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

Bakalářská práce

3D animace paže podle naměřených dat pro účely rehabilitace ve virtuální realitě

Plzeň 2020 König Alex

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 5. května 2020

König Alex

Poděkování

Tímto bych rád poděkoval panu Doc. L. Vášovi Ph.D., za odborné vedení práce, připomínky, trpělivost a vstřícný přístup. Dále děkuji pánům K. Frankovi, K. Koderovi, L. Rodinovi a paní Doc. PhDr. K. Řasové Ph.D. za spolupráci při vývoji aplikace.

König Alex

Abstract

3D animation of arm based on measured data for purposes of rehabilitation in virtual reality

This thesis is focused on the creation of arm animations, that will further be used in an application for the rehabilitation of patients with cerebrospinal multiple sclerosis. The goal was to create an Unity application that will generate animations that correspond to measured data (positions and orientations of HTC Vive trackers) and are realistic. The results were achieved using rigging and the algorithm of gradient descent. Furthermore it is possible to manually add animation of the hand using the application Blender.

Abstrakt

Tato bakalářská práce se zabývá vytvářením animace paže, která bude dále využívána v aplikaci pro rehabilitaci pacientů s roztroušenou sklerózou mozkomíšní. Cílem bylo vytvoření Unity aplikace, která by umožňovala generování animací, které co nejvíce odpovídají naměřeným datům (pozice a orientace HTC Vive trackerů) a působí co nejrealističtějším dojmem. Výsledky byly vytvořeny s využitím riggingu a algoritmu gradientního sestupu. Dále bylo umožněno manuální doplnění animace dlaně s využitím aplikace Blender.

Obsah

1 Úvod

Roztroušená skleróza mozkomíšní je chronické onemocnění poškozující nervovou soustavu. Člověk, který touto nemocí trpí může mít mimo jiné i problémy s jemnou nebo i hrubou motorikou. Rehabilitační cvičení prokazatelně pomáhá v moderování těchto příznaků a pomáhá pacientům ke znovunabytí co největší části jejich schopností.

V nynější době jsou rehabilitační cviky většinou prováděny tak, že se pacient nachází v jedné místnosti s fyzioterapeutem a s jeho větší či menší asistencí provádí pohyby. Ovšem podle závažnosti onemocnění pacient nemusí být schopen některé pohyby do určité míry vykonat. Toto může být pro pacienta značně demotivující.

Proto se začalo uvažovat o využití virtuální rehabilitace pro fyzioterapii. Pacient by v takovémto případě byl interaktivně vtažen do děje jakési "hry" a toto by jej mohlo motivovat k další snaze. Většina nyní existujících softwarů ovšem paradoxně nevyhovuje potřebám fyzioterapeutů. Často dochází k tomu, že pacienti najdou cestu, jak si pohyby usnadnit, a i když je takového "cvičení" může bavit, již nemá požadovaný rehabilitační přínos. Z tohoto důvodu byl vyvíjen nový software, podle požadavků fyzioterapeutů, konkrétně ve spolupráci s L. Rodinou pod vedením paní Doc. PhDr. K. Řasové Ph.D z University Karlovy.

Výsledkem je rehabilitační aplikace s možností doplnění nových pohybů. Práce byla rozdělena do tří bakalářských prací. Práce J. Franka s názvem "Sběr 3D dat pro rehabilitační software ve virtuální realitě" byla zaměřena na kolekci dat, která budou potřeba k vyhodnocování pohybu pacienta. Práce J. Kodery s názvem "Software pro rehabilitaci paže ve virtuální realitě" byla zaměřena na vytvoření vlastní rehabilitační aplikace, ve které bude pacient cvičit.

Bakalářská práce, která je předmětem tohoto dokumentu, byla zaměřena na vytvoření podpůrné aplikace pro vytváření animací horní končetiny, které budou dále přehrávány pacientům při provádění cviků. Aplikace je implementována v herním enginu Unity a její cílovou platformou je Windows a VR headset s trackery HTC Vive.

Všechny tři zmíněné bakalářské práce zabývající se vývojem těchto software byly vypracovány pod vedením pana Doc. L. Váši Ph.D.

2 Teorie

2.1 Medicínská teorie

V této sekci bude nejprve popsána roztroušená skleróza mozkomíšní (RS), poté přístup k její léčbě a rehabilitaci a nakonec budou popsány pohyby, které budou zahrnuty do rehabilitační aplikace.

