychli na místnost. Toho se docílí pomocí funkce *Flip Normals* v okně ProBuilder.




Obr. 21: Ukázka vytvořené místnosti

### 4.3.1 Statické objekty

Jestliže se objekt při chodu programu nehýbe, je nazýván statickým objektem. Označení objektu jako statický má hned několik výhod, mezi které patří hlavně ulehčení výpočtů a s tím související zlepšení výkonu.

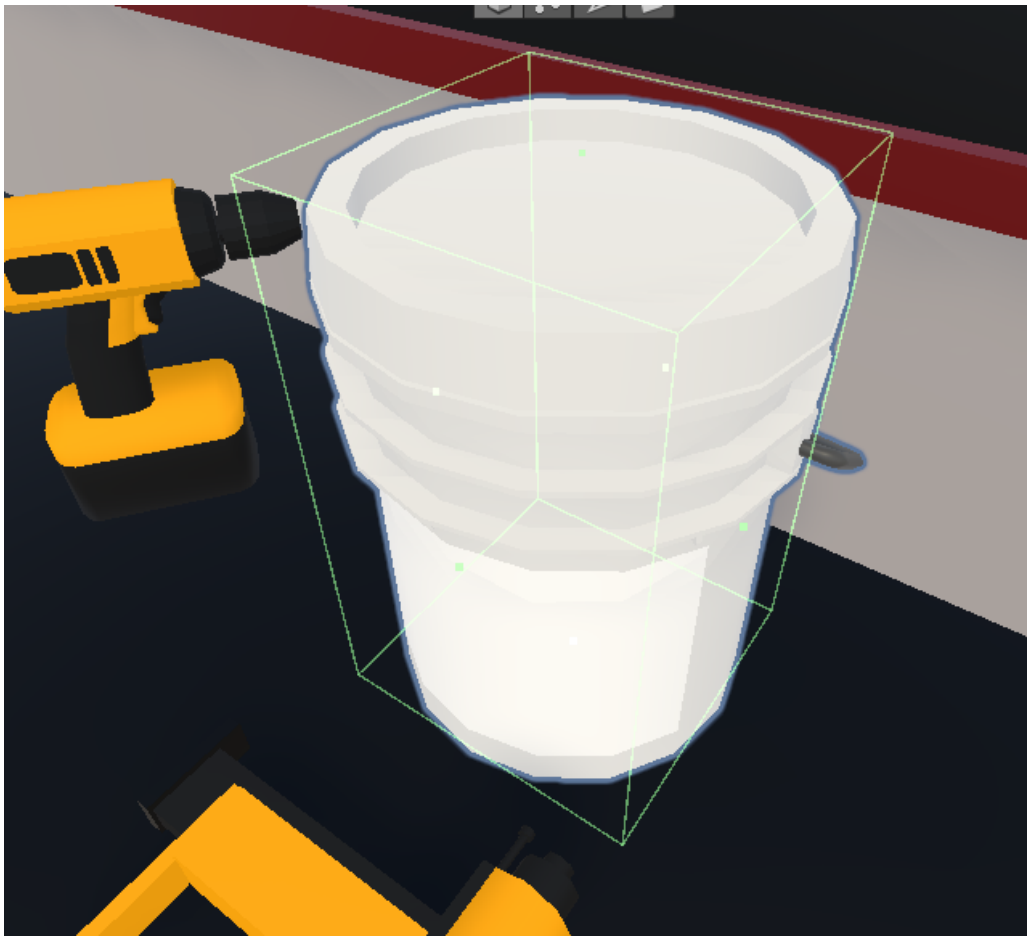
Takový objekt je možno vytvořit přímo v Unity obdobným způsobem jako místnost v minulé kapitole.

Druhou možností je vytvoření objektu v externím programu jako například Blender, Maya nebo jakýkoliv jiný program umožňující export souboru ve formátu .fbx, .dae, .3ds, .dxf, nebo .obj.

Třetí, nejlehčí možností, je importování objektu z dostupných internetových obchodů. Nejpřívětivější možností je využít přímo Unity AssetStore [15], kde po přihlášení na stejný účet jako je používaný v editoru unity stačí pouze u vybraného produktu vybrat volbu uložení a poté ho v *Package manageru* v kategorii *My Assets* vložit do projektu.

U statických objektů není nutné přidávat mnoho komponent. Jediným důležitým komponentem je Collider.

Collider je neviditelný komponent, který definuje tvar objektu za účelem zjišťovaní kolizí s jiným Colliderem. Collider nemusí být stejného tvaru jako vizuální tvar objektu.

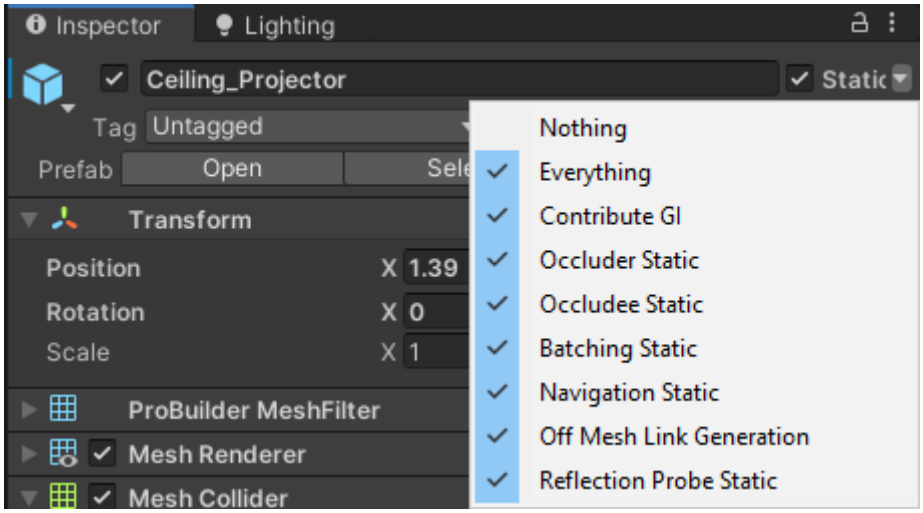


Obr. 22: Zjednodušený Collider objektu

Většina importovaných a vytvořených objektů mají při vytvoření Mesh Collider. Mesh Collider je typ collideru, který kopíruje vizuální tvar objektu. Tento Collider je více přesný, ale složitější na výpočet a při užití většího množství mesh colliderů může mít

negativní dopad na výkon programu. Proto je vhodnější využívat zjednodušených colliderů u objektů, kde není potřeba mít přesný collider.

Po vytvoření takového objektu je vhodné v kartě *Inspector* označit objekt jako statický a vybrat, jaké parametry považovat za statické.

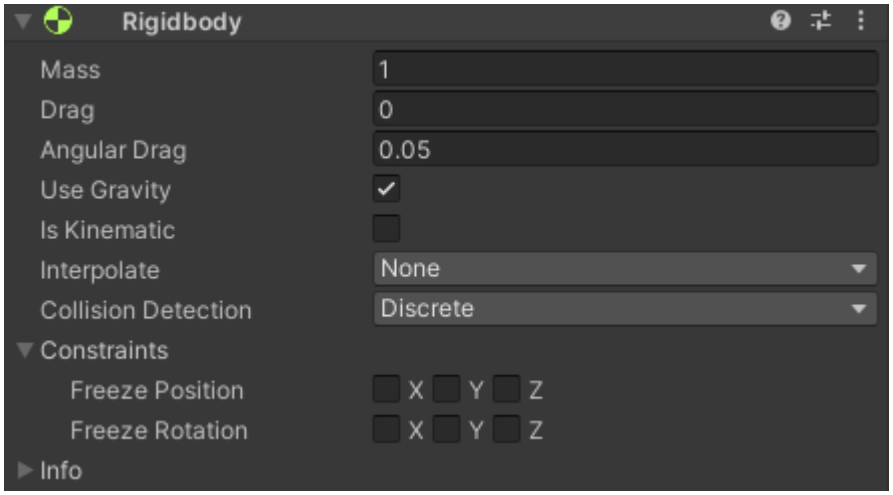



Obr. 23: Ukázka výběru

### 4.3.2 Dynamické objekty

Dynamické objekty jsou takové objekty, které se při chodu programu hýbají. Může se jednat například o nástroje, které uživatel může uchopit či součásti nábytku, které se pro konání své funkce musejí pohybovat, například dveře skříně.

