Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Software pro rehabilitaci paže ve virtuální realitě

Plzeň 2020 Jakub Kodera

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 7. května 2020

Jakub Kodera

Abstract

Arm rehabilitation software in virtual reality. The work is a part of a larger project, that is trying to achieve rehabilitation of the upper limb, in patients suffering from a neurological disorder, in virtual reality. The result of this work is an application, that allows patients to perform rehabilitation exercises. The technologies used to develop the software are discussed here, mainly the HTC Vive system and the Unity game engine. The application is designed so that it can be further expanded. The application provides patients with a vizualization of the movement, that they are to perform. The therapeut obtains data and statistics of the exercises performed by the patient.

Abstrakt

Práce je součástí projektu, který se zabývá rehabilitací horní končetiny, u pacientů trpících neurologickou poruchou, ve virtuální realitě. Výsledkem této práce je aplikace umožňující provádět rehabilitačních cvičení. Jsou zde probírány technologie použité při vývoji softwaru. Zejména systém HTC Vive a herní engine Unity. Aplikace je navržena tak, aby ji bylo možné dále rozšiřovat. Aplikace poskytuje pacientům vizualizaci prováděného pohybu. Terapeut aplikací získá data a statistiky o cvičeních vykonaných pacientem.

Obsah

1 Úvod

Práce je součástí většího projektu, jehož cílem je umožnit provádění rehabilitačních cvičení horní končetiny ve virtuální realitě. Práce je zaměřena na pacienty, kteří trpí nějakou neurologickou chorobou, která je v pohybech horní končetinou značně omezuje. Pacienti navštěvují rehabilitace, kde s pomocí terapeuta provádějí *pohyby* a *hry*, které mají za účel jejich zdravotní stav zlepšit.

Cílem této práce je vytvořit aplikaci, která pacientům umožní provádět tato cvičení ve virtuální realitě. Aplikace bude schopná pacientům prováděná cvičení vizualizovat ve virtuálním světě a terapeutům by měla poskytnout informace o pacientem vykonaných cvičeních.

Při práci probíhala spolupráce s týmem Fakultní nemocnice Královské Vinohrady (FNKV), který se touto problematikou zabývá. Na základě této spolupráce probíhala tvorba specifikace a vlastností softwaru. Tato práce přímo staví na již hotové práci, jejímž výsledkem je modul, umožňující analýzu pohybu v reálném čase [14]. Současně s touto prací probíhá vývoj dalších dvou prací [9][12], které rehabilitační aplikaci poskytují data o terapeutem předcvičených pohybech, na základě kterých proběhne vyhodnocování pacientova pohybu.

V práci bude nejprve probrána jedna z nejčastějších neurologických chorob, kterou pacienti trpí. Následně proběhne diskuze technologií použitých při vývoji aplikace. Nakonec bude popsána vlastní implementace rehabilitačního softwaru. V práci je postupně popsáno, jaké problémy bylo potřebné při její realizaci řešit.

2 Roztroušená skleróza mozkomíšní

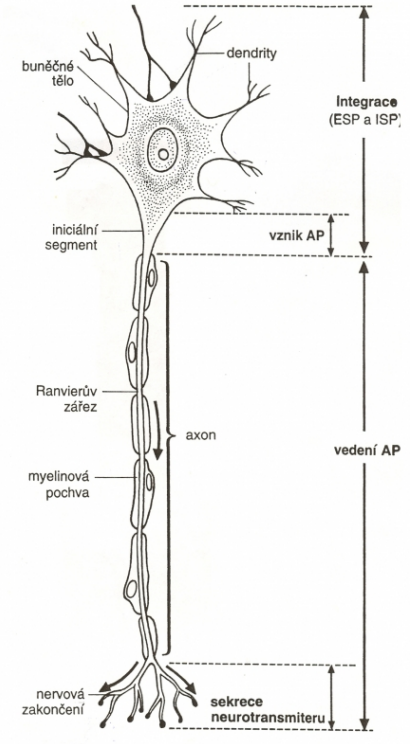
Cílem této kapitoly je seznámit čtenáře s nemocí, pro jejíž rehabilitaci byl software primárně vytvářen.

Roztroušená skleróza mozkomíšní (Sclerosis multiplex cerebrospinalis) je chronické onemocnění centrální nervové soustavy (mozku a míchy) [1]. V České republice se jedná o poměrně časté onemocnění, trpí jím přibližně 10000 až 13000 lidí [24], tzn. zhruba každý 1000, přičemž výskyt nemoci u žen je asi 2x-3x častější než u mužů [15]. Dodnes nelze jednoznačně říci co je příčinou vzniku nemoci. To má za důsledek, že jí nelze předejít. Její vývoj je dnes přičítán jak faktorům genetickým, tak faktorům zevního prostředí [1]. Na základě praxe a experimentů jsou známy mechanismy a spouštěče, které vedou k rozvinutí nemoci. Nemoc se typicky začne projevovat mezi 20.-30. rokem života. Dosud nebyl nalezen lék, který by průběh nemoci zastavil úplně. Sama o sobě není smrtelná, ovšem jelikož se jedná o onemocnění celoživotní, může pacienta v určitých oblastech značně omezovat [5].

2.1 Průběh nemoci

Roztroušená skleróza, dále jen RS, se řadí mezi autoimunitní, demyielinizační choroby [1]. Základní funkcí imunitního systému člověka je chránit tělo před infekcemi a napadání cizích, nebezpečných buněk. U autoimunitních onemocnění ovšem dochází k tomu, že buňky imunitního systému začnou napadat a ničit buňky tkání organismu. Takovýchto nemocí je celá řada, typicky se dělí podle toho jaký typ tkáně napadají [15]. Při RS dochází k tomu, že T-lymfocyty, makrofágy a další bílé krvinky imunitního systému, začnou přecházet přes hematoencefalickou bariéru (přechod mezi vlásečnicemi krevní soustavy a mozkové tkáně). Následně začnou považovat myelin za cizí a postupně napadají a ničí myelinové pochvy v bílé hmotě mozku a míchy, odtud název demyielinizační. Přičemž nelze jednoznačně určit, co tuto reakci imunitního systému způsobuje [1].

Myelinová pochva vytváří lipidový (tukový) obal okolo axonů, dlouhých výběžků neuronů (buňky nervové soustavy), které zajišťují přenos akčních potenciálů mezi jednotlivými buňkami. Pochva je přerušována Ranvierovými zářezy, viz obrázek 2.1. Myelin je v centrální nervové soustavě tvořen oligo-



Obrázek 2.1: Neuron - základní jednotka nervové soustavy[1]

dendrocyty a jeho účelem je chránit a vyživovat samotná nervová vlákna. Myelin také umožňuje přeskakování impulsů z jednoho Ranvierova zářezu na druhý, místo toho aby se vzruch musel šířit po celém vlákně, čímž šíření jednotlivých signálů značně urychluje [5].

Při rozpadu myelinových pochev dochází ke chronickému zánětu, a vzniká až několik nepravidelně rozmístěných ložisek, kterým se také říká plaky nebo jinak léze. Odtud také pochází název *roztroušená skleróza*, jelikož po odeznění zánětu vznikají v místech postižení jizvy, řecké skleros znamená tuhý. Roztroušená protože ložiska mohou vznikat na různých místech mozku a míchy. Velikost ložisek může být různá, od milimetrů po centimetry [5]. V určitých případech jsou nervová vlákna schopna alespoň částečné reparace, tzv. remyelinizace [1]. Ovšem postupem času dochází k vyčerpání regeneračních schopností oligodendrocytů. Důsledkem toho sice dojde k obnovení vzruchů, jejich vedení je ale pomalejší [5]. V pozdějších, někdy ale i v časných, fázích nemoci může také docházet k úplnému zničení samotných axonů, v takových to případech již není možná obnova funkce dané buňky [1].

Vlastní průběh nemoci je poměrně různorodý a nevypočitatelný, může se

[1] `https://velkaencyklopedie.estranky.cz/fotoalbum/biologie/biologie-lidske-telo/nervova-soustava/neuron.-.html`

u každého pacienta lišit [5]. Typickým průběhem nemoci je střídání *atak* (relapsů) a *remisí*. Ataka je období nemoci, kdy dochází k rozvoji neurologické dysfunkce, tedy období, kdy se buď zhoršují stávající příznaky, nebo dochází k rozvinutí nových. Po atakách následuje období, kdy dochází k úplnému, nebo alespoň částečnému, uzdravení (remisi). Pacient se cítí dobře a neurologický nález je také v normě [1]. U některých pacientů dochází k tomu, že první záchvat zanechá trvalé následky, ale nemoc se dále nezhoršuje, nebo může od prvního záchvatu docházet k neustálému zhoršování. Po první atace typicky dochází k dalším relapsům po dvou letech, ovšem může nastat i dříve, ale jsou i případy, kdy se další ataka neprojeví nikdy [5]. První atace nejčastěji předchází nějaký negativní vnější faktor, například virové onemocnění, psychický a fyzický stres, ale může se projevit i zcela náhle [1]. Podle průběhu se RS dělí na *benigní* a *maligní*. Benigní forma se projevuje jen malým množstvím atak s nepatrnými následky, maligní typ je charakterizován těžkými záchvaty s rychlým rozvojem invalidity [5].

2.1.1 Typy roztroušené sklerózy

Na základě průběhu nemoci, popsanému výše, se pak nemoc dělí do čtyř kategorií:

- **Relaps-remitentní** — jedná se o nejběžnější typ RS, vyskytuje se u většiny (~85%) pacientů v počátcích onemocnění [16]. Mohou se vyskytnout spontánní remise, kdy je stav nemocného stabilní, bez jakékoliv léčby. To lze s největší pravděpodobností připsat procesu *remyelinizace* [5]. Období trvá přibližně od 5 do 15let. Probíhá zde zánětlivý proces, který je stále ovlivnitelný léky. Zhruba polovina nemocných přejde do deseti let z formy relaps–remitentní do chronicko–progresivní [16].

- **Chronicko-progresivní** — neboli sekundárně progresivní. V tomto období dochází ke snížení zánětlivé činnosti a naopak růstu neurodegenerace.Typicky organismus v této fázi již vyčerpal schopnost remyelinizace a proto zde další poškození způsobené zánětem nese trvalé následky. Ataky proto nebývají tak nápadné a spíše dochází k postupnému nárůstu invalidity až imobilizaci nemocného. Léčba léky už zde přestává být efektivní a kvalita života pacienta záleží především na jeho životosprávě a rehabilitacích [16].

- **Primárně-progresivní** — touto formou onemocnění trpí přibližně 10% pacientů, převážně se jedná o muže pozdního věku (40–50 let).

U tohoto typu jsou remise velmi málo časté, a tak dochází již od počátku onemocnění k pozvolnému nárůstu invalidity [5]. Svým průběhem se zcela liší od relaps-remitentní formy, převažuje neurodegenerace a úbytek oligodendrocytů nad zánětlivou činností [16].

- **Relabující-progresivní** — nejméně často vyskytující se forma. Na druhou stranu také nejhorší ze všech popsaných. Charakterizuje se tím, že po atakách nedochází u pacientů k žádnému zlepšení stavu [5]. Nachází se zde zvýšená jak zánětlivá tak neurodegenerační činnost, které vedou k invaliditě již během pár let od počátku onemocnění [16].

2.2 Symptomy

Příznaky s sebou přináší řadu poruch, které korespondují s konkrétním místem demyelinizace v centrální nervové soustavě. Kromě místa postižení je projev choroby taky částečně ovlivněn velikostí zánětu. Typicky jsou první symptomy onemocnění mírného charakteru, které jsou dosti nespecifické, např. únava, deprese nebo bolesti končetin, pacienti jim nevěnují větší pozornost, a proto je sdělují lékařům až po delší době [16].

- **Senzitivní poruchy** — jsou poruchy citlivosti ve smyslu parestézie a dysestésie v různých částí těla především v horních a dolních končetinách. Jedná se o nepříjemný pocit brnění, píchání, svědění či pálení kůže. Tyto symptomy se vyskytují zhruba u poloviny nemocných [1] a často jsou přehlíženy jelikož po pár dnech až týdnech odezní [16].

- **Poruchy motoriky** — jedná se o *nejvýznamnější* projevy RS, způsobené poruchou pyramidové dráhy[2] jejíž funkcí je volní[3] a především jemná motorika končetin [16]. Poruchy tohoto typu se typicky neprojevují na začátku onemocnění, ale až v pozdější fázi [1]. Jedná se především o parézy (obrny) a spasticitu (porucha způsobená zvýšením tonických napínacích reflexů, závislých na rychlosti protažení svalu, tedy čím rychleji je prováděn napínací pohyb, tím větší je kladený odpor příslušných svalových segmentů [24]). Poruchy mohou postihnout kteroukoliv z končetin (i více najednou), nejčastěji se postižení objevuje u dolních končetin [16]. Časté jsou problémy při chůzi, kdy si pacient není jistý vlastní rovnováhou [24]. Pacienti mohou pociťovat svalovou slabost a zvýšení svalového napětí, které vedou až ke křečím, problémy

[2]Nervová dráha vedoucí z motorické kůry mozku a končící v míšních segmentech.
[3]Volní motorika je schopnost organismu provádět vůlí cílené pohyby.

s ohybem končetin nebo svalový třes. Projevy se mohou v pozdní fázi různě kombinovat, což je nejčastější příčinou vedoucí k invaliditě daného nemocného. Při nečinnosti svalů se projevy pouze zhoršují, dají se ovšem ovlivnit léčbou, především rehabilitacemi [16].

- **Únava** — jeden z nejběžněji vyskytujících se příznaků neurologických onemocnění, který výrazně omezuje společenský život a vykonávání běžných denních aktivit [24]. Jedná se o velmi obtěžující nespecifický symptom, kterým trpí asi 95% osob s RS. U každého se může projevovat trochu jinak. Je pociťována jako nedostatek síly, pocit vyčerpání bez odpovídající fyzické zátěže a ztráta energie [16]. Na vzniku únavy se podílí například svalová slabost, bolest, různá onemocnění, psychický stav nebo poruchy spánku [24].

- **Psychické potíže** — jedná se o změny afektivity a to zejména o deprese nebo někdy o euforie. Intelekt pacienta nebývá nemocí ovlivněn [1]. Depresivní syndromy se vyskytují zhruba u poloviny pacientů a je prokázáno, že u takovýchto pacientů je riziko sebevraždy asi 7x větší než u lidí zdravých, symptomy by se tedy neměly přehlížet. Symptomy se projevují jako nekontrolovatelný pláč, nebo v případě euforie smích. Pacienti pociťují strach, úzkost a sebelítost, což vede ke zhoršení kvality života, jelikož mají větší tendence k sociální izolaci, narušování rodinných vztahů. U některých pacientů může také docházet ke zhoršení kognitivních (paměť, myšlení, koncentrace...) funkcí [16].

- **Další** — mezi další typické příznaky patří například močové potíže, mozečkové poruchy (špatná koordinace, skandovaná řeč...). Pro více informací o těchto a dalších projevech viz [1], [16].

2.3 Léčba

Jak již bylo zmíněno roztroušenou sklerózu *není* možné zcela vyléčit.
U řady pacientů lze ovlivnit průběh nemoci. Léčebný proces se pak rozlišuje na léčbu akutního stavu, kdy dochází u pacienta k relapsům (zhoršení příznaků), a na dlouhodobou léčbu, kdy je snaha snížit počet atak a zpomalit rozvoj nemoci [16].

Dále zde léčebné zásahy pro léčbu symptomů a atak popisovat neubudu. Jedná se převážně o podávání léků pacientům, ať už za účelem zlepšení jejich aktuálního stavu, či prevenci jeho zhoršení. Pro více informací na toto téma odkážu na [1] a [16]. Popíši zde proces rehabilitací, který má v rámci kontextu práce větší význam.

2.3.1 Rehabilitace

„Rehabilitace využívá multidisciplinárních strategií ke zvýšení funkční nezávislosti, prevenci komplikací a zlepšení kvality života nemocných. Jde o aktivní proces, který pomáhá lidem k zotavení, k zachování optimální fyzické, smyslové, intelektové, psychické a sociální úrovně funkcí a k dosažení co nejvyšší úrovně nezávislosti navzdory omezení, které onemocnění způsobuje [24].“

Rehabilitační postup vychází ze stanovené diagnózy pacienta. Nejprve se určí základní onemocnění, po němž následuje analýza problému rehabilitačním týmem (lékař, fyzioterapeut, psycholog. . .) pro stanovení priorit léčby. Léčba je postupně modifikována a hodnocena [24]. Vzhledem k variabilitě onemocnění je pro každého pacienta vytvořen individuální rehabilitační program, při kterém se mimo jiné bere v potaz i fáze, ve které se nemocný nachází [16].

Častým problémem bývají depresivní syndromy. Jelikož člověk trpící depresemi není schopen spolupracovat na rehabilitačním procesu, je třeba je včas identifikovat a léčit. Samotné rehabilitace mají poměrně velký vliv na ovlivnění psychiky, a to především rehabilitace individuální, jelikož je snaha vytvářet klidné a bezpečné prostředí. Terapeuti s nemocnými vedou rozhovor, věnují jim čas a snaží se je nutit na sobě pracovat a zlepšovat se, což vede k lepšímu psychickému stavu [24].

