FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

# Zadání bakalářské práce

| | |
|---|---|
| **Název:** | Imerzivní herní a grafická aplikace s bojovými virtuálními objekty |
| **Student:** | Šimon Jajko |
| **Vedoucí:** | doc. Ing. Mgr. Petr Klán, CSc. |
| **Studijní program:** | Informatika |
| **Obor / specializace:** | Webové a softwarové inženýrství, zaměření Počítačová grafika |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | do konce zimního semestru 2022/2023 |

## Pokyny pro vypracování

Prozkoumejte, analyzujte a graficky realizujte sadu bojových virtuálních 3D objektů. Zaměřte se na bojové objekty větší délky jako jsou meče, kordy, šavle, rapíry apod. Naprogramujte prototypový herní virtuální svět, ve kterém budou bojové objekty předvedeny a kde s nimi bude možné experimentovat a trénovat formou hry pro jednoho nebo dva hráče. Postupujte podle následujících bodů:

1. Proveďte analýzu tvarů bojových 3D objektů s větší délkou.
2. Vybrané objekty graficky navrhněte a realizujte jako herní 3D modely.
3. Seznamte se s prostředím a používáním Unity 2020.
4. Navrhněte koncept herního virtuálního prostředí.
5. Naprogramujte scény virtuálního prostředí obsahující hlavní bojové objekty.
6. Realizujte několik obvyklých typů her s těmito bojovými objekty pro jednoho nebo dva hráče.
7. Testujte a vyhodnoťte jednotlivé hry a určete optimální strategie.
8. Stanovte přínos používání navržených virtuálních her a grafiky bojových objektů.

Bakalářská práce

# IMERZIVNÍ HERNÍ A GRAFICKÁ APLIKACE S BOJOVÝMI VIRTUÁLNÍMI OBJEKTY

**Šimon Jajko**

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: doc. Ing. Mgr. Petr Klán
5. ledna 2022

# Obsah

# Seznam obrázků



# Seznam tabulek

*Chtěl bych poděkovat především doc. Ing. Mgr. Petru Klánovi za vedení a konzultace po dobu práce. Také chci poděkovat rodině a přátelům za podporu a pomoc při testování.*

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 5. ledna 2022 .....

# Abstrakt

Tato bakalářská práce se zabývá tvorbou 3D aplikace pro virtuální realitu a historických objektů, které budou v jejím světě použity pro komplexní fyzikální interakce a herní prvky. Cílem je tento proces zdokumentovat a vyhodnotit jeho použitelnost a přínosnost. Práce představuje skoro kompletní proces tvorby imerzivní herní aplikace a jejích částí. To je analýza, návrh, tvorba assetů a implementace. Výsledkem je uživatelsky přívětivá aplikace jejíž hlavní přínos je v podobě zábavní a naučné aplikace. Dále je výsledkem prototypová aplikace se stejnými fyzikálními mechanikami pro více hráčů, která dokazuje nevhodnost současných volně dostupných řešení síťové synchornizace fyzikálně ovládaných objektů a mechanik ve virtuální realitě ve světě pro více uživatelů.



**Klíčová slova** interaktivní aplikace, virtuální realita, fyzikální interakce, historické objekty ve VR, Unity 2020



# Abstract

The goal of this bachelor thesis is to create a 3D application for virtual reality and historical items that will be used in the virtual environment for complex physics interactions and game elements. The main goal is to document the creation process and evaluate its usability. This thesis goes through almost the entire process of game creation. That is analysis, design, asset creation, implementation and testing. The result is a user-friendly application that is best used as an educational or game application. Another result is a prototype application with similar physics interactions that works in an environment for multiple users. It experiments with and proves that current-day available synchronization solutions aren't up for the task of a networked physics synchronization for items driven by multiple users.



**Keywords** interactive application, virtual reality, physics interactions, historical items in VR, Unity 2020

# Úvod

Virtuální realita se nedávno začala díky cenové dostupnosti prosazovat na komerčním trhu a má tak před sebou zajímavou budoucnost. Umožňuje člověku vizuálně proniknout do jiného světa a v omezené míře s ním i interagovat. Většina ovladačů má však omezený počet ovládacích prvků a jen velmi jednoduchou formu zpětné vazby. To dost omezuje možnosti mechanik, které mohou výukové pomůcky, hry a zážitkové aplikace používat. Otázkou tedy je: Jak tyto omezené možnosti ideálně využít tak, abychom dostali co nejpříjemnější a nejvěrohodnější výsledek?

Výsledek práce bude užitečný pro některé jakožto ukázka různých součástí, které je nutné při tvorbě aplikace pro virtuální realitu v systému Unity zpracovat. Samotná výsledná interaktivní aplikace a její historické prvky může být zajímavá ukázka možných využití virtuální reality.

Virtuální realita, grafická tvorba, tvorba interaktivních aplikací a historický šerm jsou všechno mými zájmy. To je důvod, proč jsem se rozhodl pro volbu tohoto, pro mě atraktivního, tématu.

Výsledkem této práce bude program pro virtuální realitu s imerzivní ukázkovou scénou tréninkové oblasti. Umožní uživateli volný pohyb, interakce a uchopení objektů historických zbraní a vybavení. Následně bude moct tyto předměty použít a vyzkoušet na prostředí a herním prvku v podobě tréninkových panáků a speciálních arén, ve kterých se simulují izolované prvky boje za pomocí trénovacích automatů.

Práce je rozdělena na několik kapitol. V první kapitole je obecný popis virtuální reality a jejích vlastností, dále imitace fyziky ve virtuálním světě a šerm jakožto speciální případ fyzikálních interakcích. V druhé kapitole se představí použitý herní engine Unity, jeho součásti a další použité technologie. Ve třetí kapitole se prochází návrhem a tvorbou vizuálních částí projektu a samotným procesem tvorby 3D objektů pomocí externích programů. V poslední kapitole jsou popsány kompletní postup tvorby výsledné prototypové aplikace, její jednotlivé části a všechna rozhodnutí a kompromisy, které byly v průběhu podstoupeny. Nakonec je zpracování a vyhodnocení herních prvků aplikace.

# Kapitola 1

# Cíl práce

Tato práce si klade čtyři hlavní cíle. Prvním cílem je prozkoumat, navrhnout a graficky zpracovat historické bojové vybavení.

Druhým cílem je navrhnout aplikaci pro virtuální realitu v systému Unity. V návrhu budou popsány principy mechanik, grafický návrh, ovládání a herní prvky.

Třetím cílem je tuto aplikaci poté implementovat. Dílčím cílem třetího cíle je zkusit implementovat tyto mechaniky do světa pro více hráčů pomocí existujícího síťového synchronizačního systému Photon PUN.

Posledním cílem je vyhodnotit použitelnost a přínos celé aplikace a jejích konceptů a najít vhodné potencionální budoucí rozšíření.

Tato práce se nezbývá návrhem ani implementací zvukých a hudebních efektů.

Kapitola 2

# Virtuální realita, fyzika a historický šerm

V této kapitole je proveden popis, analýza a porovnání zařízení pro virtuální realitu. Dále je provedena analýza existujících řešení a dalších dílčích částí.
▶ Poznámka 2.1. V rámci této práce jsem při analýze, návrhu a implementaci používal zařízení HP Reverb G2. Všechny mé poznatky ohledně následujících VR zařízeních jsou odvozeny ze zkušeností s použitím tohoto zařízení. V případech obrázků, kde se to hodí, používám jako referenci obrázky s tímto přístrojem.

## 2.1 Virtuální realita

Název je tvořen dvěma slovy, které jsou dohromady tak trochu paradoxem. Pojem 'virtuální' označuje něco co materiálně neexistuje, ale jakoby to bylo opravdové/skutečné, tedy jakási iluze. 'Realita' znamená skutečnost, tedy něco co opravdu existuje a není to jen pouhá domněnka nebo iluze.

V digitální technice pojem Virtuální realita představuje médium, které pomocí zrakových, sluchových a někdy i taktilních vjemů předstírá existenci světa, který kolem nás skutečně neexistuje.

### 2.1.1 Popis zařízení

Zařízení Virtuální reality, kterým se v této práci věnuji je výběr populárních komerčně dostupných produktů, které se skládají z HMD (Head Mounted Display) tedy ty tzv. Virtuální brýle, které obsahují displeje poskytující obrazový vjem a pár reproduktorů pro zvukový vjem. Dále jsou to dva ovladače, které se starají o sledování pohybu rukou, vstupní ovládací prvky a zpětnou vazbu.

Jelikož brýle jsou převážně jen displej a snímací zařízení pro určování polohy, musí nějak získávat zdroj obrazu a napájení. Většina zařízení tedy musí být připojena kabelem k počítači. Vyjímkou je například série zařízení Oculus Quest, který obsahuje vlastní baterii, úložiště a integrovaný procesor a může tak tedy fungovat naprosto samostatně. Takto může fungovat ale jen nenáročnými

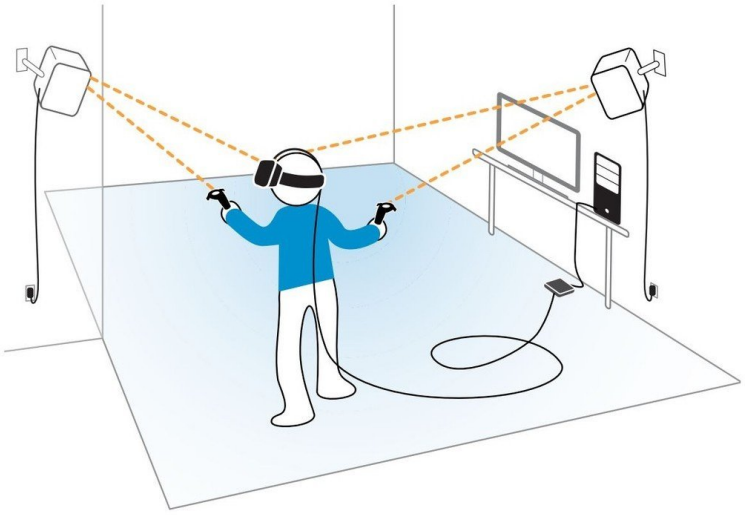
aplikacemi, jelikož integrovaný čip nenahraní dedikovanou grafickou kartu, protože virtuální realita je obecně hodně komputačně náročné médium. Důvodem je nutnost vysoké kvality a plynulosti obrazu. Rozlišení musí být pro každé oko vysoké, jelikož se display nachází velmi blízko u očí. Dále to je vysoká obnovovací frekvence (typicky 90+Hz) a velké zorné pole (typicky 100+°).
▶ Poznámka 2.2. Porovnání jednotlivých zařízení a jejich specifikací se nachází v samostatné sekci.

**Obrázek 2.1** Headset a ovladače zařízení HP Reverb G2 [1]

### 2.1.2 Rozdělení podle sledování pohybů

Tyto zařízení se dají rozdělit na kategorie podle určování polohy brýlí a ovladačů

**Outside-in Lighthouse** Systém, který používá laserové měření a tzv. majákové stanice k určení polohy HMD (brýlí) a ovladačů v reálném čase. Typicky se používají dva majáky, které se rozmístí do protilehlých rohů oblasti. Ty pomocí měření vzdálenosti k senzorům v brýlích a ovladačích určí přesnou polohu zařízení v prostoru. Sledování pohybu se přeruší, pokud majáky ztratí přímou viditelnost na zařízení. Valve a HTC jsou společnosti, které tento systém vyvinuly a používají ve svých zařízeních.



**Obrázek 2.2** Jednoduché znázornění funkčnosti majáků systému Lighthouse [2]

**Inside-out pomocí kamer** Snímající senzory jsou umístěny v samotných brýlích. V tomto případě se jedná o kamery, které snímají okolí a určují jak se jejich pozice vůči němu mění. Ovladače jsou opatřeny množstvím světelných bodů, pomocí kterých kamery určují jejich polohu. tento způsob trpí na světelné podmínky okolí. Pokud je místnost přesvětlená nebo špatně nasvětlená, brýle nebudou schopné zjistit svojí polohu. Stejně tak množství rušivých světel (např. vánoční stromek) může znemožnit určení polohy ovladačů. Dále je sledování ovladačů podmíněno jejich viditelností v zorném poli kamer. Tento systém používají například zařízení Windows Mixed Reality, které pracují s viditelným světlem a zařízení Oculus, které využívají infračervené světlo.


**Obrázek 2.3** Vzhled ovladačů HP Reverb G2 platformy WMR[3]

Pro všechny tyto zařízení je pro plynulý chod důležité, aby byly ve správných světelných podmínkách a v zorných polích sledovacích zařízení.

### 2.1.3 Ovladače

Primární funkcí ovladačů je sledovat pohyb rukou uživatele a umožnit mu vstupní ovládací prvky v podobě tlačítek, joysticků nebo touchpadů, pomocí kterých může interagovat s uživtelským rozhraním a světem. Někdy mají i funkčnost zpětné vazby v podobě haptických signálů.

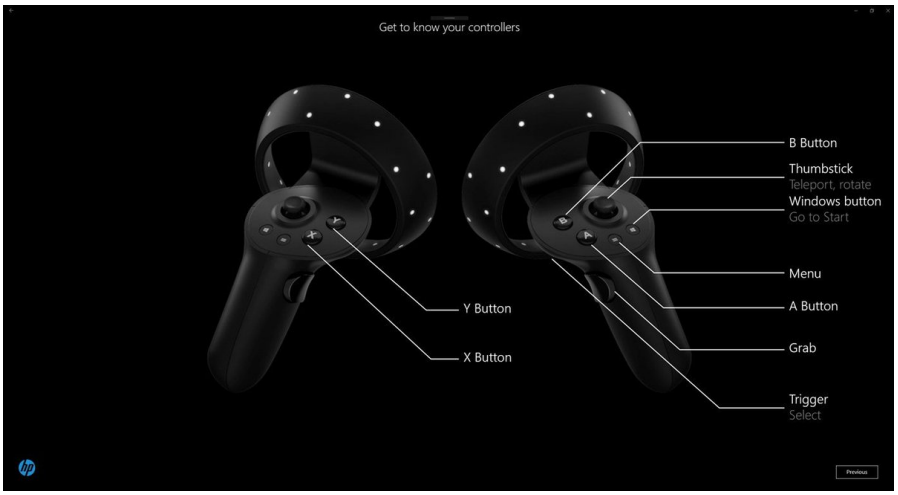
Typicky se dělí na rukojeť, ovládací prvky a senzorovou část pro sledování pohybu.

Některé ovladače (v kategorii zařízení, které popisuji se jedná o Valve Index Controllery) mají funkčnost "finger tracking", tedy sledování jednotlivých prstů, tak že v rukojeti jsou tlakové senzory na místech prostředníčku, prsteníčku a malíčku. Informace o sevření jednotlivých prstů mohou poté aplikace použít pro přesnější znázornění ruky ve virtuálním světě nebo mechaniky pracující s angažovaností jednotlivých prstů.

Kromě sledování pohybu(pozice) pomocí světelných nebo laserových senzorů umí ovladače i samotný headset zaznamenávat zrychlení a rotace pomocí interní jednotky s gyroskopy, akcelerometry a dalšími senzory.

Hlavními ovládacími prvky jsou:

- Joystick/Touchpad: Typicky používáno pro pohyb a orientaci ve virtuálním světě a pro navigaci v uživatelském rozhraní v nabídkách menu, v listech atd.
- Trigger a Grab Button: Trigger nebo někdy také Activation button představuje klik levého tlačítka myši na obrazovce při práci s počítačem, tedy aktivaci nějakého ovládacího prvku nebo předmětu. Tlačítko se nachází v přední části rukojeti na podobném místě v relaci k rukojeti jako např. spoušť u střelné zbraně. Grab button je umístěn na spodní straně rukojeti a jeho zmáčknutí reprezentuje sevření ruky a v aplikacích je to typicky sebrání předmětu nebo jiná, podobná, interakce.
- Menu Button a další tlačítka: Typicky pro zobrazení nabídky ovládacích prvků samotného řídícího softwaru headsetu nebo aplikace. Další tlačítka slouží aplikacím jako další možné ovládací prvky.




**Obrázek 2.4** Diagram ovládacích prvků ovladačů HP Reverb G2 z platformy WMR

Zpětná haptická vazba - zpětná vazba "dotykem" je základní forma předávání informací uživateli o interakcích ve virtuálním světě. V aplikacích typicky používána při kolizi s předměty nebo prostředím, při interakci s ovládacími prvky nebo jako např. intenzivní frekvence vibrací jako

indikátor stavu nízkého zdraví ve hrách. Obvykle je haptická zpětná vazba ve formě vibrace. V ovladačích je to programovatelný prvek, který je schopný vyvolat vibrace o určené frekvenci po nějakou dobu.

### 2.1.4 Omezení a nedostatky

1. Sledování pohybu:

   Omezeno viditelností ovladače v poli majáku nebo v zorném poli snímající kamery. Navíc je možné rušení ostatními prvky v oblasti, které mohou senzory mást.

2. Fyzická zpětná vazba:

   Typicky jen v podobě jednoduchého vibračního zařízení. Vhodné při kolizi nebo interakci s předměty a ovládacími prvky. Rozsah a intenzita vibrací je u některých zařízení dost špatná (např. HP Reverb G2).
   ▶ Poznámka 2.3. Nemáme jak fyzicky předat informace o váze, těžišti nebo možnost aplikovat nějakou sílu na uživatelovu ruku atd. . V této práce se zabívám interakcemi fyzikálních předmětů a tento nedostatek je jedna z mnoha věcí, které nám brání v dokonalejší imitaci reálnách fyzikálních interakcích. Máme sice možnosti jak toto zhruba simulovat, ale uživateli to lze předat jen vizuálně.

3. Kabelové připojení:

   U většiny zařízení je kvůli chodu nutné mít připojený kabel mezi počítačem a zařízením. Existence kabelu samozřejmě omezuje svojí maximální délkou oblast po které, se můžeme volně pohybovat. Dále se kabel při rotaci na místě rád zamotává kolem nohou a celkově zhoršuje volnost pohybů tím že ”překáží”.

4. Motion/Simulation Sickness:

   Obrazové a zvukové vjemy, které zařízení virtuální reality produkují jsou velmi efektivně schopny náš mozek přesvědčit o tom že se skutečně ve virtuálním světě nachází. Podvědomně víme, že se jedná o virtuální prostředí, ale některé reflexy na vizuální podněty náš mozek stejně zahájí. Např. v některých případech při pádu ve virtuální realitě nebo při naklonění virtuálního světa ztrácíme rovnováhu.