Kromě komponentu Collider dynamický objekt obsahuje také komponent rigidbody. Tento komponent objektu přidá fyzické vlastnosti, které poté objektu umožní reagovat s externími silami. Externí síly poté zapříčiní realistický pohyb tělesa, jako například gravitační působení.




Obr. 24: Komponent Rigibody

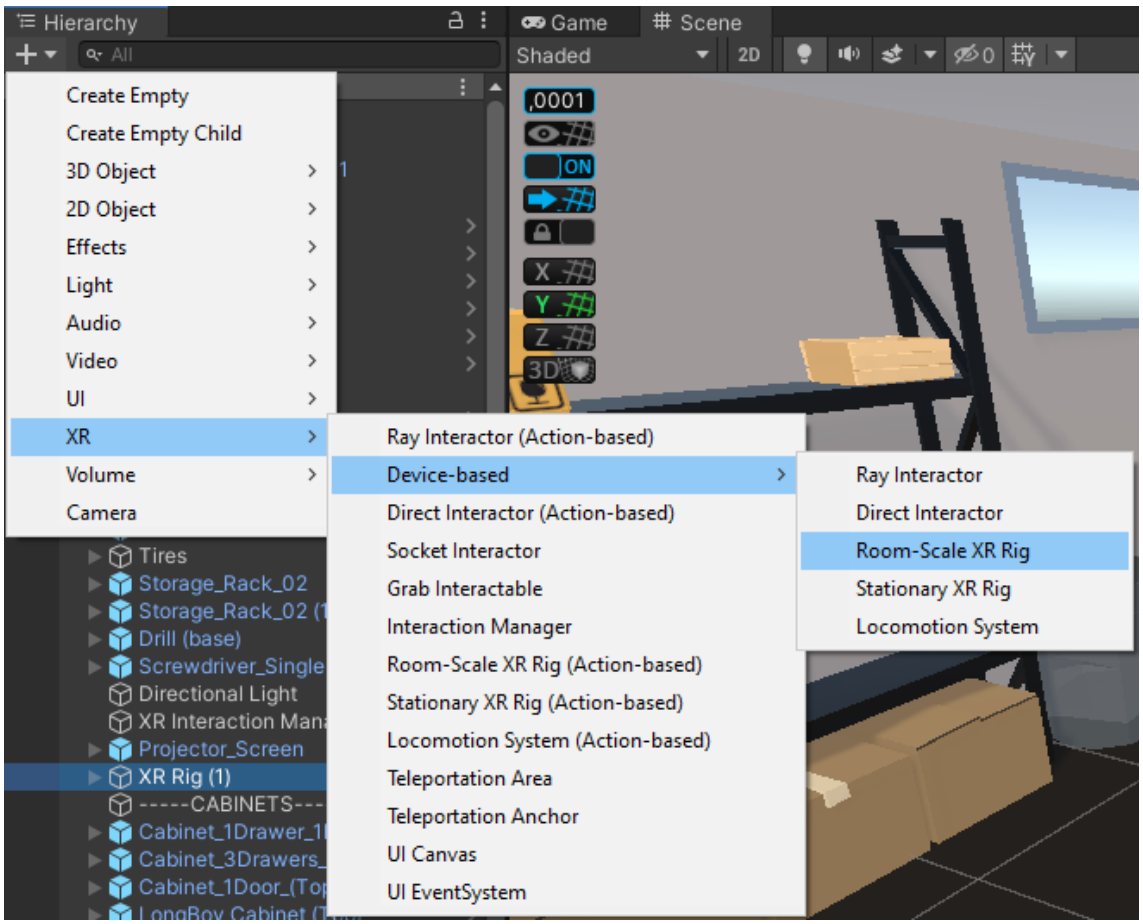
K rozšíření funkcionality tělesa se využívají takzvané skripty, které zajišťují různé chování tělesa při definovaných událostech. Ukázkou takového chování může být

například samotný pohyb tělesa, kdy skript v určité obnovovací frekvenci sleduje, zda nebyla stisknuta určitá klávesa, nebo zda nebyl aktivován jiný vstupní signál. Při splnění této podmínky zapůsobí na těleso určitou silou nebo těleso přesune určitým směrem o určitou vzdálenost. Takový skript se poté přidá jako komponent žádanému tělesu.

Pro vytváření skriptů v unity se využívá programovacího jazyka C# a jako editační program je možné využít libovolný editor. Některé editory, jako například Visual Studio Code, nabízí doplňky ulehčující práci se skripty určené pro Unity [20].

## 4.4 XR Rig

XR Rig je objekt obsažený v XR interaction toolkitu reprezentující uživatele, jehož součástí je hned několik důležitých funkcí. Přidání XR Rig objektu do scény je velice snadné. Po stisknutí pravého tlačítka myši v okně hierarchie se otevře dialogové okno, poté stačí v kategorii *XR* vybrat jednu z nabízených možností, *Room-Scale XR rig*, nebo *Stationary XR rig*.




Obr. 25: Výběr typu XR Rig objektu

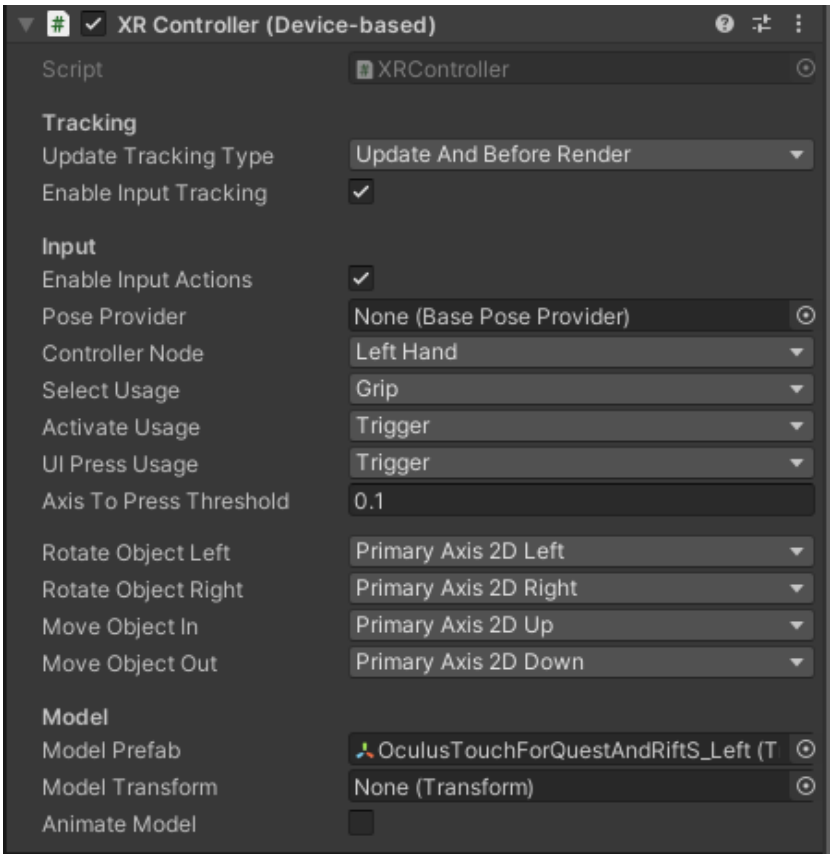
Room-Scale nabízí oproti Stationary možnost volného pohybu ve fyzickém prostoru, který se poté přenáší do prostoru virtuálního. Výběr záleží na několika faktorech, jako například účel aplikace nebo typ zařízení, pro které je aplikace vyvíjena,

jelikož ne všechny zařízení touto funkcí disponují. Room-Scale má oproti Stationary další obrovskou výhodu a tou je možnost automatické nastavení výšky headsetu od podlahy. Stationary má tuto výšku pevně danou. V případě tohoto programu byla zvolena Room-Scale varianta.