2.1.1 Roztroušená skleróza mozkomíšní

Roztroušená skleróza mozkomíšní je chronické autoimunitní onemocnění poškozující nervy. Mezi její příznaky patří například svalová slabost, ochrnutí, třes, poruchy rovnováhy, spasticita či únava. Konkrétní klinické projevy závisí na typu onemocnění a jeho průběhu.[22]

V dnešní době není známa léčba, co by onemocnění zastavila, ovšem lze zpomalit jeho postup. To ovšem za předpokladu, že léčba bude zahájena včas a bude probíhat pravidelně a dlouhobodě.[22]

2.1.2 Rehabilitace

Význam rehabilitace byl prokázán u nemocných u všech forem RS. Jedná se o aktivní proces, který pomáhá k zachování optimální fyzické, smyslové a psychické úrovně, zachování co největší nezávislosti a prevenci komplikací. Toto všechno vede ke zlepšení celkové kvality života.[21]

Fyzioterapie je zaměřena na hybnost a využívá principů učení. Vhodnou kombinací a opakováním podnětů hledá nepoškozené mozkové oblasti, které využívá pro částečnou obnovu porušené funkce. Díky tomu je možné zpomalit rozvoj onemocnění.[22]

Na samotném řízení motoriky se podílejí dva rozdílné systémy. Limbický, kde k vykonání pohybu je důležité správné emocionální naladění, a senzomotorický, kdy pohyb je realizován na základě analýzy informací. Tyto informace přicházejí z různých senzorů, jako například z oka, ucha či ze svalu.[22] Vyvíjená rehabilitační aplikace poskytuje pacientovi jak vizuální (pacient vidí animaci jak by měl vypadat prováděný pohyb), tak sluchovou (potvrzení správného vykonání cviku) zpětnou vazbu.

Je důležité, aby rehabilitační polohy a cviky byly vhodné, a vykonávané korektním způsobem podle pokynů terapeuta.[22] Proto rehabilitační aplikace kontroluje kvalitu provedení pohybu, a požaduje určitou přesnost.

2.1.3 Rehabilitační pohyby

V průběhu vývoje bylo vytvořeno několik datasetů s rehabilitačními pohyby. Tyto pohyby vždy předcvičoval pan L. Rodina, aby se jednalo o správně provedené cviky.

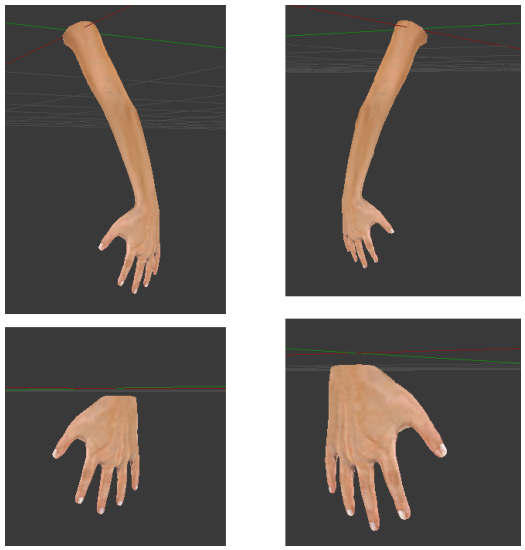
Jednalo se o diagonální pohyby paže, pravotočivé a levotočivé spirály, vstávání ze židle a opětovné sedání. Konrétní záznamy pohybů jsou k nahlédnutí na přiloženém CD.

2.2 Model

Nejprve bylo nutné získat vhodný model k animaci. Vzhledem k podstatě aplikace je potřeba, aby model byl pokud možno co nejreaelističtější a šlo jej snadno animovat.

Proto byl zakoupen již existující model levé paže[16], se kterým je poskytována i textura. Tento model bylo potřeba dále upravit pro naše požadavky. V aplikaci Blender byla doplněna kostra a připojena textura. Dále z něj byly vytvořeny modely pravé paže a modely pravé a levé dlaně. Všechny používané modely lze vidět na obrázku 2.1.

V dalších podkapitolách se nachází popis relevantní terminologie a technik.



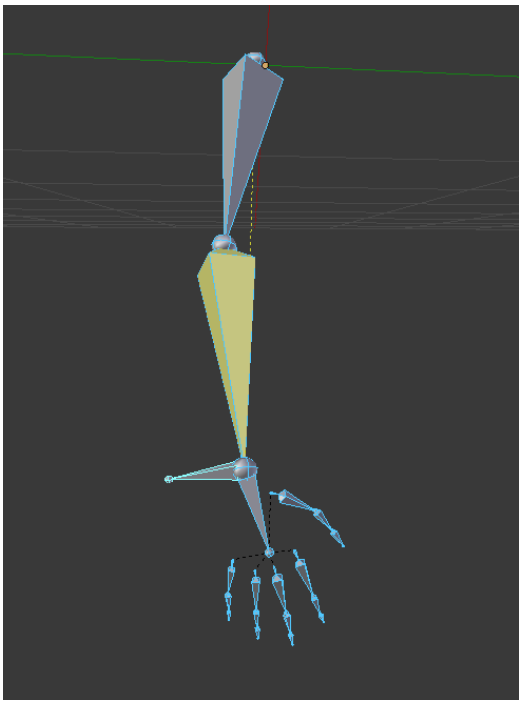
Obrázek 2.1: Modely levé paže (nalevo) a pravé paže (napravo)

2.2.1 Kostra

K objektu, který bude třeba animovat, lze vytvořit kostru, jedná se o "stick-figure", neboli určité zjednodušení jeho tvaru. Tomuto principu se říká rigging. U modelu člověka, nebo jeho části, je intuitivní kosti umisťovat podobně tak, jak vypadá reálná lidská kostra.