   To vede na problém zvaný motion sickeness nebo také simulation sickness. Podobně jako mořská nemoc, může se individuálně projevovat při nečekaných pohybech, velkých rychlostech, rychlé změně stavu kamery a u pohybů, které hráč neovládá se může začít dělat uživateli nevolno. Důvodem je typicky kolize vjemů. Vidíme a „cítíme“ něci jiného. Dále může být problém při neplynulém obrazovém výstupu (kdy se nám obraz tzv. seká).
   Ovšem stejně jako u mořské nemoci si můžeme vybudovat tzv. Sea Legs, tedy „mořské nohy“ – odolnost vůči těmto jevům. Tak ve VR si můžeme obdobně vybudovat takovouto odolnost vůči nevolnostem spojeným s orientací ve virtuální realitě.

   Základní příklady, kdy se motion(simulation) sickness může projevovat:

   a. Pohyb, který nemá hráč pod kontrolou
   b. Praktické efekty s kamerou např. třesení
   c. Trhaný obraz. Nejednotné vykreslování snímků.
   d. Nejednotně rychlí pohyb. Akcelerace, deacelerace atd.
   e. Zvláštní obrazové efekty jako např. Motion Blur

### 2.1.5 Porovnání zařízení

V následující tabulce je provnání specifikací zařízení, které jsou v momentální době komerčně oblíbené. (Rozlišení je udáváno na jedno oko a zorné pole je označeno anglickým FOV - Field of view)[4]

**Tabulka 2.1** Porovnání specifikací výběru zařízení pro VR

| Název zařízení | Sledování pohybu | Rozlišení | Frekvence | FOV | Váha |
|---|---|---|---|---|---|
| HTC Vive | Outside-in Lighthouse | 1080x1200 | 90Hz | 110° | 486g |
| Oculus Rift S | Inside-out | 1280x1440 | 80Hz | 90° | 561g |
| Oculus Quest 2 | Inside-out | 1832x1920 | 90Hz | 89° | 503g |
| HP Reverb G2 | Inside-out | 2160x2160 | 90Hz | 114° | 499g |
| Valve Index | Outside-in Lighthouse | 1440x1600 | 144Hz | 130° | 809g |
| Samsung Odyssey+ | Inside-out | 1440x1600 | 90Hz | 110° | 590g |

## 2.2 Simulace fyziky ve 3D aplikacích

Přidáním některých fyzikálních zákonu do herního světa se zvýší vnímaná realističnosti. Typicky se jedná o aproximaci prováděnou s výpočty v diskrétním prostoru. Simulaci provádějí fyzikální enginy. V tomto případě se v systému Unity používá fyzikální engine PhysX od společnosti Nvidia. Fyzikální enginy provádějí hlavně detekce kolizí, abychom mohli určit, kdy si několik předmětů zkříží cestu a simulace zákonů Newtonovské fyziky.

Obecně se jedná o zákon setrvačnosti – tělesa jsou v klidovém stavu/rovnoměrném pohybu pokud nejsou vnějšími silami narušeny, zákon síly – vektor působící síly je roven hmotnosti násobené vektorem zrychlení a zákon akce a reakce – aplikováním síly se vyvolá stejná síla, aplikovaná v opačném směru. Prakticky je to také například aplikace gravitačních sil na těleso. Tyto enginy jsou typicky parametrické, protože v některých případech chceme úmyslně použít nereálné fyzikální hodnoty pro zjednodušené nebo unikátní mechaniky. Máme tedy kompletní kontrolu nad působením sil a hmotnosmi těles.

Fyzikální enginy typicky pracují s dvěma typy fyzikálních objektů, s Rigid body a Soft body. Simulace pracuje s objektem označeným jako Rigid body jako s jedním uceleným objektem, zatímco u Soft body objektů se simulují i jednotlivé částí tohoto objektu a je tím pádem i komputačně náročnější. [5]

## 2.3 Existující řešení

Následující aplikace nějakým způsobem zpracovávají fyzikální interakce ve virtuální realitě. U každé bude její popis a analýza herních systému zpracována na základě mých poznatků ze zkušenosti s použitím aplikací.

### 2.3.1 Hellsplit: Arena

Arénová hra s hororovými prvky se středověkou tématikou, zbraněmi a vybavením. Fyzikální bojový systém je uplnatněn v arénách (ohraničeném prostoru), na postupně přicházející vlny nepřátele. Nepřátelé jsou herním prvkem, kde je cílem je eliminovat. Nepřátelé jsou v podobě

humanoidních avatarů (jedná se o horrrorovou hru, takže jsou to kostlivci a nemrtví) s prvky aktivních ragdollů. Fyzikální systém a rozsáhlý výběr zbraní umožňuje ničit nepřátele různými způsoby.[6]

**Obrázek 2.5** Oficiální ukázka z aplikace Hellsplit: Arena [7]

Jedná se o pěkně graficky zpracovanou a uhlazenou aplikaci. Hodně využívá animací pro řízení předmětů a avatarů namísto fyzikálních podnětů a má také problémy se stabilitou kolizí.

Možnost pohybu a orientace je implementován pomocí ovládacích prvků – joysticku. Zdá se ale, že je míněno, aby uživatel měl kolem sebe dost místa a mohl tak provádět výpady v reálném prostoru. V případě umělého pohybu (pomocí joysticku) hra pracuje se stálou rychlostí pohybu (poměrně rychlou) a veškerý vertikální pohyb je plynulý, takže jsem nepociťoval problémy s pohybovou nevolností.

Hra používá pro reprezentaci hráče avatar celého těla, které je plynulý, ale je převážně animační – jeho chování je definováno předpřipravenými animacemi a není přiliš ovlivněn nebo řízen fyzikálními elementy a okolím. Zdá se, že používá standartní Inverzní kinematiku k dopočtu pozic a rotací jednotlivých kostí avatara, ale i nějaké komplexnější kroutící modifikace.

Aplikace má implementován systém interakcí s objekty – držení předmětů. Lze interagovat s veškerými předměty v okolí pomocí obou rukou i naráz. V případě držených zbraní lze předměty libovolně uchopit druhou rukou např. v případě obouručního meče na zadní části rukojeti nebo na čepeli (tedy tzv. half swording – technika uchopení čepele pomocí druhé ruky za účelem silnějšího bodu/úderu). Bohužel se jedná primárně o animační efekt, tedy druhá interagující ruka nemá žádný vliv na ovládání předmětu – pozice i rotace jsou určeny jen na základě primární držící ruky.

Pro imitaci váhy držených předmětů je použito jakési odpojení a opoždění předmětu za pohyby ruky. Při pohybu držený předmět i ruka pomaleji akcelerují, pomaleji se pohybují a podobně při zastavení i pomaleji zpomaluje. Celkově je tedy předmět opožděný za polohou ruky a i se pomaleji pohybuje. Nejsem fanouškem této metody, protože se chová a působí jako kdyby znázorněné meče vážili 20kg a byly to neohrabané kusy železa.

Interagovatelné předměty se chovají podle fyzikálních zákonů a rozdíly ve váze a v těžišti jsou při kolizích znát. Navíc je implementována mechanika zabodnutí, kdy se při vyšší rychlosti a

správném směru zbraň s čepelí zabodne do vhodného materiálu nebo nepřítele a v případě nepřítele nebo volných předmětů s nimi lze pomocí zabodnuté zbraně manipulovat, zbraň zabodnout hlouběji, nebo vytáhnout. Aplikace má občas problémy se zaznamenáním kolizí na nepřátelích. Dále se občas držený předmět zasekne na kolizi v geometrii prostředí a nelze s ním dále manipulovat.

Herní prvky jsou v podobě animovaných humanoidních avatarů, které se umí po scéně pohybovat, útočit na hráče, bránit se jeho útokům a uhýbat mu. Jedná se o částečné aktivní ragdolly, které reagují na některé fyzické podněty. Je implementován i systém useknutí/rozdělení částí avatara v případě čistého zásahu na ruce, hlavu nebo nohy.

### 2.3.2 BONEWORKS

- Příběhová akční adventura, která používá pokročilé fyzikální mechaniky. Má dynamické prostředí s tématikou digitálního uměle simulovaného světa. Hra používá fyzikální mechaniky a hádanky pro navigaci a posun v prostředí (lezení po stěnách, skákání, manipulace s objekty pro zpřístupnění nové cesty atd.). Do toho jsou vloženi nepřátelé v podobě humanoidních stvoření a autonomních zbraňových systémů, se kterými může hráč bojovat pomocí fyzikálních zbraní, střelných zbraní nebo předmětů a nástrah prostředí.[8]

**Obrázek 2.6** Oficiální ukázka z aplikace BONEWORKS [9]

Jedná se o pěkně zpracovanou imerzivní aplikaci s velmi pokrokovými fyzikálními herními mechanikami. Navigace v prostředí díky zmíněným fyzikálním pohybovým mechanikám je velmi zábavná a dokonce i trochu fyzicky náročná.

Pohyb a orientace jsou zde také implementovány na použití pomocí joysticku a dalších ovládacích prvků. Navíc se ale lze pohybovat ve světě i pomocí fyzikálních předmětů a vlastního těla. To v případě lezení po speciálních stěnách, vytáhnutí sebe sama nad okraj nebo když se chceme vytáhnout na vyvýšenou pozici pomocí vhodného předmětu.

Reprezentace hráče je avatar celého těla, jedná se o jakýsi aktivní ragdoll, který reaguje na, ale i ovlivňuje, své prostředí. Části těla spolu kolidují a ačkoliv je to velmi realistické, občas to způsobí

problémy, kdy se virtuální tělo o něco zasekne. Velmi realistický působící avatar.

Propracovaný systém interakcí umožnůje libovolně držet virtuální předměty a interagovat s prostředím. Držené předměty jsou proporcionálně zatížené a používané nástroje jsou velmi responzivní. Dále je možnost si přitáhnout vzdálené předměty do ruky.

Tato hra používá fyzikální mechniky snad pro všechny aspekty její funkčnosti. Boj, pohyb, interakce, chování avatarů atd. Je to velmi dobře vyladěný systém.

Hlavními herními prvky jsou hádanky a nepřátelé. Hádanky jsou fyzikálního charakteru. Typicky se jedná o hádanky spojené s váhou předmětů a strukturální integritou nějakého objektu nebo síly působící na páku atd. Nepřátelé jsou v podobě humaniodů co jsou buď neozbrojení nebo ozbrojení a dále jsou to samostatné zbraňové systémy – zbraňové věže. Všechny lze porazit buď použitím fyzikálních mechanik (zásahem improvizovanou zbraní nebo schozením těžkého předmětu) nebo střelnými zbraněmi.

### 2.3.3 Blade and Sorcery

- Arénový sandbox zasazený ve středověkém prostředí s prvky magie. Imerzivní svět s plně fyzikálním bojem, interakcí a realisticky se chovajícími nepřáteli. [10]

**Obrázek 2.7** Oficiální ukázka z aplikace Blade and Sorcery [11]

Pohyb a orientace je pomocí ovládacích prvků ovladačů, možnost vertikálního šplhání pomocí herních rukou a předmětů. Podobné jako u předchozích aplikací. Ideálně by měl mít hráč navíc kolem sebe dostatek prostoru pro volný pohyb pro výpady a uhýbaní.

Hráč je reprezentován virtuální postavou s celým tělem. Celý avatar je velmi plynulí s realistickým chováním. V základní konfiguraci nereaguje kromě samotných rukou na žádné vnější podněty, ale v nastavení lze povolit kolize avatara.

Ve hře je systém pro interakce s objekty i možnost pro jakési telekinetické sbíraní předmětů. Tedy možnost přitáhnout do ruky vzdálený objekt. Objekty lze libovolně uchopit v oblastech

držadel a při obouručním držení předmětu mají obě ruce vliv na pohybu a natočení předmětu.

Předměty jsou rozumně zatížené, ale vyskytuje se zde zase systém odpojení předmětu a zaostávání oproti pohybům ruky. V tomoto případě to ale není až tak přehnané jako u Hellsplit: Arena.

Fyzikální chování předmětů, až na umělé zaostávání za reálnou polohou ruky, je velmi příjemné. Umělá mechanika bodání je velmi plynulá a pěkně zpracovaná. Umožňuje probodnutí a následně manipulování s několika různými předměty nebo avatary nepřátel najednou. Zdá se, že je zde i nějaká umělá mechanika pro měkké zasekávání o tvrdé materiály.

Herními prvky jsou nepřátelé v podobě lidských avatarů s aktivními ragodolly. Nepřátelé jsou vyzbrojeni zbraněmi, štíty, ale i brněním. Umí útočit, bránit se, uhýbat atd. Některé útoky jsou možná trochu moc vedeny animacemi a občas působí přehnaně. Podobně jako u předchozích, i zde je systém odseknutí částí avatara nepřátel. Dále jsou zde prvky magie, kdy může hráč házet ohnivé koule, šlehat blesky, ale i telekinese, kterou lze dálkově manipulovat s předměty a s nepřáteli.

Jde o velmi plynulou a imerzivní aplikaci (možná až moc imerzivní), jejíž systémy se budu inspirovat při návrhu svého řešení.

## 2.4 Historický šerm

▶ **Definice 2.4.** *Šerm = technika boje s ruční zbraní (typicky s mečem) proti protivníkovi/ protivníkům se zbraněmi.*

Obecně se v dnešní době pod pojmem šerm myslí následující kategorie:

**Sportovní šerm** Jedná se také o olympijskou kategorii. V dnešní době moderního boje se střelnými zbraněmi existují i sportovní aktivity, který tyto zbraně využívají nebo jejich využití simulují. Jedná se o sportovní střelbu nebo v rámci souboje je to například Paintball nebo Airsoft. Stejně tak i v průběhu historie, kdy boj s chladnými zbraněmi byl hlavním způsobem při konfliktech a válkách, tak nezávisle existoval šerm jako sport pro nácvik, zábavu a soutěž. Tedy dobová vojenská technika přešla do soutěžního, nekrvelačného prostředí a tam si dále vyvinula vlastní pravidla, nástroje a techniky pro potřeby civilního sportu. Moderní sportovní šerm je tedy nejpozdějším vývojem evoluce historického sportovního šermu. V dnešní době jsou hlavními kategoriemi boj s tzv. fleretem, šavlí a épée(kordem). Jedná se tak trochu o parodie historických zbraní. Typicky se jedná o souboj jeden na jednoho. Cílem je se jako první dotknout nepřítele hrotem zbraně do masky nebo vesty.

**Historické evropské bojové umění (HEMA)** Praktikování bojového umění s historickými zbraněmi podle historických dobových reálii - manuálů, nákresů, manuskript nebo popisů. Při zápasech se ovšem samozřejmě používají neostré varianty zbraní a moderní masky, chrániče a oblečení. Může být dost sportovního charakteru a v poslední době jsou snahy, dostat i tuto variantu jako kategorii na olympijské hry. Jednou z podkategorií by mohlo být tzv. HMB (Historical Medieval battles), kde se jedná o plnokontaktní boj a výbava se kvůli bezpečnosti skládá z moderně zpracovaných a upravených částí zbroje, které ale vypadají jako historická středověká zbroj. Příkladem této podkategorie je mistrovství světa známé jako Battle of the Nations.

**Scénický šerm (teoreticky také jako reenactment)** Poslední kategorie, která má velmi blízko k té předchozí. Také se studují historické zdroje, ale cílem je často připravit a secvičit nějaké vystoupení, nebo se zúčastnit scénické improvizované bitvy. Nedílnou součástí zde ale jsou kostými, tedy oblečení, výstroj a výzbroj, které reprezentují nějaké historické období. Toto je něco co se typicky vidí na reenactment a living history akcích, které se konají na hradech a dalších místech. Jedná se většinou hlavně o představení pro diváky a tomu jsou také přizpůsobeny pravidla. Vzhledem k tomu, že účastníci na sobě mají dobové kostými a nemají žádné moderní masky ani chrániče, tak typicky jsou údery jen zlehka a je zakázané bodání a rány na nohy a hlavu.

Poslední dvě kategorie jsou úzce provázané a jejich praktikanti mohou spadat do obou kategorií. Já osobně mám nejvíce zkušeností se scénickým šermem/reenactingem a podle toho se také budu snažit imitovat chování historických předmětů ve virtuálním světě mé aplikace.

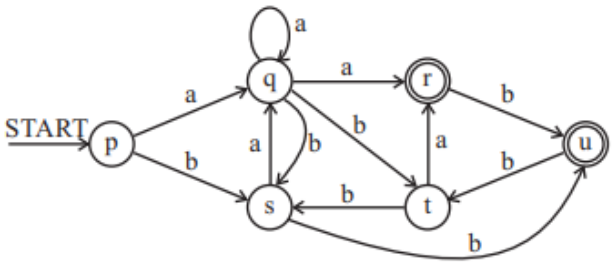
▶ Poznámka 2.5. Jedním z cílů práce je pokusit se imitovat chování a základní vlastnosti bojových předmětů - specificky historických ručních zbraní a výstroje pro boj na blízkou vzdálenost. Tedy meče, kopí, sekery, štíty, helmy atd. Kategorizací a popisem těchto konkrétních předmětů se budu zabývat v kapitole 4.1.

## 2.5 Konečné automaty a gramatiky v rámci herních prvků

Teorie formálních jazyků je úzce spojena s teoriemi automatů a gramatik a je nedílnou součástí počítačových věd. Jsou to překlady programovacích jazyků, lexikální analyzátory atd. Obecně se jedná o vyhodnocování dat. U návrhu softwaru se jedná o zachycení navrhované struktury např. pomocí stavových diagramů nebo při implementaci softwaru je to např. návrhový vzor State, který zpřehledňuje a zjednodušuje větvenou logiku funkcí pomocí principů konečných automatů.

**Konečný automat** V teorii se jedná o pětici (Q - množina stavů, Σ - abeceda, $\delta$ - přechodová funkce, $q_0$ - počáteční stav a F - množina koncových stavů). Nejdůležitější je ale přechodová funkce (zobrazení Q x Σ → Q)[1], která nám říká, jak struktura reaguje na vstupy přijímajícího jazyka.

Zapisuje se buď pomocí výpisu parametrů a hodnot funkce nebo pomocí diagramu konečného automatu. Diagram se podobá znázornění grafu – množina vrcholů(stavy) a mezi nimi jsou směrové hrany(přechody). Hrany jsou ohodnoceny symbolem z abecedy Σ. Navíc je zde znázorněn počáteční symbol pomocí šipky směřující na vrchol a koncový stav pomocí dvojitého obrysu vrcholu.