Obr. 26: XR Rig

Vložený komponent XR Rig obsahuje několik child objektů, kde každý z nich zajišťuje jisté funkce. Prvním z nich je *Camera Offset*, který udává v jaké výšce od podlahy se HMD nachází, popřípadě se tímto komponentem dají vykompenzovat odchylky umístění HMD. V případě využití Room-Scale XR Rig a nastavení *Tracking Origin Mode* na *Floor*, není nutné tento komponent nastavovat a lze ho ponechat ve své výchozí poloze.

Dalšími objekty nacházejícími se v XR Rig jsou poté ovladače. Jsou to prázdné herní objekty obsahující několik přednastavených scriptů. Nejdůležitější z těchto scriptů je *XR controller*.




Obr. 27: Komponent XR Controller

Tento script zajišťuje správné fungování ovladačů, jedná se hlavně o sledování polohy (trackování) a spravování inputů. Ovladač není v základním nastavení nijak viditelně reprezentován. Pro vizualizaci ovladače je nutné do pole *Model Prefab* vložit prefab soubor reprezentující ovladač.

Nedílnou součástí ovladače jsou poté různé typy Interactor komponentů. Interactor komponent dovoluje ovladači interagovat s ostatními objekty ve scéně. Existuje několik typů interactorů. Pro účely této aplikace byl zvolen *Direct Interactor*, který pro interakci s ostatními objekty potřebuje bezprostřední kolizi mezi Colliderem ovladače a interagovaného objektu.

Samotný objekt XR Rig poté obsahuje komponenty zajišťující pohyb ve scéně. Stejně jako u ovladačů, existuje několik způsobů ovládání. V této aplikaci bylo využito *Snap Turn Provideru*, který slouží pro rotační pohyb kolem vertikální osy uživatele po určitých krocích vyjádřených ve stupních. V praxi to tedy vypadá tak, že po zmáčknutí předem určeného tlačítka, v tomto případě se jedná o horizontální osu pravého joysticku, se objekt XR Rig otočí o 45° v žádaném směru. Pro pohyb uživatele ve scéně je využíváno komponentu *Continuous Move Provider*, který umožní plynulý pohyb uživatele pomocí jednoho z joysticků. Tento typ pohybu ve virtuální realitě je pro některé méně zkušené uživatele nepříjemný a může vyvolat nevolnost, proto je vhodnější využít druhé nabízené možnosti pohybu, systém teleportace. Tento systém ale ubírá na uživatelském prožitku a proto byla zvolena první možnost, kdy optimalizací určitých parametrů, jako například rychlost pohybu, lze snížit riziko způsobení nevolnosti.

## 4.5 XR grab interactable a Offset Grab

Objektu, který lze uchytit, je potřeba přidat komponent XR grab interactable. Při přidání tohoto komponentu je automaticky přidán komponent Rigidbody, který je potřebný pro uchycení objektu. Objekt mimo jiné potřebuje také libovolný typ collideru, který umožní interakci mezi ovladačem a objektem. Vlastnosti komponentu XR grab interactable lze libovolně měnit tak, aby byla zajištěna správná funkčnost. Příkladem může být změna parametru *movement type*, který popisuje jakým způsobem je realizován pohyb uchopené součásti. V základním nastavením je tento parametr nastaven na kinematický pohyb a objekt se může volně pohybovat skrze ostatní objekty. Změnou tohoto parametru na sledování rychlosti (*Velocity Tracking)* se pohybu skrze ostatní objekty zamezí.

Po uchycení objektu se daný objekt přesune na pozici ovladače. Offset, neboli posunutí od tohoto bodu, je možné manuálně změnit přidáním prázdného child objektu, nastaveného na uživatelem určenou pozici s nastavenou rotací. Tento child objekt je poté potřeba umístit do pole *Attach Transform* komponentu XR grab interactable. Takový typ uchycení je vhodné využít u určitých typů objektů, jako je například ruční nástroj. Pro ostatní objekty tento typ uchycení ale není vhodný a bylo by vhodnější, kdyby se poloha a rotace uchyceného objektu zachovala. Tento problém řeší skript Offset Grab, který není součástí XR interaction toolkitu, a je ho potřeba vytvořit. (viz příloha 1)

Skript rozšiřuje a přepisuje kód komponentu XR grab interactable a princip je velice jednoduchý. Při uchopení objektu se vyvolá funkce *OnSelectEntered*, ta pomocí dodatečných funkcí vytvoří nový Vector3 a zapíše do něj aktuální umístění ovladače, který objekt uchopil. Stejně je tomu s rotací, která se zapíše ve formě Quaternionu. Vytvořené proměnné se poté zapíší do pole *Attach Transform* jako tomu bylo u komponentu XR grab interactable. Při upuštění objektu se hodnoty pole *Attach Transform* vynulují a vytvořené proměnné se přepíší na nulovou hodnotu.

Z důvodu rozsáhlosti kódu je kompletní zdrojový kód k dispozici v příloze 1.

## 4.6 Spawner

K tomu, aby uživatel měl neomezené množství určitého objektu, lze využít několika způsobů. Nejjednodušším způsob je ten, že po vykonání nějaké akce, například zmáčknutí tlačítka v UI, se na určitém místě ve scéně objeví konkrétní materiál. Tohoto způsobu bylo využito v blueprint systému popsaného v kapitole 4.10. Dalším řešením, které je unikátní pro aplikaci ve virtuální realitě, je využití komponentu *XR base interactor*.