Jednotlivé kosti mají stromovou strukturu. Dá se tedy použít termín strom kostí. Zároveň mezi kostmi existuje vztah rodič-potomek. Potomek může být rodičem dalších kostí, ale jedna kost nemůže mít více rodičů. Tato vazba znamená, že pokud je transformován rodič, transformují se i všichni potomci. Dále pojmem armatura označujeme kost, která je rodičem všech ostatních kostí.[15] Tato kost tedy je kořenem celého stromu. Je vhodné jako kořen zvolit tu kost, která je nějakým způsobem v rámci modelu významná, jelikož její transformací se budou transformovat i všechny zbylé kosti modelu.

Kostra objektu paže byla vytvářena v programu Blender a lze ji vidět na obrázku 2.2.



Obrázek 2.2: Vytvořená kostra paže

Vzdálenosti kostí

Defaultně jsou kosti přímo napojeny na svého rodiče, v tomto případě jejich polohu ovlivňuje pouze konec rodičovské kosti. Děti lze od jejich rodiče odpojit, a pokud by bylo potřeba, aby tyto odpojené kosti udržovaly nějakou vzdálenost od jiných, lze v aplikaci Blender nastavit tzv. "Limit Distance Constraint". Toto omezení umožní kosti, aby si zachovávala vzdálenost buď menší nebo větší než aktuální, nebo stále stejnou od zadaného cíle. To znamená, že umístění kosti je omezeno na prostor vně, uvnitř nebo na povrchu koule se středem v zadaném cíli.[3]

Toto nastavení bylo využito pro kosti prstů, kdy odpojené kosti udržují stále stejnou vzdálenost od kosti zápěstí.

Inverzní kinematika kostí

V první fázi vývoje rehabilitační aplikace byly využívány manuálně vytvořené animace celé paže. Pro usnadnění animování se využíval princip inverezní kinematiky zabudovaný do aplikace Blender. Kosti lze v aplikaci Blender nastavit tzv. "IK Solver Constraint". Toto nastavení bude ovlivňovat pozice několika kostí. Tento počet se zadává pomocí atributu "Chain length". V praxi to znamená, že transformace aplikovaná na tuto kost, bude přenášena na chain-length kostí. Tedy pokud pohneme kostí s nastaveným "IK Solver Constraint", pohne se i specifikovaný počet nadřazených kostí. Zde lze dále nastavit tzv. "Pole Target", nebo-li kost, která bude ovlivňovat osovou rotaci kostí.[4]

2.2.2 Skinning

Výše zmíněné vytvoření kostry usnadňuje animaci modelu. S jejím využitím je možné během animace neurčovat na každém snímku zvlášť pro každý vrchol v modelu jeho novou pozici, ale celý model je spojen s kostrou. Podle rotace a pozice kostí se odvozuje, kde se budou jednotlivé body modelu nacházet. Transformovat je tedy během vytváření animace potřeba jen kosti.

Aby tento postup bylo možné využít, je potřeba vytvořenou kostru navázat na příslušný model. Procesu, který popisuje jak toto provést, se říká skinning. Každý vrchol v modelu je ovlivňován všemi kostmi z kostry. S jakou mírou daná kost ovlivňuje vrchol, je určeno pomocí vah, kdy každému vrcholu přísluší pro něj relevantní váhy od všech kostí ve stromě kostí. Tyto váhy jsou normalizovány na interval <0,1> a součet vah pro jeden vrchol přes všechny kosti musí být roven 1.[20]

Přiřazování vah bylo tradičně považováno za manuální práci, kdy se váhy nanášely na model formou vybarvování. Nyní existuje několik různých principů k automatickému nanášení vah. Z těchto algoritmů lze popsat například přístupy "bone heat"[2] a "bone glow"[20].

Takto přiřazené váhy jsou dále využívány některým z algoritmů, které provádějí vlastní deformaci modelu. Zde dochází ke kombinaci jednotlivých transformací tak, aby vznikl co nejlépe vypadající modifikovaný model. Kombinace může probíhat například algoritmy "linear blend skinning" nebo "dual quaternion skinning"[11].

Blender využívá metodu "bone heat" k výpočtu vah kostí, a pro kombinaci vzniklých vah může používat jeden z výše zmíněných algoritmů.[18]

Bone heat

Přístup k přiřazování vah kostem "bone heat" vypočte váhy tím způsobem, že se chová k modelu jako k izolovanému tepelně vodivému objektu. Teplota kosti, pro které se budou počítat váhy, je nastavena na 1, a teplota zbylých je 0. Konkrétní váha této kosti pro jeden vrchol se určí podle toho, kolik tepla se do daného vrcholu přeneslo.[2]

Bone glow

O něco lepším přístupem k výpočtu vah je takzvaný přístup "bone glow". Zde se váhy jedné kosti pro jednotlivé vrcholy počítají na základě toho, kolik světla z dané kosti na daný vrchol dopadne, pokud by se daná kost stala zdrojem světla. Váhy, které tímto způsobem vzniknou, jsou ihned použitelné pro další zpracování, oproti vahám, které vzniknou při některých jiných technikách, které se často musejí dále manuálně upravovat.[20]