**Obrázek 2.8** Ukázka diagramu přechodové funkce končného automatu

[1] funkce deterministického konečného automatu

Na vstupy a proměnné programu lze nahlížet jako na jakýsi jazyk, který určuje chod a pořadí provádění operací.

V této aplikaci používám konečné automaty a gramatiky pro návrh a implementaci herních prvků v podobě trénovacích panáků. Ty reagují a řídí se podle jazyka, který je složen z parametrické reprezentace akcí hráče a časových událostí světa. Stavy jsou poté reprezentace akcí, které automat umí vykonávat.

# Kapitola 3

# Použité technologie

## 3.1 Unity

Unity je multiplatformní herní engine vyvinutý společností Unity Technologies. První verze byla vydána roku 2005. Umožňuje vývoj 2D a 3D aplikací libovolného žánru a zaměření. Práce probíhá hlavně v grafickém prostředí Unity Editoru a také tvorbou skriptů v jazyce C#, pomocí kterého se přistupuje k aplikačnímu rozhraní enginu.[12]

Výhodou enginu je to, že je pro jednotlivce bezplatný a existuje množství dokumentace a návodů pro jeho použití jako například Dokumentace Unity[13] nebo platforma Unity Learn[14].

### 3.1.1 OpenXR

Open XR je aplikační rozhraní pro multiplatformní kompatibilitu napříč nejpoužívanějšími VR headsety. Zjednodušuje vývoj VR a AR aplikací. Programátor nemusí vytvářet různé verze pro specifické rozhraní různých headsetů. Místo toho vytvoří jedno univerzální řešení pracující s rozhraním OpenXR, které se dá aplikovat na jakýkoliv headset.[15]

### 3.1.2 PhysX

PhysX je open-source engine pro simulaci fyzikálních procesů v reálné čase pomocí více vláken. Vyvinut společností Nvidia. Je používaný ve velkém množství her, protože jeho SDK (software development kit) je dostupný zdarma pro využití ve všech placených i neplacených aplikací.[16]

#### 3.1.2.1 Rigidbody

Jedná se o vlastnost, která umožní objektu aby se zapojil do základního fyzikálního chování, které engine simuluje. Je to tedy typ tělesa použitého pro fyzikální simulace, který se jako celek chová jednotvárně. Jeho tvar ani objem se s účinkem působících sil nijak nemění.

#### 3.1.2.2 Joint

Jedná se o vlastnost mezi dvěma fyzikálními objekty, která omezuje jejich relativní pohyb. Obecně se jedná o způsub jak k sobě dva objekty kontrolovaně připojit. V aplikacích se používají např. pro panty dveře, kdy pant umožní objektu dveří volnou rotaci po některé z os, ale jejich pozice je omezená na bod, kde jsou spojeni. Dále např. pro fyzikálně simulované aktivní ragdolly nebo řetězy.

#### 3.1.2.3 Collider

Jedná se o komponentu, která se v herních enginech 3D a 2D aplikací používá pro detekci srážek dvou a více objektů. Také se požívají při výpočtu fyzikálních simulací. Collider popisuje fyzický tvar objektu, typicky zjednodušeně, aby se ušetřil výpočetní výkon a aby simulace byly konzistentnější.

### 3.1.3 Photon PUN

Photon Unity Networking(PUN) je balíček pro Unity s frameworkem pro tvorbu multiplayerových aplikací. Dále nabízí i propojené služby pro funkčnosti živého chatu a přenosu hlasu v reálném čase (voip).[17]

### 3.1.4 C#

Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý společností Microsoft. Byl představen a vydán v roce 2000 zároveň s platformou .NET Framework. Je založen na dalších objektově orientovaných jazycích a to C++ a Java.[18]

## 3.2 Blender

Blender je open-source software sloužící primárně k modelování 3D objektů. Mezi další funkce patří animování, texturování, rigging, exportování modelů, renderování, editace videí a přidávání efektů, editace rasterů a mnoho dalších. Využívá se v herním průmyslu pro tvorbu assetů a ve filmovém průmyslu. [19]

### 3.2.1 Modelování

Způsob tvorby 3D objektů pomocí definování a manipulace jeho možných vrcholů, hran a ploch.

### 3.2.2 Texturování

Metoda přidání textury (2 dimenzionální obrázek) na 3D objektu pomocí definovaní jeho rozkladu. Objekt se rozloží do tzv. UV roviny a části textury se podle tohoto rozložení namapují na jednotlivé vrcholy a plochy.

### 3.2.3 Rigging

Rigging je metoda přidání kostry (složené z kostí) objektu. Jednotlivé kosti poté získávají vliv nad částmi meshe a objekt lze snadno animovat pomocí pózování kostry.

### 3.2.4 Animace

Způsob přidání zdánlivého pohybu předmětům. Počítačová animace je převážně tvořena definováním tzv. klíčových snímků na časové ose, kdy počítač dopočítá chování předmětu mezi těmito polohami v čase pomocí různých křivek.

## 3.3 Substance Painter

Jedná se o software „malování“ ve 3D. Jeho ovládání a princip je podobný s Adobe Photoshopem. Jeho primární funkcí ale je vytvářet, malovat a generovat textury na 3D objekty. Také umí zpracovávané objekty renderovat.[20]

## 3.4 Character Creator

Character Creator je nástroj pro rychlou tvorbu, import a kustomizaci realisticky vypadajících postav, jejich koster a animací. Dále umožňuje jednoduchý export do všech hlavních softwarů pro tvorbu 3D animací, modelů a her.[21]

## 3.5 Mixamo

Mixamo je bezplatný webový nástroj pro získání postav, rigů a univerzálních animací z rozsáhlé databáze. V momentální době je ve vlastnictví firmy Adobe. Dále nabízí funkce pro vytoření rigu pro vlastní postavu a namapování existujících animací na libovolnou postavu. [22]

Kapitola 4

# Návrh a tvorba světa, prostředí a objektů

Tato kapitola se věnuje představení a kategorizování historických předmětů, tvorbě jejich digitálních replik a tvorbě prostředí pro výslednou aplikaci.

## 4.1 Historická výzbroj a výstroj

Cílem práce je ztvárnit pokud možno co nejvěrohodněji některé historické bojové předměty. Nápad byl takový, že by aplikace mohla působit trochu jako interaktivní muzeum. Tedy že máme ve světě plně funkční repliky předmětů, které si můžeme bezmezně prohlížet a v rámci prostředí i vyzkoušet.

Co se týče kategorizace předmětů, tak z historického hlediska byla dost prostá. Obecně řečeno pokud zbraň měla jílec, záštitu a čepel, byla typicky zaznamenána a označována prostě jen jako meč. Podrobná typologie a kategorizace jednotlivých součástí je spíše posedlostí moderní doby (řekněme zhruba od viktoriánského období). Názvy specifických historických předmětů tedy nejsou u velkého množství případů dobové, ale spíše moderní.

Bez této podrobné kategorizace bychom ale nebyli schopni rozlišit mezi předměty z různých období a lokací. Samotná kategorizace jen podle lokace nálezu nebo dobového záznamu není nejspolehlevější, jelikož místo nalezení nemusí odpovídat místu vývoje a výroby. Dobový časový záznam také nemusí být úplně vypovídající opravdovému věku a původu. Přeci jenom i v dnešní době používáme desítky i stovky let staré předměty. V historickém kontextu by tedy nebylo vůbec zvláštní z generace na generaci zdědit a používat stovky let starý nástroj nebo předmět.

Typologie tedy většinou rozlišují podle dominantních charakteristik předmětů. Například asi nejznámější typologie středověkých mečů je Oakeshott typologie[23], která se zaměřuje na evropské meče a rozděluje je na typy hlavně podle tvarů, rozměrů a vlastností jejich čepelí, dále podle záštit a hrušek.

Rozhodl jsem se ztvárnit zajímavé předměty z několika různých historických období. Prvním z nich je starověké Řecko, specificky zbraně a zbroj pro hoplitu – bojovníka některého z městských

států, který je součástí sevřené bitevní řady – falangy. Jeho hlavní výzbrojí je velký kulatý štít Hoplon (nebo také Aspis), dále dlouhé kopí Dory(Doru) typicky s bronzovým hrotem na spodní straně a jako poboční zbraň rovný dvojbřitý meč Xiphos nebo zahnutý meč Kopis, který je vhodnější spíše na sekání. Nejdůležitější ochranný prvek v jakékoliv bojové situaci je helma. V tomto případě je to ikonická bronzová uzavřená helma korinthského typu s nánosníkem a chrániči tváří. Dále ochrana těla a to hrudní plát nebo linothorax (brnění ze lnu – přesná forma a složení nejsou známy) a holenice pro ochranu nohou.

Další ztvárněné předměty jsou z období středověké Evropy. Do období raného středověku spadá kulatý štít, který je asi nejčastěji spojován se severskými nájezdníky. Tou dobou se ale štíty tohoto typu používaly skoro po celé Evropě, jelikož jsou jen dalším vývojem předchozích štítů se středovou rukojetí. Další zbraní, která je co se období týče univerální, je sekyra jakožto poboční zbraň. Samozřejmě existují i varianty s dlouhou násadou a větší čepelí používané obouruč. Sekery určené do bojového prostředí mají jiný profil a tvar čepele (typicky mnohem užší a tenčí) než ty určené jakožto nástroje na sekání a kácení. Dalším z předmětů je helma typická pro řadového pěšáka vrcholného středověku – železný klobouk, nejprve jako vývoj z helmy známé jako lebka přidáním krempy a dalších ochranných plátů navíc, dále již jako jednolitá konstrukce. Poslední dvojice předmětů z vrcholného a pozdního středověku je jednoruční meč a malý štít Buckler (česky pukléř). Typická kombinace spíše pro civilní osobní ochranu než pro válečné využití.

## 4.2 Vytváření grafických herních assetů

**Herní asset** termín, který označuje základní elementy, ze kterých jsou hry, jejich systémy a prostředí poskládány. Jedná se o ucelené jednotky, které lze ve scénách nebo i v jiných projektech použít vícekrát. Typicky jsou vytvářeny v externích specializovaných programech, ale herní enginy mají často také omezené nástroje umožňujíci jejich tvorbu. Dají se klasifikovat na Grafické assety – např. modely, animace, samotné textury nebo shadery. Skripty – kódy systému a jeho komponent (těmi se zabývám v kapitole 5) a Zvukové/hudební assety – zvukové klipy, efekty, hudba atd. (tato práce se zpracováním hudby ani zvuků nezabývá).

Tvorba herních assetů skýtá několik kompromisů — hry jsou obecně komputačně náročnější médium. Typicky vyžadují, aby se za vteřinu obraz scény vykreslil z pohledu kamery řádově více jak šedesátkrát. Každé vykreslení vyžaduje, aby se všechny viditelné objekty a jejich vrcholy, hrany, plošky a všechny fragmenty mezi nimi správně vykreslily a vystínovaly. Dá se tedy říci, že čím komplexnější objekty jsou, tím déle bude trvat než se ve scéně vykreslí a budou zabírat více paměti.

Objekty tvořené do her jsou tedy typicky, co se týče meshe, trochu surovější a detail se převážně získává pomocí texturování a technik s ním jako je např. bump a normal mapping, které ovlivňují osvětlovací model tak, aby se zdálo, že povrch má detaily, které jeho samotná struktura modelu ve skutečnosti nemá.

Aplikace a hry pro stolní počítače a konzole si díky tomu, že monitor je relativně malý a pozorovatel ve větší vzdálenosti od něj, mohou dovolit tuto techniku využít naplno a slabší části neukazovat vůbec například omezením ovládacího schématu.

Virtuální realita má v tomto nevýhodu, že umožňuje uživateli ohromnou svobodu, co se prozkoumávání a inspekce okolí týče. Navíc jsou displeje s vysokým rozlišením přímo u očí pozorovatele. Je tedy důležité objekty dělat detailně (při tvorbě fotorealistického světa), co se struktury týče a méně se spoléhat na techniky texturování, protože iluze detailu se při některých úhlech pozorování ztrácí.

Obecně je tvorba objektů do her kompromis mezi kvalitou a časem. Pro hry se mohou vytvářet řádově stovky/tisíce modelů. Tedy jsou objekty, které jsou hlavní tváří aplikace — ty se kterými

se uživatel skoro určitě setká a věnuje jim nejspíše největší pozornost a pak jsou tu objekty, které jsou neméně důležité, ale figurují spíše v pozadí nebo jako komplement hlavním předmětům a prostředí.

V mém případě jsem nejvíce času a detailu věnoval předmětům zbraní a zbroje, které jsou pro aplikaci zásadní. Jako vedlejší objekty jsem zpracovával ploty, cedule, stany a trénovací panáky.

Všechny objekty jsem zpracoval podle níže popsaného workflow. Hlavní objekty ale samozřejmě dostaly více času a pozornosti pro detaily.

### 4.2.1 Workflow

**Workflow – „proud práce“** schéma znázorňující postupné použití programů a jejich funkcionalit, které dohromady tvoří komplexní činnost.

▶ Poznámka 4.1. Teoretický český překlad „Pracovní postup“ podle mě úplně nevystihuje to, že se jedná především o koncept postupu a ne přímo o manuál přesných instrukcí. Proto v práci používám cizí pojem workflow a ne jeho český ekvivalent.

V mém případě jde tvorba modelů pro tuto aplikaci rozdělit na několik kroků. Toto specifické workflow není nijak unikátní, naopak je převzané z volně dostupných naučných prostředků.

#### 4.2.1.1 Reference

Cílem prvního kroku je nalézt reference pro daný předmět

**Reference** Materiál, který obsahuje informace o předmětu. Používá se pro porovnání a zpřesnění tvorby vyhotovovaného předmětu. Jsou to typicky obrázky, plánky, rozměry a nebo samotný fyzický předmět.

Jakožto zdroje referencí používám dochovalé muzejní exponáty, moderní repliky, typologické systémy a literaturu. Pro organizaci digitálního referenčního materiálu jsem použil volně přístupný minimalistický software PureRef [24], který umožňuje do svého okna jednoduše přidávat a volně manipulovat s obrázky.

#### 4.2.1.2 Model

Poté co jsou připravené podklady a vím jaký má přesně zpracovávaný objekt tvar a rozměry, můžu vytvořit jeho mesh – jeho strukturu vrcholů a hran. Na to používám modelářský software Blender [19].

Pro tvorbu meshe existuje množství postupů. Pro hlavní objekty jsem používal metodu, kterou bych nazval „subsurface modeling“ – definování komplexního tvaru pomocí jednoduššího meshe a nedestruktivního modifikátoru, který rozděluje plochy na více „podploch“. Pro vedlejší objekty, které jsou obecně méně detailní, jsem pracoval s primitivním modelem, který jsem postupnou úpravou a modifikací předělal na finální.

Dalším dílčím krokem je tzv. UV Unwrapping — potřebujeme určit, jak se na mesh v budoucím kroku namapuje textura. Pomocí tzv. švů určím jak se mesh rozloží do plochy.

#### 4.2.1.3 Textury

Modelová data z předchozího kroku přesunu do programu Adobe 3D Substance Painter. Jedná se o program jehož hlavní předmětem je malování, generování a vytváření textur a renderování již hotového meshe.

Používá k tomu systém hierarchie vrstev a kanály pro různé složky. Vrstva je položka, která obsahuje nějaký materiál/kresbu a efekty. Vrstvy jsou tedy v hierarchii, kde nejvyšší vrstva má při generování výsledku přednost. Vrstvy se navzájem mohou prolínat a maskovat oblast vlivu. Kanál určuje jaké složky daná vrstva ovlivňuje. Jedná se o kanály barvy, výšky, hrubosti, kovovosti, normály, emise a průhlednosti.

Každý materiál objektu má svou hierarchii vrstev. Typicky využívám hojně možnosti maskování, multiplikace a odečítání vrstev mezi sebou. Po tom co jsem hotov exportuji textury jednotlivých vrstev (které používají shadery v Unity) každého materiálu.

#### 4.2.1.4 Export/Import

Pro export dat z Blenderu do Unity se mi nejvíce osvědčil formát .fbx. Vzhledem k tomu, že každý software má jinak orientovaný prostor, tak je důležité před exportem mít škálu a všechny modifikátory aplikovány. Při importu objektu do Unity je nutné extrahovat a vygenerovat objekty materiálů. Do nich lze následně jednoduše dosadit do správných složek jednotlivé exportované textury ze Substance Painteru z předchozího kroku.

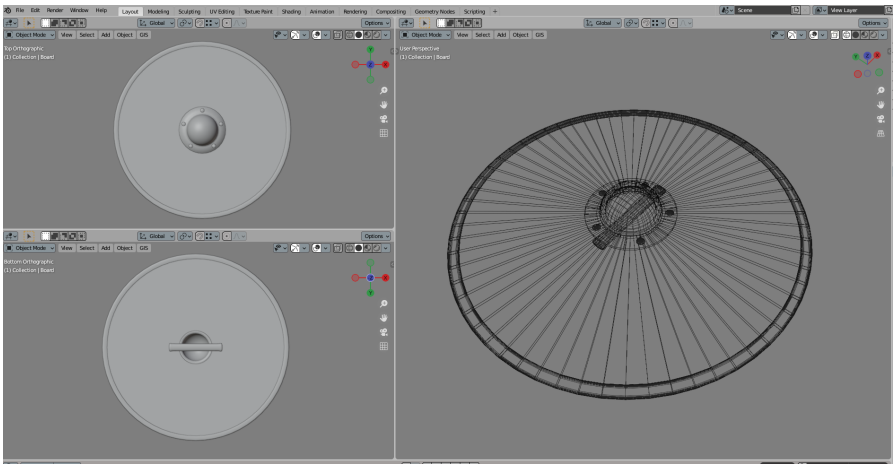
#### 4.2.1.5 Příkladné použití

Ukázka jednotlivých kroků při tvorbě raně středověkého kulatého štítu:

**Reference** Zdrojem informací rozměrů a konfigurace jednotlivých součástí je publikace archeologických nálezů a kategorizace raných anglosaských štítů [25]. Podklady pro vzhled jsou nápady vzhledu štítů s možnou dobovou ikonografii.

**Obrázek 4.1** Referenční tabule v aplikaci PureRef pro tvorbu štítu

**Model** Jelikož se jedná o dost jednoduché tvary, použil jsem metodu postupné modifikace primitivních tvarů pro jednotlivé části. Štítový puklíř vznikl z rozpůlené koule roztažením jejích okrajů a zkosení přechodu mezi ním. Pro vytvoření tloušťky jsem tvar zduplikoval a odsadil. Samotné prkno vzniklo z válce a pomocí jednoduché booleanovské operace jsem do něj uprostřed udělal otvor, kde se připojuje k puklíři. Ochranné lemování okraje vzniklo z duplikace okrajové řady štítového prkna a zvětšení. Nýty v puklíři jsou hodně zjednodušené a zploštěné poloviční koule napozicované a zduplikované modifikátorem Řada s počátkem uprostřed puklíře a pomocí relativního odsazení a rotací o 360/5°. Následně jsem pro každou část vytvořil vlastní materiál a UV souřadnice. Švy byly vedeny podél hran okrajů.