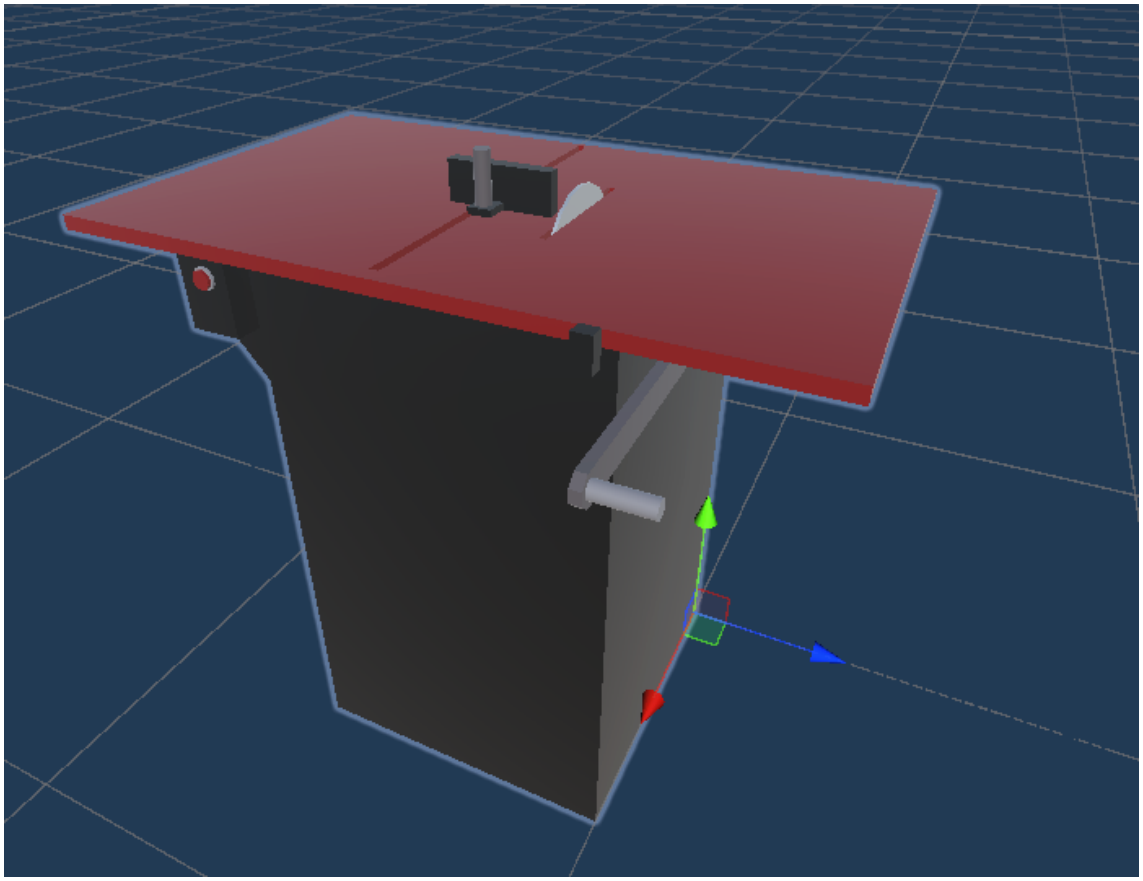
Tento komponent obsažený v XR interaction toolkitu ve své základní podobě nemá žádné reálné využití, ale slouží jako základní šablona pro vytváření uživatelem definovaných funkcí. Princip je podobný jako to mu bylo u komponentu Offset grab v přechozí kapitole 4.5.

```
public class ObjectSpawning : XRBaseInteractable
{
    public GameObject ObjectToSpawn;
    public Transform SpawnPoint;

   protected override void OnSelectEntered(XRBaseInteractor interactor)
   {
       GameObject Object = Instantiate(ObjectToSpawn,SpawnPoint.position,
SpawnPoint.rotation);
       XRBaseInteractable SObject = Object.GetComponent<XRBaseInteractabl
e>();
       interactionManager.ForceSelect(interactor, SObject);
   }
}
```

Jak už bylo řečeno, tento skript vychází z *XR base interactor* komponentu, který obsahuje několik funkcí specifických pro interakci mezi ovladačem a objektem. Funkce, které bylo využito pro tento účel, se nazývá *OnSelectEntered*. Tato funkce se vyvolá v případě, že uživatel při kontaktu s objektem zmáčkne tlačítko *select* na ovladači, neboli se pokusí objekt uchytit. V ten moment se pomocí funkce *Instantiate* vytvoří nový předem určený objekt a poté funkce *ForceSelect* zajistí aby se vytvořený objekt uchytil.

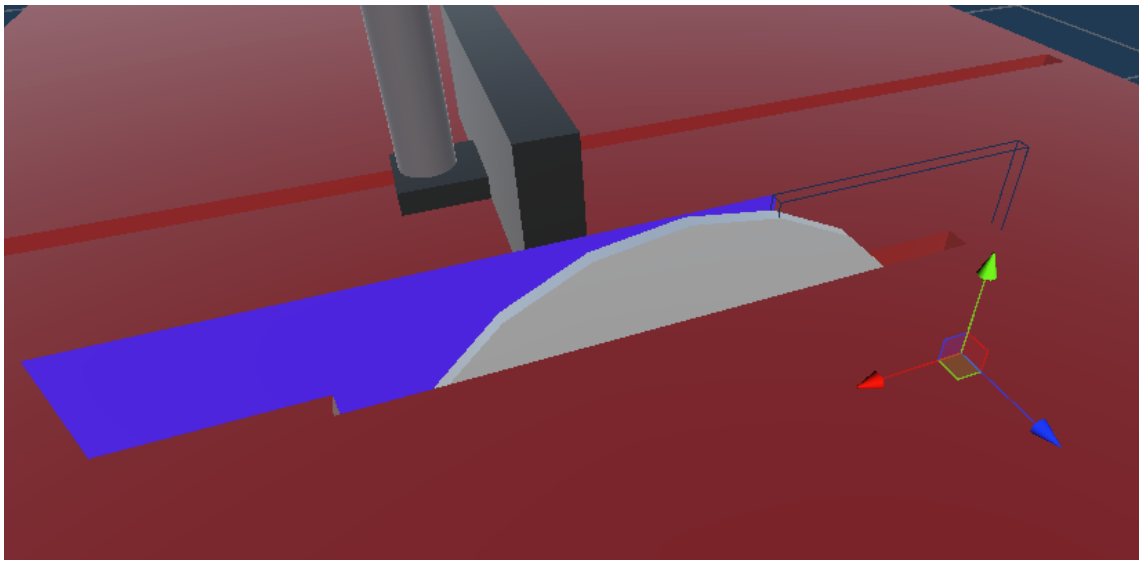
## 4.7 Stolní pila



Obr. 28: Stolní pila

Tento objekt, jehož hlavní funkcí je rozříznout objekt na dva podobjekty, byl vymodelován pomocí nástroje ProBuilder a předlohou mu byla obyčejná, volně dostupná stolní pila. Objekt obsahuje několik funkcí, kde většina z nich slouží jako funkce podpůrné, které mají za účel aby objekt jako celek působil co nejvíce jako reálná stolní pila.

Hlavní funkce, kterou je rozdělení objektu na více objektů, byla realizována pomocí Open Source Frameworku Ezy-Slice [21]. Realizace této funkce je taková, že pila obsahuje dva objekty, které mají vypnutý komponent *meshrenderer* a nejsou viditelné při běhu programu.



Obr. 29: Ezy-Slice aktivátor

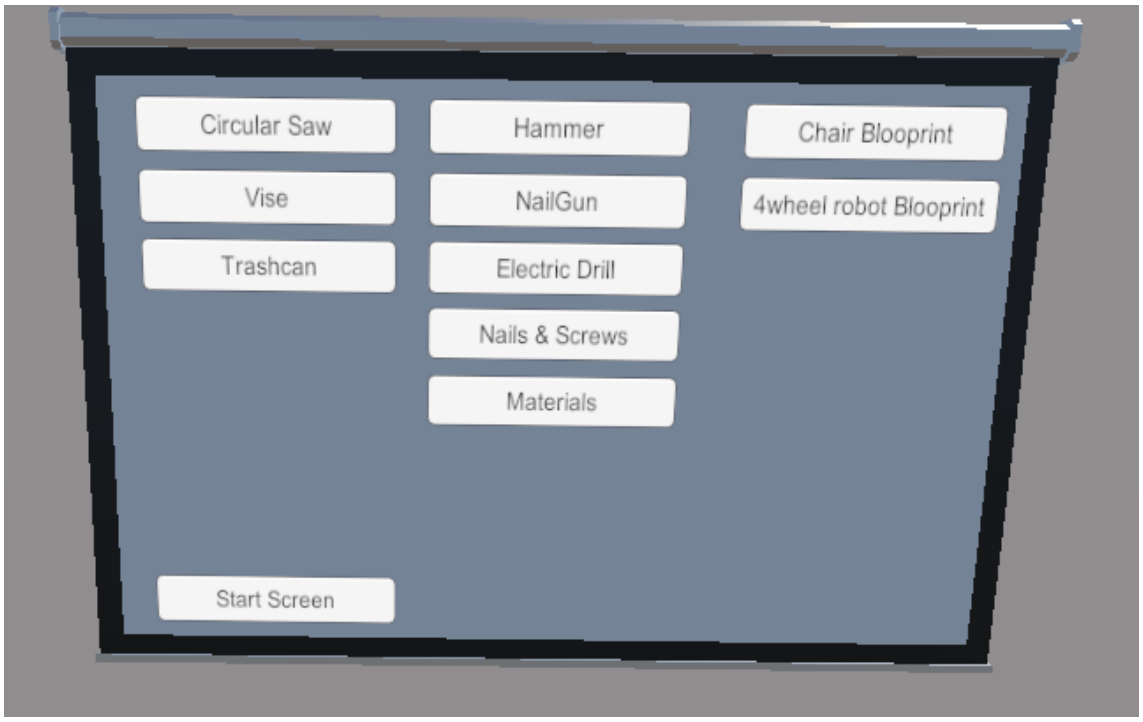
Jeden z těchto objektů slouží jako aktivátor (na obr. 29 vybraný objekt) a pomocí skriptu obsaženém v tomto frameworku při kontaktu s řezaným objektem zaktivuje skript obsažený v druhém objektu, kterým je jednoduchý plane objekt (na obr. 29 objekt fialové barvy). Tento script poté najde všechny validní objekty, se kterými je v kontaktu, a provede rozříznutí. Rozříznutí probíhá tak, že se odstraní hlavní řezaný objekt a vyvolá se funkce *SlicedHull*, která vygeneruje nové objekty s modifikovaným mesh komponentem. Stejná funkce poté vytvořeným objektům přidá další komponenty, které byly obsaženy v původním objektu.