Rehabilitace jsou zaměřeny především na spasticitu, zvýšení svalové síly a poruchu koordinace[4] [16]. U těchto poruch nemocný váhá, jak má vlastně daný pohyb udělat a dochází tedy ke zpomalení již naučených činností a zhoršení obratnosti. Cílem rehabilitace je tedy motorické řízení a učení. Nejprve dojde k plánování pohybové sekvence a po prvním úspěšném výkonu je snaha o naučení dané dovednosti opakujícím se tréninkem, kterým dochází ke zdokonalování provedení pohybu (přesnost, rychlost, plynulost) [24].

Pacient s terapeutem provádí několik cvičení (pohybů). Nejprve je pacientovi vysvětlen princip řízení daného pohybu a princip jeho terapie. Následuje definice pohybu a nastavení pacienta do výchozí polohy pro vykonání. Samotný pohyb pak bývá doprovázen s pomocí terapeuta, který pacienta navádí a pomáhá k tomu, aby byl pohyb vykonáván korektně. Je důležité, aby pohyby byly vykonávány správně, jelikož při špatném provádění cvičení může dojít spíše k prohloubení symptomů [24].

Mezi jednotlivá cvičení patří například správné sezení, zvedání se ze sedu do stoje, chůze, diagonály horních končetin, kreslení spirál nebo špetkový úchop.

[4] Jedná se tedy o poruchu především motorické.

3 Virtuální realita

3.1 Obecně

Virtuální realita, dále také VR, je trojrozměrné počítačem generované prostředí, ve kterém se uživatel může volně pohybovat a interagovat s ním. Jedná se tedy o jedno z dalších rozhraní mezi člověkem a počítačem. Na rozdíl od jiných rozhraní, uživatel ve VR se stane přímo součástí vygenerovaného světa. Uživatelé jsou tak přímo vnořeni do vymodelovaného prostředí, ve kterém mají možnost manipulovat s ostatními 3D objekty a ovlivňovat tím dění kolem sebe [2].

Co odlišuje VR od ostatních počítačových rozhraní je snaha úplně izolovat uživatele od okolního světa. Cílem je přesvědčit mozek, že vjemy působící z virtuálního prostřední na naše smysly (zrak, sluch, čich, hmat ale i například rovnováha) jsou skutečné, přestože nejsou [17].

Zde však narážíme na jistou technologickou bariéru, neboť i když jsou dnešní technologie schopny poskytnout věrohodný obraz a zvuk, může dojít ke konfliktu smyslů. Zrak mozku předává informace o tom, že ve virtuálním světě dochází ke zrychlení, ale ostatní smysly, především centrum rovnováhy vnitřního ucha a svalový vzruch, mozku naopak posílají signály o tom, že ne. Tomuto efektu se říká *motion sickness* a typicky způsobuje nevolnost, závratě nebo únavu [13].

Jak působíme a ovlivňujeme virtuální realitu závisí hlavně na platformě. Některé zatím vydané VR brýle byla navržena k používání v sedě, a pohyb je umožněn ovladači, stejně jako u PC či konzole. Typicky je uživateli umožněna teleportace ve vykresleném okolí nebo pohyb pomocí tlačítek daného ovladače. Existují ale již zařízení pro volný pohyb. Uživatel pak typicky definuje prostor ve kterém se bude pohybovat. Velikost takového prostoru je pak samozřejmě limitována samotným rozměrem místnosti a také systémem, který daná platforma používá pro sledování pohybu.

Jak tomu u nových technologií bývá (a virtuální realita stále ještě je relativně nová, přestože už tu s námi pár let je, viz sekce 3.1.1), problémem pro běžného uživatele může pořád být cenová nedostupnost. Jelikož nejčastěji je k samotné VR sadě zapotřebí dostatečně výkonný počítač nebo herní konzole, může se cena kompletní sady pro virtuální realitu vyšplhat až na desítky tisíc korun.

3.1.1 Historie a současnost

Historie

Přestože by se mohlo zdát, že VR je poměrně nová technologie, její počátky sahají až do šedesátých let minulého století. Jako první koncept virtuální reality tak jak ji známe dnes, můžeme považovat tzv. „Divadlo zážitků“. Jednalo se o prototyp přístroje Sensorama umožňující promítání krátkých filmů, při kterém mohl divák kromě obrazu a zvuku vnímat i vůni. V roce 1968 vznikl první zobrazovací systém, který byl připevněn na hlavu. Jednalo se o poměrně primitivní uživatelské rozhraní a vizuální zážitek. Hardware tohoto systému byl tak těžký, že musel být přimontován ke stropu [7].

V roce 1979 vyvinul Eric Howlett systém LEEP (Large Expanse Extra Perspective). Systém vytvořil stereoskopický obraz s dostatečně širokým zorným polem, aby vytvořil přesvědčivý pocit prostoru. LEEP byl jeden z prvních systémů, který připomínal moderní soupravu pro VR, tak jak ji známe dnes. Pojem *virtuální realita* byl popularizován v 80. letech Jaronem Lanierem, který je považován za jednoho z průkopníků VR. Jeho společnost vytvořila produkty jako „DataGlove“ rukavice pro sledování dlaní, nebo „EyePhone“, což byl VR headset [7].

Na počátku 90. let vyvinula NASA systém VIEW (Virtual Interactive Environment Workstation), navazující na systém LEEP, vytvořený za účelem virtuálního tréninku astronautů. Tento systém využíval zpracování *binaurálního 3D zvuku*[1] v reálném čase. NASA také využila VR pro řízení robotů na Marsu v předpokládaném reálném čase navzdory zpoždění signálu mezi planetami. V roce 1991 vytvořila skupina Virtuality stejnojmenný VR systém pro hraní video her. Jednalo se o první masově vyráběné zábavní VR stroje. Headset v reálném čase zobrazoval 3D stereoskopické[2] obrázky. Bylo také možné propojit více strojů po síti a hrát tak hry pro více hráčů. Ve stejném roce byl také vydán Sega VR headset. Jednalo se o LCD headset se sluchátky a senzory, které umožňovaly sledovat jeho polohu. V roce 1992 bylo vynalezeno prostředí CAVE (Cave Automatic Virtual Environment). Jednalo se o tmavou krychlovou místnost, na jejíž stěny se promítalo pomocí zadní projekce (technika, kdy je projektor schovaný za plátnem a neruší tak uživatele). Tento systém má každopádně řadu nevýhod, největší z nich jeho nepřenositelnost, a tak v komerčním světě příliš neujal [7].

[1] Zvuk, který je do levého a pravého ucha vysílán s mírně odlišnou frekvencí.
[2] Technika vytvoření iluze hloubky.

VR dnes

Největší posun ve vývoji ovšem technologie zaznamenala v posledním desetiletí. Vzhledem k tomu, že člověk ke vnímání okolí používá asi ze 70% zrak a 20% sluch nebude překvapením, že současné systémy se zaměřily právě na tyto smysly. Faktem je, že člověk reaguje na zvukové podněty rychleji než na podněty vizuální. Pro skutečně pohlcující virtuální zážitek je potřeba zvuky prostředí a prostoru nezanedbat. Přestože audio-vizuální podněty hrají ve VR největší roli, výzkum a vývoj se zabývá i ostatními smysly. Například všesměrové trenažéry, umožňující uživateli pocit, že se skutečně pohybuje ve virtuálním světě, nebo technologie dotykové zpětné vazby [2].

Virtuální realitu dnešní generace lze datovat od roku 2010, kdy se VR začalo znovu dostávat do povědomí širší veřejnosti. Především to bylo díky zařízení Oculus Rift, které vzniklo jako prototyp Palmera Luckeyho a přineslo s sebou *90 stupňové zorné pole*, kterým žádné dosavadní systémy nedisponovaly. Dalším pokrokem lze považovat průlom v oblasti tzv. *displejů s nízkou odezvou* objevené a volně sdílené společností Valve Corporation. Displeje s nízkou odezvou jsou podstatnou součástí dnešních zařízení z důvodu vyšší ostrosti výsledného obrazu a snížení vlivu motion sickness [7]. V následujících letech se již objevují další zařízení, kde za zmínku stojí především HTC Vive, Playstation VR, Google Daydream View. V neposlední řadě nový Oculus Quest nebo méně známý Pimax, které nabízejí sledování dlaní bez použití dodatečných ovladačů či rukavicí [2].

Dnešní zařízení pro virtuální realitu typicky sestávají ze dvou částí, kterými jsou sledované objekty (HMD, ovladače...) a zařízení, které sleduje polohu a orientaci těchto objektů viz 3.1 [2]. *HMD* (Head-mounted display) je důležité zařízení, které zprostředkovává uživateli audio-vizuální vstup do světa VR. Jedná se o nejvíce rozeznatelnou část VR. Je to headset nebo jakési brýle, typicky se skládající ze dvou displejů, čoček a v některých případech i poloprůhledných zrcadel. Čočky, případně i zrcadla, se používají k roztažení zobrazovaného obrazu, který musí pokrývat co největší zorné pole. Některé typy headsetů používají místo vlastního displeje mobilní telefon například Samsung Gear VR. Helma je nejčastěji připojena k externímu zařízení (počítač nebo konzole), které zpracovává výsledný obraz [21]. Dnes již existují headsety, které jsou schopné fungovat jako *samostatná* jednotka, například Oculus Quest.

3.1.2 Využití

Virtuální realita se v dnešní době používá v mnoha odvětvích. Největší výhodou VR je, že dění ve virtuální realitě nemá trvalé následky na realitu

skutečnou, tzn. tam kde je příliš nebezpečné, nákladné nebo nepraktické dělat něco ve skutečnosti, je virtuální realita odpovědí. Největším odvětvím které využívá VR je bezpochyby zábavní průmysl. Především herní průmysl značně dominuje, dnes již není neobvyklé, že nově vydaná hra, kromě možností hraní pomocí herního počítače či konzole, přináší také podporu pro VR. Do zábavního průmyslu ovšem patří také například interaktivní film, či různé typy zážitků jako je prohlídka památek nebo chůze na Marsu. Virtuální realita má také řadu aplikací v serióznějších oborech. Využívá se pro vizualizaci vědeckých informací ve fyzice nebo chemii. Má své využití i ve vojenských simulacích či vzdělání. V neposlední řadě se také VR používá v lékařství, kde například chirurgové mohou trénovat provádění operací, nebo provádět diagnostiku. Koneckonců aplikace tohoto projektu vznikla za účelem experimentální rehabilitační léčby.

Jak se náklady na virtuální realitu snižují a postupně se stává běžnější, můžeme předpokládat, že se do popředí dostanou vážnější použití a pravděpodobně budou vznikat nové, dosud neprozkoumané aplikace. Virtuální realita by mohla podstatně změnit způsob, jakým propojujeme naše digitální technologie [7][17].

3.2 Volba systému VR

Dnes se již na trhu vyskytuje spousta headsetů pro virtuální realitu. Při výběru jaký z nich použít pro realizaci projektu, poměrně suverénně vyhrál VR headset *HTC Vive*, viz sekce 3.3. Důvodů pro zvolení této soupravy bylo hned několik. Hlavním důvodem proč byl Vive zvolen byl fakt, že umožňuje sledovat zařízení nazývaná *trackery*, viz sekce 3.3.2. Valná většina ostatních systémů používá pouze headset a ovladače. To je ovšem k řešení problému pro rehabilitační cvičení nedostačující, jelikož kromě polohy a orientace horní končetiny[3] hraje svoji důležitou roli i poloha a natočení hrudníku. Dalším rozhodujícím faktorem byla velmi jednoduchá integrace s herním enginem Unity, viz sekce 4.3. Nemalou roli sehrál i fakt, že HTC Vive byl již před začátkem projektu dostupný jak na Fakultě aplikovaných věd Západočeské univerzity, tak ve Fakultní nemocnici Královské Vinohrady. Kromě zvoleného headsetu jsme v počátcích také experimentovali se senzory.

Senzory jsou v tomto případě myšleny dva, a to *akcelerometr* a *gyroskop*. Gyroskop je zařízení, které využívá gravitaci Země k určení orientace. Akcelerometr je kompaktní zařízení určené k měření zrychlení. Když objekt,

[3]Ovladačem by se dala sledovat pouze pozice dlaně/zápěstí, což by stejně nestačilo jelikož je třeba sledovat i polohu paže.

do kterého je integrován, přechází z klidového stavu do jakékoli rychlosti, je akcelerometr navržen tak, aby reagoval na vibrace spojené s takovým pohybem [10]. Většina dnešních mobilních zařízení má v sobě tyto senzory zabudované. Pro práci se senzory vznikla aplikace[4], která s pomocí android aplikace Sensorstream IMU+GPS[5] zapisovala data z těchto senzorů do souborů. Tím jsme získali orientaci a zrychlení mobilního telefonu v konkrétním čase. K řešení problému sledování pohybu horní končetiny ovšem potřebujeme zjistit její polohu a orientaci. Polohu je možno ze zrychlení získat jeho dvojí integrací, ovšem z důsledků integrace poměrně nepřesnou. Telefony se také projevily jako poměrně nepraktické pro upevnění na ruku, a tak bylo pro zjednodušení práce od tohoto nápadu odstoupeno. Výhodou senzorů by každopádně bylo odstínění od konkrétního VR headsetu, jelikož by odpadla závislost na HTC Vive trackerech. Pro bližší informace o senzorech, více viz bakalářská práce [9].

3.3 HTC Vive

3.3.1 Vznik

HTC Vive byl vyvinut společností HTC, zaměřena na výrobu mobilních zařízení, a společností Valve, zaměřena především na vývoj počítačových her. Předtím než Vive vůbec vznikl, obě společnosti začaly zkoumat virtuální realitu samostatně. Už v roce 2012, kdy Oculus Rift odstartoval novou generaci VR headsetů, Valve pracoval na svém vlastním řešení. Vnikly první prototypy sestávající se z HMD, kamery a AprilTags[6] pro sledování polohy zařízení. Hlavním problémem, na který společnost narazila, byl fakt, že obraz, který displej zobrazoval, byl rozmazaný. Valve bylo jasné, že pro hraní her je nutné, aby displej zobrazoval obraz ostřeji a s nízkou odezvou.

Valve sice měla základní fungující řešení, ale vlastní headset v plánu neměla. Nejprve bylo snahou společnosti spolupracovat s Oculusem. Jejich spolupráce ovšem z neznámých důvodů dlouho nevydržela, s největší pravděpodobností došlo k odlišným vizím jak by měla VR vypadat. Společnosti HTC VR velmi připomínalo počátky chytrých telefonů a bylo jasné, že to bude nové technologické médium a chtěla být v popředí jeho vývoje. A tak došlo ke schůzce mezi Valve a HTC, která byla očividně úspěšnější.

[4]Zdrojové kódy dostupné na `https://bitbucket.org/frankkuba/vr/src/master/`.

[5]Aplikace volně ke stažení na `https://play.google.com/store/apps/details?id=de.lorenz_fenster.sensorstreamgps`. Umožňuje sběr dat ze senzorů a pomocí UDP protokolu je po síti posílá na konkrétního klienta.

[6]Čárové kódy, připomínající jednodušší QR kódy.

Práce na první vývojářské edici headsetu začala hned po podepsání dohody. Vývoj probíhal iteračně, postupně byly probírány návrhy a nápady jak by měl headset vypadat. Bylo odstoupeno od nepraktického systému sledování založeného na AprilTags a vznikly dvě alternativy tzv. *dot tracking* a *laser tracking*. U systému dot tracking jsou headset a ovladače pokryté „tečkami“, kamera pak zaznamenává snímky, ze kterých se následně pomocí strojového vidění určí poloha teček a tedy poloha jednotlivých zařízení. Laser tracking se projevilo jako přesnější, a tak probíhaly práce na prototypování tohoto systému. Více o tom jak tento systém funguje, viz sekce 3.3.3.

Zhruba šest měsíců po začátcích spolupráce bylo pozváno pár vývojářů do Valve kanceláří, aby si Vive vyzkoušeli. Na základě odezvy byla vydána první verze na konci roku 2014 pro menší skupinu zákazníků. V roce 2015 byl Vive poprvé představen širší společnosti na MWC (světová mobilní konference) v Barceloně. O rok později pak na stejné konferenci byla vydána první verze pro veřejnost, která od svého prvního představení přinesla pár převážně estetických změn (headset byl více stabilní, senzory byly zakryté...) [18].

3.3.2 Specifikace

Základní balení HTC Vive obsahuje HMD, dva ovladače a dvě základní stanice. Cena tohoto balíčku je 599$. Kromě základních komponent je možno dokoupit další příslušenství, zde zmíním především trackery a adaptér pro bezdrátové připojení HMD, umožňující volnější pohyb bez omezujících kabelů [6]. Základní souprava s trackery, viz obrázek 3.1.

HMD

HMD má obnovovací frekvenci 90Hz a zorné pole 110 stupňů. Zařízení používá dva OLED panely s celkovým rozlišením 2160×1200 (na každé oko tedy 1080×1200) [8].