**Obrázek 4.2** Vyhotovený model štítu v aplikaci Blender

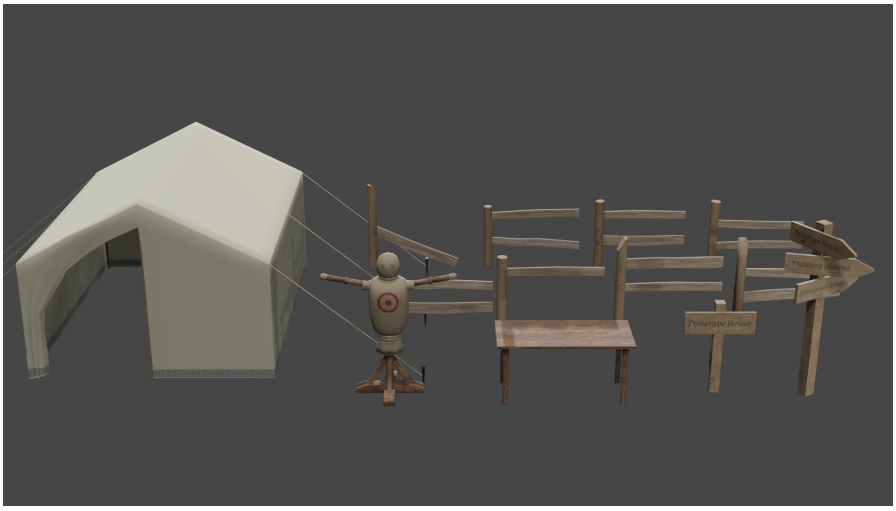
**Textura** Model byl nahrán do programu Substance Painter. Pokud bych v tuto chvíli chtěl na předmětu vytvořit nějaké hloubkové detaily, jako jsou např. škrábance, poškození nebo nějaký embos, tak bych namaloval pomocí štětců, alfa šablon a textur do hloubkového kanálu materiálů detaily. Tento štít má znázornit nedotčený exponát a všechny výškové detaily budou tvořeny pouze samotnými materiály – dřevo, kůže atd. Pro některé materiály používám tzv. chytré materiály – již hotová a znovu použitelná hierarchie vrstev. Pro materiál kovu jsem použil existující vícekanálový materiál oceli s upravenou škálou. Stejně tak tak jsem vytvořil i textury pro materiál surové kůže na okraji. Dřevěnou desku štítu jsem zpracoval po částech v různých vrstvách. V jedné složené vrstvě mám materiál dřeva, který aplikuji jen na výškový, hrubostní a normálový kanál. Pomocí dalších dvou vrstev, které jsou jen na barevném kanálu, jsem namaloval pomocí různých štětců a šablon masku barvy popředí symbolu „erbu“. „Erb“ je inspirován ikonografií z reference. Držadlo je materiál dřeva a materiál kovu, který pomocí masky a hloubkového kanálu se zápornou hodnotou znázorňuje zapuštěné nýty/hřebíky.


**Obrázek 4.3** Otexturovaný štít v aplikaci Substance 3D Painter

**Export** Poté jsem pro každý materiál exportoval textury, které používá osvětlovací model v Unity (který je docela standardní) – Albedo (barevná), Metallic, Roughness a Normal. Model z Bledneru jsem nahrál do adresáře Unity projektu, extrahoval a vygeneroval objekty materiálů a exportované textury do nich dosadil.

**Obrázek 4.4** Finální podoba štítu v prostředí Unity

### 4.2.2 Vytvořené objekty prostředí



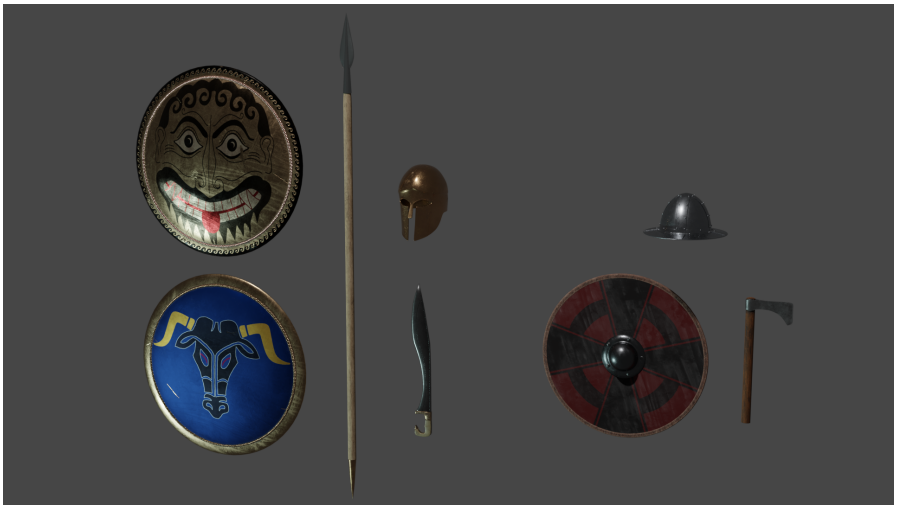
**Obrázek 4.5** Render vytvořených modelů prostředí

**Plot, stůl, rozcestník a cedule** Všechny objekty byly vytvořeny pomocí primitivních meshů. Referencí byly různé obrázky reálných objektů z internetu.

**Stan** Referencí je rekonstrukce stanu s dřevěnou kostrou a plachtou. Textura a vzory na stanech jsou vymyšlené a mají jen tématicky oba stany odlišit. Na plachtu stanu je aplikovaná simulace látky tak, aby stan působil vizuálně zajímavě. Simulace je všude volná kromě vrcholu stanu a bodů, kde je stan uchycen lanovými kolíky. Navíc má simulace látky také kolize s colliderem postavy hráče a s collidery jeho rukou, aby bylo zajištěno trochu imerze při interakci.

**Figurína** Vizuálně je založena na figurínách, které se typicky používají ve hrách pro hráče jakožto úvodní nepřítel, na kterém si mohou bezpečně vyzkoušet ovládací schéma a své schopnosti. Má sloužit jako tématický interaktivní prvek, jehož účelem je, aby na něm byly zkoušeny zbraně. Ekvivalent boxovacího pytle. Tento můj model má ale dlouhé a segmentované „ruce“, které v oživlé aktivní variantě herního prvku budou použity pro animování útoků a obrany figuríny.

### 4.2.3 Vytvořené hlavní předměty



**Obrázek 4.6** Render vytvořených historických předmětů (velikost některých předmětů není z důvodu viditelnosti úplně správná)

**Aspis/Hoplon** Pro jeden model jsem vytvořil dvě varianty tohoto štítu. Jeden s vyobrazením Gorgony (mytické stvoření – dcery mořského boha Forkýse – jednou z nich je Medúsa), což se z muzejních exponátů a vyobrazení zdá jako dost oblíbený motiv. Druhý motiv je namalované vyobrazení býčí hlavy. Každá varianta je tedy samostatná sada textur a obsahuje různé embosy a efekty škrábanců a poškození.

**Doru** Kopí bylo hlavní zbraní pro boj ve formacích od pravěku až po raný novověk, kdy ho plně vystřídali palné zbraně. Referencí rozměrů a tvaru hlavice jsou moderní repliky.

**Korintská helma** Ikonická přilba, jedná se o dřívější variantu, která má ještě překryté uši. Podle některých typologií je tato varianta datována kolem roku 500BC. Referencí jsou muzejní exponáty a moderní repliky.

**Kopis** Také nazývaný jako Falcata (varianta spojovaná s Ibérským/Pyrenejským poolostrovem). Tvar a hrubé rozměry podle muzejních kousků a moderních replik.

**Kulatý štít** Podrobně v kapitole 4.2.1.5

**Sekera** Sekera typicky jako poboční zbraň. Narozdíl od meče má výhodu v ceně, jednoduchosti se zacházením (srovnání čepele s osou úderu je mnohen snažší u objektů, které jsou symetrické a středově nevyvážené) a možnosti použití hlavice jako háku k manipulaci s nepřátelovým štítem nebo zbraní. Zároveň ale nemůže moc bodat, má mnohem menší řeznou plochu a typicky i dosah. Tvar a profil sekery je založen na typu B z Petersonovy typologie skandinávských seker a mečů z obdobý raného a vrcholného středověku.

**Železný klobouk** Referencí rozměrů a tvaru je moderní replika, kterou vlastním.

**Jednoruční meč a puklíř** Nejsem autorem těchto modelů. Jsou vytvořeny podle vyobrazení ze zachovalého historického manuálu pro zacházení s touto kombinací zbraní zvaném i.33. Tyto dva předměty jsem používal už pro experimentování a testování v počáteční fázi projektu a jedná se o pěkné modely, tak jsem je z časových důvodů ve finální verzi scény zanechal. Jsou volně dostupné z webové databáze modelů Sketchfab [26].

## 4.3 Návrh prostředí

Při navrhování ukázkového prostředí jsem si dal pár bodů, které bych měl s prostředím splnit.

1. Rozmístění, viditelnost a vizuální aspekt objektů prostředí by mělo navádět k hlavním prvkům aplikace.
2. Prostředí by nemělo být lineární, mělo by být dostatečně rozsáhlé k prozkoumání a přirozeně ohraničené.
3. Mělo by být tématické a v rámci tohoto tématu konzistentní.

Na základě těchto podmínek jsem si udělal několik náčrtů z nichž tento se mi zdál nejlepší a zpracoval jsem ho.

**Obrázek 4.7** Náčrt navrhovaného prostředí

Tematicky se jedná o tréninkový tábor, který je umístěn ve skalnaté rokli, která jej obklopuje ze tří stran. Herní oblast je tedy ohraničena neprostupnými keři a skalami.

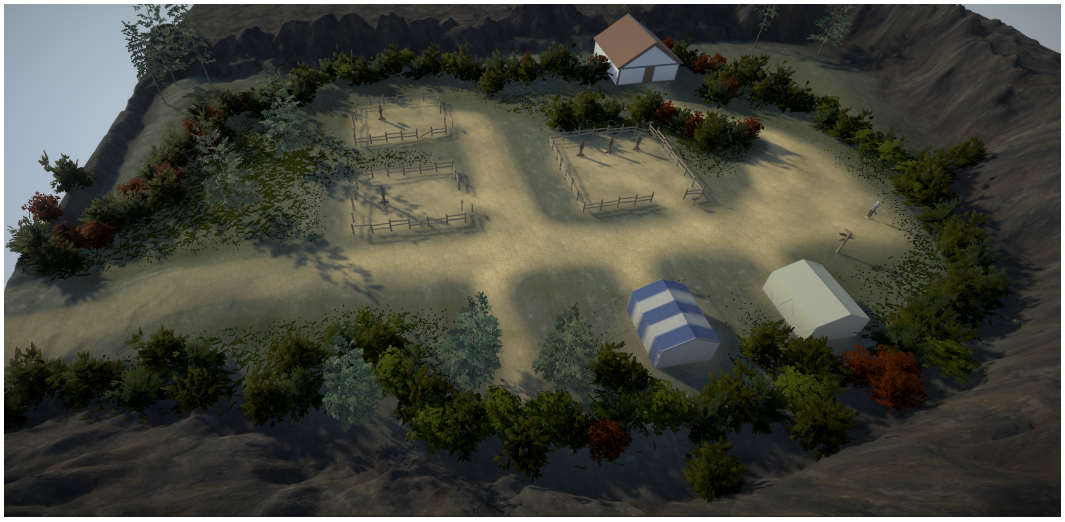
Z počáteční pozice hráč spatří všechny hlavní prvky. Nejblíže přístřešky/stany se zbraněmi a zbrojí, blízko nich oblast s interaktivními prvky, kde si může hráč předměty vyzkoušet a nejdále jsou arény s aktivními herními prvky. Prostředí je ale větší a na boku obsahuje napůl schovaný „prototypový domek“, kde se nachází předměty z průběhu implementace a testování herních prvků. Tento prvek tedy trochu odporuje třetímu bodu tím, že není konzistentní se zbytkem prostředí. Jelikož se ale jedná o ukázkové prostředí, přišlo mi vhodné toto zde porušit a přidat na ukázku do scény množství netématických objektů a prvků.

Pro realizaci tohoto návrhu jsem využil nástroj tvorby terénu v prostředí Unity a balíček rozšíření k terénu[27], který je volně dostupný z Unity Asset Store[28]. Umožnuje formovat terén pomocí skulpturovacích nástrojů a texturovat pomocí štětců a vrstev. Dále jsem využil i jeho možnost instanciovat efekty prostředí jako jsou tzv. stromy — instance meshů a trávy — instance plošek s texturou. Nástroj umožňuje přidávat tyto instance do terénu také pomocí štětců.

Při tvorbě prostředí jsem pracoval s měřítkem 1 jednotka Unity = 1 reálný metr. Podle toho jsem také škáloval a aranžoval importované vytvořené modely.

K tvorbě prostředí jsem využil výše vytvořené modely a volně dostupné assety z Unity Asset Store[28] : modely stromů[29], modely keřů[30], textury trav a květin[31], textury zeminy a

travnaté plochy[32] a HDR pozadí oblohy [33].



**Obrázek 4.8** Finální podoba navrhnutého prostředí

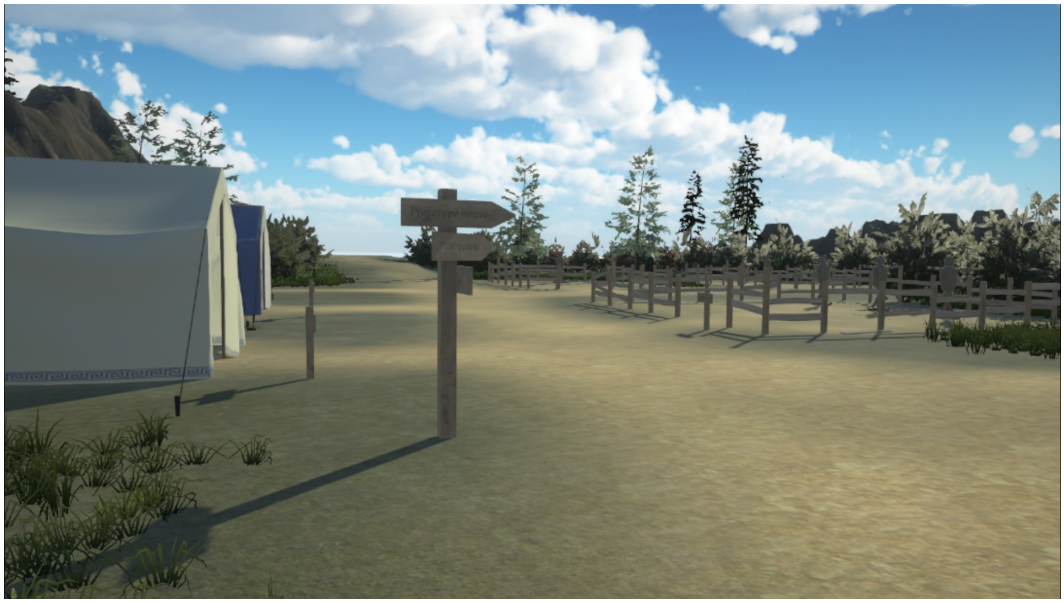
Hned po spuštění jsou před hráčem na levé straně stany, ve kterých jsou umístěny herní předměty – zbraně a zbroj. Stany jsou každý vzhledem unikátní a představují nějaké historické období, ze kterého předměty uvnitř pocházejí nebo jsou pro něj ikonické. Samotné stany by měly okamžitě být ve scéně vizuálně to nejzajímavější a tedy podněcovat k prozkoumání a vyzkoušení nalezených předmětů.

Na počátečním místě se v blízkosti hráče také nachází rozcestník, který má pomoci s okamžitou orientací a přiblížit rozložení prostředí. U každého z prvků je navíc rozmístěna odpovídající cedule pro usnadnění orientace.

Hned naproti východům ze stanů se nachází ohraničená oblast s pasivními trénovacími panáky, která tedy slouží k okamžitému vyzkoušení sebraných předmětů ze stanů na interaktivním prvku.

Dále směrem od počátku a trénovací oblasti se nachází tréninkové arény s aktivními trénovacími panáky. Jedná se o dvě arény, každá zaměřena na jinou aktivitu. V té první, bližší, se nachází útočný panák, který po aktivaci na hráče provádí výpady z různých gardů a směrů. Cílem je se výpadům bránit. Je to hlavně cvičení reakcí a pohotovosti. Druhá aréna obsahuje panáka co se brání, zde je cílem vymyslet a provést fintu, která obejde štít obránce.

Poslední prvek prostředí je napůl schované stavení tzv. prototypový barák, ve kterém se nachází předměty a prvky z průběhu prototypování ,implementace a testování. Slouží převážně jako zajímavost, kterou lze snadno po zorientování a krátkém prozkoumání okolí najít a která do scény přidá spoustu „obyčejných objektů na hraní“.



**Obrázek 4.9** Finální podoba navrhnutého prostředí z pohledu hráče

## 4.4 Imerze

Technická kvalita — obecně jsou to prvky, které vtáhnou uživatele do děje a které do nás vnášejí pocit reálnosti (ne nutně realističnosti). Je to téma úzce propojené s virtuální realitou, o kterém se vedou debaty. Virtuální realita implicitně obsahuje množství imerze díky tomu, že se jedná o iluzi, která náš mozek „ošálý“ a vnutí nám pocit reálnosti okolního virtuálního prostředí.

Naopak ztráta imerze je typicky způsobena nekonzistencí prvků ve stanovených pravidlech virtuálního světa. Celý pojem imerze a její skutečné pojetí v reálných příkladech je silně subjektivní a pro každého jsou tedy imerzivní jiné prvky. Imerzivní může být tedy také to že uživatel má na základě vizuální strany předpoklad výsledku nějaké akce, a ten se naplní. Tedy aplikace umožňuje to, co se zdá že by mělo jít.

Prvky imerze, které zpracovávám a které jsou podle mých zkušeností imerzivní jsou zpětná vazba při interakcích, plynulý systém interakcí s objekty. Vrstvené zpracování „rukou“ – fyzikální a animační vrstva. Efekty prostředí jako je např. interagovatelná simulace látky na plachtách stanu atd.