Mezi podpůrné funkce patří spouštěcí tlačítko, které obsahuje komponent *XR base interactor*. Ten při vyvolání funkce *OnSelectEntered* změní materiál tlačítka na svítivý materiál, indikující zapnutí pily a zaktivuje jinak vypnutý aktivační objekt.

Další podpůrnou funkcí pily je páčka nastavující výšku řezného kotouče. Realizace je taková, že samotná páčka obsahuje Offset Grab, komponent umožňující uchopení, a zároveň je její pohyb limitován *Hinge Joint* komponentem, který páčce umožní pohyb pouze kolem předem určené osy. Pohyb páčky v ose Y je poté přenesen na objekt kotouče společně s aktivačním a řezacím objektem.

## 4.8 UI

UI, neboli User Interface, je v případě toho projektu realizováno jako projekce na plátně, které je docíleno umístěním *Canvas* komponentu na plátno.




Obr. 30: UI Canvas

Jedná se o sérii panelů popisujících základní funkcionalitu jednotlivých interaktivních objektů ve scéně, kdy hned první panel seznamuje uživatele se základním ovládáním programu. Ovládání UI je docílené pomocí *Ray Interactoru*, který se spustí po uchopení laserového ukazovátka. Přepínání mezi jednotlivými panely je poté docíleno pomocí tlačítek umístěných na jednotlivých panelech tak, aby z každého panelu byla možnost vrácení se do hlavní nabídky.

UI zároveň slouží kromě popisu jednotlivých objektů ke spuštění blueprintů, které jsou podrobně popsány v kapitole 4.10.

## 4.9 Spojování materiálů pomocí šroubů a hřebíků

Hlavním interaktivním prvkem je volné spojování objektů pomocí spojovacího materiálu za využití podpůrných systémů jako je například stolní pila popsána v kapitole 4.7, dílenský svěrák nebo nastřelovací hřebíkovačka.

Hlavním principem je vytvoření jednoho objektu, obsahujícího ostatní přidané objekty, který se poté chová jako celek při zachování fyzických vlastností a unikátního collideru. Jednotlivé části tohoto systému jsou dále popsány v příslušných kapitolách.

Variace materiálů rozmístěných po scéně mají nejrůznější tvary, díky kterým je možné vytvoření širokého množství výrobků.

### 4.9.1 Hřebíky a šrouby

Implementace reálného chování hřebíků a šroubů ve VR tak, aby jejich používání bylo co nejpřesvědčivější, je velice komplikované a je zapotřebí tuto aktivitu v určitých oblastech ulehčit.

Největším ulehčením bylo samotné přidržování před prvním průnikem do materiálu. V programu to bylo vyřešeno tak, že při kontaktu s materiálem po upuštění se poloha šroubu či hřebíku zamrazí a zároveň se konkrétní spojovací materiál přiřadí k žádanému materiálu jako child objekt. Objekt se poté i se spojovacím materiálem chová jako celek a při uchopení se společně s materiálem hýbe i hřebík či šroub.

Z důvodu rozsáhlosti kódu je kompletní zdrojový kód k dispozici v příloze 2.

### 4.9.2 Vrtačka a Kladivo

Po uchopení šroubku či hřebíku k materiálu se zaktivuje příslušný skript, díky kterému je možné šroub zašroubovat pomocí vrtačky hlouběji do materiálu, popřípadě zatlouct hřebík pomocí kladiva.

Jak šroub, tak hřebík jsou principiálně stejné. Na hlavičce je umístěný collider, který zjišťuje, zda se hlavičky dotýká nástroj. V případě kladiva se po kontaktu s nástrojem hřebík posune o určitou hodnotu směrem hlouběji do materiálu, tento proces lze opakovat dokud se hlavička šroubu nedotkne materiálu, v ten moment se tento skript deaktivuje a není dále možné hřebík zatloukávat. K tomu, aby se hlavička hřebíku nezatloukla hluboko do materiálu, je collider této hlavičky mírně zvětšen ve směru hřebíku.

V případě šroubu je místo kladiva použita vrtačka a pohyb šroubu je plynulý. Hlavička šroubu zjišťuje, zda se ji nedotýká vrtačka. V případě, že se vrtačka šroubu dotýká, skript zjišťuje, zda je na nástroji stisknuto aktivační tlačítko, v případě že ano, šroub se začne pohybovat směrem hlouběji do materiálu. Zastavení šroubu je totožné jako u hřebíku, jestliže se hlavička dotkne materiálu, script se deaktivuje. Jako vizuální pomůcka slouží část scriptu, která při kontaktu s nástrojem změní materiál šroubu tak, aby bylo vidno že je šroub správně vybrán. Při porušení kontaktu s vrtačkou se materiál šroubu změní na materiál původní.

### 4.9.3 Nailgun

Kromě manuálního umístění hřebíku na materiál a následovné zatlučení je možné využít hřebíkovačku která při uchopení a následovném zmáčknutí aktivačního tlačítka vystřelí hřebík. Chování hřebíku je dáno komponentem rigidbody, a tak na hřebík působí gravitační síla. Iniciální pohyb hřebíku je dán přidáním síly ve směru hřebíkovačky.

Jestliže se hřebík dotkne collideru, který je označen příslušným tagem, hřebík se zapíchne a zamrazí se. Aby se zamezilo zamoření scény zbytečnými komponenty, které nemají žádný účel, hřebík se po určité časové prodlevě odstraní.

V případě když hřebík narazí do použitelného materiálu, hřebík zamrzne, posune se o určitou vzdálenost směrem do materiálu a poté se chová stejně, jako hřebík umístěný na materiál ručně. Jelikož se při doteku s materiálem hřebík posune o určitou vzdálenost do materiálu, je automaticky vyvolány skript zajišťující spojení materiálu, který je popsán v kapitole 4.9.4. Díky této mechanice se nástroj hodí na rychlé spojení dvou materiálu.

### 4.9.4 Princip spojování (Parentování)

K vytvoření sestavy tvořené z více objektů bylo využito metody parentování, neboli přiřazení objektu jako child objekt již existujícímu nadřazenému objektu.

Parentování zajišťuje spojovací materiál, který při prvním kontaktu s materiálem zjišťuje, zda je materiál součástí sestavy. V případě že není, příslušný materiál přiřadí jako svůj parent objekt.

Šroub či hřebík poté ignoruje kolizi s již připojeným materiálem a hlídá, zda se nedotkne dalšího materiálu. V případě, že se dotkne, nastane obdobný proces, kdy prvně zjistí, zda je materiál součástí sestavy či nikoliv. Jestliže materiál není součástí sestavy, materiál přiřadí jako svůj child objekt.