Z obnovovací frekvence vyplývá, že Vive běží na 90FPS (Frames Per Second, neboli počet snímků za sekundu). Skutečný počet FPS pak je samozřejmě ovlivněn výkonem a zatížením především grafické karty počítače, ke kterému je Vive připojen. V případě větší zátěže a tedy poklesu FPS používá systém techniku tzv. *interleaved reprojection*. To znamená, že za účelem zachování plynulosti vykreslovaného obrazu je počet snímků za sekundu limitován na 45 a je tedy vykreslován pouze každý druhý snímek. Později byla přidána technologie tzv. *asynchronous reprojection*. Ta se aktivuje v případě nižšího poklesu FPS. Funguje tak, že se vezme předchozí snímek, který se na základě aktuálních dat orientace příslušně otočí a posune a následně je

Obrázek 3.1: Komponenty pro virtuální realitu technologie HTC Vive[7]

zobrazen. Výsledkem je tak plynulejší a příjemnější obraz než při interleaved reprojection [3].

Vive používá Fresnelovy čočky, vzdálenost těchto čoček je možné upravit pro dosažení co nejlepší ostrosti obrazu. Vzadu se nachází tzv. Link Box, přes který je headset pomocí USB a HDMI kabelu připojen k počítači. Vpředu headsetu se nachází kamera, která uživateli umožňuje sledovat jeho okolí, aniž by musel headset sundat. Součástí headsetu není souprava pro audio, každopádně se zde nachází 3,5mm jack konektor pro připojení externích sluchátek. Na headsetu jsou infračervené senzory, které detekují světelné a laserové paprsky základních stanic a určují jeho aktuální polohu a orientaci v prostoru, viz sekce 3.3.3. Mezi další senzory patří G-senzor, gyroskop a přibližovací senzor [8].

Ovladače

Ovladače disponují trigger tlačítkem na spodní straně, dvěma tlačítky na bocích, dvěma systémovými tlačítky na vrchu, mezi nimiž se nachází trackpad. Ovladače jsou zároveň schopny poskytovat zpětnou vazbu v podobě vibrací. Jsou bezdrátové a podobně jako headset mají několik senzorů pro detekci jejich polohy a orientace [8].

[7] `https://static.turbosquid.com/Preview/2017/09/02__14_05_49/HTC_Vive_set_01.jpg4F083E2C-EA62-415B-BF6B-D43229B2402DZoom.jpg`

Základní stanice

Základní stanice se typicky připevňují na zeď do výšky alespoň dvou metrů. Pro správnou funkčnost by na sebe obě stanice měly „vidět“, respektive by mezi nimi neměla být žádná překážka. Pokud tomu tak není, je nutné propojit je synchronizačním kabelem. Samotné stanice s počítačem vůbec nekomunikují a jsou pouze připojeny k napájení. Dokáží snímat 360-stupňovitý pohyb brýlí, ovladačů a trackerů v prostoru až 15m^2. Uživatel se může pohybovat a orientovat kdekoliv v dosahu stanic. Umožňují takzvanou *full-room experience* spočívající ve volném fyzickém pohybu ve zvoleném prostoru. Pokud by se uživatel dostal na okraj daného prostoru, je mu v rámci bezpečnosti zobrazena ve virtuálním světě zeď indikující tuto skutečnost [8].

Trackery

Trackery obsahují, podobně jako headsetu a ovladače, senzory. Je možné je připevnit na libovolný objekt či část těla, za účelem snímání jeho polohy či orientace. Pro zajištění sledování trackerů, musí být pro každý tracker k počítači připojen tzv. dongle[8]. Trackery hrají v rámci našeho projektu důležitou roli, jelikož je díky nim zajištěno sledování hned několika objektů: horní končetiny, hrudníku a židle.

3.3.3 Systém sledování zařízení

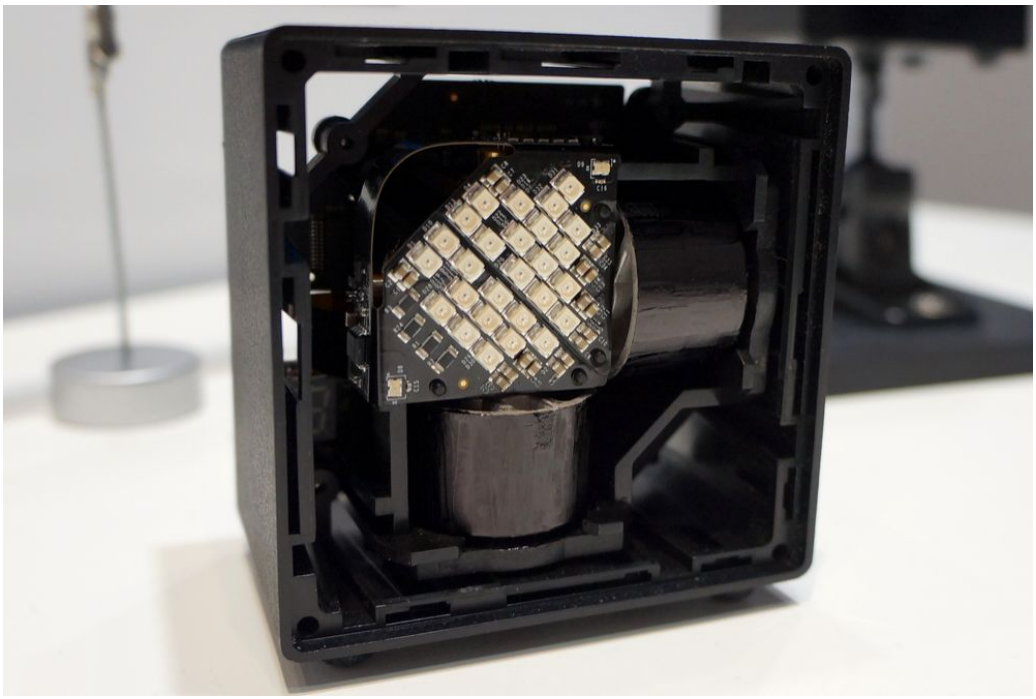
Na rozdíl od většiny ostatních dosavadních systémů HTC Vive ke sledování pozice a orientace jednotlivých objektů (HMD, ovladač, trackery, viz sekce 3.3.2) nepoužívá kamery ani strojové vidění. Systém sledování zařízení se u Vive nazývá *SteamVR Lighthouse* [11].

SteamVR je API (Application Programming Interface, neboli programové rozhraní), které komunikuje se samotným hardware systému Vive a poskytuje vývojářům funkce, které mohou používat pro práci se systémem. Jedná se tedy o vrstvu mezi hardware a samotným software aplikace, která s HTC Vive pracuje. API je vyvíjeno společností Valve a je integrováno do služby Steam.

Základní jednotkou Lighthouse sledování je tzv. *base station*, neboli základní stanice, viz sekce 3.3.2. Tyto stanice v sobě mají panel s LED diody a dva rotory, jeden umístěný horizontálně a jeden vertikálně, viz obrázek 3.2.

[8] `https://www.vive.com/us/support/wireless-tracker/category_howto/using-the-dongle.html`

[9] `https://www.vrfocus.com/wp-content/uploads/2015/06/LighthouseBaseStation_1.jpg`



Obrázek 3.2: Náhled do struktury základní stanice HTC Vive[9]

Stanice pak provádí následující sekvenci. Nejprve jsou za pomocí diod do prostoru vyslány infračervené paprsky, sloužící k synchronizaci. Sledovaná zařízení, pokrytá četnými foto senzory, tyto paprsky zachytí a odstartují si interní časovač. Následně proběhne rotace horizontálního (na obrázku 3.2 spodního) rotoru, který do prostoru začne vyzařovat laserové paprsky zleva doprava tzn. ve vertikální rovině[10]. Foto senzor zařízení laserový paprsek zaznamená a zastaví interní časovač. Jelikož se rotory otáčí s konstantní rychlostí a je tedy známá doba otočení zleva doprava je výsledný vertikální úhel senzoru k dané stanici dán vztahem

$$y = \frac{\text{čas\_dopadu\_laseru}}{\text{celkový\_čas\_otáčky}} \cdot \text{FOV}$$

kde *čas_dopadu_laseru* je uplynulá doba, od které byl senzorem zaznamenán synchronizační puls do zasažení laserem a *celkový_čas_otáčky* je 8,3 milisekund. *FOV* (Field Of View) je pak zorné pole[11] základní stanice. Stejným principem pak proběhne synchronizace a otočení vertikálního rotoru,

[10]Vzhledem k tomu, že stanice nemají zorné pole o velikosti 180 stupňů, jsou nejčastěji nakloněny směrem do herního prostoru. Z toho vyplývá, že i samotné roviny, které stanice pokrývá, jsou nakloněné.

[11]Zorné pole stanic HTC Vive je na základě `https://www.vive.com/eu/support/vive/category_howto/tips-for-setting-up-the-base-stations.html` 120 stupňů.

pro zjištění horizontálních úhlů senzorů k základní stanici[12]. Tuto sekvenci stanice provádí s frekvencí 60Hz [23][4][22]. Vizualizaci toho, jak sekvence probíhá je možno vidět na [11].

Data ze senzorů jsou odeslána počítači, kde je proveden samotný výpočet polohy a orientace daného objektu. Jelikož je jeden sledovaný objekt pokryt větším množstvím senzorů (v řádu desítek) a typicky je jich paprsky zasaženo více, lze na základě znalosti umístění daných senzorů na objektu zjistit jeho polohu a orientaci vůči základní stanici. Jedná se o triangulační problém [23][4].

Další možností by bylo přidání druhé stanice, typicky se používají stanice dvě, kde druhá je přidána hlavně za účelem redundance a ne přesnosti, viz dále. Je-li více senzorů, tak stačí stanice jedna, viz předchozí odstavec. Stanice paprsky pokrývá dvě roviny, vertikální a horizontální. Průsečíkem těchto rovin vznikne přímka (vektor), na které daný senzor leží. Přidá-li se druhá stanice, jsou přímky dvě. Průsečíkem obou přímek lze následně získat polohu daného senzoru[13] [23][4].

Vive primárně pro získání pozice používá IMU (Inertial measurement unit) zařízení využívající akcelerometr a gyroskop, viz sekce 3.2. Výše popsaný systém je založen především na časování, a jelikož mezi jednotlivými synchronizačními pulzy je určitá časová prodleva, pohybuje-li se daný objekt příliš rychle, byla by do jeho snímané pozice zanesena značná chyba. Proto jsou do výpočtu zahrnuty i data z IMU. Z principu systému však je nutné, aby zařízení byla pro základní stanice viditelná [23][22]. Pokud je objekt před stanicí zakrytý, spoléhá systém pouze na data z IMU a jak je v sekci 3.2 popsáno, ta jsou v čase zanesena poměrně rychle rostoucí chybou[14]. Důsledkem toho se pak může stát, že se daný objekt ve virtuální realitě začne od své pozice vzdalovat.

[12]Do signálů, které stanice vysílají, je zakódovaná informace o tom o jakou stanici se jedná a ze kterého rotoru byl vyslán.

[13]Reálně bude výpočet složitější, jelikož je nutné znát vzájemnou polohu obou stanice a velikost sledovaného prostoru. Tato data se získávají při kalibraci systému.

[14]Optický lighthouse systém se tuto chybu snaží co nejvíce redukovat.

4 Unity

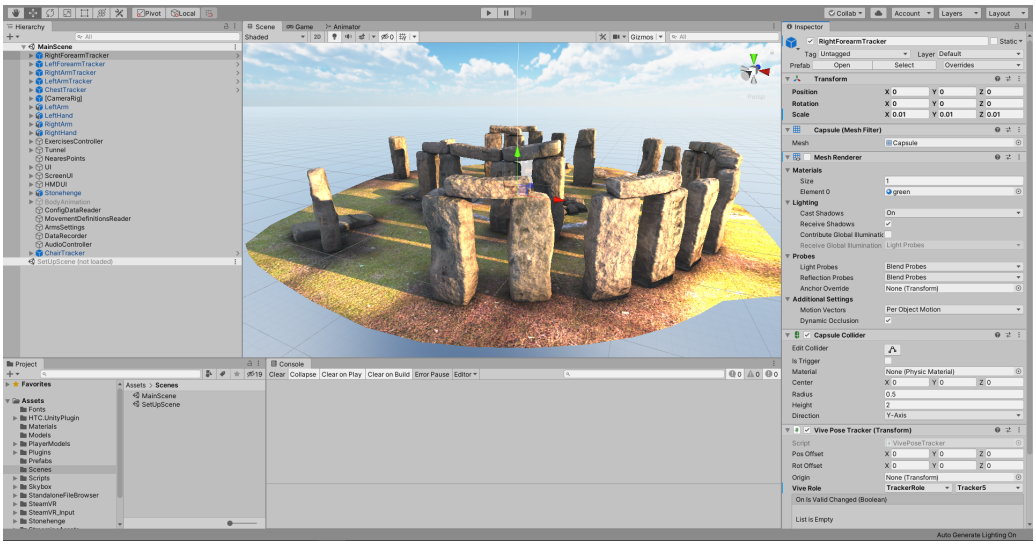
4.1 Obecně

Unity je jeden z nejpopulárnějších herních enginů. Je vyvíjený společností Unity Technologies od roku 2005. Herní engine je software, který vývojářům poskytuje nezbytnou sadu funkcí pro vytváření her. Jedná se o framework (jakousi kostru) pro vývoj her, který podporuje a sdružuje několik klíčových oblastí, jako je například vykreslování 3D grafiky, fyzika, detekce kolizí, animace nebo audio. Vývojář pak tyto funkce využívá, aniž by se musel starat o jejich implementaci, což typicky značně zjednoduší postup vývoje, jelikož je možné se zaměřit především na vývoj samotné herní logiky [20]. Samotný Unity engine je napsaný v programovacím jazyce C++, pro vývoj aplikací se používá jazyk C# (C-sharp).

Unity podporuje celou řadu platforem. Umožňuje vytvářet především hry pro desktopové počítače, mobilní zařízení, herní konzole, web a v neposlední řadě pro *virtuální realitu*. Dále je podporována většina populárních operačních systémů, pro které lze aplikace sestavit. Je tedy poměrně jednoduché vytvořit jednu aplikaci, která může podporovat více platforem najednou. Unity je sice zaměřená pro vývoj videoher, každopádně vzhledem k vlastnostem a nabízeným funkcím, které herní engine poskytuje, lze engine využít i pro tvorbu aplikací z jiných oblastí. Nejčastěji jsou tyto aplikace nějakým způsobem provázány s prací s 3D grafikou, ať už pro vizualizaci dat nebo tvorbu prototypů či provádění různých simulací.

Herní engine Unity byl již v počátcích projektu zvolen jako vhodné prostředí pro vývoj aplikace. Ve své podstatě sice software není hrou, určité prvky her ovšem obsahuje, přeci jen v rámci rehabilitací existují typy her nebo aktivit, které pacienti vykonávají za účelem zlepšení hybnosti. Tato práce přímo navazuje a staví na bakalářské práci [14], jejíž výsledkem je .NET knihovna pro analýzu prováděného pohybu v reálném čase, která byla vytvořena přímo pro možnost její integrace do aplikace tvořené v Unity.

Důležitou roli při vývoji aplikace v Unity hrají takzvané *assety*. Jsou to veškeré objekty (zdroje, data), které jsou v projektu použity a společnou integrací tvoří samotnou aplikaci. Některé assety je možné vytvořit přímo z Unity editoru (skripty, scény, primitivní herní objekty. . .), jiné jsou typicky externí a je možné je do projektu importovat (3D modely, audio soubory, knihovny. . .). Veškeré assety se v Unity projektu nacházejí ve složce Assets. Assety se při sestavení aplikace stanou přímo součástí spustitelného .exe sou-



Obrázek 4.1: Editor herního enginu Unity

boru, někdy je ale zapotřebí k assetům přistupovat ze souborového systému. Takové soubory se pak ukládají do speciální složky Assets/StreamingAssets.

4.1.1 Editor

Unity editor je volně dostupný[1] nástroj, viz obrázek 4.1, vyvíjený Unity Technologies, který umožňuje vývojářům vytvářet vlastní aplikace s využitím Unity herního enginu.

Práce v editoru se skládá ze dvou základních částí a to je tvorba herního světa a programování jeho vlastností. V samotném editoru se pouze tvoří herní svět, samotné psaní kódu probíhá v externím integrovaném vývojovém prostředí, které lze typicky s editorem propojit. Nejčastěji používané prostředí bývá Visual Studio, kód je ovšem možné psát v jakémkoliv textovém editoru. Tvorba světa v editoru probíhá pomocí takzvaného drag and drop vývoje. To je způsob, kdy vývojář jednotlivé objekty/komponenty přetahuje například pro umístění ve scéně nebo předání reference mezi komponentami, viz následující sekce. Tento styl vývoje je poměrně jednoduchý, jelikož vývojáři umožňuje svět „naklikat“. Pro vývoj v editoru existuje spousta tutoriálů, dokumentace a manuálů od samotné firmy, či bohaté komunity, která okolo vývoje v Unity vznikla.

V editoru se nachází několik základních podoken. Budou popsána jednotlivá okna dle obrázku 4.1, v editoru lze zapnout i další okna, tyto však

[1]Firma nabízí i placené licence, které poskytují určité rozšiřující možností, které jsou vhodné spíše pro firmy a v rámci projektu nebyly zapotřebí.

byly při vývoji nejčastěji používané. Asi nejdůležitější okno je okno Scene, nacházející se uprostřed. Zde se vytváří grafické prostředí aplikace. Rozmisťují se zde herní objekty, osvětlení a kamera. Při spuštění aplikace (tlačítko „play“ uprostřed nahoře) se okno přepne do pohledu Game. To slouží k testování provedených změn. Vývojář zde vidí jak bude výsledná hra/aplikace vypadat.