Kapitola 5

# Návrh a implementace prototypu

Tato kapitola se věnuje návrhu a implementaci samotné aplikace a jejích jednotlivých systémů.

## 5.1 Cíl implementace

Cílem je vytvořit interaktivní aplikaci pro virtuální realitu v systému Unity. Předmětem aplikace je interakce předmětů pomocí fyzikálních mechanik a herní prvky, které jsou na těchto mechanikách postaveny.

Cílovou platformou jsou jen zařízení virtuální reality. Není možné vytvořit rozumné ovládací schéma pro takto volnou manipulaci s předměty i pro ostatní zařízení jako je např. klávesnice a myš nebo konzolový ovladač.

## 5.2 Práce s Unity

Použitá architektura se odvíjí od herního enginu. V tomto případě se tedy jedná o herní engine Unity. Pracuji tedy s jeho strukturou a pro implementaci prvků používám tzv. herní objekty a jejich komponenty. To znamená tvorbu skriptů – komponent objektů, pomocí kterých jsem schopný ovládat a vytvářet logiku chování objektů. Dále to je manipulace s hierarchií scény a referencování objektů komponentám.

Zdrojem informací o funkčnosti a použitelnosti editoru Unity pro mojí práci jsou: oficiální dokumentace Unity[13], webová stránka Unity Learn[14] s detailními ukázkami a tutoriály a publikace Unity Game Development[34]

## 5.3 Verzování

Práce na takovémto projektu je časově rozsáhlejší a typicky probíhá iterativně ve verzích – práce je vykonávána v samostatných blocích, kde výsledkem každého bloku je nějaká funkční změna. V případě týmového projektu probíhá práce na více samostatných částech najednou a je třeba pracovní verze nějak synchronizovat. K tomu se nejčastěji používají distribuované systémy pro sdílení souborů jako je např. GIT.

Unity editor a jeho projektové soubory ale moc dobře nefungují se systémy jako je GIT. Unity vytváří a spravuje ohromné množství dočasných souborů, které jsou důležité pro nastavení prostředí a reference mezi objekty. Všechna tato nastavení prostředí se ukládají do binárního formátu a mohou vzniknout nevyřešitelné merge konflikty nebo ztráty referencí. Celé projekty jsou navíc typicky paměťově náročné, což komplikuje operace.

Jedním z řešení je použití balíčku Github for Unity[35], který tyto problémy řeší a umožňuje integrovanou správu verzování z prostředí Unity.

## 5.4 Návrh herních systémů

Následující navrhované systémy jsou imitace a úpravy systémů ze zanalýzovaných aplikací již existujících řešení.

Jedná se o interaktivní aplikaci. Tato interakce bude probíhat pomocí pohybu po světě a možnosti interakce s předměty pomocí systému držení a manipulování s předměty. Pro toto budeme potřebovat zpracovávat vstupy aplikace (VR) – číst pozice a rotace headsetu a ovladačů, dále také hodnoty ovládacích prvků na ovladačích.

Také budeme potřebovat systém pro pohyb a orientaci ve virtuálním prostředí (typicky se nazývá Locomotion – česky pohyb). Pro zpracování avatara bude třeba vytvořit systém pro jeho ovládání a zpracování relevantních informací. Pro možnost interakce a držení předmětů bude třeba vytvořit systém pro správu a fungování rukou a dále systém pro samotnou logiku držení a manipulace s předměty v rámci fyzikálního enginu. Dále v rámci organizace a použitelnosti předmětů ve hře potřebujeme systém umělého držení předmětů v prostoru. Předměty ve světě se skládají z vytvořených historických a testovacích předmětů, které jsou různorodé s různými účely a tedy bude potřeba mít nějakou strukturu, která definuje jejich vlastnosti a chování.

Pro funkčnost herního prvku budeme potřebovat systém, která bude provozovat konečný automat ovládající akce tréninkového panáka, a dále ovladač, který bude zajišťovat provedení a animace těchto akcí. Pro zajištění chodu herního prvku v prostředí je potřeba systém, který bude herní prvky spravovat – systém „arén“, který bude aktivovat panáky na základě herních událostí, zjišťovat statistiky „tréninku“, předávat informace hráči a spravovat informační panel se statistikami. Pro komunikaci a správu jednotlivých systému potřebujeme ještě správce hráče – systém, který bude zajišťovat komunikaci mezi manažerem inputu a manažery avatara a interakcí rukou, dále komunikace se systémy herních prvků – s manažerem arén a také pro správu post precesssing efektů.

### 5.4.1 Unity XR Plugin a XR Interactions Toolkit

Jedná se o balíčky, který obsahují řadu komponent a systémů pro tvorbu XR[1] aplikací. Jedná se tedy o framework se systémy pro víceplatformní zpracovávání vstupů XR ovladačů skrz Unity Input systém. Nabízí také základní funkčnost vybírání, držení a organizaci předmětů nebo UI elementů a základní možnost pohybu a orientace. Dále nabízí haptickou zpětnou vazbu skrz XR ovladače a také VR Rig s kamerou – objekt, který mapuje reálné pozice headsetu a ovladačů na herní objekty ovladačů a hlavy s kamerou ve virtuálním prostředí.

Tento balíček využiji v projektu jakožto rozhraní pro zpracování vstupů a haptické zpětné vazby. Dále využiji některé komponenty pro tvorbu Locomotion systému – pohybu a orientace. Také

[1] XR = VR a AR

použiji připravený XR Rig jakožto výchozí bod objektu hráče.

## 5.5 Input

Jakožto rozhraní pro získání vstupních hodnot a volání funkcí pro zaslání haptických impulzů používám XR Plugin a jeho struktur XR Controlleru.

Hlavní vstupy, které se zpracovávají.

1. Joystick, touchpad – typicky použito pro ovládání pohybu a orientace nebo navigace v UI. Dalším z užití je pro ovládání animací pohybu avatara. V rozhranní XR controlleru se jedná o tzv. InputFeature primary2Daxis a je definován vektorem o velikosti 2 – $x$ a $y$ složky pozice joysticku.
2. Grip – tlačítko primárně používáno pro ovládání a animování ruky a interakci s předměty. Lze číst jen jako spínač – 0 nebo 1, ale typicky se spíše pracuje s desetinnou hodnotou od 0 do 1, která se více hodí na přechodové animace. V rozhranní XR controlleru se k němu přistupuje pod hodnotou *grip* nebo *gripButton* (spínač).
3. Trigger – další frekventovaně použité tlačítko, hlavně jako aktivátor předmětů, prostředí a UI elementů. Má obdobné vlastnosti jako tlačítko Grip. Lze k němu přistupovat hodnotu *trigger* nebo *triggerButton*.

Výsledný Input Manager spravuje dvě instance vlastních tříd pro přístup k rozhranní XR Controllerů levého a pravého ovladače a na požádání od manažerů rukou nebo manažera hráče vrací momentální hodnoty vstupů nebo zasílá haptický impuls.

## 5.6 Locomotion

Implementovaný systém pohybu a orientace využívá některých komponent a logiky XR Interaction Toolkitu. Důvodem použití tohoto řešení je fakt, že pohyb není hlavním předmětem této aplikace a takto implementovaný a modifikovaný systém je více než dostačující pro vytvořené prostředí.

Obecně základním stavebním kamenem jakékoliv VR aplikace je nějaký VR Rig.

**VR Rig** Objekt, který zpracovává poziční a gyroskopická data připojeného headsetu a ovladačů tak, aby se jejich pozice a rotace namapovaly na herní objekty ve virtuálním prostoru, které reprezentují hlavu(kameru) a ovladače. Tento objekt hlavy tedy slouží jako zdroj kamery, která se používá k vykreslování obrazu z virtuálního prostředí na displeje headsetu.

XR Interaction Toolkitu nabízí takovýto připravený rig, který rovnou po přidání do scény a spuštění funguje a pohyb headsetu a ovladačů se promítá do pohybu příslušných objektů a kamery ve virtuálním prostředí.

Dalšími použitými komponentami[2] jsou:

1. Locomotion System – objekt, který spravuje to, aby Locomotion Providery získávaly přístup k provedení pohybu nebo rotace rigem jen po jednom.
2. Locomotion Providery – komponenty, které implementují nějaký druh pohybu nebo orientace.

[2] z XR Interaction Toolkitu

a. Snap Turn Provider – nakonfigurovaný tak, aby se při pohybu pravého joyticku/touchpadu do stran okamžitě provedl ve stejném směru obrat o 90°.

b. Continuous Move Provider – provider, který zajišťuje plynulí pohyb postavy. Nakonfigurovaný tak, aby reagoval na levý joystick a hýbal se relativně vůči orientaci kamery.

Tento systém je, co se týče namapovaného ovládání a možností pohybu, standardní v rámci aplikací pro virtuální realitu z první osoby. Absence teleportování, jakožto možnosti pohybu, je z důvodu imerze. Teleportace není z principu kompatibilní s imerzivním bojem s ručními zbraněmi.[3]

Dále je použita nativní komponenta Character Controller, která realizuje kolize s collidery ve scéně bez nutnosti fyzikální simulace a také konfigurovatelný vertikální pohyb např. po rampě. Jeho interní collider ale zůstává vždy ve středu objektu rigu a jeho výška neodpovídá výšce kamery.

Ještě jsem tedy vytvořil vlastní skript, který každý Update přepočítá pozici a výšku interního collideru character controlleru na pozici a výšku kamery, aby bylo možné pod překážkami podlézt.

## 5.7 Avatar

Avatar (v tomto případě full body avatar) je model hráčské postavy, který se mapuje tak, aby jeho pozice a póza odpovídaly hráčově v reálném světě a tím se zvyšovala imerze. Existuje zde ale několik problémů. Jelikož ze zařízení virtuální reality dostáváme pouze pozice a rotace hlavy a rukou, musíme pozice sposty částí těla dopočítávat a odhadovat. Tyto odhady mohou způsobit nesrovnalost s pózou uživatele a tím z pohledu uživatele způsobit ztrátu imerze. Příkladem jsou nohy, nemáme absolutně žádné informace o poloze pánve natož o polohách samotných nohou.

Řešením jsou externí zařízení pro sledování jednotlivých částí těla. Ty ale nejsou standartním doplňkem pro zařízení virtuální reality a proto je velké množství aplikací nezpracovává.

V aplikaci používám pro ovládání avatara kombinaci vlastních skriptů, které jsou insipirovány principy z webových článků[36] a komponenty z balíčku Unity Animation Rigging.

Samotný avatar je pozicován každý frame podle polohy hlavy/kamery.

Pro dopočítávání informací o rukou používám tzv. IK – Inverzní kinematiku. Obecně jde o dopočítání prostředního bodu podle znalosti počátku a konce. Samozřejmě takto má prostřední bod nekonečné množství míst kde může být, proto se používá ještě jeden bod, kterému se říká hint, který svojí pozicí „napovídá“ přesné místo středního bodu.

Ve standartním případě se IK používá mezi dvěma spojenými kostmi. Počátek je kořen první kosti, konec je místo, kde má být vrchol druhé kosti a hint je potom nějaký bod v prostoru, který určuje jak zalomený a jakým směrem bude spoj mezi kostmi. Přesně takto funguje i u ruky avatara, kde konec(hand target) je proměnná polohy, kterou v manažeru ruky nastavuji tam, kde chci mít herní ruku avatara. Také používám rotační korekci mezi řetězem kostí ruky, aby se koncová rotace ruky realisticky distribuovala i do ostatních částí ruky.

Pro zobrazení nohou počítám s předpokladem, že tělo je vždy rovné a pánev je tedy fixní vzdálenost od hlavy. Stejně u rukou se dopočítává pozice nohou pomocí IK. V tomto případě se ale konec (target – pozice nohy) zjistí vysláním paprsku směrem dolů a zjištěním nejbližší

[3] samozdřejmě pokud se nejedná o případ, kdy je to jedna z hlavních herních bojových mechanik

„země“. Pro samotný výpočet se používá funkčnosti Animátoru, který provádí tzv. IK Pass a lze mu předat potřebné parametry.

Komponenta Animátor je použit ale hlavně pro animace pohybu a rukou. Unity umí pracovat s tzv. Univerzálními humanoidními avatary tak, že odlišné ale podobné rigy dokáže namapovat na ten univerzální a libovolně mezi nimi používat jejich animace. To využívám pro animace pohybu nohou v různých směrech z webové stránky Mixamo[22]. Také pro vlastní animace sevření rukou vytvořené v Blenderu[19]. Animátor herního avatara je složen z několik vrstev, které jsou oddělené pomocí Avatar masek. Jedna pro každou ruku a jedna vrstva pro nohy. Nohy se animují prolínáním čtyř animací pohybu do různých stran na dvourozměrném grafu podle vektoru vstupu joysticku získaného z Input manažera. Podobně se animují ruce. Podle získané hodnoty vstupu *grip* se prolne mezi pózou otevřené a plně sevřené ruky.

## 5.8 Herní ruce

Herní ruce se skládají ze 3 samostatných vrstev. Vstupní vrstva, Fyzikální a Animační/Vizuální vrstva.

Při různých stavech se projeví jiná vrstva jako zdroj pozice herní ruky. Vstupní vrstva je čtení informací přímo ze vstupu, tedy použití přímé polohy a rotace ovladačů.

Fyzikální vrstva je fyzikální objekt, který je aktivní jen pokud se pozice vstupní vrstvy dostane do kolize s jiným objektem a zabrání, aby ruce vizuálně pronikly do jiného objektu. Také umožňuje manipulovat s předměty samotným pohybem ruky.

Animační/Vizuální vrstva rukou je aktivní, když je držen nějaký předmět, pozice a póza ruky je v tom případě na objektu. Samotné virtuální ruce jsou tvořeny pomocí avatara, kde části kostry, které odpovídají rukám, se snažím pozicovat tak, aby byly v pozici některé z vrstev aplikačních rukou.

## 5.9 Manažer interakcí ruky

Cílem systému je zpracovávat možné interakce ruky s předměty v prostředí. Spravuje tedy sebrání a pouštění předmětů, vytváření a zobrazování indikátorů a informačních objektů. Také aktualizuje data pro avatara pro animační vrstvu rukou a v případě držení předmětu aktualizuje manažera drženého předmětu.

Každá ruka má tedy jednu instanci manažera. Systém pracuje s funkcemi input manažera, aby zjistil, zda je jím spravovaná ruka ve světě aktivní. Pokud ano, tak provádí jednoduchý logický loop. Pokud ruka nedrží předmět, tak sleduje blízké okolí a hledá předměty, které by se daly držet. Pokud některé najde, tak u nejbližšího z nich vytvoří jakési UI v podobě indikátoru pro sebrání a info panel s informacemi o předmětu. Pokud v tuto chvíli uživatel podrží grip button, předmět se sbírá a spravuje pomocí manažera drženého předmětu, který je aktivován a aktualizován z tohoto manažera. Pokud je držen a přestane být grip button input, předmět se pouští.

Také spravuje interakce se sockety, tedy pokud je nedržící ruka poblíž socketu s předmětem, stane se z něj také kandidát na sebrání. Stejně tak když hráč pouští předmět a nachází se poblíž socketu, postará se manažer o transfer předmětu do držení socketu.

Nachází se zde ovládací prvky, které dovolí předmět sebrat pouze v momentě stisknutí grab buttonu, tedy pokud uživatel zatne ruku a poblíž se nenachází žádný předmět, následné přiblížení

se k validnímu sebratelnému předmětu se sepnutým grip buttonem ho nesebere.

Oba manažery rukou navzájem komunikují, aby se dali zpracovávat případy, kdy se předmět dá držet pomocí dvou rukou nebo když by se měl držený předmět prohodit z jedné ruky do druhé. Při sebrání předmětu se zkontroluje, jestli je tento předmět už držený. Pokud je to předmět, který se dá držet pomocí obou rukou, tak se upraví parametry v manažeru drženého předmětu a drží se obouručně. Pokud lze předmět držet jen jednou rukou, tak se předmět v druhé ruce pustí a sebere se v nové ruce.

Jelikož manažer ruky je autorita co se aktivovanosti ruky a jejího chování týče, tak spravuje i animační stránku avatara. Aktualizuje a předává informace manažeru avatara, aby byly vizuální ruce na správném místě.


**Obrázek 5.1** Ukázka systému interakcí v aplikaci.

Na obrázku 5.1 je znázorněna funkcionatila systému manažerů rukou. Manažer levé ruky drží předmět (meč Kopis) a má tedy pozitivní grip button input. Manažer pravé ruky předmět nedrží a ani nemá grip button input. Našel ale v okolí předmět (přilbu), který lze sebrat a tak u něj vytvořil indikátor místa držení a info panel s informacemi o předmětu.

Pseudokód logického loopu manažera interakcí rukou:

```
public void Update()
{
    //check if controller device is active and accessible
    if(inputManager.DeviceIsValid())
    {
        // check if controller grip button is pressed
        if(inputManager.GetInputGripButton())
        {
            // grip button is pressed and an item is held -> update held item
            if(heldItemManager.IsHoldingItem())
            {
                // update items position, rotation a behaviour
                heldItemManager.UpdateHeldItem();
            }
```

```
            // grip button is pressed and hand is empty -> check for nearby
            // grabable items and grab the closest one
            else
            {
                // check if there is a selected nearby item and if the hand
                // can currently grab an item.
                if(CanGrabItem())
                {
                    // Grab nearby selected item into hand
                    Grab();
                }
            }
        }
        // controller grip button is NOT pressed
        else
        {
            // check if hand is holding any item
            if(heldItemManager.IsHoldingItem())
            {
                // hand is no longer sqeezed -> Drop item from hand
                Drop();
            }
            // controller grip button is NOT pressed and hand is NOT holding
            // any item -> find a potential object to grab and display some
            // info about it
            else
            {
                // find nearby items (sockets or free standing items) and mark
                // the closest one as a selected item
                ScanForGrabbableItems();
                // update position, data and animation of an infographic object
                // and a grab position indicator
                UpdateSelectedItemInfo();
            }
        }
        // Update animation hand targets for avatar hand manager
        heldItemManager.UpdateHands();
    }
}
```

## 5.10 Interaktivní předmět

Objekt, který pro držený předmět definuje jeho vlastnosti. Jedná se o klasifikaci objektu, definování možností uchopení (která ruka?, obouruční držení?), váha, parametry pro filtrování pohybu, parametry pro haptickou zpětnou vazbu, parametry držení předmětu – místa uchopení předmětu, konstanty pro spoj držící předmět, ofset transformu kontroleru oproti předmětu. Dále také parametry pro různé stavy držení a chování, funkce zpracovávající callbacky kolizí fyzikálního systému a delegát, pomocí kterého může objekt spravující držený předmět zpracovávat jeho kolize.