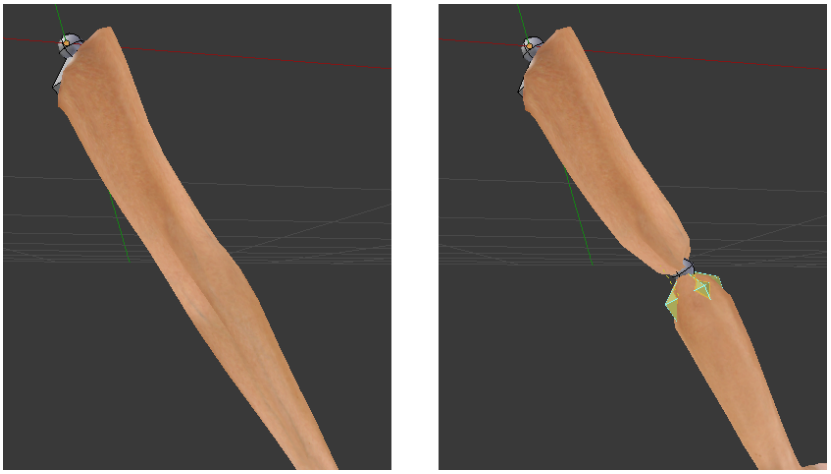
Linear blend skinning

Standardním přístupem ke kombinaci transformací vzniklých vlivy různých kostí je přístup "linear blend skinning" (LBS). Tento algoritmus je založen na tom, že transformace jsou reprezentovány maticemi, které jsou mezi sebou lineárně kombinovány.[11]

Tato kombinace lze popsat vzorcem 2.1. Kde $\bar{p_i}$ popisuje výslednou transformaci vrcholu i, p_i aktuální transformaci vrcholu i, n je počet kostí, jejichž váhy se berou v potaz, w_{ij} je váha přiřazená j-té kosti u i-tého vrcholu,
a T_j je 4x4 matice, reprezentující globální transformaci kosti ve vztahu ke své klidové pozici.[19]

$$\bar{p_i} = \sum_{j=1}^{n} w_{ij} T_j p_i \tag{2.1}$$

Tento přístup ovšem může produkovat artefakty na deformovaném modelu. Problémem je především ztráta objemu okolo kloubů, kde v některých případech může vzniknout takzvaný "candy wrapper" efekt. V některých případech lze tomuto předejít vhodným riggingem, v jiných to ovšem možné není. Tento efekt lze vidět na obrázku 2.3.



Obrázek 2.3: "Candy wrapper" efekt nasimulovaný na modelu paže

Dual quaternion skinning

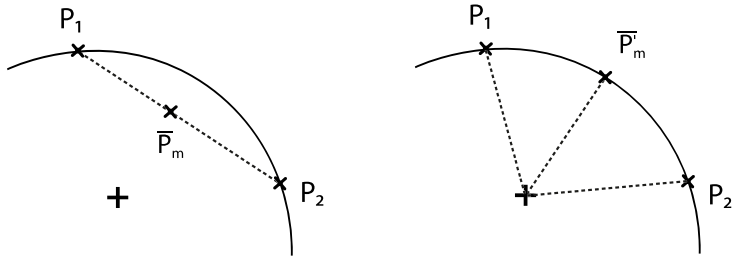
Aby se předešlo ztrátě objemu, lze využít přístupu "dual quaternion skinning" (DQS). Kde místo toho, aby se transformace popisovaly maticemi, jsou vyžity duální kvaterniony.

Duální kvaternion reprezentuje jen rotaci a translaci, na rozdíl od transformační matice, která obsahuje i škálovací koeficient, který může způsobvoat právě výše zmíněné artefakty.[19]

Tento postup se dá popsat vzorcem 2.2. Kde n je opět počet kostí, jejiž váhy se berou v potaz. Dále $\dot{q_i}$ reprezentuje duální kvaternion transformace kosti, kterému je přiřazena váha w_i. Dále dochází k normalizaci pomocí $\left\|\sum_{i=1}^{n}(w_i \dot{q_i})\right\|$, aby vznikl kvaternion, který transformuje kost z klidové pozice do požadované deformované.[19]

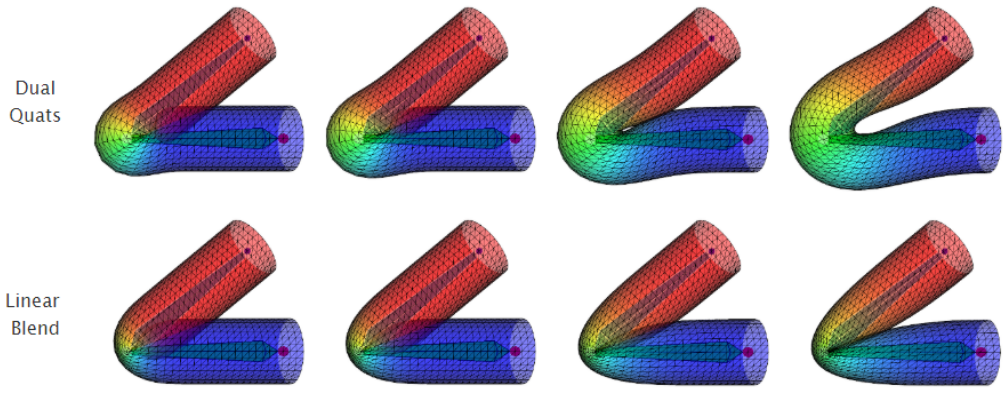
$$\dot{q_i} = \frac{\sum_{i=1}^{n}(w_i \dot{q_i})}{\left\|\sum_{i=1}^{n}(w_i \dot{q_i})\right\|} \tag{2.2}$$