Vlevo uprostřed se nachází okno Hierarchy. Zde jsou zobrazeny všechny použité scény a herní objekty nacházející se v dané scéně. Objekty lze v rámci scény strukturovat do stromové hierarchie. Vlevo dole je okno Project, ve kterém je vidět struktura dat daného projektu. Jedná se typický souborový systém, je sem možno importovat externí assety. Z hlediska vývoje je poměrně důležité okno Console, nacházející se dole uprostřed. Zde se vypisují chybové a varovné hlášky, které v průběhu hry nastanou, případně je zde možné vypsat nějaké informace k ladění projektu. Vpravo se pak nachází okno Inspector. Zde se ukazují informace o aktuálně zvoleném herním objektu z hierarchie.

4.1.2 Scény

Scéna v Unity představuje kolekci herních objektů, které spolu nějakým způsobem spolupracují a v danou chvíli vytvářejí herní prostředí. Je to vlastně jakýsi kořen stromové hierarchie těchto objektů. Scén může být v rámci projektu více, v danou chvíli je však aktivní pouze jedna a mezi scénami se pak lze programově přepínat. Při sestavení projektu se pak určí jaké scény do něj mají být zahrnuty a nastaví se jejich indexace. Při spuštění aplikace se pak spustí scéna s indexem 0.

4.1.3 Herní objekty

Herní objekt (`GameObject`) je asi nejdůležitější koncept v rámci Unity Editoru. Herní objekty jsou jakési kontejnery, které samy o sobě nic nedělají. Z inspektoru editoru jim lze přiřadit tzv. *komponenty*. Každá komponenta přidává objektu nějakou vlastnost nebo chování, například to může být nastavení kolizí (`Collider`), způsob jeho grafického vykreslení (`Mesh Renderer`) nebo skript. Unity má spoustu komponent již v sobě vestavěných. V inspektoru pak lze také měnit některá nastavení daných komponent. Pro každou komponentu existuje v Unity Enigne stejnojmenná třída, a lze tak předat odkaz na konkrétní instanci komponenty skriptu a volat jejich metody, viz 4.1.4. Objekty a i jejich komponenty, mohou být ve scéně, ale lze je jak v editoru tak programově deaktivovat. Deaktivované objekty pak sice jsou

součástí scény, ale žádným způsobem ji neovlivňují. Pokud je na objektu deaktivována komponenta, tak ji sice objekt má, ale neposkytuje objektu svoje vlastnosti.

Nejdůležitější komponentou herních objektů je jeho transformace `Transform`. Tato komponenta je součástí každého objektu a nelze ji odstranit. Obsahuje informace o tom jaká je aktuální pozice objektu ve scéně, jaká je jeho rotace a velikost. Modifikací této komponenty lze pak objekt různě orientovat nebo měnit jeho rozměry. Obsahuje také metody pro práci s hierarchií objektu v rámci scény, lze tedy například získat referenci na svého předka, nebo reference na potomky.

Pro složitější objekty Unity umožňuje vytvoření takzvaného *prefabu*. To je jakási šablona pro další tvorbu a vkládání těchto objektů do scén. Obsahuje pak všechny komponenty a potomky daného objektu, ze kterého byla šablona vytvořena.

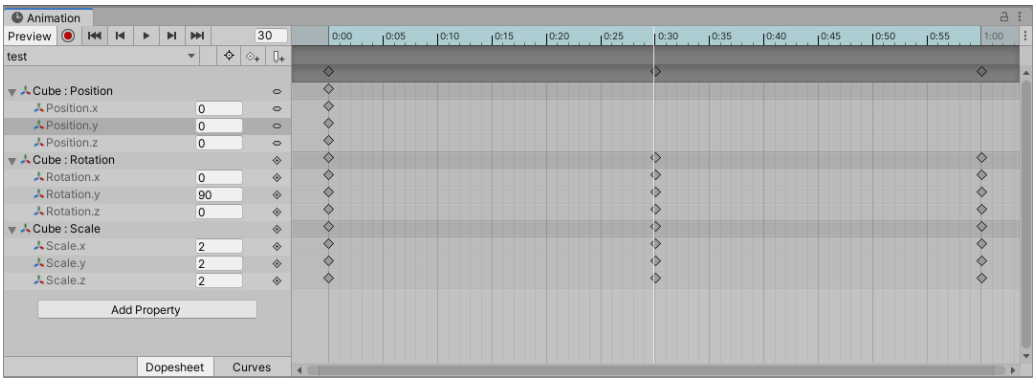
4.1.4 Skriptování

Kromě využívání vestavěných komponent je nutné do aplikace zavést nějakou vlastní logiku, k tomu slouží skripty. Skript je tedy naše vlastní komponenta, ve které můžeme definovat chování určitého objektu, měnit jeho vlastnosti v čase nebo reagovat na vstup od uživatele. K tomu aby byl kód skriptu vykonáván, musí být přidán jako komponenta některého aktivního herního objektu ve scéně. Pro programování vlastních skriptů je používán objektově orientovaný programovací jazyk C#. Jako skript se pak ve skutečnosti bere třída, která je potomkem třídy `MonoBehaviour` z Unity Engine API.

Třída `MonoBehaviour` skriptu poskytuje řadu metod, které jsou následně volané z enginu. Skripty mají svůj „životní cyklus“, na základě kterého jsou v konkrétním pořadí dané metody volány. Metody pak lze roztřídit na základě toho do jaké části cyklu patří. Jedná se o metody inicializační, starající se o herní fyziku, herní logiku, vykreslování a ukončování skriptu. Nejčastěji používané metody jsou `Start()`, což je metoda, která je volána pouze jedou za celou dobu existence skriptu a používá se typicky k inicializaci proměnných[2], a metoda `Update()`, která je volána jednou za každý snímek probíhající aplikace. Celý životní cyklus skriptů je možné najít v [19] v sekci Order of Execution for Event Functions.

Skriptům je možné definovat atributy, tak jak je tomu v objektově orientovaných jazycích zvykem. Je-li atributu nastaven modifikátor `public`, nebo

[2]Dalo by se říci, že se jedná o konstruktor skriptu, ovšem voláním této metody instance skriptu nevzniká, o to se stará samotný Unity Engine



Obrázek 4.2: Okno pro tvorbu animací v Unity Editoru

je mu předána vlastnost `[SerializeField]`[3], je možné nastavit hodnotu či předat referenci na konkrétní instanci z inspektoru Unity Editoru. V editoru je také možné nastavit pořadí v jakém jsou jednotlivé skripty vykonávány.

4.2 Animace

Unity pro práci s animacemi používá systém nazývaný *Mecanim*. V Unity existuje ještě jeden starší systém, který je označován jako Legacy a již se příliš nepoužívá, v enginu zůstal především pro zachování zpětné kompatibility starších aplikací. Z tohoto důvodu bude dále popsána práce pouze se systémem Mecanim.

Unity používá koncept animačních klipů (`AnimationClip`). Animace je vlastně pouze postupná změna pozice, rotace nebo velikosti nějakého vykreslovaného modelu v čase. Animační klip si je pak možno představit jako „nahrávku“, která obsahuje data o příslušné animaci. Animace se nejčastěji vytvářejí v externích softwarech, ve kterých se tvoří samotný model, jako je například Autodesk Maya, Autodesk 3ds Max nebo Blender. Do Unity je pak možné tyto objekty naimportovat i s jejich animacemi.

Vlastní animace je možné tvořit i v samotném Unity Editoru. K tomu slouží okno Animation, viz obrázek 4.2. V okně pak lze nad vybraným herním objektem ze scény vytvořit animaci.

Animace je tvořena pomocí *klíčových snímků* (`Keyframe`) a *animačních křivek* (`AnimationCurve`). Klíčové snímky se z okna dají nahrát kliknutím na červené tlačítko. To umožní zaznamenat transformaci[4] objektu ve zvoleném čase animace. Křivky pak určují jakým způsobem se bude dopočítávat

[3]Výhodou tohoto přístupu je, že atribut může být nastaven jako privátní, ale jeho hodnotu je stále možné měnit z editoru. Není jej však možné měnit z jiného skriptu.

[4]Komponenta animovaného herního objektu, viz sekce 4.1.3.

(interpolovat) hodnota transformace mezi dvěma po sobě jdoucími klíčovými snímky. Například na obrázku 4.2 je 30. snímek animace klíčový a objekt má y rotaci nastavenou na 90 stupňů. V prvním snímku, který je také klíčový, je tento úhel nastaven na 0, pokud by tedy byla křivka mezi těmito snímky pro daný úhel lineární, bude rotace v 15 snímku nabývat hodnoty 45 stupňů.

Pro ovládání animací slouží v Unity *Animator Controller*. V editoru pro něj existuje vlastní okno. Lze si tento kontrolér představit jako konečný stavový automat, kde jednotlivé stavy jsou přidané animační klipy. Přechody mezi klipy jsou umožněny pomocí parametrů, které mohou být různého typu (float, int, bool) a přechod se provede je-li splněna nějaká podmínka se zvoleným parametrem[5]. Na animovaný herní objekt se musí přidat komponenta `Animator`, které je nutné předat referenci na vytvořený kontrolér. Animace je pak možné ovládat pomocí metod, které tato komponenta nabízí.

Výše popsaný systém byl používán v prvotních fázích projektu. Roli animovaného objektu v našem případě hraje 3D model horní končetiny. Měli jsme naměřené pohyby a pro ně v Blendru ručně vytvořené animace[6]. V Unity projektu se pak pro tento objekt vytvořil `Animator` a Animator Controller, do kterého se jednotlivé animace přidaly. Následně se pak ze skriptu získala reference na `Animator` a volala se metoda pro přehrávání animace aktuálně vykonávaného pohybu.

Postupem vývoje jsme ovšem jako jednu z největších možností aplikace chtěli implementovat definování vlastního pohybu bez nutnosti jejího překladu a znovu sestavení. Šlo by tedy o to, že se naměří nový pohyb, ke kterému by se automaticky vytvořila animace a následně by byla přidána k rehabilitační aplikaci. Ukázalo se ovšem, že pomocí systému Mecanim přidávat animace za běhu programu nelze. Vznikl tak software, který dokáže z naměřených dat o pohybu automaticky vytvořit animaci pro model horní končetiny, jehož výstupem jsou data o vytvořené animaci. Software pracuje na principu inverzní kinematiky a je výsledkem bakalářské práce Alexe Königa [12]. Animační data jsou pak poskytnutá rehabilitační aplikaci přes *StreamingAssets* a jsou v aplikaci využívána k animování aktuálně prováděného pohybu. Konkrétní implementace, viz sekce 5.3.4.

[5]Například se definuje parametr typu int a v přechodu mezi dvěma animacemi se nastaví, že k přechodu dojde pokud je jeho hodnota větší než nějaká předem definovaná hodnota.

[6]Model a jeho animace jsou práce Alexe Königa [12].

4.3 VR integrace

Pro integraci virtuální reality s Unity typicky vývojáři daného headsetu poskytují ostatním vývojářům kteří chtějí pro daný headset tvořit software tzv. SDK (Software Development Kit), tedy sadu vývojových nástrojů, které umožňují práci s daným VR systémem. Jednotlivé knihovny jsou pak dostupné buď na Unity Asset Store, nebo je možné je do projektu importovat ze stránek vývojáře.

V projektu je pracováno s headsetem HTC Vive, pro který existuje vývojová sada SteamVR. Ta je dostupná přímo z Unity Asset Store odkud ji je možno přímo do Unity projektu naimportovat. Po dokončení vlastního importu, je víceméně integrace hotová a dále se jedná pouze o využívání možností, které knihovna poskytuje. Důležitou součástí knihovny je prefab `[CameraRig]`, který reprezentuje kameru, kterou bude vykreslováno na displej samotného headsetu. Prefab také obsahuje ohraničení zvoleného herního prostoru v rámci Vive. Z důvodu omezené práce s ovladači nebyly další možnosti knihovny využívány.

V projektu je ještě částečně pracováno s další knihovnou, a to Vive Input Utility, která je také dostupná z Unity Asset Store. Knihovna je používaná především za účelem svázání herního objektu s nějakým sledovatelným objektem (headset, ovladače, trackery) systému HTC Vive. Poskytuje skript `VivePoseTracker`, který je možné přidat jako komponentu herního objektu, který bude představovat příslušný sledovaný objekt. V inspektoru skriptu pak lze nastavit roli. Roli je pak nutné svázat s konkrétním zařízením.

5 Software

Cílem této kapitoly je seznámit čtenáře se softwarem, který je výsledkem této bakalářské práce. Nejprve bude popsána hlavní myšlenka daného softwaru a jaké náležitosti bylo zapotřebí brát v úvahu při jeho vytváření, viz sekce 5.1. Následně proběhne popis toho co software umí a nabízí a jakým způsobem s ním může interagovat pacient a terapeut a jakou roli v rámci softwaru každý z nich hraje, viz sekce 5.2. Poté bude popsáno, jakým způsobem jsou jednotlivé vlastnosti a komponenty aplikace implementovány v rámci enginu Unity, viz sekce 5.3. Testování aplikace bude popsáno v sekci 5.4. Nakonec proběhne diskuze možných nedostatků, které se ve výsledné aplikaci nachází, a možných budoucích rozšíření, která by při další práci na projektu mohla být implementována, viz sekce 5.5.

5.1 Návrh

Pacienti s neurologickými potížemi při rehabilitacích provádějí různá cvičení (pohyby a hry), pod dohledem terapeuta, za účelem zlepšení hybnosti horní končetiny. Cílem pacientů by mělo být dané pohyby provádět co nejlépe bez „podvádění“[1].

Základní myšlenka celého projektu je následující. Bude nutné nasbírat data o jakémsi „perfektním“ pohybu. Perfektním pohybem je myšlen jakýkoliv rehabilitační pohyb, který je vykonán zdravým člověkem, nejlépe takovým, který ví, jak by měl být pohyb proveden správně. Nejčastěji se bude pravděpodobně jednat o terapeuta, který s pacientem rehabilitace provádí. Získání dat o tomto pohybu je úkolem aplikace Jakuba Franka [9], která umožňuje nahrání pohybu za pomoci systému HTC Vive. Pohyb je definován polohou a rotací několika senzorů v čase. Senzory podchycující pohyb jsou čtyři, a to HMD a tři Vive trackery, jeden umístěný na zápěstí, jeden na paži a poslední na hrudníku. Výstupem aplikace jsou pak soubory, jeden pro každý senzor, ve kterých se nacházejí data o provedeném pohybu, tedy jednotlivé časové snímky rotace a polohy senzoru.

Úkolem pacienta je pak co možná nejlépe předcvičený pohyb replikovat. Cílem rehabilitační aplikace, která je výsledkem této bakalářské práce, je především tedy umožnit pacientům provádět definované pohyby a poskytnout terapeutům informace o vykonaných pohybech.

[1] Pacienti si například pomáhají při pohybu paže vzhůru zvednutím hrudníku.

V rehabilitační aplikaci bude potřeba umožnit terapeutovi definovat a přidat nový pohyb. Aplikace pak bude data definující pohyb načítat do struktur. Je tedy potřebné v aplikaci připravit datové struktury, které budou reprezentovat jednotlivá cvičení. Následně je třeba ovládat a analyzovat pohyb prováděný pacientem. Za účelem analýzy pohybu v reálném čase vznikla knihovna AAPD (Automatic Arm Position Detection), jež je výsledkem bakalářské práce Vítka Poóra [14] a dále upravována Jakubem Frankem. Knihovna poskytuje rozhraní, které provádí analýzu prováděného pohybu a poskytuje o něm rehabilitační aplikaci informace. Jedná se o aktuální fázi a kvalitu pohybu. Na základě těchto údajů bude pak možné určit, zda je prováděný pohyb správný nebo není. O pohybu bude možné sbírat další statistky.

Samozřejmě bude nutné pacienta i terapeuta informovat o tom, co se aktuálně v aplikaci děje, pomocí uživatelského rozhraní. Největší výhodou aplikace oproti regulérním rehabilitacím pak je možná vizualizace předcvičeného pohybu terapeutem pacientovi. Další výhodou je poskytnutí dodatečných informací o vykonaném pohybu terapeutovi.

5.2 Možnosti softwaru

Software je rozdělen do dvou herních scén. První, konfigurační, která se načte při spuštění aplikace, a druhá, sloužící k provádění rehabilitačních cvičení. Jak bylo výše zmíněno, aplikace je zaměřena pro používání dvěma uživateli, terapeutem a pacientem. V následujících sekcích bude popsáno, jakým způsobem mohou uživatelé v jednotlivých scénách pracovat a jakou hrají v kontextu aplikace roli.

5.2.1 Použití HTC Vive

Pro podporu virtuální reality používá rehabilitační aplikace systém HTC Vive, viz sekce 3.3. Ke správnému fungování je používán headset a 6 trackerů. Trackery jsou používány pro sledování pozice a rotace následujících částí těla a objektů: levé zápěstí, levá paže, pravé zápěstí, pravá paže, hrudník a židle, na které pacient bude sedět a provádět cvičení. Teoreticky je možné v danou chvíli používat trackery pouze 4, jelikož jednotlivá cvičení jsou rozdělena podle toho, zda se jedná o cvičení zaměřené na pravou nebo levou horní končetinu. Z toho tedy vyplývá, že při provádění cvičení na pravou ruku není nutné mít nasazené (nemusí být ani zapnuté) trackery pro levou ruku a naopak. Pro jednotlivé trackery byla 3D tiskárnou vytisknuta podstava, která jde k trackeru přišroubovat a provléknout jí páskou se suchým zipem,

takže je pak možné tracker připevnit na horní končetinu. Pro připevnění k hrudníku je použit hrudní popruh jaký je používán například pro kamery GoPro.