Pseudokód podoby parametrů a funkcí skriptu definujícího interaktivní předmět:

```
public class InteractableItem
{
    public enum ItemType { Generic, Weapon, Shield,
                           EquippableHead, EquippableBody }

    public enum HandlingType { Ambidextrous, LeftOnly, RightOnly }

    // item info
    public string itemName;
    public string itemTypeName;
    public string itemSetName;

    // double handed item params
    public bool doubleHanded = false;
    public float doubleHandedSecondaryInfluence = 0.5f;

    // item grip params
    public Transform primaryGrip;
    public Transform secondaryGrip;

    // item to hand alignment offset
    public Vector3 itemHandAlignment = Vector3.zero;

    // filtering params
    ...

    // physics joint params
    ...

    // haptics params
    ...

    // additional physical mechanics params
    ...

    // other
    ...

    // collision callback delegation functions
    public delegate void OnCollisionDelegate(Collision collision);
    public OnCollisionDelegate onCollisionDelegate;

    // on collision callback
    private void OnCollision(Collision collision)
    {
        onCollisionDelegate(collision);
    }
}
```

Herní předměty mají definované chování jen pomocí parametrů a jednotlivé poddruhy

předmětů sami o sobě žádnou speciální funkčnost neposkytují[4] a tak jsem si mohl dovolit takto neobjektový přístup.

Tedy takovýto herní objekt se skládá z meshe (např. ty vytvořené v předchozí kapitole), ručně vytvořených colliderů, které odpovídají tvaru objektu, tohoto InteractibleItem skriptu, který definuje jejich chování manažeru drženého předmětu a ještě Rigidbody komponentu. Tato komponenta zapojuje předmět do aktivních fyzikálních simulací. Její parametry jsou nastaveny následovně:

Hmotnost je nastavena individuálně napříč předměty. Gravitace je pro všechny předměty povolena a žádný není implicitně kinematický. Interpolační mód je na Extrapolate, ale teorieticky by neměl moc ovlivňovat chování, jelikož díky nastavení by měl za každý frame proběhnout jeden fixed update a fyzikální data pro každý update by měla být dostatečně čerstvá. Detekce kolizí je nastavena na Continuous Dynamic, což je nejlepší nastavení pro kolize aktivních colliderů s jinými aktivními collidery.

## 5.11 Správa drženého předmětu

Systém, který obstarává logiku chování držených předmětů v různých stavech a při kolizích.

Tento systém také zpracovává samotnou logiku interakce s předměty, kterou volá manažer rukou:

1. Sebrání předmětu – při kterém vytvoří a nakonfiguruje potřebné komponenty pro fyzikální držení a zaznamená důležité reference k drženému předmětu. Také zpracovává případy, kdy by se sebraný předmět náhle držel obouručně a v tom případě nastaví potřebné příznaky a získá informace o drženém předmětu od manažera druhé ruky. Stejně tak v případě, kdy přebírá předmět z druhé ruky, se zprostředkuje předání informací a sekvence upuštění v jedné ruce a sebrání v druhé.
2. Upuštění předmětu – kdy odstraní reference a fyzikální spoje a přenechá předmětu veškerou kinetickou energii, aby bylo možné předměty házet. V případě, že je předmět držen obouručně a v primární ruce se pouští, tak systém přesune zodpovědnost-primárnost držení na druhou ruku a předá jí všechny potřebné informace, reference a delegáty.

V normálním stavu držení může systém držet předměty několika způsoby v závislosti na jejich typu.

1. Pokud se drží obyčejný předmět nebo zbraň pomocí jedné ruky, tak je update předmětu jen prosté přesunutí a orotování předmětu na polohu vstupní vrstvy rukou - ovladače. Výsledná poloha drženého předmětu je ale ještě ovlivněna filtrováním (následující sekce), které je provedeno mezi těmito dvěma body.
2. V případě obouručního držení zpracovává logiku primární držící ruka. Při držení a používání nástrojů pomocí dvou rukou v reálném světě je ovládání předmětu dáno hlavně dominantní rukou, která určuje rytmus a směr pohybu předmětu a druhá, pomocná, ruka slouží hlavně pro stabilizaci nebo protipohyb za účelem zrychlení pohybu vrcholu (příkladem je práce s krumpáčem, demoličním kladivem nebo lopatou). Obdobně je zpracováno obouruční držení. Pozice a rotace osy zbraně je určena primární drženou rukou. Celková rotace zbraně je určena prolnutím rotace primární ruky a vektorem mezi sekundární a primární rukou na základě parametru vlivu sekundární ruky. V reálném světě má sekundární ruka stoprocentní vliv na předmět, ale při virtuálním držení není žádná fyzická spojnice, která by obě ruce udržela ve správné pozici na předmětu a proto používám jen částečný vliv, aby byl výsledný předmět stabilnější.

[4] až na malou vyjímku v případě umělé mechaniky zaseknutí

```
// vector from secondary hand to primary hand
Vector3 secToPrimVec = itemPosition - otherHand.position;
// rotation of vector direction
Quaternion rawRot = LookRotation(secToPrimVec, itemTransform.up);
// apply influence param on final rotation
Quaternion doubleHandRot = Lerp(itemRotation, rawRot, secHandInfluence);
```

Dále je ještě upravena rotace vizuální stránky sekundární ruky, aby odpovídala směru ze kterého předmět drží. To je docíleno projekcí předového vektoru ruky na horizontální rovinu rukojeťi předmětu v místě držení.

3. Posledním případem speciální logiky držení předmětu je v případě tzv. připoutaného štítu. V aplikaci je to Hoplon/Aspis. Jedná se o štít, který je k ruce připoután v oblasti lokte a zároveň je držen za držadlo na kraji štítu. Důvodem tohoto dvoubodového uchopení je rozložení váhy a znemožňuje nepříteli manipulovat se štítem pomocí tlaku na okraje, jako je tomu u štítů se středovým držením, zároveň ale tento způsob ztrácí něco do rozsahu pohybu a nelze použít až tak efektivně k ofenzivním manévrům.

   Držadlo štítu je pozice, podle které se v aplikaci štít drží. Aby byl štít připoután i k předloktí, musí manažer požádat skrze manažera hráče manažera avatara o pozici lokte, aby mohl vypočítat správné natočení štítu v ruce.

   Z hlediska výpočtu je princip podobný předchozímu výpočtu držení předmětu pomocí dvou rukou.

V aplikaci je ještě jeden samostatný systém pro držení předmětů. V textu ho označuji pod názvem Socket. Jedná se o jakousi jednomístnou zásuvku, která je schopna volně držet předmět v prostoru. V aplikacích virtuální reality je takovýto obdobný systém použit pro inventáře a držáky výstroje. Jeho použití v této aplikaci je obdobné.

Používám tyto objekty jako držák pro helmy na hlavně a jako pochvu pro zbraň na levém boku avatara. Také je v prototypovém domě použit jako držák pro helmu z multiplayerového prototypu. Tento objekt tedy umí na vyžádání převzít a držet předmět nebo držený předmět upustit.

## 5.12 Filtrování pohybů

Cílem systému držení předmětů je, aby byly předměty pokud možno co nejresponzivnější, aby vizuální stránka co možná nejvíce odpovídala reálné poloze ovladače, tedy aby uživatel přibližně věděl, kde se jeho virtuální ruka nachází i bez nutnosti vizuálního kontaktu. Paradoxně to tedy teď překazíme tak, že budeme pohyby ruky filtrovat a redukovat tak, abychom trochu simulovali váhu.

V existujících řešení, které jsem analyzoval v kapitole Existující řešení, se typicky používá metoda, kde se držené předměty "odpojí" a opožděně se táhnout za rukou. To z mé zkušenosti vede na jakýsi konflikt smyslů, kdy se naše fyzická ruka nachází jinde než ruka virtuální. Ideálním řešením by bylo změnit uměle váhu a těžiště drženého ovladače a tedy virtuální váhu simulovat fyzicky, ale logicky to není něco, co by bylo momentálně proveditelné a asi ani bezpečné. Myšlenkou této metody je tedy simulovat váhu a setrvačnost zbraně. Problém je, že historické meče a další zbraně nebyly vůbec nijak přehnaně těžké, jak by se z filmů, seriálů a obecně médií zdálo. Trénovaný "bojovník" je schopen hýbat (se správnou technikou) se zbraní skoro stejně rychle a obratně, jako kdyby žádnou zbraň nedržel.

Proto nechám pohyb co nejresponzivnější. Místo toho bude moje filtrování zaměřeno převážně na rotace a to specificky na osách rotace, které jsou mechanciky na ruce slabé, aby se předešlo rychlému máchání, které by bylo s větším a těžším objektem nemožné. Filtracní parametry jsou pro každý předmět individuálně odlišné. Navíc pokud se jedná o předmět, který se typicky používá pomocí dvou rukou — předmět který je těžší nebo benefituje z dvou bodů kontaktu pro zrychlení a lepší charakteristiky (např. kopí, dlouhé obouruční meče atd), tak pokud ho držíme jen jednou rukou, filtrace je mnohem "agresivnější" než když ho držíme pomocí dvou.

Jedná se o lineární filtrování se stropem. Funkce přečte momentální hodnoty a pokud přesahují osový limit, velikost pohybu se ořízne na hodnotu limitu. Tímto způsobem bude předmět pořád co možná nejrespozivnější na malé změny, ale omezí se maximální rychlost pohybu na některých osách. Nepoužívám tedy realističtější simulace, kde by filtrování působilo hlavně při akceleraci a deceleraci, protože ta by způsobila, že je držený objekt opožděný za pozicí rukou a vizuálně působí loudavě a zpomaleně.

Vstupem do filtrovací funkce jsou dva transformy[5]. Momentální pozice předmětu a pozice ovladače (tedy tam, kde by měl po aktualizaci předmět být).

V případě omezení úhlové rychlosti získám rotaci, která mezi těmito pozicemi je a tu rozdělím podle jednotlivých os. Tedy v podání eulerových úhlů získám úhly Pitch, Roll a Yaw. Vzhledem k definované orientaci předmětu omezuji rotace na ose y (roll) a na ose z (yaw). Omezení pohybu probíhá pomocí funkce lineární interpolace tak, že se interpoluje mezi momentální rotací předmětu a rotací ovladače s parametrem $t$, který se rovná limit rotace osy/ úhel rotace na ose. ($t$ je omezeno v rozsahu [0-1])

```
newRotation = Lerp(itemRotation, targetRotation, angleLimit / angle);
```

Další zpracované fitrování je zjemnění pohybu při změně stavu (při sebrání předmětu do ruky, nebo ze zaseknutého stavu do normálního držení – přechod předmětu z jedné pozice do pozice ruky plynulím způsobem). Tato funkce je omezena časově a vždy trvá dobu stanovenou parametrem *duration*. Pro provedení samotného pohybu je použita funkce lineární interpolace, ale použitý parametr $t$ dává funkci spíše podobu konkávy, co se rychlosti pohybu předmětu týče, protože parametr počátku interpolace se v průběhu pořád mění na aktuální pozici předmětu.

```
newLocation = Lerp(currentItemLocation, handLocation, elapsedTime / duration);
```

## 5.13 Fyzikální mechaniky

Fyzikální mechaniky spočívají ve využití fyzikálního enginu a jeho součástí k dosáhnutí požadovaného chování.

### 5.13.1 Úvod a konfigurace

Nejprve je třeba fyzikální engine nakonfigurovat. V tomto případě se nejedná o nic komplikovaného. PhysX je v základním nastavení velmi schopný engine. Pro mé účely potřebuji obětovat trochu výkonu za přesnost. Fyzikální simulace neprobíhají ve stejném rytmu jako zbytek herního loopu, který se typicky orientuje podle toho, jak rychle se stíhají renderovat jednotlivé framy.

[5] komponenta držící informace o pozici, rotaci, škále atd. objektu ve světě

Fyzická simulace tedy naopak probíhá v pravidelném rytmu[6], který je definován parametrem Fixed Timestep (defaultně se jedná 50 updatů za vteřinu).

Pro účely aplikace pro vr je „doporučený postup“ použít stejnou frekvenci jako je obnovovací frekvence cílového zařízení. V mém případě se jedná o zařízení HP Reverb G2, které má obnovovací frekvenci 90Hz (standartní frekvence pro většinu zařízení). Fixed timestep je definován jako prodleva mezi jednotlivými updaty fyziky. Tedy nová, modifikovaná, hodnota je $1/refresh\ rate = 1/90 = 0.011111$.

Další modifikací je zvýšení počtu iterací řešitele fyzikálních událostí. Řešitel je funkce, co iterativně aproximuje výsledek nějaké fyzikální interakce, výsledek jedné iterace se použije jako vstup do další a zvyšuje se přesnost dané aproximace. Čím více iterací, tím přesnější je výsledek, ale zároveň je i větší výkonostní zátěž. Tato aplikace je zaměřena na fyzikální interakce, tedy základní hodnota 6 by v některých případech nemusela být dostačující.

### 5.13.2 Použití

Pro základní kontrolu nad kolizemi a snížení množství nepotřebných kolizí ve scéně jsou všechny předměty rozděleny do různých fyzikálních vrstev. Tyto vrstvy jsou navzájem nastaveny tak, aby podle typu spolu buďto kolidovaly a nebo kolize ignorovaly.

Obecná „fyzikálnost“ předmětů je zajištěna pomocí komponenty Rigidbody, která předmět zapojí do fyzikálních simulací a bude na něj aplikovat všechny vnější síly jako např. gravita a síly vytvořené uměle pomocí volání funkcí. Tato komponenta má pár důležitých parametrů.

1. Hmotnost – určuje hmotnost předmětu a tedyi sílu/enerigii, kterou je třeba na rozpohybování předmětu.
2. Gravitace – je to bool, který značí jestli bude na předmět aplikována gravitační síla (její velikost a směr je konfigurovatelná).
3. Kinematický objekt – kinematický objekt nereaguje na žádné vnější síly a je kontrolován jen pomocí skriptů a animací. Pořád ale ovlivňuje ostatní fyzické předměty.
4. Interpolace – mód interpolace, který se použije pro získání pozice fyzikálního objektu v případě, kdy Update nemá čerstvá data z fyzikální simulace ve Fixed Updatu, tedy když musí hádat, jak vlastně fyzikální simulace dopadla/dopadne. Vzhledem k tomu, že aplikace má nakonfigurován stejný počet fyzikálních updatů za vteřinu jako je její obnovovací frekvence - cílový počet snímků za vteřinu. Tedy simulace by měly být dostatečně čerstvé pro každý snímek. I tak ale pro případ výpadků používám extrapolaci.
5. Typ detekce kolize – pro případ aktivních colliderů, které budou kolidovat s jinými aktivními collidery je nejvhodnější režim Continuous Dynamic. Aktivní collider je takový collider, který se po scéně hýbe.

Jediným rozumným způsobem jak v tomto fyzikálním enginu spojit dva fyzikální objekty nebo ovlivňovat jejich vzájemné chování je pomocí tzv. Spoje[7]. Hlavním cílem spoje je udržet spojené předměty u sebe. Místo spoje je definováno argumentem Anchor a connectedAnchor, jedná se o pozice v lokálním prostoru každého s připojených objektů. Spoj tedy aplikuje síly na oba předměty tak, aby pozice Anchor bylo stejná jako connectedAnchor. K tomu používají tzv. „pohony“, pro každou osu[8] jeden a jeden pro rotační pohyby. Každý takovýto pohon má atribut pro

[6] za předpokladu, že stíháme generovat dostatečný počet snímků za vteřinu

[7] popisuji komponentu Konfigurovatelný spoj. Všechny ostatní typy spojů jsou jen omezenými speciálními případy tohoto konfigurovatelného spoje

[8] osy prostoru – $x$, $y$ a $z$

sílu kterou může ve své ose použít pro udržení spojení a atribut pro tlumení aplikované síly. Spoj dále umožnůje limitovat vzdálenost a rotace na jednotlivých osách, popřípadě úplně některé osy pohybu a rotace zamknout a žádný pohyb nedovolit.[37]

Systém držení předmětu fyzikálním způsobem se skládá ze dvou částí. Prvním je samotný držený objekt, který je tedy rigidbody. Tím druhým je neviditelný objekt, jakési Zápěstí[9]. Jedná se také o fyzikální objekt, který je ale kinematický. Toto zápěstí je připojeno k předmětu právě pomocí Spoje. Tedy předmět se bude (z definice spoje) snažit dostat svůj Anchor k Anchoru Zápěstí a bude podle toho na něj aplikována síla. Zápěstí, díky tomu že je kinematické, bude ignorovat veškeré síly které na něj spoj aplikuje aby dostal svůj Anchor k Anchoru předmětu.

Efektivně tedy Zápěstí ovládá pozici předmětu. Pokud se Zápěstí pohne pomocí skriptu, předmět se také pohne aby se jejich anchory znovu „spojily“. Tato korigující síla ale není nekonečná a pokud se v cestě mezi Zápěstím a drženým předmětem objeví jiný statický fyzikální objekt, předmět se o něj zastaví a nebude moci pokračovat. Síly a vlastnosti se konfigurují pomocí atributů spoje. Ten je v tomto případě nakonfigurován tak, aby se držený předmět mohl pohybovat se šesti stupněmi volnosti (6dof). Síly aplikované pro korekci polohy a rotace jsou dostatečně velké, aby byl pohyb předmětu k zápěstí dostatečně responzivní a přímý. Zároveň ale nejsou všechny stejné, síly aplikované na korekci polohy jsou větší než síla aplikovaná na korekci rotace. Tedy v případě že předmět narazí na jiný, silnější, fyzikální předmět, jeho rotace ustoupí, ale pozice se spíše zachová. Tzn. při dvou stejných držených předmětech bude jeden schopný přetlačit ten druhý např. použitím zákonů páky.

Při se sebrání předmětu se na jeho *gripPosition* pozici vytvoří nový objekt Zápěstí se Spoj do kterého se jakožto spojený předmět předá sebraný předmět. Z InteractableItem skriptu sebraného předmětu se získají atributy sílý a tlumení, které se do Spoje zapíší.

Při upuštění předmětu se objekt Zápěstí a komponenta spoje odstraní a předmět tak bude pokračovat ve svém pohybovém momentu.