Tento přístup lze vizualizovat jako nahrazení lineární interpolace interpolací sférickou. Tedy, pokud by na vrchol působily dvě kosti, jeho nová pozice bude ležet na kružnici, místo na přímce, spojující transformace vycházející z vlivů každé z kostí zvlášť.[19] Toto je znázorněno na obrázku 2.4, kde P_1 a P_2 jsou původní transformace a $\bar{P}_m$ je transformace vzniklá jejich kombinací.

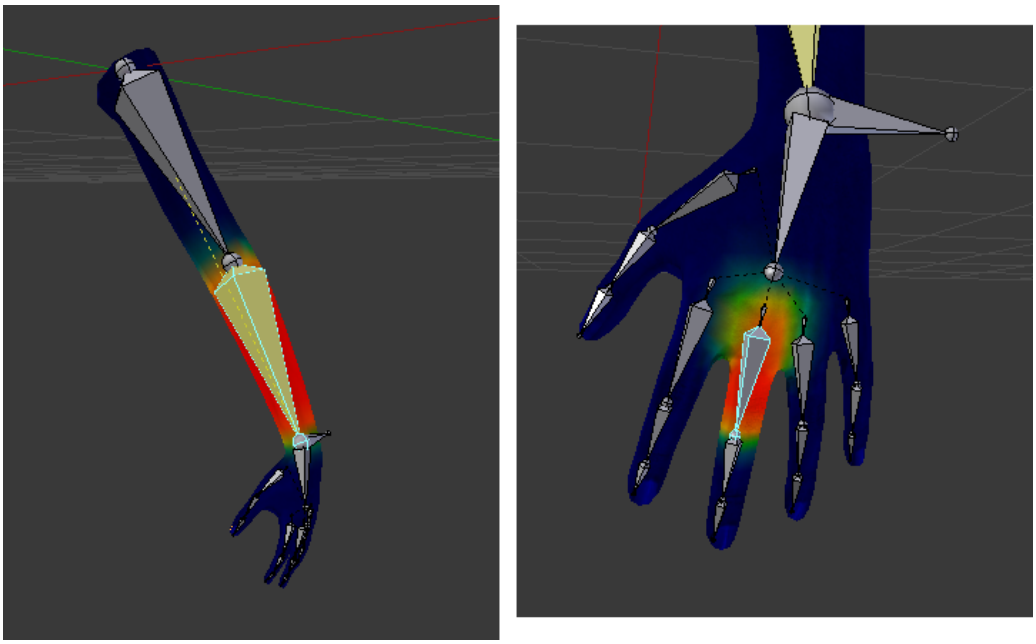


Obrázek 2.4: Vizualizace metod LBS a DQS.[19]

Výsledný objekt může vypadat například jako na obrázku 2.5, kde je deformace znázorněna na jednoduchém modelu válce se dvěmi kostmi. Jednotlivé sloupce se od sebe liší rozprostřením vah po modelu (červená znační hodnotu 1, modrá hodnotu 0). První řádka zobrazuje skinning s využitím metody DQS a druhá s využitím metody LBS.



Obrázek 2.5: Porovnání výsledných deformací pomocí metod LBS (druhá řádka) a DQS (první řádka). Sloupce se od sebe liší rozložením vah.[19]



Obrázek 2.6: Znázornění váh pro kosti předloktí (vlevo) a prvního článku prostředníku (vpravo)

2.2.3 Textura

Textury se používají pro vytvoření větší realističnosti modelu. Stínováním a modelováním se dá dosáhnout jen určité úrovně realizmu, kdy například kovové či plastové povrchy budou vypadat přijatelně, ale materiály jako lidská kůže či dřevo uměle. Důvodem je, že většina reálných objektů má mikrostrukturu – texturu, která poskytuje lidskému oku potřebné informace o typu objektu. [13]

Textury lze na objekty nanášet více či méně automaticky. Výběr vhodné metody záleží na složitosti modelu a textury. Pro mapování na složitější objekty lze využít metoda UV mapování. Tento přístup byl použit pro nanesení dodané textury na model paže za pomoci aplikace Blender.

UV mapování

UV mapování je jednou z metod nanášení bitmapové textury na 3D modely, umožňující složitějším modelům získat přirozený vzhled.