Důležitým aspektem pro funkčnost aplikace je správná pozice a orientace trackerů nasazených na příslušných částech těla, jelikož při analýze pohybu prováděného pacientem dochází k porovnávání aktuální pozice a rotace trackeru vůči nejbližšímu bodu fáze pohybu předcvičeného terapeutem. To znamená, že pro správné vyhodnocování pohybu je nutné, aby měl pacient trackery nasazené stejně jako je měl nasazené terapeut při nahrávání daného pohybu. Pokud by tomu tak nebylo, bude logicky pohyb vyhodnocován špatnou kvalitou, přestože by pohyb byl vykonáván správně. Za tímto účelem vznikla konvence mezi všemi aplikacemi projektu[2] pro pozice a rotace trackerů, kterou je *nutné* dodržovat[3]. Konvence je následující. Trackery na zápěstí jsou umístěny stejným způsobem jako náramkové hodinky a jejich orientace je taková, že LED dioda indikující stav trackeru směřuje směrem k dlani. Trackery na paži jsou umístěny pod deltovým svalem a dioda směřuje směrem k lokti. Hrudní tracker by měl být umístěn zhruba uprostřed hrudní kosti mezi prsními svaly a dioda směřuje směrem k hlavě. V aplikaci je nutné ještě příslušné objekty reprezentující trackery spárovat s trackery skutečnými, viz příloha A.2.3. Ovladače systému HTC Vive nejsou v aplikaci nikde použity. Důvodem je fakt, že většina pacientů má problémy s jemnou motorikou, viz sekce 2.2, a tudíž velmi často nejsou schopni ani některé předměty uchopit.

5.2.2 Role pacienta

Úlohou pacienta v rehabilitační aplikaci, je provádět předem definovaná rehabilitační cvičení. Cvičeními jsou myšlené jak pohyby, předcvičené terapeutem a přidané k aplikaci, tak rehabilitační hry. Pacient je tedy uživatel, který má na sobě sestavu HTC Vive, tak jak bylo popsáno v předchozí sekci. Je vhodné, aby pacient měl dostupný nějaký zdroj zvuku, sluchátka nebo lépe reproduktory[4], jelikož aplikace pro určitá oznámení používá audio signály, viz dále.

První, tedy konfigurační, scéna je z pohledu pacienta vcelku nezajímavá a bude tedy popsána v následující sekci. Po provedení konfigurace se pacient

[2]Rehabilitační aplikace, aplikace pro měření rehabilitačního pohybu [9], aplikace pro automatickou tvorbu animací rehabilitačního pohybu [12].

[3]Dodržením této konvence dojde ke značnému snížení zanesené chyby vyhodnocování.

[4]Reproduktory jsou asi lepší volba, jelikož pacient stále může komunikovat s terapeutem.

objeví v hlavní herní scéně aplikace. Ve scéně se nachází několik prvků, které může pacient vidět. První čeho si nejspíše všimne je samotné virtuální prostředí, reprezentované 3D modelem[5] anglické památky Stonehenge[6]. Model slouží pouze k oživení prostředí a nelze s ním nijak interagovat. Prostředí je herní objekt, tzn. bylo by poměrně jednoduché přidat prostředí více a umožnit mezi nimi přepínat. Dalším prvkem jsou herní objekty reprezentující Vive trackery. Konkrétní reprezentace trackeru může být různá dle konfigurace, viz sekce 5.3.5. U modelu se nachází popisek, který určuje, kde by se měl tracker nacházet (levé zápěstí atd.). Pokud jsou trackery nasazeny korektně (ve smyslu rotace), pak by měl být popisek čitelný, jinak bude pravděpodobně nějakým způsobem převrácený. Před pacientem je ve scéně plátno, na které se budou vypisovat aktuální pokyny a informace o prováděných cvičeních.

Terapeut pacientovi postupně spouští cvičení. V aplikaci se aktuálně nachází jedna rehabilitační hra, sbírání míčů, viz obrázek 5.1. Jedná se o velmi jednoduchou hru, kde úkolem pacienta je posbírat kuličky (herní objekty reprezentující „míče“), které se ve scéně zobrazí. Pacient se musí dotknout kuliček virtuální dlaní, která se mu při hře zobrazí u zápěstního trackeru. Kulička, kterou je třeba sebrat je obarvena modře, sebrané jsou zelené a kuličky, které budou na řadě červené. Při sebrání kuličky je přehrán zvukový signál. Pokud se náhodou kulička, která se má sebrat, nenachází v zorném poli pacienta, zobrazí se mu šipka[7], ukazující směrem ke kuličce.

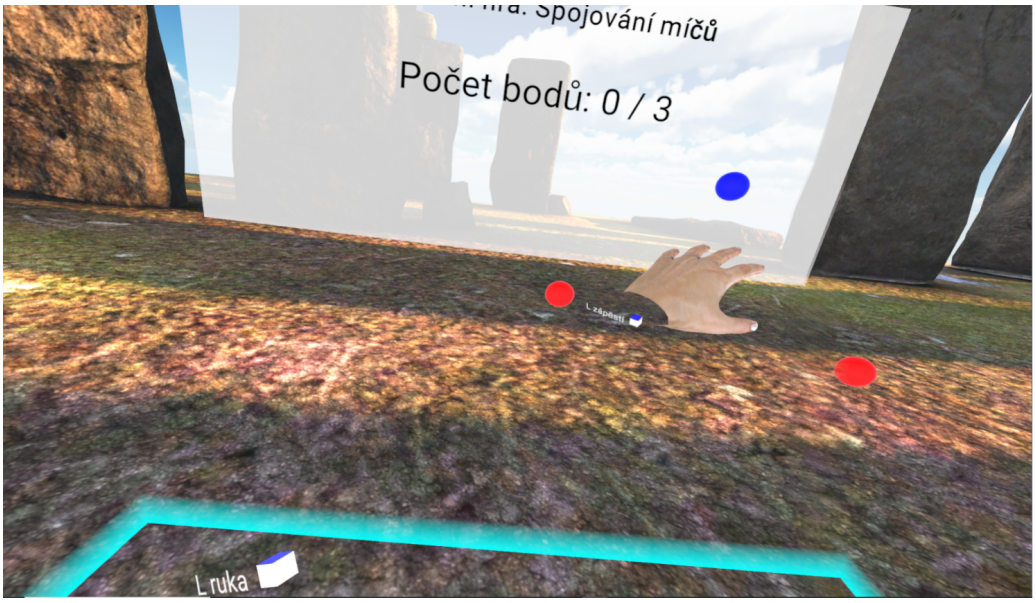
Dalšími cvičeními jsou pohyby. Před spuštěním pohybu by se měl pacient dostat do neutrální polohy. Neutrální poloha se používá k transformaci tunelu používaného pro analýzu pohybu, viz sekce 5.3.3. Pro naměřené pohyby byla stanovena konvence neutrální polohy, která je zaznamenána aplikací pro sběr dat pohybů [9]. Poloha by měla být v sedě s rovnými zády s dlaněmi položenými na kolenou.

Po spuštění pohybu se zobrazí několik nových prvků, viz obrázek 5.2. Je zobrazena vizualizace tunelu. Jedná se o trajektorie, reprezentované barevnými křivkami, které zobrazují, jakým způsobem probíhal naměřený pohyb pro každý tracker. Ukazuje pacientovi, jak by měl trackery pohybovat pro dosažení „perfektního pohybu“. Dalším prvkem je 3D model horní končetiny, který bude prováděním pohybu animován. Animace je založena na předcvičeném pohybu, to znamená, že nebude přímo odpovídat aktuálně prováděnému

[5]Model není naší prací. Byl stažen z: `https://sketchfab.com/3d-models/stonehenge-37cfc2bb99944703b5d57ea281030ca6` a následně importován do Unity projektu.

[6]`https://www.mundo.cz/anglie/stonehenge`

[7]Pokud je tak nastaveno terapeutem, viz sekce 5.3.5



Obrázek 5.1: Ukázka prvků hry sbírání míčů

pohybu, ale bude ukazovat, jak by paže měla vypadat v ideálním případě.

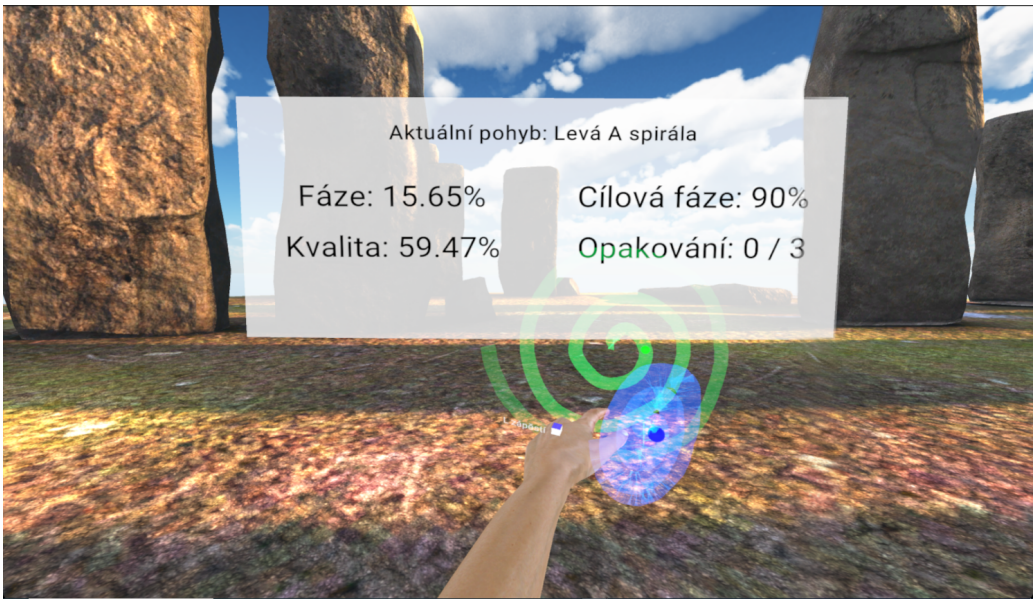
Vyhodnocovaní pohybu začne v momentě, kdy se pacient dostane z neutrální polohy do počáteční polohy pohybu, viz sekce 5.3.2. Je při tom přehrán zvukový signál indikující, že pohyb začal. Cílem pacienta je dostat se po křivce s dostatečnou kvalitou do nějaké definované fáze. Kvalitu jakou má pohyb mít a cílovou fázi stanovuje terapeut. Po provedení pohybu s dostatečnou kvalitou je přehrán zvuk indikující úspěch, a pacient může provádět další opakování, pokud ještě nějaké následuje. Počet opakování pohybu je také stanoven terapeutem.

Pohyby jsou rozděleny na dva typy, podle toho zda je počáteční a koncová fáze pohybu ve stejné poloze či nikoliv. Pokud tomu tak je a pacientova fáze pohybu, při snaze dostat se do specifikované fáze, nějakým významným způsobem klesne, je přehrán zvukový signál oznamující selhání. Pokud nastane tato situace, pak se musí pacient vrátit do počáteční polohy a zkusit pohyb znovu, tzn. opakování pohybu se již nebude počítat.

Po vykonání každého cvičení se na plátně zobrazí jeho shrnutí a informace o tom, jaké cvičení následuje (pokud takové je).

5.2.3 Role terapeuta

Terapeut hraje v rámci rehabilitační aplikace podstatnou roli. Jeho úkolem je aplikaci nastavovat a ovládat. Jelikož pacienti nemohou mít ovladače, je veškerý vstup od uživatele prováděn klávesnicí, a tedy terapeutem. Nastavo-



Obrázek 5.2: Ukázka prvků u pohybů

váním je myšleno zejména sestavování posloupnosti rehabilitačních cvičení a úprava parametrů v konfiguračních souborech. Konfigurace a její možnosti budou dále popsány v sekci 5.3.5. Terapeut má možnost k aplikaci připojit a definovat nový libovolný rehabilitační pohyb, s využitím vlastností všech tří aplikací[8] projektu, viz příloha A.2.4.

První, konfigurační scéna, slouží k změření poměru rozměru těla pacienta ku rozměru těla terapeuta[9]. Tento poměr je pak použit pro úpravu poloh naměřeného pohybu tak, aby pohyb mohl být vykonáván i pacientem[10]. Rozměr se měří od hlavy, tedy headsetu, k pravému zápěstnímu trackeru. Logicky vyplývá, že by terapeut i pacient měli být při měření ve stejné poloze. Byla zavedena konvence, že je připaženo, jelikož to je poloha, která by pacientům neměla dělat větší potíže, na rozdíl například od upažení. Aplikace si rozměry mezi spuštěními pamatuje, takže je možné rozměr terapeuta změřit pouze jednou a následně měřit pouze pacienty.

Ve scéně se nachází tabule, stejně jako v hlavní scéně, na které je popis zda se aktuálně jedná o terapeuta či pacienta. Dále jsou zde vypsány jednotlivé rozměry uživatelů v metrech. Přepnutí mezi pacientem a terapeutem zajišťuje klávesa S. Změření rozměru těla je provedeno stisknutím mezerníku.

[8]Rehabilitační, nahrávací a aplikace pro tvorbu animací.

[9]Respektive uživatele jenž prováděl předcvičení pohybů.

[10]Například pokud by byl terapeut vysoký a pacient nízký, mohlo by dojít k situaci, že pacient by pohyb ani nemohl vykonat, protože by nemusel dosáhnout tak daleko jako terapeut.

Pro přepnutí do hlavní scény slouží klávesa Enter.

V hlavní scéně, kromě toho že na monitoru uvidí to co pacient, má terapeut ještě vlastní uživatelské rozhraní, které se zobrazuje pouze jemu a nikoliv pacientovi, viz obrázek 5.3. Jedná se o čtyři menší okna, která je možné minimalizovat kliknutím na bílé tlačítko. V levém okně se nacházejí informace o prováděných cvičení, které se vypisují i na plátno ve scéně.

Obrázek 5.3: Uživatelské rozhraní zobrazované pouze na monitoru terapeuta

Dole jsou dvě okna, která slouží k nahrávání a přehrávání pohybů prováděných pacienty. Levé okno slouží pro nahrávání pohybu, nejprve je nutné zvolit složku, kam se pohyb nahraje. Pak v případě, že pacient provádí pohyb je možné pohyb začít nahrávat. Z pravého okna je pak možné vybrat složku nahraného pohybu a ten přehrát. Tuto akci je možné provést kdykoliv, kdy se neprovádí nějaké cvičení. Pohyb se pak přehrává tak, že se ve scéně zobrazí pohyb trackerů a headsetu, tak jak jimi hýbal pacient, a vizualizace tunelu, který pacient pacient při daném pohybu používal.

Z pravého okna pak může terapeut měnit nastavení 3D modelu horní končetiny. Je zde možné nastavit jeho průhlednost pomocí posuvníku. V okně je výpis o jaký index nastavení se jedná. Je možné mít pro model více nastavení a pak jedno konkrétní používat. Myšlenka je taková, že každý pacient bude preferovat nastavení trochu jinak a takto je možno udělat nastavení pro více pacientů. Mezi nastaveními se přepíná klávesami 0–9 z horní řady klávesnice (tedy ne z numerické). V okně je ještě možné dopsat nějakou poznámku, tag, který umožní lehčí zapamatování, které nastavení přísluší jakému pacientovi, oproti indexu. S nastavením modelu se ještě pojí jeho poloha. Tu terapeut může měnit pomocí šipek a numerických kláves 2 a 8. Šipky vlevo a vpravo slouží k posunu ve směru horizontálním, nahoru a dolu ve směru

vertikálním a numerické klávesy v prostoru před či za pacienta. Posuny se vztahují k pravé ruce a levá ruka je pak symetricky posouvána s pravou. Takto je možný posun v daném směru o jeden centimetr, pro větší posun je pak možné použít klávesy s kombinací levý SHIFT pro posun o 10cm nebo levý CTRL posun o 5cm. Tato nastavení jsou perzistentní a při dalším spuštění aplikace se použije nastavení používané při předchozím vypnutí aplikace.

Terapeut v hlavní scéně spouští jednotlivá cvičení mezerníkem. Po skončení všech cvičení se aplikace vypíná kombinací levý ALT + F4, nebo klávesou ESCAPE. Po vypnutí aplikace se do složky ExerciseSummaries[11] zapíše soubor, obsahující statistiky provedených cvičeních. Soubor je tvořen pokaždé nový a obsahuje v názvu datum a čas vypnutí aplikace.