Spoje dále využívám pro tréninkové panáky, aby se chovali jako boxovací pytle a vraceli vždy do vspřímené polohy. Také pro dveře prototypového baráku, kde se Spoj chová jako pant a umožňuje rotaci jen kolem vertikální osy v jednom směru a jen o 90°.

## 5.14 Umělé fyzikální mechaniky - manipulace s enginem

Následující mechaniky, které ovlivňují chování předmětů, jsou tzv. umělé. Tedy nejsou výsledkem simulace fyzikálního enginu, protože ten je není schopen sám o sobě provést, ale pro jejich fungování je třeba chování tohoto fyzikálního enginu v některých případech modifikovat.

### 5.14.1 Zaseknutí

Cílem této mechaniky je přidat ostrým objektům při zásahu pocit velké síly/energie. Objekt se při prudkém zásahu pod kolmým úhlem do cílového objektu zasekne. Pokud je v této zaseknuté fázy puštěn, zůstane zaseknutý a nebude na něj aplikovaná žádná gravitační síla. Odseknout lze znovusebráním nebo dostatečnou silou způsobenou jiným předmětem. Pokud ale zaseknutý objekt je pořád držen, jde odseknout dostatečně velkým pohybem držící ruky.

[9] nejlepší analogie na kterou jsem zde přišel

Aktivuje se, pokud nějakou zbraní, která je definována jako ostrá, velkou silou (s velkou rychlostí) a pod kolmým úhlem dopadu (osa čepele zbraně je kolmá na povrch dopadu) zasáhneme nějaký vhodný objekt (např. dřevěné prkno). Následně probíhá aplikování této mechaniky v několika fázích.

1. Nejprve se při aktivaci změní fyzikální chování spoje mezi zápěstím a zbraní tak, aby neměla žádný rotační pohyb a logika držení předmětu omezuje pohyb jen na rovinu místa objektu zásahu.
2. Po krátkou dobu se měří vzdálenost, kterou reálná ruka s ovladačem od zaseknuté zbraně urazí a tím se tedy určí jakási hloubka zaseknutí.
3. Z hloubky zaseknutí se vypočítá vzdálenost, kterou musí ruka urazit nebo do které se musí zpět k rukojeti přiblížit, aby se ze zasekunutí zbraň vyprostila.

V případě že předmět upustíme aniž by se vyprostil ze zaseknutého stavu, přechází do tzv. volného zaseknutí. Rigidbody předmětu se nastaví jako Kinematický, tedy jeho fyzikální simulace reaguje jen na účelnou aplikaci sil pomocí funkcí. A také se v hierarchii stává potomkem objektu do kterého se zasekl, aby jeho pozice a rotace zůstali stejné vůči objektu i při jeho pohybu.

Při znovusebrání volně zaseknuté zbraně se ze zaseknutí vyprostí a vrací se do normálního stavu držení.

Tato mechanika funguje i v dalším případě, kdy předmět nedržíme, ale vrhneme ho – když předmět koliduje s jiným objektem, ale není držen v ruce. Aktivace je zase podmíněna rychlostí a správným úhlem mezi čepelí a povrchem. V tomto případě ale předmět aplikuje impulzivní sílu na základě rychlosti na předmět v místě dopadu a rovnou přechází do volně zaseknutého stavu.

To tedy znamená že logiku volného zaseknutí a odseknutí zpracovává samotná zbraň a ne manažer držené zbraně.

### 5.14.2 Bodání

Cílem mechaniky je umožnit zbraním s ostrou špičkou probodnout vhodný materiál a manipulovat s ním.

Mechanika se aktivuje, pokud se ostrou zbraní bodne do objektu se speciálním materiálem a špička je pod vhodně kolmým úhlem k povrchu. V tu chvíli je čepel schopna projít skrz nabodnutý objekt a ovlivňovat jej svými pohyby. Pokud se čepel vytáhne z předmětu, zabodnutí končí a zbraň se vrací do normálního stavu držení.

Pokud hráč upustí zbraň ve chvíli, kdy probodává jiný předmět, zbraň přechází, podobně jako u mechaniky zasekávání, do stavu volného zabodnutí, kdy se stává kinematickým rigidbody a potomkem nabodnutého předmětu. Pokud se takto volně zabodnutá zbraň sebere, zbraň se z předmětu vytahuje a přechází do normálního stavu držení.

Pro funkčnost je nutný další pomocných objekt. Všechny zbraně, které jsou schopny bodat, potřebují nový fyzický herní objekt s minimální váhou, který ignoruje kolize s čepelí a který je pomocí volného spoje připojen ke zbrani. Anchor jejich spoje je ve špičce čepele a pomocí zamykání přebytečných os pohybu a lineárního limitu je pohyb předmětu omezen pouze na osu a délku čepele.

Tento objekt působí zároveň jako indikátor bodnutí do jiného předmětu a jako médium, které bude spojem připojeno i k zabodnutému předmětu a bude s ním manipulovat.

Mechanika funguje následovně:

1. Při aktivaci nabodnutí se začnou ignorovat kolize mezi collidery čepele a nabodnutým předmětem. V tu chvíli je indikátor bodnutí na špici čeple a je tlačen nabodnutým předmětem.
2. Vytvoří se nový volný spoj na objektu indikátoru bodnutí s nabodnutým objektem. V tu chvíli se pozmění hodnoty síly pružiny, škál hmotnosti a hmotnosti připojeného objektu na spoji mezi zbraní a indikátorem za účelem zlehčení manipulace se zabodnutým předmětem pomocí držené zbraně.
3. Při pohybu indikátoru bodnutí s předmětem po čepeli se simuluje odpor aplikováním síly v opačném směru.
4. Pokud se indikátor bodnutí s nabodnutým předmětem vzdálí od hrotu čepele (zbraň je tedy z předmětu vytažena) tak se spoj ruší a mechanika zabodnutí končí.

V aplikaci byly logicky vhodné, jakožto možné cíle pro zabodnutí, jen objekty tréninkových panáků a prototypový tréninkoví panáci a jejich části z prototypového domu.

**Obrázek 5.2** Ukázka mechaniky zabodnutí

## 5.15 Vizuální zpětná vazba

V aplikace se nachází speciální[10] forma vizuální zpětné vazby v případě kolize zbraní. Jedná se o vizuální efekt zajiskření, což je jen přehrání particle effectu vytvořeného pomocí Unity Particle Systému. Tento efekt se přehraje v místě kolize orientován ve směru normály povrchu.

[10] Samotný obraz prostředí promítaný do brýlí virtuální reality je forma vizuální zpětné vazby

## 5.16 Fyzická zpětná vazba - Haptics

Zpětná vazba dotykem. V ovladačích virtuální a rozšířené reality je typicky v oblasti držadla součástka složena z nějakého vibračního aktuátoru / motoru, která je schopna získaný elektrický signál přeměnit na vibrace nebo mechanické impulzy, které uživatel cítí jako vibraci. Různá zařízení mají různé rozsahy amplitud vibrací.

Pro správu haptické aktivace používám již vytvořený Input Manager, který spravuje přístup k vstupům a funkcím objektů ovladačů. XR Interactions Toolkit umožňuje zasílat haptické signály ovladači pouze pokud se k ovladači přistoupí jako k XR ovladači s haptickými schopnostmi. Funkce má dva hlavní parametry a to Amplituda $x$ a dobu trvání $y$. Tedy po zavolání funkce se spustí v ovladači vibrace o amplitudě $x$ po dobu $y$ vteřin.

Haptickou zpětnou vazbu používám v aplikaci v několika případech:

1. Při kolizích:
   a. Když hráč s drženým předmětem koliduje s jiným předmětem nebo prostředím. Krátký impuls o amplitudě v rozsahu střední až vysoká, která je závislá na rychlosti a síle dopadu.
   b. Při tření. Když hráč s drženým předmětem kontinuálně koliduje s jiným objektem. Vibrace trvá tak dlouho, dokud je nějaký pohyb a předměty kolidují. Amplituda je jednotná na nižší úrovni
2. Při rychlých pohybech:

   Pokud hráč s předmětem pohne vysokou rychlostí (jde i o úhlovou rychlost, takže např. švihne mečem), tak dostane zpětnou vazbu o amplitudě ve středním rozsahu na základě rychlosti a omezení, která jsou individuálně nastavena pro každý předmět. Cílem je předat uživateli informaci o rychlosti, s jakou danou zbraní manipuluje a také jistou senzaci z tohoto rychlého pohybu.
3. Při interakci s objekty:

   V případě, kdy ruka sebere nějaký předmět, uživatel dostane krátkou vibraci o malé intenzitě. Účelem je předat potvrzení uživateli o sebrání předmětu .

## 5.17 Herní prvek - trénovací automaty

Ve scéně se nachází dva druhy herních prvků. Jeden pasivní, kdy se primárně jedná o neaktivní trénovací panáky, které mají díky použitým fyzikálním komponentám a mechanikám chování boxovacího pytle a jsou ideální na vyzkoušení všech zbraní a vybavení. Aktivní herní prvky jsou dvě arény s aktivními trénovacími panáky. Jedna je zaměřena na obranu a druhá na útok. Oba jsou řízeni tím samým stavovým automatem, každý má akorát nastavené jiné priority.

Cílem aktivního útočného trénovacího panáka je imitovat techniky boje s mečem, napadat hráče a nutit ho se bránit. Tyto techniky začínají v nějakém gardu[11] a končí v jiném gardu nebo v bodu výpadu. Pro stížení bude moci automat podnikat finty. Finta je v principu zmatení nepřítele přesvědčením ho o použití nějaké techniky, i když se ve skutečnosti použije jiná technika. To znamená rychlá změna gardu před provedením útoku. Cílem tréninku pro hráče je správně přečíst směr a možnou techniku, a použít vlastní vhodnou techniku, případně včas zareagovat na fintu.

[11] gard = pozice zbraně, která buď pasivně brání některé z kvadrantů, je to výchozí, přechodová nebo koncová pozice techiniky a nebo obojí naráz.

Trénink zároveň ukazuje světlé a stinné stránky jednotlivých zbraní a vybavení např. helma obecně uchrání hlavu uživatele, ale se zvýšenou ochranou je zhoršená viditelnost (v reálném světě helmy zhoršují i sluch a pokud jsou kompletně uzavřené tak i dýchání).

Cílem aktivního obranného trénovacího panáka je bránit se před hráčovými útoky a přinutit ho vymyslet nějakou fintu. Akce obrany pro panáka spočívá v umístění obraného předmětu (štítu) mezi střed čepele zbraně a místo možného zásahu, pokud by zbraň pokračovala ve stejné výšce. Tento trénink je tedy hlavně o donucení hráče ke kreativitě a celkově je spíše doplňkem k hlavnímu tréninkovému prvku aplikace, což je výše zmíněný útočný trénovací panák.

### 5.17.1 Návrh automatu

Skript ovládající akce trénovacího panáka je vytvořen a ovládán podle logiky stavového automatu. Jeho chování tedy lze zadefinovat pomocí konečného automatu.

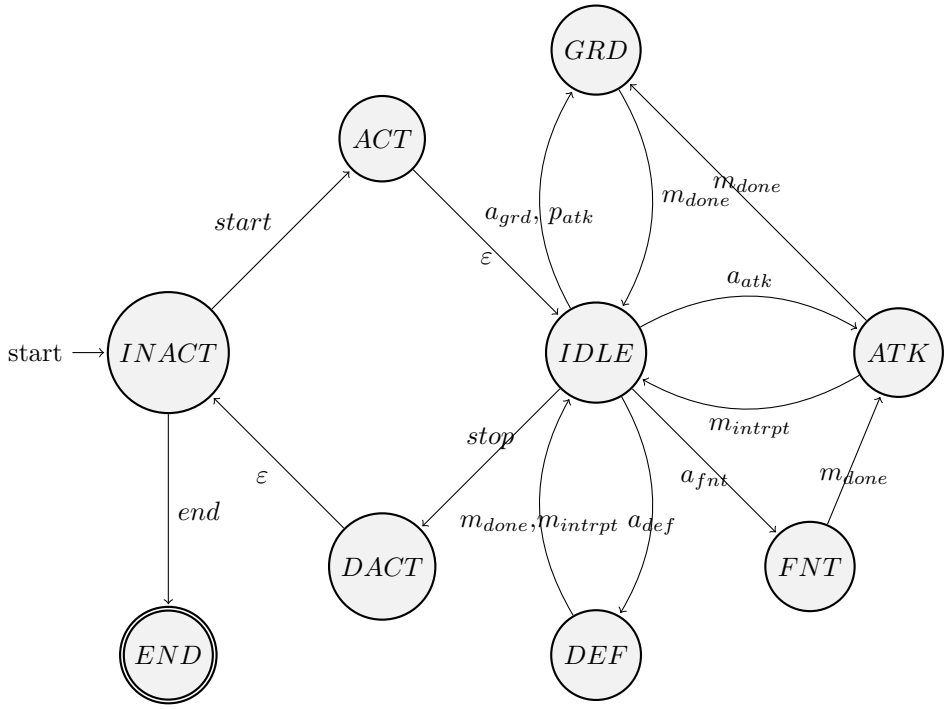
Jeho stavy představují aktivity, které automat vykonává. Abeceda, kterou se automat řídí představuje vstupy do automatu, jeho řídící logiku a časové a herní události.

$M = (\{INACT, ACT, DACT, END, IDLE, GRD, ATK, FNT, DEF\},$
$\{start, stop, end, a_{atk}, a_{fnt}, a_{grd}, a_{def}, p_{atk}, m_{done}, m_{intrpt}\}, \delta, INACT, \{END\})$

| Množina stavů | Význam v aplikaci |
|---|---|
| Inactive(INACT) | Neaktivní panák, čeká na signál *start* |
| Activate(ACT) | „Fyzická“ animace aktivování panáka |
| Deactivate(DACT) | „Fyzická“ animace deaktivování panáka |
| End(END) | Ukončení aplikace |
| Idle(IDLE) | Čekání a zvolení následující akce |
| Attack(ATK) | Stav kdy probíhá útok. Čeká se na příznaky úspěchu/neúspěchu |
| Defend(DEF) | Stav kdy probíhá pohyb do obranné pozice. Čeká se na příznaky |
| Guard(GRD) | Pohyb do nového gardu. Získání příznaku „může zaútočit“ |
| Feint(FNT) | Provedení finty, zrychlený přesun do nového gardu a okamžitý útok |

| Abeceda | Význam v aplikaci |
|---|---|
| *start* | Hráč vstoupil do oblasti tréninkové arény |
| *stop* | Hráč opustil oblast nebo vypršel čas tréninku |
| *end* | Signál ukončení aplikace |
| Action-Attack($a_{atk}$) | Vybrána akce útok – zvolení cíle a začátek pohybu |
| Action-Defend($a_{def}$) | Vybrána akce obrana – nalezení a pohyb do místa obrany |
| Action-Feint($a_{fnt}$) | Vybrána akce finta – zvolení nového gardu a rychlí přesun |
| Action-Guard($a_{grd}$) | Vybrána akce změna gardu – zvolení a pohyb do nového gardu |
| Prepare-Attack($p_{atk}$) | Není získán příznak „může útočit“, ale je v úmyslu zaútočit |
| Move-Done($m_{done}$) | Kontroler ruky vrátil příznak, že pohyb proběhl vpořádku |
| Move-Interrupt($m_{intrpt}$) | Kontroler ruky vrátil příznak, že při pohybu došlo ke kolizi |

$\delta$ :




Podobu jazyka, který autom příjmá, lze možná snadněji přiblížit na regulárním výrazu:

$$V = (start(((a_{grd} + p_{atk})m_{done})^* + (a_{atk}(m_{intrpt} + (m_{done}m_{done})))^*$$
$$+(a_{fnt}m_{done}(m_{intrpt} + (m_{done}m_{done})))^* + (a_{def}(m_{done} + m_{intrpt}))^*)^* stop)^* end$$

Tedy automat se během jednoho běhu aplikace může „aktivovat“ libovolněkrát. Aktivace tedy značí vstup hráče do oblasti arény a započnutí tréninkové události. V průběhu tréninkové události automat provádí sérii možných kdy se pohybuje mezi garby a provádí útoky, finty nebo obranu. Pokud hráč v průběhu z oblasti odejde nebo vyprší doba tréninku, automat se deaktivuje. Pokud přijde signál ukončení aplikace, automat přechází do koncového stavu a příjmá.

▶ Poznámka 5.1. Takto definovaná přechodová funkce a regulární výraz nejsou z hlediska fungování aplikace úplně správně, ale jsou takto znázorněny hlavně kvůli čitelnosti. Ve skutečnosti mohou signáli *stop* a *end* přijít kdykoliv a ne jen ve stavech $IDLE$ a $INACTIVE$.

### 5.17.2 Realizace

Pro funkčnost volného pohybu rukou panáků byl vytvořenému modelu v Blenderu přidán ještě rig. Po importu do Unity byl vytvořen podobný systém dopočítávání polohy ruky jako pro hráčův avatar.

Jedním z vytvořených systémů je ovladač rukou panáka, který je schopný držet předmět a hýbat s rukou (obdobně jako funguje hráčův manažer drženého předmětu) a na vyžádání pohnout drženým předmětem danou rychlostí do stanovené lokace. V rámci toho musí zpracovat kolize drženého předmětu a být schopný vracet informace o svém stavu.

Dalším ze systémů je implementace stavového automatu, který funguje tak jak je popsán v předchozí podsekci. Reaguje na vstupy, časové událost a stav drženého předmětu ovladače ruky a v rámci stavů pracuje s ovladačem rukou k vykonání svých akcí.

Posledními ze součástí systému herních prvků je info panel a manažer arény. Info panel jen zobrazuje a spravuje daný text, také se aktivuje nebo deaktivuje v závislosti na vzdálenosti od hráče. Při aktivaci se natáčí tak, aby hráč viděl vždy napřímo.

Manažer arény spravuje aktivního trénovacího panáka, tréninkové událost a sběr a vypisování výsledků a skóre. Pokud se hráč přiblíží do oblasti arény (je ohraničena plotem), započne tréninková událost (v základním nastavení trvá minutu a půl) při níž se aktivuje přítomný trénovací panák. V průběhu trénovací události komunikuje s manažerem hráče a s automatem a získává od nich statistiky útoků, zásahů, blokování atd. Trénovací událost se ukončí po skončení časovače a nebo pokud hráč opustí oblast. Po skončení se deaktivuje trénovací panák a na info panel se vypíší statistiky a skóre.