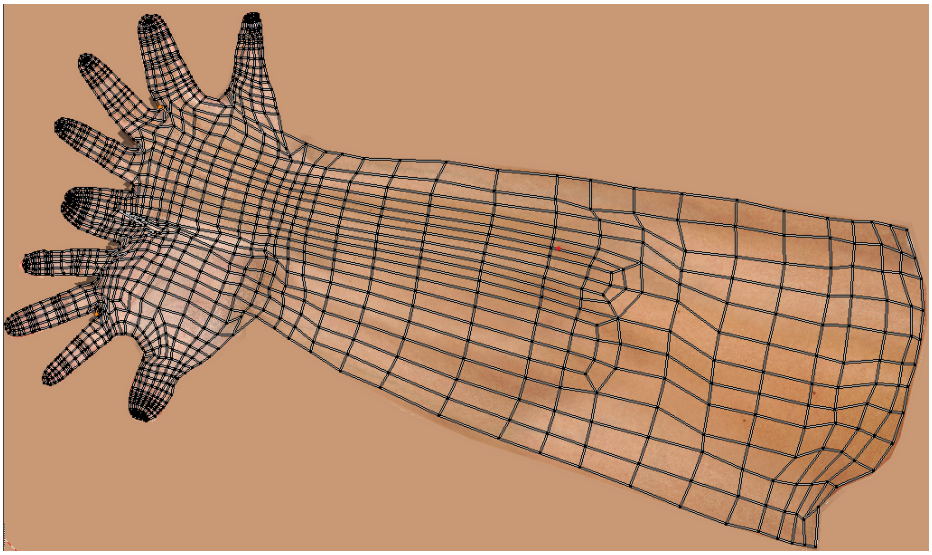
Pro každý vrchol v modelu se uchovávají jeho souřadnice v textuře, které mu přiřazují jeden její pixel. Tyto souřadnice se označují jako U a V. Tato přiřazovací funkce tedy transformuje 3D objekty do 2D prostoru. Dvojice "vrchol – UV souřadnice" jsou uchovávány v UV mapě. Všechny UV souřadnice se nacházejí na intervalu <0, 1>.

Procesu mapování vrcholů na pixely se říká “UV unwrap”. Blender umožňuje tento proces udělat automaticky, pomocí metody “Smart UV Project”, kdy je analyzován vybraný objekt a automaticky vytvořena jeho UV mapa. Toto nemusí být vždy vhodná volba, jelikož program nemusí správně vyhodnotit, které strany 3D objektu je vhodné rozpojit a které ne. Proto je nejspolehlivějším přístupem udělat UV unwrap manuálně.

Unwrap si lze představit jako opačný proces skládání kostky z kusu papíru. Při tomto úkolu se nejprve vystřihne potřebný tvar, poté lze z něj složit kostku, a tu opět rozložit do plochy. Při UV unwrap se ovšem začíná se 3D objektem.

Nejdůležitějším krokem je tedy vybrání tzv. švů. Jako šev se označuje hrana, podél které bude daný 3D objekt pomyslně rozstříhnut. Švy je vhodné umístit do té části objektu, kde bude nejméně nápadná případná nesouvislost textury. Zvýrazněné švy na objektu paže lze vidět na obrázku 2.8.

Tímto postupem lze vytvořit 2D reprezentaci 3D objektu. Toto rozložení pro objekt ruky je vidět na obrázku 2.7. Paži s namapovanou texturou lze vidět na obrázku 2.1.



Obrázek 2.7: UV mapování vrcholů objektu paže na texturu.

Obrázek 2.8: Objekt paže se zobrazenými vrcholy a hranami, švy zvýrazněny červeně

2.3 Unity

Unity je zvolený game engine pro vývoj jak aplikace pro vytváření animací, tak rehabilitačního softwaru.

Game engine je framework, který se primárně využívá pro vytváření her. Podporuje práci práci jak ve 3D tak 2D, umožňuje užití fyziky, animace, interaktivity mezi objekty či zvuku. Vývoj je typicky umožněn pro více cílových platforem. Urychluje a zefektivňuje práci na vývoji nové hry.[7]

K Unity je umožněno s pomocí rozšíření SteamVR připojit HTC Vive a vyvíjet pro virtuální realitu. Mimo to je zde možné vytvořit i přehledné GUI pro desktopové aplikace. Jelikož vývoj animačního softwaru i rehabilitační aplikace bude probíhat ve stejném prostředí, je možno zaručit, že vytvořené animace bude možné použít bez dalších úprav.

2.4 Trackery v prostoru

Vive tracker je reálný objekt, s jehož pomocí je možné vytvářet vlastní VR kontrolery. Lze jej upevňovat pomocí standardního šroubku na stativ. [9] Tracker je schopný poskytovat údaje o své pozici a rotaci. Dva takovéto trackery jsou umístěny na paži a jeden na hrudníku při záznamu dat.

Pro tuto práci je důležité pochopit jak se tracker nachází v prostoru.

Pokud bychom jej položili na stůl plochou stranou dolů a "nožičkou" s LED diodou k sobě, míří jeho lokální osa[10]:

- X směrem doleva
- Y směrem nahoru
- Z směrem dopředu (nebo-li od nás)

Jedná se tedy o pravotočivý souřadný systém. Znázornění os lze vidět na obrázku 2.10. Jelikož Unity pracuje v levotočivém souřadném systému, bude na tyto osy ještě aplikována příslušná transformace. O tyto transformacae se postará Unity a Steam VR.