5.3 Implementace

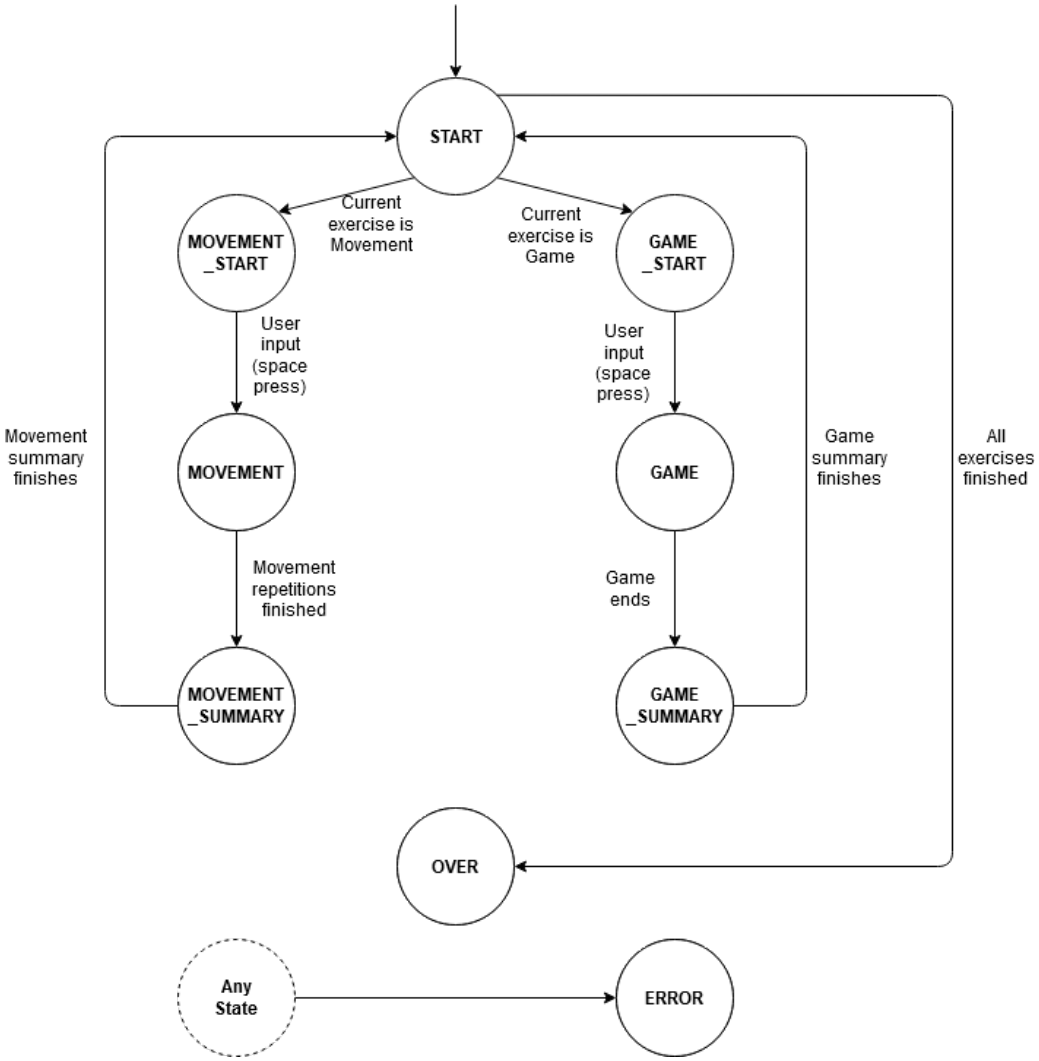
Implementace rehabilitační aplikace proběhla v herním enginu Unity[12], viz kapitola 4. Jak již bylo zmíněno, aplikace je tvořena dvěma herními scénami. V první probíhá krátká konfigurace a ve druhé je možné provádět samotnou rehabilitaci. První scéna je velmi jednoduchá a obsahuje pouze jeden skript `SetUpManager`, který pouze změří vzdálenost mezi trackerem na zápěstí a HMD kamerou, voláním metody `Vector3.Distance()` pro zvoleného uživatele, viz sekce 5.2.3. Změřené hodnoty uloží pomocí metody `PlayerPrefs.SetFloat()`, a jejich poměr je předán nastavení knihovny AAPD. Při dalším spuštění aplikace jsou rozměry zpětně získané metodou `PlayerPrefs.GetFloat()`.

Dále bude popsána hlavní funkcionalita druhé scény, nebude zde popisovaná veškerá implementace, ale pouze části, které jsou důležité nebo nějakým způsobem zajímavé. Veškeré zdrojové kódy aplikace jsou dostupné na repositáři `https://gitlab.com/frankkuba/vr-arm-motion` a v příloze práce, viz sekce A.3.

Herní objekty tvořící druhou scénu je možné rozdělit do několika kategorií. Objekty reprezentující jednotlivé trackery, 3D modely (horní končetiny, dlaně, židle, a vlastní virtuální prostředí), objekty tvořící uživatelské rozhraní, a pak typicky prázdné objekty, které slouží pouze jako kontejnery pro běžící skripty. Skripty těchto objektů typicky načítají konfiguraci, nebo dále ovládají chod aplikace.

[11] Adresář se vytvoří v kořenovém adresáři aplikace.

[12] V projektu jsou použity různé metody a třídy poskytované z API tohoto engine, viz [19], v textu nebude zmiňováno zda se jedná o vlastní třídu/metodu nebo zda je z Unity API.



Obrázek 5.4: Graf stavů aplikace a jejich přechodů

Data jsou aplikaci předávána pomocí konfiguračních souborů, viz sekce 5.3.5, ze kterých jsou načítány a tvořeny instance příslušných datových struktur (budou popsány v dalších sekcích). Skripty si pak kromě herních objektů předávají reference na datové struktury.

Důležitou strukturou aplikace je třída `AppStateManager`, obsahující statickou vlastnost (C# property) `CurrentAppState` typu `AppState`. `AppState` je enum, jehož hodnoty reprezentují jednotlivé stavy aplikace. Skripty pak na základě aktuální hodnoty vlastnosti provádí různé akce a podle dění v aplikaci stav mění. Hodnoty a jejich přechody je možné reprezentovat grafem, viz obrázek 5.4.

5.3.1 Uživatelské rozhraní

Uživatelská rozhraní, dále zkráceně GUI (Graphical User Interface), jsou v aplikaci řešena pomocí Unity komponenty `Canvas`. Potomci herního objektu s touto komponentou jsou jednotlivé objekty tvořící prvky GUI (texty, tlačítka...). Pro zobrazení jednotlivých textů GUI byla použita knihovna TextMesh Pro, která oproti nativním textům Unity poskytuje výrazně lepší kvalitu vykreslování.

V komponentě `Canvas` je možno určit, kde se bude zobrazovat, tzv. render mode.

- Screen Space - Overlay — GUI je vykresleno přes obrazovku displeje.
- Screen Space - Camera — GUI je vykresleno před vybranou kamerou.
- World Space — GUI je vykresleno ve scéně je viditelné všemi kamerami.

V aplikaci je využíváno všech tří možností. Overlay je využíváno pro zobrazení GUI terapeutovi, tedy pouze na obrazovku monitoru. Camera se používá pro zobrazení pomocné šipky před HMD pacienta, při hře sbírání míčů. World Space je používáno pro zobrazování pláten v obou scénách.

Pro každou ze tří možností pak existují skripty, které zajišťují vykreslování textů, případně obsluhu tlačítek. Bude popsána implementace plátna ve scéně, pro GUI monitoru je implementace velmi podobná.

Plátno reprezentuje herní objekt UI, který má jako komponentu předaný skript `UIController`. Objekt jako potomky obsahuje prázdné herní objekty, které slouží pouze pro sdružování textů, které jsou zobrazeny společně. Skript tedy při inicializaci (metoda `Start()`) nejprve podle názvu metodou `Find()` najde tyto potomky. Následně pak v metodě `LateUpdate()`, na základě aktuálního herního stavu `AppStateManager`, aktivuje příslušného potomka a zbytek deaktivuje. Z potomků se najdou příslušné instance textů TextMesh Pro a je jim předaná hodnota formátovaného řetězce metodou `Language.GetString()` (třída `Language` bude popsaná v sekci 5.3.5). Data pro formátované řetězce jsou typicky nějaké hodnoty o prováděných cvičeních. Reference na cvičení jsou `UIControlleru` předána ze skriptu `ExerciseController`, viz sekce 5.3.2.

5.3.2 Cvičení

Pro práci s cvičeními vzniklo několik datových struktur a skriptů. Jelikož mají všechna cvičení určité společné vlastnosti, vznikla abstraktní třída

`Exercise`. Společnými vlastnostmi cvičení jsou: jejich definice, abstraktní třída `ExerciseDefinition`, statistky cvičení, abstraktní třída `ExerciseStats` a jméno daného cvičení. Třída `Exercise` poskytuje základní konstruktor, kterému je jako parametr předána reference na definici cvičení. Instance této třídy reprezentují cvičení, která se budou v průběhu rehabilitace provádět.

`ExerciseDefinition` specifikuje data, která jednoznačně definují cvičení. Každá definice cvičení obsahuje ID, unikátní řetězec, pomocí kterého lze definice rozlišovat. Dále obsahuje stranu, enum `Side`, obsahující hodnoty `RIGHT` a `LEFT`, která určuje na kterou horní končetinu je cvičení zaměřeno. Z těchto hodnot je tvořen konstruktor třídy.

`ExerciseStats` slučuje společné statistiky jednotlivých cvičení, což je mezi všemi cvičeními pouze čas trvání prováděného cvičení. Třída také poskytuje abstraktní metodu `WriteToXML()`, která pomocí předaného `XMLWriteru` umožňuje zápis statistik do souboru.

Konkrétní cvičení, definice a statistiky pak dědí od těchto abstraktních tříd, viz dále.

Cvičení ovládá skript `ExerciseController`, jenž je komponentou stejnojmenného herního objektu. Skriptu je z inspektoru předána reference na instanci `UIController`, viz sekce 5.3.1, a instanci `ConfigDataReader`. Při inicializaci si skript převezme referenci na seznam cvičení z `ConfigDataReader` pro rehabilitaci. Pokud načtení cvičení neproběhlo bez problémů, přepne skript aplikaci do stavu `ERROR`. Pokaždé když se aplikace dostane do stavu `START`, vybere ze seznamu další cvičení a uloží si na něj referenci do atributu `CurrentExercise`, kterou dále předá `UIController`. Podle typu instance cviční pak přepne aplikaci do stavu `MOVEMENT_START` či `GAME_START` a předá tak zodpovědnost pro kontrolu daného cvičení jinému kontroléru, viz následující podsekce. Při vypnutí aplikace skript zapíše do souboru statistiky o provedených pohybech a hrách.

Pohyby

Pro práci s pohyby vznikly datové struktury `Movement`, `MovementDefinition`, `MovementStats`, které dědí od příslušných abstraktních tříd popsaných výše, a skript `MovementController`, který ovládá průběh pacientem vykonávaného pohybu. Tyto třídy již abstraktní nejsou a pouze rozšiřují implementaci jejich předků. Všechny zatím definované pohyby splňují stejné vlastnosti, takže nebylo třeba pro každý tvořit vlastní třídu/skript. Dělí se pouze na dva typy, který specifikuje jejich definice, viz dále.

`Movement` obsahuje oproti generickému cvičení několik dalších atributů.

Jsou to atributy načítané z konfiguračního souboru, viz sekce 5.3.5. `PhaseTarget`, cílová fáze pohybu do které se musí pacient dostat, `QualityThreshold`, reprezentující kvalitu pohybu, kterou pacient musí po celou dobu provádění pohybu splňovat, `RepetetionsToHit`, počet opakování tohoto pohybu, které má pacient vykonat, a pak atributy indikující, zda se má nějaká část vizualizace tunelu schovat či nikoliv, viz sekce 5.3.3. Další atributy třídy jsou modifikovány při průběhu pohybu. `InitialStartingPosition` indikující, zda se pacient dostal do počáteční polohy, a pak jsou zde aktuální hodnoty fáze `CurrentPhase`, kvality `CurrentQuality` a počtu splněných opakování `CurrentRepetitions`.

Definici pohybů rozšiřují následující atributy. `DataPath`, což je souborová cesta k adresáři, kde se nacházejí naměřená data předcvičeného pohybu terapeutem. `AnimationData`, třída obsahující načtená data pro práci s animací 3D modelu horní končetiny, viz sekce 5.3.4. A atribut `HasApex`, který indikuje zda má pohyb vrchol či nikoliv. Pohyb má vrchol, jestliže se z počáteční polohy dostane do jakéhosi maxima, ze kterého se následně vrací stejnou trajektorií zpět do počáteční polohy, kde opakování pohybu končí[13]. Vrchol pohybu automaticky určí knihovna AAPD. Tento atribut pohyby dělí na dvě skupiny, a s každou je potřeba pracovat trochu jinak, viz dále. Tím, že lze definiční data pohybů předávat jako soubory, které lze poměrně jednoduše načítat, je zajištěna možnost definice nového pohybu bez nutnosti překladu aplikace.

Kromě doby trvání vykonání pohybu jsou v průběhu sbírány ještě další dvě statistiky. Jedná se o průměrnou kvalitu vykonaného pohybu a průměrné vzdálenosti pozic trackerů pacienta při vykonaném pohybu od pozic trackerů předcvičeného pohybu terapeutem[14]. Průměrná kvalita je počítána váženým průměrem, kde váha pro každou aktuální kvalitu je doba trvání této kvality, tedy `Time.deltaTime`, jelikož čas mezi voláními Unity metody skriptů `Update()` má proměnlivou délku. Třída také překrývá metodu `WriteToXML()`, kde je implementována funkcionalita zápisu statistik pohybu do xml souboru.

`MovementController` je skript, a tedy komponentou stejnojmenného herního objektu, který je potomkem herního objektu `ExerciseController`.

[13]Pro ilustraci máme například naměřené pohyby diagonál a spirál. Diagonála je pohyb, kdy pacient hýbe horní končetinou například od kotníku a snaží se ji zvednout nad hlavu. Z pozice nad hlavou se pak vrací zpět ke kotníku, tzn. pozice nad hlavou je vrchol pohybu. Kdežto u spirály je provedena spirála pouze jedním směrem a není nutné se zpětně vracet stejnou trajektorií.

[14]Pokud by tedy vzdálenosti byly nulové, tak to znamená, že pacient naprosto přesně kopíroval pohyb terapeuta. Což je samozřejmě ve skutečnosti nemožné, pacient by se každopádně měl snažit aby vzdálenosti byly co nejmenší.

Z inspektoru je skriptu předána reference na transformaci trackerů (bez trackeru židle), reference na 3D modely horní končetiny a instanci skriptu `TunnelController`, blíže popsán v sekci 5.3.3. Skript pak v metodě `Update()` provádí obsluhu na základě toho v jakém stavu `AppState` se aplikace nachází. Pokud se aplikace nenachází ve stavu `MOVEMENT_START` či `MOVEMENT`, tak skript pouze deaktivuje modely horních končetin a nic jiného nedělá.

Ve stavu `MOVEMENT_START` je především provedena inicializace struktury `Tunnel`. Nejprve je na základě definice pohybu vytvořen *`IDataProcessor`*, který je společně s pohybem předán metodě `CreateNewTunnel()` ze třídy `TunnelController`. Tyto struktury jsou součásti knihovny AAPD, viz [14][9]. `Tunnel` a jeho procesor dohromady tvoří analyzátor aktuálně prováděného pohybu. Tunel si lze představit jako jakési okolí (prostor) okolo naměřených dat předcvičeného pohybu, ve kterém se může pacient pohybovat, aby byl jeho pohyb započítán. S vyšším nárokem na požadovanou kvalitu (`QualityThreshold`) se tento prostor zužuje. V tomto stavu je ještě na základě strany `Side` pohybu připraven příslušný model horní končetiny a trackery, ze kterých bude prováděna analýza pohybu. Je zde také připravena instance třídy `AnimationPlayer`, pro přehrávání animace, viz sekce 5.3.4.

Po provedení inicializace se pak čeká na vstup od terapeuta, tedy na stisk mezerníku. V tu chvíli je aplikace přepnuta do stavu `MOVEMENT` a je volána metoda `TunnelController.StartProcessing()`, viz sekce 5.3.3. V každém volání metody `Update()` je prováděna následující sekvence. Nejprve je datovém procesoru předána transformace všech trackerů, voláním metody `IDataProcessor.Input()`. Pak je volána metoda `IDataProcessor.Process()`, která na základě předaných dat z trackerů spočítá aktuální *fázi* a *kvalitu* pohybu. Jsou to hodnoty z intervalu

$$\langle 0; 1 \rangle$$

kde fáze je poměrná část předcvičeného pohybu ve kterém se pacient nachází. Kvalita je hodnota reprezentující jak dobře je předcvičený pohyb replikován.

Nejprve se čeká na to, dokud se pacient nedostane do počáteční fáze pohybu. Do počáteční fáze pohybu se pacient dostane tak, že jeho aktuální fáze je v rozmezí 5% od začátku pohybu, nebo vyšší než 95% pro pohyb s vrcholem[15]. Poté co se pacient dostane do počáteční polohy se zahájí sběr statistik. V průběhu pohybu se kontroluje, zda se pacient dostal nad požadovanou fázi a pokud ano, je mu započítané opakování a čeká se na to až se dostane zpět do počáteční polohy, odkud může provádět opakování další. U pohybů s vrcholem je logika trochu složitější. Opakování je započítáno

[15]Pohyb s vrcholem začíná a končí ve stejné poloze.

až poté, co se pacient dostane zpět do počáteční polohy za podmínky, že se nejprve dostal do požadované fáze. Pokud se pacient začne v rámci pohybu vracet[16], a přitom se nedostal za vrchol nebo požadovanou fázi[17], není započítáno opakování a pacient ho musí provést znovu.