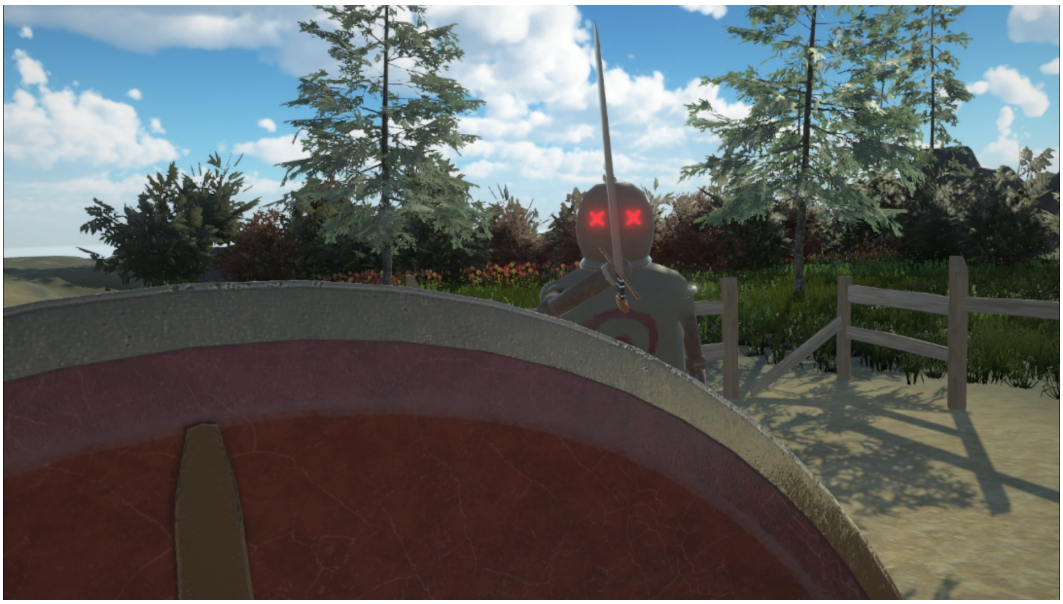
Skórovací systém je jednoduchý, procentuálně hodnotí absolutní „úspěšnost“ (od 0 do 100).

Pro trénink obrany se jedná o vážené hodnocení jednotlivých příznaků.

$$defenceScore = \frac{attacksBlocked + 0.5 * attacksDodged}{attackCount} * 100 \tag{5.1}$$

Pro trénink útoku je skóre zaměřeno na množství zásahů, ale také na snahu štít obranného panáka co nejvíce obcházet a nezasekávat se o něj.

$$attackScore = min(attackHits * 4 - attacksBlocked, 100) \tag{5.2}$$



**Obrázek 5.3** Ukázka obranného tréninku

### 5.17.3 Možné rozšíření

Výsledný systém je poněkud „hrubý“, ale jakožto test formy funčnosti splnil očekávání. Mám tedy návrh možného rozšíření prvku .

Místo lineárně definovaných technik by se opravdové historické techniky a útoky z dobových manuskiptů nahrály do animačních dat pomocí technologie Motion Capture předvedené trénovanou osobou. Poté by se daly namapovat na virtuálního avatara a do pozic rukou přidat virtuální zbraň s fyzikálními vlastnostmi. Mohla by tak vzniknout referenční knihovna těchto technik, které by se daly takto ve virtuálním prostoru libovolně rychle přehrávat, pozastavovat a případně s nimi fyzikálně interagovat a pokusit se jim ubránit nebo je přelstít. Virtuální realita je zatím ještě plně nevyvynuta, ale už teď se jedná o jednu z nejlepších možností jakožto médium pro detailní vizuální referenci a inspekci.

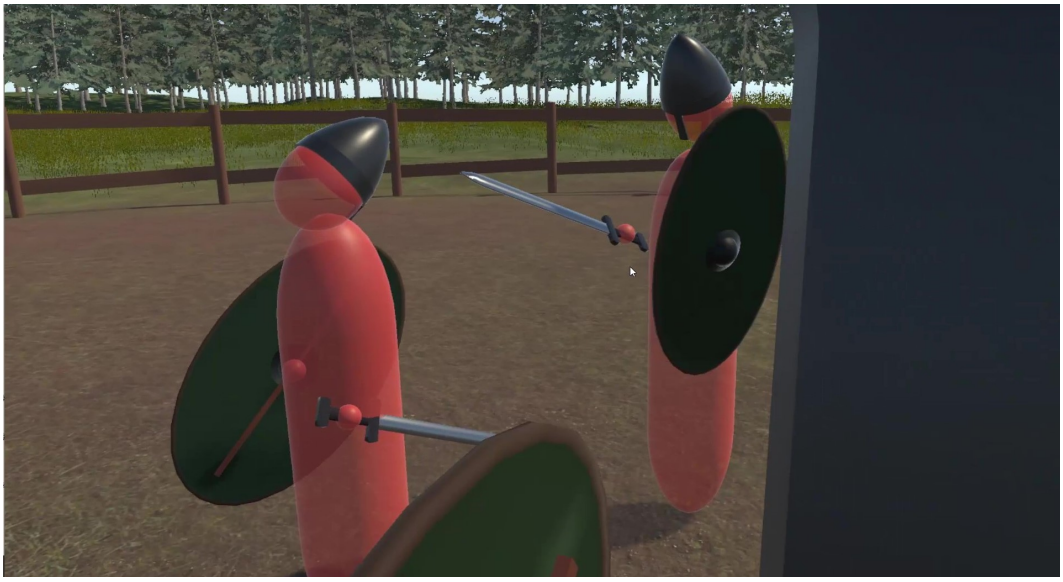
## 5.18 Svět pro více hráčů

Jedná se o samostatný projekt-experiment, který je zjednodušenou formou hlavní aplikace a jeho jediným cílem je prozkoumat fyzikální interakce v prostředí pro více hráčů.

Prostředí aplikace je vytvořené pomocí podobných assetů jako u hlavní aplikace. Bylo vytvořeno i low-poly vybavení – meč, štít a přilba (je k nalezení v hlavní aplikaci v prototypovém domě). Herní postavy jsou tvořeny pomocí objektu koule pro reprezenataci hlavy, kapsle, která se přepočítává na výšku postavy, jako tělo a palé koule jako ruce. Všechny s průhledným barevným materiálem. Vybavení je také trvale součástí postavy.

Pro tvorbu světa pro více hráčů je použit systém Photon PUN, jedná se o server-klient relaci a systém nabízí množství komponent a skriptů pro ulehčení práce se synchronizací mezi klienty.

V implementované prototypové aplikaci má každý hráč svoji instanci s VR Rigem a všemi systémy a za každého dalšího hráče co se připojí na server se vytvoří jakási loutka. Hráč pozice své hlavy a rukou posílá na server, a ten je distribuuje mezi všechny ostatní klienty. Tyto informace se použijí pro aktualizování hráčovi loutky u klientů ostatních hráčů. Pozice všeho vybavení hráče se obdobně propisuje do pozic vybavení jeho loutky.



**Obrázek 5.4** Ukázka z multiplayerového prototypu s více hráči.

Hlavním problémem je samotná synchronizace světa. Každý hráč je autoritou svého vlastního vybavení (štítu, meče) a jejich projekce do loutky u ostatních klientů je z principu opožděná a tedy nepřesná. Pokud hráč interaguje s předmětem některé z loutek (střetne se se svým mečem s mečem jiného hráče), opačná interakce na straně druhého hráče bude v jiném stavu světa (meč druhého hráče a loutka prvního budou v odlišných lokacích než z pohledu prvního hráče). Tedy tato interakce může proběhnout u každého z hráčů jinak. Do toho vstupuje nedeterminističnost fyzikálního enginu, která také může způsobit rozdílné výsledky akce u každého z hráčů.

Potencionální řešení použitím centralizované autority pro řešení fyzikálních interakcí by nejspíše bylo problémové z hlediska plynulosti pohybů a kontroly, kterou by měl hráč nad svými předměty. Při výpadcích nebo opoždění by následná synchronizace působila na hráče vizuálně nepřívětivě. Hráč by obecně neměl skoro žádnou moc nad „svými věcmi“.

V přiloženém médiu se nachází videoklip s ukázkou playtestu zkoušení mechanik s více hráči.

# Závěr

Cílem práce bylo navrhnout a vytvořit 3D aplikaci pro virtuální realitu se zaměřením na fyzikální interakce v systému Unity a tvorbou 3D objektů v grafických aplikacích a projít celým procesem tvorby aplikace herního charakteru od návrhu, tvorby dílčích objektů a implementace a prozkoumat míru, s jakou lze takto komplexní úlohu pro současné komerční technologie dosáhnout.

V rámci práce byl jeden z dílčích cílů zjistit jestli je možné tyto fyzikální interakce dostat i do virtuálního světa pro více uživatelů pomocí již existujících systémů. Dospěl jsem k výsledku, že existující systémy nejsou vhodné pro takto synchronizačně náročnou mechaniku jako je fyzika. Volnost pohybů, a tedy i množství možných interakcí, které navíc přináší virtuální realita, tuto nevhodnost ještě zhoršuje.

Výsledná aplikace pro jednoho uživatele umožňuje uživateli využít předností virtuální reality volností pohybu ve virtuálním světě. Vytvořené historické předměty a prostředí lze prozkoumat a použít je pro interakci s ostatními objekty a s herními prvky v podobě tréninkových panáků pro základní elementy boje.

Z výsledné interaktivní aplikace a z kompromisů, které musely být při tvorbě použity vyplývá, že v současné době není samozřejmě možné používat virtuální realitu jako plnohodnotnou simulaci fyzikálních interakcí, ale je naprosto dostačující jako aplikace zábavního nebo naučného typu. Tedy jako hra nebo jako naučná a poznávací aplikace se základními, ne příliš komplexními, fyzikálními prvky.

Z výsledných fyzikálních mechanik aplikace a herních prvků se nabízí možnost rozšíření přidáním knihovny technik z historických manuskriptů namapováním dat získaných technologií snímání pohybu (Motion Capture) na virtuální postavu. To by umožňovalo vizuální referenci technik a případně i cvičnou interakci s touto virtuální postavou.

Dále by v budoucnosti bylo ideální se vrátit a provést revizi síťového systému světa pro více uživatelů a buďto využít nějaký jiný, vhodnější, systém nebo nový systém přenosu informací a synchronizace navrhnout a implementovat neboť tato aplikace v podobě zábavní a naučně poznávací aplikace by nejvíce benefitovala z více hráčských postav v jednom světě.

# Bibliografie


1. *Carbon-front* [online obrázek]. [B.r.] [cit. 2021-12-10]. Dostupné z: https://vr.tobii.com/wp-content/uploads/2020/09/Carbon_front-1024x576.jpg.
2. *htc-vive-manual-main* [online obrázek]. [B.r.] [cit. 2021-12-10]. Dostupné z: https://www.vrheads.com/sites/vrheads.com/files/styles/large/public/field/image/2016/11/htc-vive-manual-main.jpg.
3. *HP-Reverb-G2-Controllers* [online obrázek]. [B.r.] [cit. 2021-12-10]. Dostupné z: https://uploadvr.com/wp-content/uploads/2020/11/HP-Reverb-G2-Review-Controllers-scaled.jpg.
4. *VR Headset Comparison December 2021* [online]. 2021 [cit. 2021-12-06]. Dostupné z: https://benchmarks.ul.com/compare/best-vr-headsets.
5. EBERLY, D.H. *Game Physics*. Taylor & Francis, 2010. ISBN 9780123749031. Dostupné také z: https://books.google.cz/books?id=Ax47nwEACAAJ.
6. *Hellsplit: Arena* [soft.]. Deep Type Games, 2019 [cit. 2021-12-06]. Dostupné z: https://store.steampowered.com/app/1039880/Hellsplit_Arena/.
7. *hellsplit-arena-preview* [online obrázek]. [B.r.] [cit. 2021-12-10]. Dostupné z: https://cdn.cloudflare.steamstatic.com/steam/apps/1039880/ss_5b1b6534839e37a3e19699f62a0a034683a15b78.600x338.jpg?t=1624358632.
8. *BONEWORKS* [soft.]. Stress Level Zero, 2019 [cit. 2021-12-06]. Dostupné z: https://store.steampowered.com/app/823500/BONEWORKS/.
9. *boneworks-preview* [online obrázek]. [B.r.] [cit. 2021-12-10]. Dostupné z: https://cdn.cloudflare.steamstatic.com/steam/apps/823500/ss_784f12a1c9818045adef23c4b713f3039d803e48.600x338.jpg?t=1581381377.
10. *Blade and Sorcery* [soft.]. Warp Frog, 2018 [cit. 2021-12-06]. Dostupné z: https://store.steampowered.com/app/629730/Blade_and_Sorcery/.
11. *blade-and-sorcery-preview* [online obrázek]. [B.r.] [cit. 2021-12-10]. Dostupné z: https://cdn.cloudflare.steamstatic.com/steam/apps/629730/ss_9f437e2fd1f4ebc780337cb820550cae250cccdd.600x338.jpg?t=1635526484.
12. *Unity* [soft.]. San Francisco: Unity Technologies, 2004. Verze: 2020.3.22f [cit. 2021-12-06]. Dostupné z: https://www.unity.com/.
13. *Unity Documentation* [online]. San Francisco: Unity Technologies, 2021. Verze: 2020.3 [cit. 2021-12-06]. Dostupné z: https://docs.unity3d.com/.
14. *Unity Learn* [web service]. San Francisco: Unity Technologies, 2019 [cit. 2021-12-06]. Dostupné z: https://learn.unity.com/.

15. *OpenXR* [online]. Oregon: Khronos, 2019 [cit. 2022-01-04]. Dostupné z: https://www.khronos.org/openxr/.
16. *PhysX Documentation* [online]. Nvidia, [b.r.] [cit. 2022-01-04]. Dostupné z: https://docs.nvidia.com/gameworks/#gameworkslibrary/physx/physx_v3_3.htm.
17. *Photon PUN* [soft.]. Hamburg: Photon Engine, 2003 [cit. 2021-12-06]. Dostupné z: https://www.photonengine.com/pun.
18. *Microsoft Documentation* [online]. Redmond, Washington: Microsoft, [b.r.] [cit. 2022-01-04]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/.
19. *Blender* [soft.]. Amsterdam: Blender Foundation, 1998. Verze: 2.93.5 [cit. 2021-12-06]. Dostupné z: https://www.blender.org/.
20. *Adobe Substance 3D Painter* [soft.]. San José: Adobe Systems, 2021. Verze: 7.2.2 [cit. 2021-12-06]. Dostupné z: https://www.substance3d.com/.
21. *Character Creator 3* [soft.]. Reallusion, 2021. Verze: 3.44.4709.1 [cit. 2021-12-06]. Dostupné z: https://www.reallusion.com/character-creator/.
22. *Mixamo* [soft.]. San José: Adobe Systems, 2008 [cit. 2021-12-06]. Dostupné z: https://www.mixamo.com/.
23. OAKESHOTT, E. *The Sword in the Age of Chivalry*. Boydell Press, 1998. History of the sword. ISBN 9780851157153.
24. *PureRef* [soft.]. Idyllic Pixel, [b.r.] [cit. 2021-12-06]. Dostupné z: https://www.pureref.com/.
25. DICKINSON, Tania; HÄRKE, Heinrich. Early Anglo-Saxon Shields. In: 1992. ISBN 0-85431-260-9. Dostupné z DOI: 10.1017/S0261340900028125.
26. MALARSKI, Mateusz. *i.33 Sword and Buckler* [herní asset]. 2019 [cit. 2021-12-06]. Dostupné z: https://sketchfab.com/3d-models/i33-buckler-and-sword-6f10087d36b84a1290387fecfe44dcd6.
27. UNITY TECHNOLOGIES. *Terrain Tools Sample Asset Pack* [herní asset]. Unity Asset Store, 2019 [cit. 2021-12-06]. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/nature/terrain-tools-sample-asset-pack-145808.
28. *Unity Asset Store* [web service]. San Francisco: Unity Technologies, 2010 [cit. 2021-12-06]. Dostupné z: https://assetstore.unity.com/.
29. FORST. *Conifers [BOTD]* [herní asset]. Unity Asset Store, 2021 [cit. 2021-12-06]. Dostupné z: https://assetstore.unity.com/packages/3d/vegetation/trees/conifers-botd-142076.
30. NOBIAX / YUGHUES. *Yughues Free Bushes* [herní asset]. Unity Asset Store, 2020 [cit. 2021-12-06]. Dostupné z: https:assetstore.unity.com/packages/3d/vegetation/plants/yughues-free-bushes-13168.
31. ALP8310. *Grass Flowers Pack Free* [herní asset]. Unity Asset Store, 2019 [cit. 2021-12-06]. Dostupné z: https:assetstore.unity.com/packages/2d/textures-materials/nature/grass-flowers-pack-free-138810.
32. TEXTURE HAVEN. *Terrain Textures - 4K* [herní asset]. Unity Asset Store, 2020 [cit. 2021-12-06]. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/terrain-textures-4k-179139.
33. PROASSETS. *Free HDR Sky* [herní asset]. Unity Asset Store, 2016 [cit. 2021-12-06]. Dostupné z: https://assetstore.unity.com/packages/2d/textures-materials/sky/free-hdr-sky-61217.
34. GEIG, M. *Unity 2018 Game Development in 24 Hours, Sams Teach Yourself*. Pearson Education, 2018. Sams Teach Yourself. ISBN 9780134998893.

35. *Github for Unity* [online]. San Francisco: Unity Technologies, 2018. Verze: 1.4.0 [cit. 2021-12-06]. Dostupné z: https://unity.github.com/.

36. *WireWhiz* [blog]. 2019 [cit. 2021-12-06]. Dostupné z: https://wirewhiz.com/.

37. VLADIMIR, T. *Luna Tech Series: A Deep Dive into Unity Configurable Joints* [online]. Luna Labs, 2020 [cit. 2021-12-06]. Dostupné z: https://medium.com/luna-labs-ltd/luna-tech-series-a-deep-dive-into-unity-configurable-joints-96c49138b9b7.

# Obsah přiloženého média

- `readme.txt` — stručný popis obsahu média
- `exe` — adresář se spustitelnou formou implementace
  - `manual.txt` — informace ke spuštění a ovládání aplikace
  - `openXRBuild` — zakomprimovaná verze spustitelné implementace
  - `mpPrototype` — zakomprimovaná spustitelné verze implementace světa pro více hráčů
- `media`
  - `playtest.mp4` — videoklip ukazující test aplikace pro více hráčů
- `src`
  - `thesis` — zdrojová forma práce ve formátu LaTeX
- `text` — text práce
  - `thesis.pdf` — text práce ve formátu PDF