V situaci, kdy je materiál součástí sestavy, šroub nastaví sebe i svůj parent objekt jako child objekt příslušnému objektu. Tento proces se může libovolněkrát opakovat, ale pro zachování jednoduchosti byl počet objektů, který lze k sobě jedním šroubem či hřebíkem připojit, nastaven na dva.

K tomu, aby výsledná sestava šla uchopit, byl přidělen objektu umístěného nejvýše v hierarchii komponent OffsetGrab, který aktualizuje obsah platných colliderů při každém přidání nového objektu.

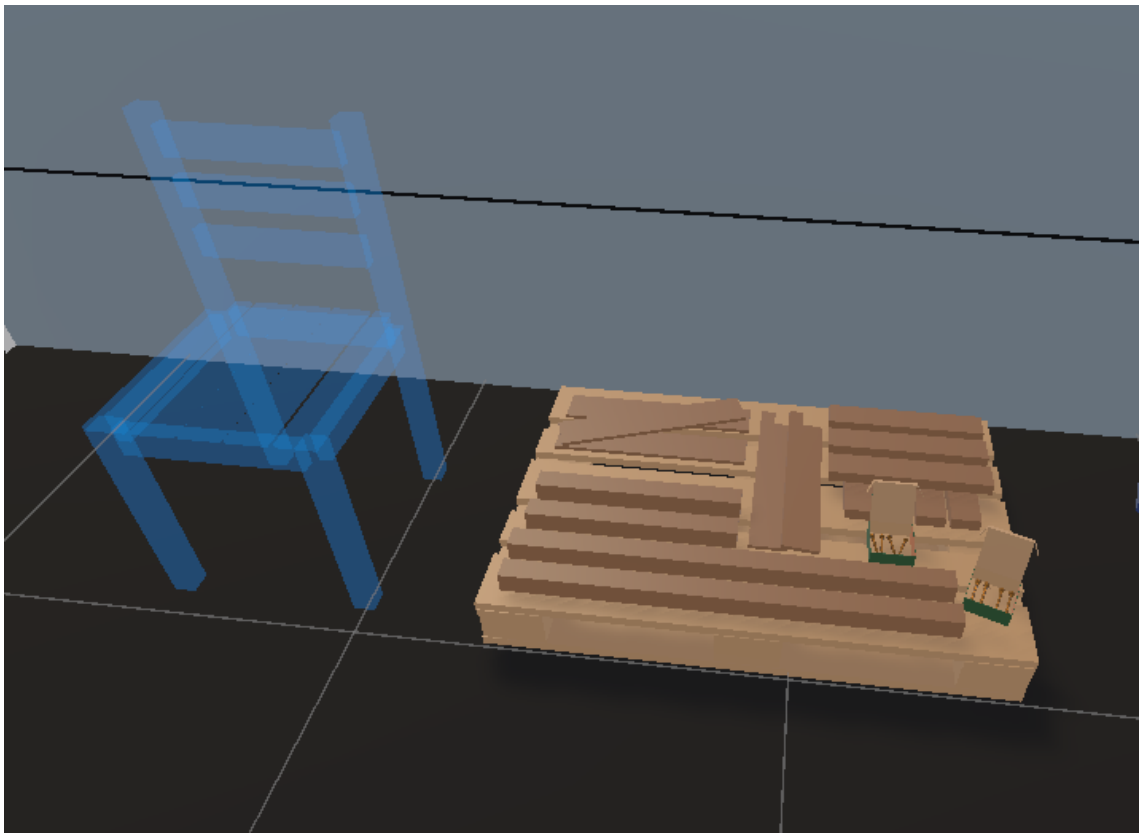
Každý objekt v sestavě má svůj vlastní collider, ale již neobsahuje komponent Rigidbody. Tento komponent je opět obsažen pouze v nejvyšším objektu hierarchie a díky tomu, že příslušné child objekty obsahují collider, se výsledný fyzikální model rovná modelu celé sestavy.

## 4.10 Blueprint systém

Kromě volného vytváření sestav byla do programu přidána možnost vytvoření předem vymodelované sestavy. Konkrétně se jedná o sestavu židle a jednoduchého čtyřkolového robota vytvořeného formou stavebnice.

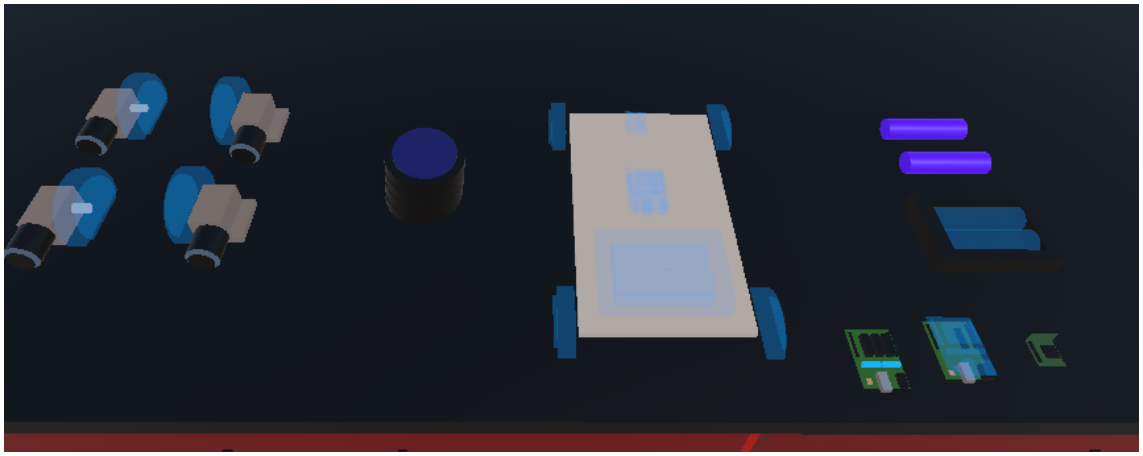
Princip tohoto systému je velice jednoduchý. V případě židle se po zmáčknutí aktivačního tlačítka v prostoru na předem určeném místě objeví sestava obsahující namísto jednotlivých částí objekty, které jsou obohaceny vytvořeným socket komponentem. Ten po vložení správného materiálu materiál přichytí na předem určenou polohu. Tento systém je podrobně popsán v kapitole 4.10.2.

Kromě samotného blueprintu se objeví i paleta obsahující všechny potřebné materiály, které jsou ke kompletaci potřebné.



Obr. 31: Blueprint židle

Jestliže uživatel vybere možnost robota, tak se na stole objeví jeho jednotlivé součásti. Oproti předchozí sestavě se zde ale nenachází objekt složený ze socketů, ale samotné sockety jsou již obsaženy na komponentech. Například základní deska sestavy obsahuje sockety přijímající pouze motory kol. Tyto motory na sobě poté mají socket přijímající pouze kolo, atd.



Obr. 32: Blueprint jednoduchého robota

### 4.10.1 Spawnování objektů pomocí UI

Pro spuštění této aktivity je potřeba v uživatelském prostředí umístěného na plátně projektoru otevřít kartu vybraného blueprintu a zmáčknout tlačítko *Spawn*. Na příslušném místě ve scéně se poté objeví Samotný blueprint a potřebné materiály.

Jednotlivé sestavy jsou v souborech programu uložené jako prefab soubory a jejich vyvolání se poté uskuteční jednoduchým skriptem umístěném na tlačítku UI, pomocí příkazu funkce *Instantiate.*

Po zkompletování sestavy, v případě židle i správného spojení jednotlivých objektů pomocí přiložených šroubů, je třeba zmáčknout tlačítko *Finish* umístěného na UI, kterým se spustí skript. Tento skript každému jednotlivému objektu v sestavě odstraní komponenty rigidbody a offsetgrab. Kromě toho skript také vytvoří prázdný objekt a všechny objekty umístí pod tento nově vytvořený objekt, kterému přidá příslušné komponenty, mezi ně patří Collider, RigidBody a Offsetgrab. Tím se odstraní veškeré pozůstatky blueprint systému a objekt se nyní chová jako každý jiný objekt umístěný ve scéně.