Nakonec je volána metoda pro animování modelu horní končetiny, viz sekce 5.3.4, a zkontroluje se, zda byl dosažen počet opakování. Po dokončení opakování je aplikace přepnuta do stavu `MOVEMENT_SUMMARY` a je volána metoda `TunnelController.HideTunnel()`.

Hry

U většiny her je potřebná určitá forma interakce s uživatelem. Jelikož v aplikaci není pracováno s Vive ovladači, viz sekce 5.2.1, je vývoj her značně omezen. Jak již bylo zmíněno byla provedena implementace pouze jedné velmi jednoduché hry. Každopádně při implementaci bylo snahou systém her navrhnout tak, aby bylo možné další hry v budoucnu přidat. Z principu her však na rozdíl od pohybů není možno přidat bez potřeby překladu aplikace, jelikož je minimálně potřebné definovat jejich logiku. Další hry by pravděpodobně nebylo možné ovládat dostatečně genericky, takže je rozšíření o něco komplikovanější a bude popsáno níže.

Pro práci s hrami vznikla abstraktní třída `Game`, potomek `Exercise`, od které budou konkrétní hry dědit. Třída poskytuje následující abstraktní metody: `ReadGameData()`, která má za úkol načíst herní data z předaného XML uzlu. Třída poskytuje konstruktor, který tuto metodu volá. Cvičení, která mají být provedena, se načítají z jednoho souboru, viz sekce 5.3.5, ve skriptu `ConfigDataReader` se tvoří jejich instance. Očekává se tedy, že konkrétní data pro hru budou načítána při její inicializaci. Metoda `ShowCanvasGameUI()`, jejímž parametrem je herní objekt, jehož potomci jsou `gui` prvky. Metoda GUI prvky podle jména vyhledá vypíše na ně aktuální informace o průběhu hry. Metoda `ShowCanvasSummaryUI()` funguje stejně jako metoda `ShowCanvasGameUI()`, ale vypíše na `gui` informace o shrnutí hry.

Tyto metody pro práci s GUI volá skript `UIController`, který tak přenechává zodpovědnost o vykreslení informací hře, a tak tedy jejímu programátorovi. To znamená, že při implementaci vlastní hry je potřeba definovat i vlastní UI prvky, viz dále.

Třída `GameDefinition` je koncipovaná jako enum v programovacím jazyce Java. Jsou zde tedy konstanty definic jednotlivých her obsahující ID a stranu, na kterou končetinu je hra zaměřena. Na základě ID, a tedy dané

[16]Aktuální fáze pohybu začne významněji klesat.

[17]Požadovaná fáze může být teoreticky před vrcholem.

konstanty, se pak při vytváření instance hry pozná, jaká se má vytvořit konkrétní hra. Pro budoucí hry je tedy nutné definovat vlastní konstantu. Třída `GameStats` pouze sbírá počet bodů dosažených při hře. Pro komplexnější hry by bylo nutné vytvořit vlastní třídu a od této dědit, a tak definovat další statistiky.

Pro práci s GUI a hrami je skript `GameUIController`, který propojí konkrétní třídu hry s herními objekty UI předanými z inspektoru. Myšlenka je taková, že nově definovaná třída nesoucí data hry bude propojena[18] s definovanými UI objekty. Reference na skript je předána `UIControlleru`, který volá metodu `GameUIController.Get...UI()`, která podle třídy vrací svázaný objekt. Získaný objekt je pak předán metodě `Game.ShowCanvas...UI()` pro vykreslení UI.

Nově přidané hře bude nutné definovat logiku. Skript `GameControllerActivator` propojuje, stejným způsobem jak bylo popsáno výše, třídu hry s herním objektem jehož komponentou je skript, který herní logiku provádí. Při přepnutí aplikace do stavu `GAME_START` skript vybere podle třídy hry, které se bude provádět, objekt pro ovládání hry a ten aktivuje, zbytek deaktivuje. Tím je zajištěno, že pro konkrétní hru běží její nadefinovaná obsluha.

Hra spojování míčů je implementovaná třídou `BallConnectingGame` a skriptem `BallConnectingGameController`. Třída implementuje metody zděděné z třídy `Game`. Data pro hru tvoří pozice míčů, které je nutné ve scéně nasbírat. Pozice míčů je možné definovat v externí aplikaci, vytvořené Jakubem Frankem. Výstup aplikace je soubor obsahující nadefinované pozice, který je třídou načítán. Skript funguje na podobném principu jako `MovementController`. Tedy na základě herního stavu jsou prováděny různé obsluhy. Při spuštění hry, tedy stisknutím mezerníku terapeutem, jsou ve scéně vytvořeny míče. To je zajištěno Unity metodou `Instantiate()`. Pozice objektu (míče) je pak nastavena jako jeho načtená pozice přenásobená poměrem rozměrů uživatelů a posunutá vůči aktuální pozici pacienta[19]. Aplikace je přepnuta ze stavu `GAME_START` do `GAME`.

Pak se postupně kontroluje, zda došlo ke kolizi mezi objektem míče a objektem středu dlaně. Pokud došlo, tak je míčům změněna barva a čeká se na kolizi s dalším míčem v pořadí, dokud nedojde k sebrání všech míčů. Po jejich sebrání je aplikace přepnuta do stavu `GAME_SUMMARY`.

[18] Pomocí slovníků (`Dictionary`), kde klíč je třída a hodnota UI objekt.

[19] Je posunuta vůči pozici hrudníku, jelikož se pacient s největší pravděpodobností nenachází ve stejné oblasti, jako byl terapeut při definování pozic míčů.

5.3.3 Tunel

Jak bylo popsáno v sekci 5.3.2 je pro analýzu prováděného pohybu používán tzv. `Tunnel` z knihovny AAPD. Pro práci s touto strukturou byla v rehabilitační aplikaci ještě vytvořena nadstavba v podobě skriptu `TunnelController`. Skript poskytuje skriptu `MovementController` metody pro sestavení tunelu a práci s jeho vizualizací.

Metoda `CreateNewTunnel()` vrací referenci na vytvořenou instanci nového tunelu na základě předané instance pohybu (`Movement`) a datového procesoru (`IDataProcessor`). Je volána metoda `DataCollection.LoadData()` knihovny AAPD, která načte data předcvičeného pohybu ze souboru konkrétního trackeru. Pole[20] kolekcí dat je pak společně s instancí `IDataProcessor` předáno konstruktoru `Tunnel`. Nad tunelem se ještě volá metoda pro načtení dat reprezentující neutrální polohu předcvičeného pohybu.

Metoda `StartProcessing()` transformuje data předcvičeného pohybu na základě aktuální transformace hrudního trackeru pacienta a následně zobrazí vizualizaci tunelu. Transformace dat je provedena voláním metody `Tunnel.TransformTo()`. Data je nutné transformovat, jelikož pacient se může nacházet v prostoru jinde, než kde se nacházel terapeut při nahrávání pohybu[21].

Vizualizaci provádí metoda `DisplayTunnel()`. Je provedena vizualizace trajektorie předcvičeného pohybu, kterou má pacient následovat. Základní vizualizace je řešena pomocí Unity komponenty `LineRenderer`. Komponenta mezi předanými body vykreslí přímku. Pro každý tracker tak existuje jedna komponenta vizualizující jakým způsobem jím bylo pohybováno v průběhu předcvičení. Body pro vizualizaci jsou získané z `Tunnel.RawData`, která vrací již transformovaná data pohybu. Přestože provádění pohybu je spojitá činnost, jeho zaznamenání je však diskrétní, jelikož transformace trackerů jsou vždy zaznamenány pouze v jednotlivých snímcích metody `Update()`. Vykreslením úsečky mezi dvěma po sobě jdoucími body pak zajišťuje spojitou vizualizaci.

Jednotlivé body, které jsou použité k vizualizaci, mohou být různé na základě konfigurace a typu pohybu. Pokud pohyb nemá vrchol, viz sekce 5.3.2, pak jsou vždy zobrazeny všechny body. Pokud vrchol má, tak je ještě možné nastavit aby se zobrazovala buď jen část pohybu od počátku do vrcholu, nebo je možné nastavit, že se bude přepínat mezi trajektorií od počátku do vrcholu a z vrcholu do konce[22], tzn. je vizualizována trajektorie na základě

[20] Pro každý tracker jedna kolekce.

[21] To by mělo značný vliv na analýzu pohybu, která je postavená na principu hledání nejbližšího bodu.

[22] Při pohybu s vrcholem je sice pohyb tam a zpět prováděn po relativně stejných

fáze pohybu, ve které se pacient nachází. Získání rozsahu bodů umožňuje metoda `GetBounds()` a přepínání `SwapTunnel()`.

Vizualizované křivky je možno vypnout pro jednotlivé trackery jak již bylo popsáno u atributů `Movement`. Do křivek jsou ještě vložené kostky. Kostky jsou umístěné na bodech tvořící křivku. Jelikož bodů je hodně, je kostka zobrazena pouze na každém 20. bodu. Kostka má obarvené stěny a je orientovaná tak, jak byl orientovaný tracker při nahrávání pohybu. Je-li nastaven model trackeru také na kostku, pak by při provádění pohybu měla její orientace (a tedy orientace trackeru) odpovídat kostkám ve křivce.

5.3.4 Animace

Jak bylo popsáno v sekci 4.2 bylo nutné, za účelem definování vlastního pohybu, pracovat s animacemi 3D modelu horní končetiny trochu jinak, než je tomu v Unity zvykem. Tvorbu vlastních animací pohybů provádí aplikace Alexe Königa [12]. Animační data jsou předána rehabilitační aplikaci, ve které je s nimi dále pracováno.

Třída `AnimationData` reprezentuje načtená data animace. Vlastní animace je vlastně tvořena pouze lokálními rotacemi kostí v jednotlivých snímcích. Soubor nesoucí data je pak strukturován podle snímků následovně. První řádka snímku obsahuje rotace ramenní a předloketní kosti. Následuje 21 řádek, kde na každé řádce je rotace jedné z kostí dlaně. Struktura první řádky je tedy <rotX; rotY; rotZ> <rotX; rotY; rotZ> a zbylé řádky snímku <rotX; rotY; rotZ>. Třída metodou `ReadData()` z předaného souboru data načte, a kolekci snímků uloží.

Pro samotné přehrávání animací pak slouží abstraktní třída `AnimationPlayer` a její konkrétní implementace `RotationAnimationPlayer`. Třída z konstruktoru převezme objekt, který bude animovat, a definici pohybu (`MovementDefinition`), ze které získá referenci na `AnimationData`. Přehraní animace v konkrétní fázi prováděného pohybu provádí metoda `PlayAnimation()`. Metoda volá `AnimationData.GetBonesRotations()`, která vrátí příslušný snímek rotací dle předané fáze. Následně je kostem modelu nastavena jejich lokální rotace ze získaného snímku[23].

trajektoriích, pohyb ovšem nikdy nebude tak perfektní aby byly identické.

[23]Animace je pouze závislá na aktuální fázi pohybu pacienta, tzn. pokud se fáze nemění, tak objekt není animován, respektive se nemění rotace jeho kostí.

5.3.5 Konfigurace

Konfigurace rehabilitační aplikace je prováděna nastavováním parametrů několika konfiguračních souborů. Všechny konfigurační soubory jsou typu `XML`. Pro každý soubor existuje třída/skript, která se stará o jeho načítání. Soubory se musí nacházet v adresáři *StreamingAssets*. Dále budou popsána struktura souborů a jejich parametry.

config.xml

Soubor *config.xml* slouží jako hlavní konfigurační soubor aplikace. O jeho načítání se stará třída `ConfigDataReader`. Hlavní účel souboru je definování posloupnosti cvičení, která budou pacientem prováděna, tzn. samotnou rehabilitační sekvenci. Dále se zde nastavují nějaká dodatečná globální nastavení (třída `GlobalSettings`). Možná jednoduchá struktura souboru:

```
<exercises>
  <lang>czech</lang>
  <hide_hint_arrow>false</hide_hint_arrow>
  <tracker_model>Cube</tracker_model>
  <hide_cubes_in_tunnel>false</hide_cubes_in_tunnel>
  <show_tunnel_only_to_apex>
    false
  </show_tunnel_only_to_apex>

  <exercise type="game" id="ball_l">
    <name>threeballs</name>
  </exercise>
  <exercise type="movement" id="diag1_l" hideArm="true">
    <phase_target>0.9</phase_target>
    <quality_threshold>0.3</quality_threshold>
    <repetitions>5</repetitions>
  </exercise>
</exercises>
```

Parametry cvičení byly popsány v sekci 5.3.2. Tag `lang` specifikuje, jaký bude použit jazyk textů GUI. V aplikaci jsou podporované dva jazyky: čeština a angličtina, viz sekce 5.3.5. Tag `hide_hint_arrow` indikuje, zda bude zobrazována šipka při hře spojování míčů. Při hodnotě false bude šipka zobrazena, při hodnotě true se šipka zobrazovat nebude. Tag `tracker_model` definuje, jaký bude model trackerů v hlavní scéně. Modely jsou dva: Cube, model trackerů bude krychle s obarvenými stěnami, stejné jako krychle

v trajektoriích tunelu, viz sekce 5.3.3, a Capsule, model trackerů je barevný plošný tvar, jehož barva odpovídá barvě trajektorie tunelu pro daný tracker. Tag `hide_cubes_in_tunnel` umožňuje schovat kostky v trajektoriích tunelu. Hodnota tagu `show_tunnel_only_to_apex` indikuje, zda bude zobrazena trajektorie tunelu pouze od začátku do vrcholu tunelu, nebo jestli bude mezi trajektoriemi přepínáno.

movement-definitions.xml

V tomto souboru jsou jsou definovány všechny pohyby, které je možné použít v sadě cvičení souboru `config.xml`. Definice ze souboru načítá skript `MovementDefinitionsReader`. Struktura souboru s jedním definovaným pohybem:

```
<movements>
  <movement>
    <id>diag1_l</id>
    <side>left</side>
    <data_path>Data/diagonal1-left/</data_path>
    <animation_path>
      AnimationData/diagonal1-left.csv
    </animation_path>
    <has_apex>true</has_apex>
  </movement>
</movements>
```

Parametry definující pohyb jsou popsané v sekci 5.3.2. Souborové cesty musí být relativní vůči adresáři StreamingAssets.

language.xml

V souboru jsou definovány jednotlivé řetězce použité pro texty GUI v aplikaci. Aktuálně je podporovaná čeština a angličtina. Je zde možné přidat překlady pro další jazyky. V souboru jsou definované názvy cvičení v daném jazyce.

```
<languages>
  <English>
    <string name="diag1_l">Left 1 diagonal</string>
  </English>
  <Czech>
    <string name="diag1_l">Leva 1 diagonala</string>
  </Czech>
```

```
</languages>
```

Soubor načítá třída `Language`, která poskytuje statickou metodu `GetString()`. Metoda podle jména řetězce vrací jeho načtenou hodnotu. U názvů cvičení musí hodnota atributu `name`, tagu string, odpovídat definovanému `id`.

tunnel-config.xml

V souboru je možné nastavit různé parametry analyzátoru pohybu knihovny AAPD. Nastavují se zde především váha pozice a rotace jednotlivých trackerů. Váha může být hodnota z intervalu (0;1>, která reprezentuje jaký bude mít vliv při analýze prováděného pohybu.

```
<settings>
  <coefficients>
    <data_type name="head">
          <position x="1" y="1" z="1"/>
          <rotation x="0.5" y="0.5" z="0.5"/>
    </data_type>
  </coefficients>
</settings>
```

5.4 Testování

Aplikace byla vyvíjena a její funkcionalita testována v laboratoři pro virtuální realitu na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd. Vývoj aplikace probíhal iteračním způsobem. Jednotlivé verze softwaru byly dodávány týmu FNKV. Tým FNKV s jednotlivými verzemi prováděl experimentální měření s cílovou skupinou pacientů. Tím tak byla ověřována dosud implementovaná funkcionalita.

Při měření se samozřejmě objevila celá řada chyb, které při testování v laboratoři nalezeny nebyly. Chyby softwaru bylo vždy snahou do další předávané verze opravit. Zároveň probíhala s týmem diskuze o tom, co by se v další verzi mělo objevit nebo co je nutné vylepšit. V dalším vývoji pak vždy byl brán zřetel na odezvu pacientů, kteří se do experimentů zapojili, a terapeuta, který experimenty prováděl. Na základě diskuzí a odezvy uživatelů byla implementovaná nová domluvená funkcionalita, která byla nejprve otestovaná ve VR laboratoři a po odladění nedostatků znovu dále předána.

Tým FNKV provedl pilotní studii, ke které byla využita jedna z verzí vytvářeného softwaru. V této studii proběhlo patnáct jedno hodinových re-

habilitačních schůzek. Některá cvičení, například pohyb vstávání, proběhla v rehabilitační aplikaci. Studie se zúčastnily tři ženy, v průměrném věku 59 let, trpící roztroušenou sklerózou typu relaps remitentní, viz sekce 2.1.1. Výsledkem této studie bylo určité zlepšení v provádění pohybů, ale zároveň došlo ke zhoršení rychlosti jejich provedení. Pacienti se po studii cítí výrazně lépe. Výsledky tak naznačují, že by software mohl pacientům, trpícím touto chorobou, v rehabilitacích napomáhat. Nicméně pro ověření tohoto výsledku je nutné provést studii s větším počtem pacientů. Aktuálně je v přípravě odborný článek, který bude výsledky této studie shrnovat.