Pokud se jedná o pozice trackerů na paži, jsou umístěny tak, že pokud je paže spuštěna volně podél těla, LED diody obou trackerů směřují podél paže dolů. Tracker reprezentující pozici předloktí je umístěn na horní straně zápěstí, a tracker reprezentující pozici nadloktí ze strany pod deltoidem. Dále tracker, který snímá pozici hrudníku je umístěn LED diodou směrem nahoru a svou plochou stranou leží na hrudníku. Toto umístění je vidět na obrázku 2.11.

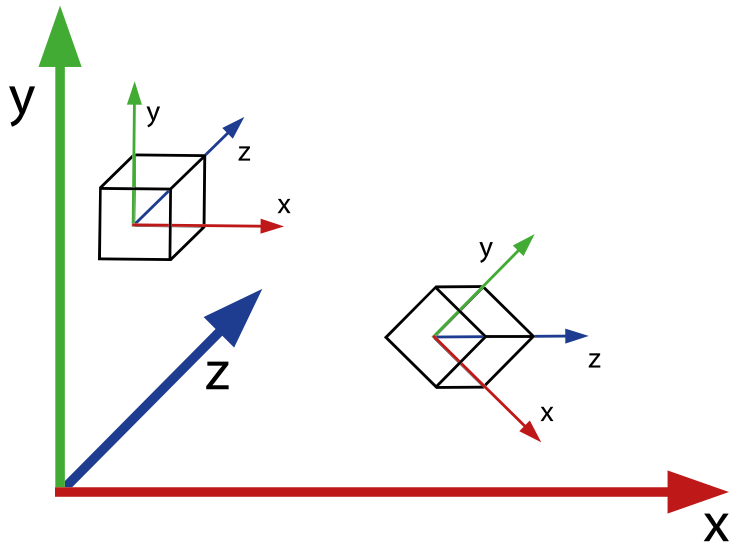
Dále je vhodné vysvětlit tyto pojmy: globální a lokální soustava souřadnic a eulerovské úhly.

Globální a lokální soustava souřadnic

Globální soustava souřadnic je fixována na celý prostor, ve kterém se pohyb měřil. Lze ji reprezentovat třemi ortogonálními osami x, y a z se stanovenou orientací. Dále je pevně stanoven počátek (tedy bod [0,0,0]). Globální soustava souřadnic je absolutní.

Oproti tomu lokální soustava souřadnic je fixována na nějaký objekt. Její počátek je obvykle stanoven ve středu objektu. Opět existují tři ortogonální osy x, y a z. Tato lokální soustava souřadnic rotuje spolu s objektem, pokud je transformován ve vztahu k soustavě globální. Lokální soustavy souřadnic jsou relativní.[6]

Znázornění globální a lokální soustavy souřadnic lze vidět na obrázku 2.9.

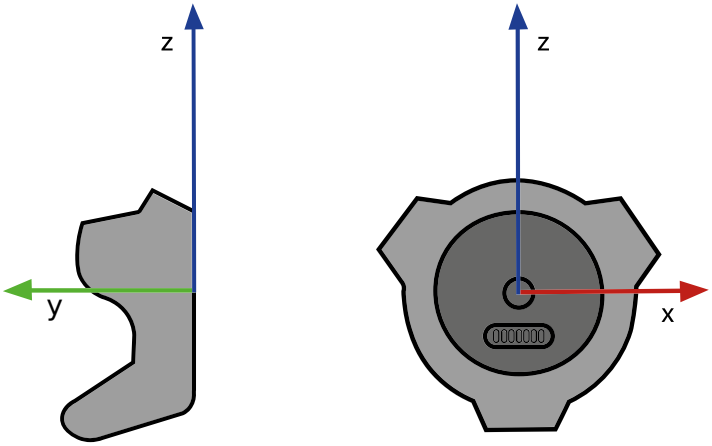


Obrázek 2.9: Globální soustava souřadnic s dvěmi krychlemi s lokálními soustavami souřadnic.

Eulerovské úhly

S pomocí eulerovských úhlů lze reprezentovat jakoukoli rotaci v prostoru jako tři po sobě jdoucí rotace okolo souřadnicových os. Těmto rotacím se říká "yaw" (ϕ), "pitch" (θ) a "roll" (ψ) (česky precese, nutace a rotace). Tyto úhly jednoznačně udávají zrotování objektu v prostoru, ale nelze zpětně jednoznačně získat úhly. [17] A to proto, že může existovat několik různých kombinací rotací okolo souřadnicových os, které ve výsledku dají stejné zrotování objektu.

Postup aplikace rotací je takový, že nejprve otočíme souřadnicový systém okolo jeho osy z o úhel ϕ. Poté otočíme okolo jeho osy x o úhel θ. A nakonec otočíme okolo jeho osy z o úhel ψ.[17]



Obrázek 2.10: Tracker v prostoru se znázorněnými osami[10]

Obrázek 2.11: Umístění trackerů na paži

Obrázek 2.12: Umístění trackerů na člověku

2.5 Vstupní data

Vstupní data pro aplikaci pro generování animací jsou poskytována aplikací, kterou vyvíjel J. Frank v rámci své bakalářské práce. Těmito daty jsou naměřené informace o pohybu, ze kterých je třeba vytvořit animaci.