### 4.10.2 Exclusive Name Socket

Tento skript přepisující základní *XR socket interactor* změní interactor tak, aby přijímal pouze objekt s konkrétním názvem. Před vybráním objektu se vyvolá funkce *CanSelect,* která vrací boolean proměnnou pomocí které se rozhodne, zda socket objekt příjme či nikoliv. Do této funkce byla přidána další podmínka zjišťující, zda se název objektu pokoušející se do socketu přichytit rovná žádanému názvu.

Funkce *OnSelectEntered* se vyvolá v moment, kdy se objekt úspěšně přichytí do socketu. V ten moment se změní poloha objektu tak, aby souhlasila s předem určenou

polohou a deaktivuje se komponent *MeshRenderer*, který zajišťuje vizuální podobu objektu.

```
public class NameSocket : XRSocketInteractor
{
    public string targetName;
    private bool CanPickUp;
    public Transform TargetTransform;
    public bool PickedUp = false;

    public override bool CanSelect(XRBaseInteractable interactable)
    {
        if(interactable.name == targetName)
        {
            CanPickUp = true;
        }
        else if(interactable.name != targetName)
        {
            CanPickUp = false;
        }
        return base.CanSelect(interactable) && CanPickUp;
    }
    protected override void OnSelectEntered(XRBaseInteractable interactab
le)
    {
        interactable.transform.position = TargetTransform.position;
        interactable.transform.rotation = TargetTransform.rotation;
        this.GetComponent<MeshRenderer>().enabled = false;
        interactable.gameObject.layer = 1;
        PickedUp = true;
    }
}
```

### 4.10.3 Pohyb jednoduchého robota

Po úspěšném zkompletování jednoduchého robota je možné sestavu umístit na zem dílny a robota aktivovat, ten se poté začne po scéně hýbat.

Pohyb robota je realizován jako pohyb mezi předem určenými body ve scéně. Ve funkci *Awake* se prve naleznou všechny body umístěné ve scéně. Tyto body se poté umístí do pole *spots*. Náhodně se vybere jeden z těchto bodů a změní se rotace tak, aby mířil na robot tento bod. Pomocí funkce *MoveTowards* se poté robot rozpohybuje. Jestliže se vzdálenost mezi robotem a vybraným bodem zmenší na hodnotu menší než 0.5, vybere se nový bod.

```
public class CarMovement : MonoBehaviour
{
    public GameObject[] spots;
    private int random;
    void Awake()
    {
        for(int i=0; i<4; i++)
        {
            spots[i] = GameObject.Find("CarPoint"+i);
        }
        random = Random.Range(0, spots.Length);
    }
    void FixedUpdate()
    {
        Vector3 rotation = new Vector3(spots[random].transform.position.x
, transform.position.y, spots[random].transform.position.z);
        transform.position = Vector3.MoveTowards(transform.position, rota
tion, 0.5f * Time.deltaTime);
        Vector3 direction = (spots[random].transform.position - transform
.position);
        direction.y = transform.position.y;

        transform.rotation = Quaternion.LookRotation(direction);

        if(Vector3.Distance(transform.position, spots[random].transform.p
osition)<0.5f)
        random = Random.Range(0, spots.Length);

    }
}
```

# 5 ZÁVĚR

V teoretické části byla stručně shrnuta nejpopulárnější dostupná zařízení a vývojové enginy používané k vývoji aplikací pro virtuální realitu.

V praktické části byl vytvořen model dílny s aktivními a interaktivními prvky ve vývojovém prostředí Unity. Aplikace byla vyvíjena na široké množství dostupných zařízení, ale primárním cílem bylo zařízení Oculus Rift S, na kterém byla aplikace periodicky testována.

Vytvořená aplikace reprezentuje co nejpřesněji reálnou dílnu a k tomu směřují i dostupné aktivity, mezi které patří převážně úprava dostupného materiálu a následné spojování více materiálů k sobě pomocí spojovacího materiálu. Jako doplňková aktivita k aktivitě předchozí slouží takzvaný „blueprint“ systém, který uživatele přímo vede k vytvoření předem určené součásti. Celá aplikace byla vyvíjena tak, aby byla snadně rozšiřitelná o nové systémy navazující na stávající, základní.

Přístup, který byl zvolen k vytvoření této aplikace nebyl nejpraktičtější. Místo programování jednotlivých systémů od počátku by bylo z časového hlediska vhodnější využít již existující, zpravidla placené, systémy dostupné z Unity Asset Storu a tím ušetřit desítky hodin práce. Vytvoření vlastních systémů ale autorovi přineslo porozumění a kontrolu nad použitými systémy, které jsou velice užitečné při rozšiřování aplikace.

Pro autora byla práce přínosem, jelikož mu přinesla zkušenosti v oboru vývoje aplikací a programování, ve kterém měl minimální zkušenosti. Díky tomuto programu našel profesní zaměření, kterému by se chtěl i po studiu věnovat.

# 6 SEZNAM POUŽITÉ LITERATURY


[1] Degrees of freedom [online]. Google, 2018 [cit. 2021-5-6]. Dostupné z: https://developers.google.com/vr/discover/degrees-of-freedom

[2] ISHII, Hirotake. Augmented Reality: Fundamentals and Nuclear Related Applications [online]. In: . Prosinec 2010, s. 13 [cit. 2021-5-6]. Dostupné z: https://www.researchgate.net/publication/241686793_Augmented_Reality_Fundamentals_and_Nuclear_Related_Applications

[3] Vive: Products [online]. HTC [cit. 2021-5-12]. Dostupné z: https://www.vive.com/us/product/

[4] Index [online]. Valvesoftware [cit. 2021-5-12]. Dostupné z: https://www.valvesoftware.com/cs/index

[5] Oculus: Porovnání [online]. Oculus [cit. 2021-5-12]. Dostupné z: https://www.oculus.com/compare/

[6] VIVE Tracker (3.0) [online]. HTC Corporation [cit. 2021-5-12]. Dostupné z: https://www.vive.com/us/accessory/tracker3/

[7] Tesla suit: the suit [online]. VR Electronics [cit. 2021-5-12]. Dostupné z: https://teslasuit.io/the-suit/

[8] Hapitc gloves for virtual reality and robotics: Haptx gloves [online]. Haptx [cit. 2021-5-12]. Dostupné z: https://haptx.com/

[9] TRENHOLME, D., SMITH, S.P.: Computer game engines for developing first-person virtual environments, Virtual Reality (2008) 12: 181.

[10] Unity XR platform has a new architecture – Unity Copenhagen. In Youtube [online]. 23.10.2019 [cit. 2021-5-12]. Dostupné z: https://www.youtube.com/watch?v=Stqk1GxlSK0. Kanál uživatele Unity.

[11] WANG, S., MAO, Z., ZENG, C., GONG, H., LI, S. and CHEN, B.: A new method of virtual reality based on Unity3D, 2010 18th International Conference on Geoinformatics, Beijing, 2010, pp. 1-5.

[12] LINOWES, Jonathan. Unity 2020 Virtual Reality Projects: Learn VR development by building immersive applications and games with Unity 2019.4 and later versions. 3rd ed. Packt Publishing, 2020. ISBN 1839217332.