Výslednou verzi softwaru, této bakalářské práce, bylo v plánu ve Fakultní nemocnici zprovoznit a následně provést další experimentální měření. To se bohužel zatím nestalo z důvodu vypuknutí pandemie virové choroby Covid-19.

5.5 Nedostatky a potencionální budoucí rozšíření

Jedním nedostatkem aplikace je fakt, že aktuálně dokáže podporovat pouze jednoho terapeuta. První, konfigurační, scéna umožňuje změření rozměrů pacienta a terapeuta. Je předpokládáno, že měřený terapeut je ten, který prováděl předcvičení *všech* naměřených pohybů přidaných k aplikaci. To v průběhu vývoje nebyl problém, jelikož probíhala spolupráce pouze s jedním terapeutem FNKV. Pokud by však došlo k situaci, kdy budou do aplikace přidány pohyby naměřené několika různými terapeuty a pohyby by následně byly zahrnuty do sestavy cvičení pacienta, nastal by problém. S největší pravděpodobností by terapeuti byli různých fyzických rozměrů. Jelikož je možné změřit pouze jednoho terapeuta, nedošlo by ke správnému nastavení pohybů naměřených ostatními terapeuty. Nejjednodušším řešením tohoto problému by bylo rozměr terapeuta měřit přímo před prováděním předcvičovaného pohybu a údaj uložit společně s daty naměřeného pohybu. V rehabilitační aplikaci by pak bylo prováděno měření rozměrů pouze pacientů a u každého pohybu by byl určen jeho poměr s konkrétním terapeutem.

Aplikace neprovádí kontrolu všech vstupních hodnot. Pokud je tedy zadána nějaká hodnota parametru špatně, aplikace se s tímto stavem nemusí vypořádat a nebude tak fungovat správně. To znamená, že aplikace funguje s předpokladem, že dostane korektní hodnoty parametrů. To samozřejmě není ideální situace a je to určitě funkcionalita, kterou by bylo potřebné dodělat.

S tím je spojený další nedostatek a to, že aplikace spoléhá na dodržo-

vání několika konvencí. Nejdůležitější z nich je správné nasazení trackerů. Bylo by vhodnější aby aplikace sama dokázala automatickou detekci špatně nasazeného trackeru a uživatele na to upozornila.

Aplikace aktuálně načítá sadu cvičení pouze z jednoho konfiguračního souboru, viz sekce 5.3.5. Terapeut typicky pro pacienta připraví sadu cvičení na základě toho, jakou část horní končetiny s ním chce procvičit. Z toho důvodu by asi bylo vhodnější umožnit terapeutovi výběr z více souborů. Nemusel by pak sadu cvičení pokaždé definovat znovu, ale pouze by došlo k definici jedné sady, která by se dala použít několikrát.

Jedním z možných rozšíření je přidání dalších rehabilitačních her. V aplikaci je vytvořen systém, kdy je další hry možné doprogramovat, viz sekce 5.3.2. Každopádně jak je v sekci poznamenáno, je u her problém především interakce s uživatelem, kvůli nemožnosti použití Vive ovladačů. Možným řešením tohoto problému by mohlo být prozkoumání technologie *Leap Motion*, která umožňuje sledování pohybů dlaní a prstů bez nutnosti použití jakéhokoliv ovladače či rukavice. Pacienti by tak mohli lépe interagovat s virtuálním prostředím aplikace, což by mohlo otevřít i další, nejen herní, možnosti. Dalším řešením by mohl být přechod ze systému HTC Vive na Oculus Quest, který možnost sledování dlaní umožňuje také. S přechodem na jiný systém by však nejdříve bylo nutné vyřešit problém nahrazení Vive trackerů, na kterých celý projekt stojí.

Osobně si myslím, že by bylo potencionálně lepší, kdyby byl projekt tvořen pouze jednou aplikací. Aktuálně projekt tvoří aplikace pro nahrání pohybů, generaci animací, generování pozic míčů hry a rehabilitační. Pokud by došlo k integraci všech aplikací v jednu, bylo by nutné umožnit jejich ovládání z uživatelského prostředí. Tím by se dle mého názoru značně ulehčila práce terapeuta. Nemusel by ovládat aplikace čtyři ale pouze jednu. Zároveň by nemusel přistupovat ke konfiguračním souborům přímo z nějakého souborového průzkumníka, ale aplikace by se o ně starala automaticky.

Určitě by stálo za to, poskytnout terapeutům podrobnější analýzu pohybu. Aplikace umožňuje terapeutům nahrávat data pohybů prováděných pacienty. Z těchto dat lze řadu věcí změřit/spočítat. Bylo by pak především na týmu FNKV, jaké hodnoty by se jim hodily.

6 Závěr

Výsledkem práce je funkční, konfigurovatelná a především dále rozšiřitelná rehabilitační aplikace, implementovaná v herním enginu Unity. I přes určité nedostatky byly splněny hlavní cíle práce. Výsledná aplikace umožňuje terapeutům sestavit pro pacienty sadu rehabilitačních cvičení. Pacienti tato cvičení provádí ve 3D virtuálním prostředí pomocí systému HTC Vive. Cviční je možné rozdělit do dvou kategorií, pohyby a hry.

Pohyby fungují na principu analýzy dat o pozici a rotaci senzorů systému HTC Vive v reálném čase. Pacientem prováděný pohyb je porovnáván se správně vykonaným pohybem figuranta (terapeuta). Data správně vykonaného pohybu jsou v aplikaci pacientovi ve virtuálním světě vizualizována v podobě trajektorie, kterou pacient musí při pohybu následovat. Aplikace umožňuje terapeutovi definovat jakýkoliv další pohyb, který je možné k aplikaci přidat bez nutnosti aplikaci znovu sestavit.

Implementace her byla, především z důvodu omezené interakce s pacientem, značně limitovaná. Součástí aplikace je jedna jednoduchá hra, nicméně byl navržen systém, který umožňuje další hry doprogramovat.

Výstupem aplikace jsou data a statistiky cvičení, která byla pacientem vykonána. Největším přínosem aplikace je především vizualizace pohybů, kterou pacient v reálném světě nezíská. Vizualizace by tak měla pacientovi daný pohyb značně zjednodušit.

Seznam zkratek

3D trojdimenzionální 14, 15, 24, 29, 32, 34, 37, 38, 42, 43, 47, 53

AAPD Automatic Arm Position Detection 32, 38, 42, 43, 46, 50

API Application Programming Interface 21, 27

FNKV Fakultní nemocnice Královské Vinohrady 7, 50–52

FOV Field Of View 22

FPS Frames Per Second 19

GPS Global Positioning System 18

GUI Graphical User Interface 40, 44, 45, 48, 49

HMD Head-mounted display 16, 18, 19, 21, 31, 38, 40

IMU Inertial measurement unit 18, 23

RS Roztroušená skleróza 8, 10–12

SDK Software Development Kit 30

UDP User Datagram Protocol 18

VR Virtuální realita 14–18, 30, 50

Literatura

[1] Ambler, Z. *Základy neurologie*. Galén, spol. s r.o., 2006. ISBN 80-246-1258-5.

[2] Bardi, J. *What is Virtual Reality? [Definition and Examples]* [online]. Marxent, 2019. [cit. 2020/04/07]. Dostupné z: https://www.marxentlabs.com/what-is-virtual-reality/.

[3] Brennan, D. *SteamVR Update Adds Asynchronous Reprojection* [online]. Road To VR, 2016. [cit. 2020/04/18]. Dostupné z: https://www.roadtovr.com/steamvr-update-adds-asynchronous-reprojection/.

[4] Buckley, S. *This Is How Valve's Amazing Lighthouse Tracking Technology Works* [online]. Gizmodo, 2015. [cit. 2020/04/19]. Dostupné z: https://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768.

[5] *Co je to RS?* [online]. 2020. [cit. 2020/03/01]. Dostupné z: http://ereska.cz/rs/index.html.

[6] Corporation, H. *HTC Vive product* [online]. HTC Corporation, 2020. [cit. 2020/04/17]. Dostupné z: https://www.vive.com/eu/product/#vive%20series.

[7] Education, V. *Virtuální realita – historie a současnost* [online]. VR Education, 2020. [cit. 2020/04/07]. Dostupné z: https://vreducation.cz/virtualni-realita-historie-a-soucasnost/.

[8] Computing History, T. C. *HTC Vive Virtual Reality* [online]. The Centre for Computing History, 2020. [cit. 2020/04/17]. Dostupné z: http://www.computinghistory.org.uk/det/52936/HTC-Vive-Virtual-Reality/.

[9] Frank, J. *Sběr 3D dat pro rehabilitační software ve virtuální realitě* [online]. Jakub Frank, 2020. [cit. 2020/04/16].

[10] Goodrich, R. *Accelerometer vs. Gyroscope: What's the Difference?* [online]. LiveScience, 2018. [cit. 2020/04/12]. Dostupné z: https://www.livescience.com/40103-accelerometer-vs-gyroscope.html.

[11] Heaney, D. *How VR Positional Tracking Systems Work* [online]. UploadVR, 2019. [cit. 2020/04/19]. Dostupné z: https://uploadvr.com/how-vr-tracking-works/.

[12] König, A. *3D animace paže podle naměřených dat pro účely rehabilitace ve virtuální realitě* [online]. Poór, Vítek, 2020. [cit. 2020/04/24].

[13] McIntosh, C. W. *Understanding Simulator Sickness* [online]. Children's Hospital of Philadelphia, 2018. [cit. 2020/04/07]. Dostupné z: https://injury.research.chop.edu/blog/posts/understanding-simulator-sickness.

[14] Poór, V. *Automatická analýza pohybu ramene pro účely rehabilitace ve virtuální realitě* [online]. Poór, Vítek, 2019. [cit. 2020/04/24]. Dostupné z: https://bitbucket.org/Custom666/aapd/src/master/AAPD_BakalarskaPrace.pdf.

[15] *O roztroušené skleróze* [online]. Nadační fond impuls, 2020. [cit. 2020/03/01]. Dostupné z: https://www.nfimpuls.cz/index.php/roztrousena-skleroza/o-roztrousene-skleroze/161-roztrousena-skleroza-mozkomisni-co-by-mel-nelekar-vedet.

[16] Snopková, N. *Kvalita života pacienta sroztroušenou sklerózou* [online]. 2020. [cit. 2020/03/18]. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/17937/1/Nikola%20Snopkova%20%28MVS%202%29%20Kvalita%20zivota%20pacienta%20s%20roztrousenou%20sklerozou.pdf.

[17] Society, V. R. *What is Virtual Reality?* [online]. Virtual Reality Society, 2017. [cit. 2020/04/07]. Dostupné z: https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html.

[18] Souppouris, A. *How HTC and Valve built the Vive* [online]. Engadget, 2016. [cit. 2020/04/16]. Dostupné z: https://www.engadget.com/2016-03-18-htc-vive-an-oral-history.html.

[19] Unity Technologies. *Unity User Manual (2019.3)* [online]. Unity Technologies, 2020. [cit. 2020/04/24]. Dostupné z: https://docs.unity3d.com/Manual/index.html.

[20] Unity Technologies. *Game engines—how do they work?* [online]. Unity Technologies, 2020. [cit. 2020/04/24]. Dostupné z: https://unity3d.com/what-is-a-game-engine.

[21] Xinreality. *Head-mounted display* [online]. Xinreality, 2018. [cit. 2020/04/09]. Dostupné z: https://xinreality.com/wiki/Head-mounted_display.

[22] Xinreality. *Lighthouse* [online]. Xinreality, 2017. [cit. 2020/04/19]. Dostupné z: https://xinreality.com/wiki/Lighthouse.

[23] Yates, A. *SteamVR's "Lighthouse"for Virtual Reality and Beyond* [online]. Tested, 2015. [cit. 2020/04/19]. Dostupné z: `https://www.youtube.com/watch?v=xrsUMEbLtOs`.

[24] Řasová, K. *Fyzioterapie u neurologicky nemocných (se zaměřením na roztroušenou sklerózu mozkomíšní)*. CEROS Centrum komplexní neurorehabilitační péče pro nemocné s roztroušenou sklerózou, o.p.s, 2007. ISBN 978-80-239-9300-4.

A Přílohy

A.1 Seznam obrázků

A.2 Uživatelská příručka

A.2.1 Sestavení

Aplikace byla vyvíjena v Unity editoru verze 2019.3.0, která je dostupná ke stažení na adrese `https://unity3d.com/get-unity/download/archive`. Po otevření projektu aplikace[1] v Unity editoru je pro sestavení nutné kliknout na `File > Build Settings`. Objeví se okno, ve kterém je možné provedení sestavení. Pro správné fungování aplikace je nutné do aplikace zahrnout obě scény, `SetUpScene` a `MainScene`, přičemž je nutné, aby scéna SetUp byla jako první[2]. Následně je nutné zvolit platformu `PC, Mac & Linux Standalone`. Po zvolení cílové platformy je možné kliknout na tlačítko `Build`. V zobrazeném dialogu je nutné zvolit adresář do kterého se aplikace sestaví. Po zvolení adresáře proběhne sestavení aplikace.

A.2.2 Spuštění

Po sestavení aplikace je ji možno spustit dvojklikem na soubor `VR Rehabilitation.exe`, nacházející se ve zvoleném adresáři. Možnosti aplikace byly

[1] Projekt je dostupný na CD v příloze této práce, nebo na git repositáři `https://gitlab.com/frankkuba/vr-arm-motion`.

[2] Přetažením je možné scény prohodit.

popsány v sekci 5.1. Konfigurace v sekci 5.3.5.

A.2.3 Binding trackerů

Při konfiguraci je nutné spárovat všechny trackery s příslušnou rolí, jinak software nebude fungovat správně a bude docházet k prohazování trackerů. Pro spárování trackerů stiskněte po spuštění aplikace současně *pravý SHIFT + B*. Pro svázání jednotlivých rolí je potřeba zvolit požadovanou kartu, poté vybrat požadované zařízení a přiřadit mu předem určenou roli. Zařízení se vybírá z přehledové mapy scény v pravém dolním rohu kliknutím levým tlačítkem myši. Pokud již nějaké zařízení spárováno, lze toto nastavení smazat pomocí kliknutí na křížek v příslušném řádku. Před opuštěním konfigurace je *nutné* uložit změny.

Zavedenou konvenci pro bindingu zařízení je možno vidět v tabulce A.1.

Umístění	Binding role	Barva podstavy
headset	head	
hrudník	tracker1	popruh na hrudník
levé zápěstí	tracker2	bílá
levá ruka	tracker3	červená
pravá ruka	tracker4	fialová
pravé zápěstí	tracker5	žlutá
židle	tracker6	modrá (úchyt na židli)

Tabulka A.1: Tabulka zavedené konvence pro binding trackerů

A.2.4 Definice vlastního pohybu

Dále popisovaný postup předpokládá, že již byla získána data z aplikací[3] pro nahrání pohybu [9] a data animace nahraného pohybu [12]. Uvedené souborové cesty jsou relativní ke kořenovému adresáři sestavené aplikace.

Složku s daty pohybu je potřeba nakopírovat do složky VR Rehabilitation_Data/StreamingAssets/Data/. Soubor s daty animace je potřeba nakopírovat do složky StreamingAssets/AnimationData/. Následně je potřeba do souboru movement-definitions.xml nacházející se ve složce VR Rehabilitation_Data/StreamingAssets přidat nový záznam o pohybu, viz sekce 5.3.5. Na závěr je nutné definovat název pohybu. Ten se definuje v souboru VR Rehabilitation_Data/StreamingAssets/language.xml, viz sekce 5.3.5. Tímto

[3] Aplikace se nacházejí na CD v příloze práce.

je pohyb přidán do aplikace a je možné ho zapojit do cvičení ze souboru VR Rehabilitation_Data/StreamingAssets/config.xml, viz sekce 5.3.5.

V případě že by se nejednalo přidání zcela nového pohybu, ale například pouze přeměření, není nutné definovat další pohyb ale pouze u stávajícího změnit data.

A.3 Obsah DVD

- `Animation Generation Build.zip` — soubor obsahují adresář se sestavenou aplikací, vyvinutou Alexem Königem, pro generaci animací nových pohybů
- `Bakalarska prace.pdf` — soubor s textem této práce
- `Ball Configuration Project.zip` — soubor s adresářem Unity projektu aplikace, vyvinutou Jakubem Frankem, pro definici pozic míčů hry
- `Data Recorder Project.zip` — soubor s adresářem Unity projektu aplikace, vyvinutou Jakubem Frankem, pro nahrávání pohybů
- `Movement Videos.zip` — soubor obsahují adresář s videi, která ukazují předcvičování pohybů, které jsou součástí rehabilitační aplikace
- `VR Rehabilitation Build.zip` — soubor obsahující adresář s již sestavenou rehabilitační aplikaci
- `VR Rehabilitation Project.zip` — soubor obsahující adresář Unity projektu rehabilitační aplikace