[13] Compare Unity plans: Pro vs Plus vs Free [online]. [cit. 2021-5-13]. Dostupné z: https://store.unity.com/compare-plans

[14] XR Develepment | Unreal Engine Documentation [online]. Epic Games [cit. 2021-5-12]. Dostupné z: https://docs.unrealengine.com/en-US/SharingAndReleasing/XRDevelopment/index.html

[15] Unity Asset Store: The Best Assets for Game Making [online]. [cit. 2021-5-13]. Dostupné z: https://assetstore.unity.com/

[16] XR Interaction Toolkit: manual [online]. Unity Technologies [cit. 2021-5-13]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@1.0/manual/index.html

[17] About Oculus: Oculus Desktop [online]. Unity Technologies [cit. 2021-5-13]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.xr.oculus.standalone@2.38/manual/index.html

[18] ProBuilder [online]. Unity Technologies [cit. 2021-5-13]. Dostupné z: https://unity3d.com/unity/features/worldbuilding/probuilder

[19] About ProGrids: ProGrids [online]. Unity Technologies [cit. 2021-5-13]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.progrids@3.0/manual/index.html

[20] Visual Studio Code and Unity [online]. Microsoft, 2017 [cit. 2021-5-13]. Dostupné z: https://code.visualstudio.com/docs/other/unity

[21] Ezy-slice: An open source mesh slicer framework for Unity3D Game Engine [online]. David Arayan, 2018 [cit. 2021-5-13]. Dostupné z: https://github.com/DavidArayan/ezy-slice

# 7 SEZNAM ZKRATEK

| | |
|---|---|
| VR | (virtual reality) – virtuální realita |
| HDM | (head mounted device) - zařízení nošené na hlavě |
| XR | (extended reality) - rozšířená realita |
| UI | (user interface) - uživatelské rozhraní |
| SDK | (software development kit) - sada vývojových nástrojů |

# 8 SEZNAM PŘÍLOH

1) Zdrojový kód funkce OffsetGrab
2) Zdrojový kód funkce šroubu
3) První ukázkový snímek obrazovky z programu
4) Druhý ukázkový snímek obrazovky z programu
5) Třetí ukázkový snímek obrazovky z programu
6) QR kód odkazující na demonstrační video

# PŘÍLOHY

1) Zdrojový kód funkce OffsetGrab

```
public class OffsetGrab : XRGrabInteractable
{
    private Vector3 interactorPositon;
    private Quaternion interactorRotation;

    protected override void OnSelectEntered(XRBaseInteractor interactor)
    {
        base.OnSelectEntered(interactor);
        StoreInteractor(interactor);
        MatchAttachmentPoints(interactor);
    }
    private void StoreInteractor(XRBaseInteractor interactor)
    {
        interactorPositon = interactor.attachTransform.localPosition;
        interactorRotation = interactor.attachTransform.localRotation;
    }
    private void MatchAttachmentPoints(XRBaseInteractor interactor)
    {
        bool hasAttach = attachTransform != null;
        interactor.attachTransform.position = hasAttach ? attachTransform
.position : transform.position;
        interactor.attachTransform.rotation = hasAttach ? attachTransform
.rotation : transform.rotation;

    }
    protected override void OnSelectExited(XRBaseInteractor interactor)
    {
        base.OnSelectExited(interactor);
        ResetAttachmentPoints(interactor);
        ClearInteractor(interactor);
    }
    private void ResetAttachmentPoints(XRBaseInteractor interactor)
    {
        interactor.attachTransform.localPosition = interactorPositon;
        interactor.attachTransform.localRotation = interactorRotation;
    }
    private void ClearInteractor(XRBaseInteractor interactor)
    {
        interactorPositon = Vector3.zero;
        interactorRotation = Quaternion.identity;
    }
}
```

2) Zdrojový kód funkce šroubu

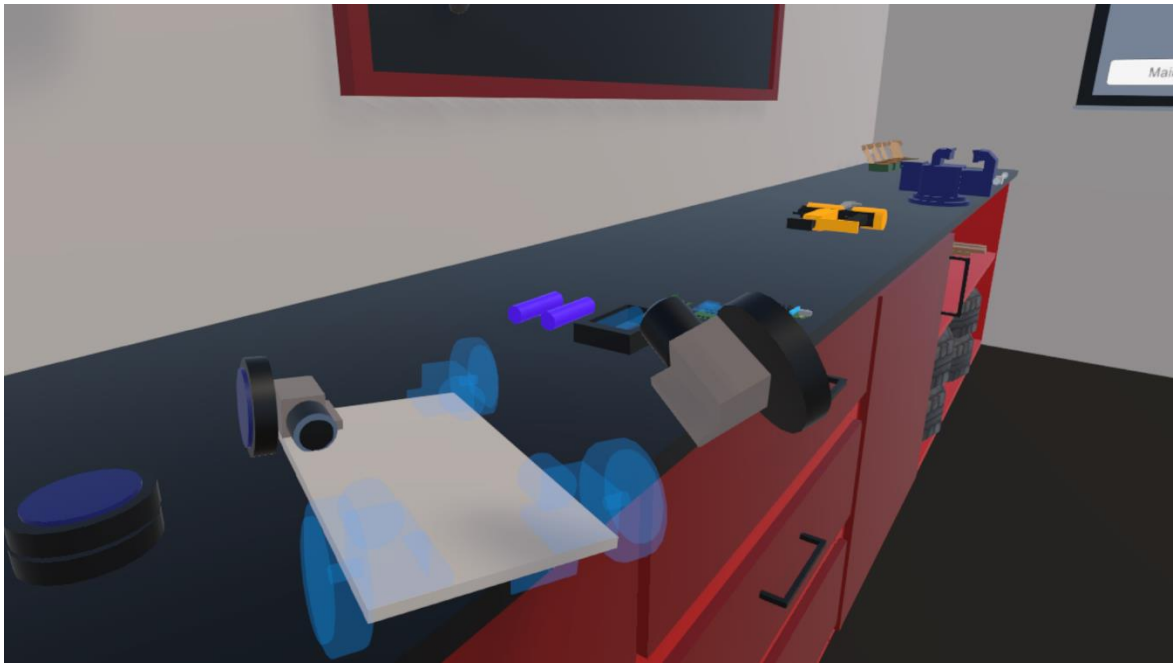
```
public class Screw : MonoBehaviour
{
    public Material originalMaterial;
    public Material MaterialOnTouch;
    private bool istouched = false;
    private Transform Screwdriver;
    public float ScrewSpeed;
    private float xAngle, yAngle = 0;
    public GameObject screwhead;

   void OnCollisionEnter(Collision coll)
   {
       if(coll.collider.tag == "Bit")
       {
          gameObject.GetComponent<Renderer>().material = MaterialOnTouch;
          istouched = true;
          Screwdriver = coll.collider.transform.root;
       }
   }
   void OnCollisionExit(Collision coll)
   {
       if(coll.collider.tag == "Bit")
       {
         gameObject.GetComponent<Renderer>().material = originalMaterial;
         istouched = false;
         Screwdriver = null;
       }
   }
  void OnCollisionStay(Collision coll)
  {
      if(coll.collider.tag == "Bit")
      {
        if(istouched == true)
        {
              if(coll.collider.GetComponent<Rotate>().enabled == true)
              {
                  screwhead.GetComponent<BoxCollider>().enabled = true;
                  this.transform.Rotate(xAngle,yAngle,ScrewSpeed);
                  this.transform.Translate(0,0,(-
Time.deltaTime * 0.1f), Space.Self);
              }
        }
      }
  }
}
```

3) První ukázkový snímek obrazovky z programu
   - na snímku je zobrazena pravá strana dílny, kde se nachází plátno s UI, stolní pila, vozík a příslušné materiály sloužící k práci na stolní pile.


4) Druhý ukázkový snímek obrazovky z programu
   - na snímku je prakticky zobrazena možnost interakce se zásuvkami a dvířky stolu

5) Třetí ukázkový snímek obrazovky z programu
   - snímek pořízený při kompletaci jednoduchého robota



6) QR kód odkazující na demonstrační video