Záznam jednoho pohybu se nachází v 5 souborech. Tyto soubory jsou:

1. Arm.csv
2. Forearm.csv
3. Head.csv
4. Chest.csv
5. Neutral.csv

Trackovací informace obsažené v těchto souborech jsou poskytovány trackery a VR headsetem HTC Vive.

Arm.csv obsahuje trackovací informace získané z trackeru na nadloktí, Forearm.csv obsahuje trackovací informace získané z trackeru na předloktí a Chest.csv informace z trackeru na hrudníku. Data v Head.csv představují trackovací informace získány z headsetu. V souboru Neutral.csv se nachází neutrální pozice, ze které bude pacient začínat cvičit tento pohyb. Pro tuto aplikaci není poslední soubor potřeba.

Všechny vstupní soubory, až na soubor Neutral.csv, jsou stejně dlouhé a jejich i-tá řádka udává i-tý snímek pohybu.

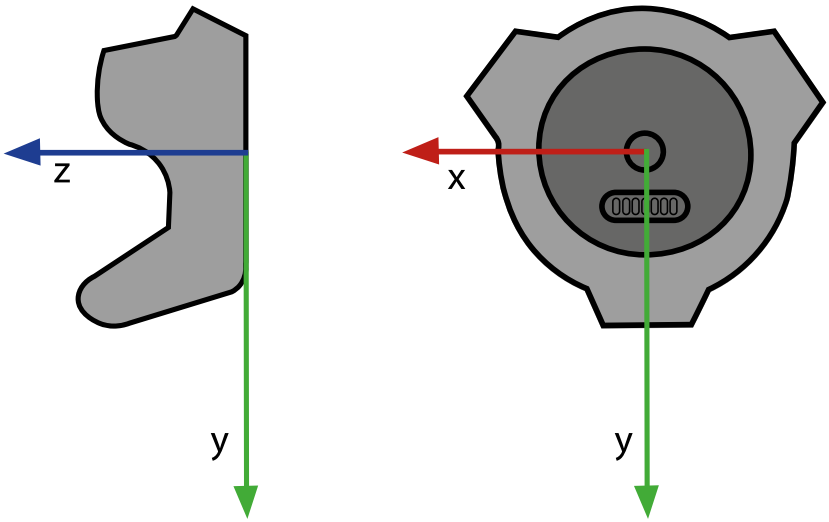
Naměřená data jsou transformována tak, že pokud bychom tracker položili na stůl plochou stranou dolů a "nožičkou" s LED diodou k sobě, míří jeho lokální osa:

- X směrem doprava
- Y směrem k nám
- Z směrem nahoru

Znázornění těchto os je vidět na obrázku 2.13.

2.5.1 Formát souboru

Všechny používané soubory mají společný formát. Jelikož se jedná o .csv soubory, dají se rozdělit do sloupců. Jednotlivé sloupce jsou odděleny znakem středníku (';'). V prvním sloupci se nachází časová značka kdy byla



Obrázek 2.13: Lokální osy trackeru v Unity

data v řádce zaznamenána. V dalších 3 sloupcích se nachází souřadnice pozice trackeru v prostoru v pořadí os x, y a z. Poslední 3 sloupce obsahují eulerovské úhly rotací v globální soustavě souřadnic. Opět jsou zapsány v pořadí os x, y a z.

2.6 Animace

Animace je postup, kdy se dynamicky vytváří série snímků zachycujících množinu objektů. Každý ze snímků je považován za modifikaci snímku předchozího.[13]

Obecně existují dva přístupy k animaci. Lze animovat s využitím pevně nastavených keyframes, mezi kterými se pro vytvoření využívá automatická generace, nebo lze každý snímek generovat na základě nějakých požadavků, kde se musejí objekty nacházet, či jaké síly na ně působí.

Pro první verzi programu byly veškeré animace vytvářeny ručně v Blenderu za pomoci keyframes. Tyto animace paže byly vytvářeny pouze jako provizorní, v době, kdy tento projekt byl v začátcích. Animace dlaně jsou takto dělány stále, jelikož pro ně neexistují žádná zaznamenávaná data. Animace paže se nyní generuje automaticky podle vstupních dat (viz 2.6.1).

Blender také umožňuje vytváření animací manipulováním s objekty pomocí skriptu. Aby bylo umožněno lepší uživatelské prostředí, byl zvolen přístup standalone aplikace, kdy je možné vytvořit GUI přístupnější cílovým uživatelům. Skriptování je ovšem stále využito pro exportování dat animace dlaně z Blenderu.

Cílem vyvinuté aplikace je vytvořit animaci paže, která se co nejvíce přiblíží naměřenému pohybu. Proto je vhodné využít animací s využitím kostí.

Animace s použitím kostí je animace, při které se nastavují hodnoty rotace a pozice v prostoru kostem, a povrch modelu, který je spojený s danou kostrou, se hýbe v závislosti na ní. Tato animace je používána zejména pokud chceme, aby se model hýbal a zároveň měl jasně definovanou strukturu pohybu. Kosti se hýbou rigidně, ale přesto se model může ohýbat a přetáčet plynuleji.[8]

2.6.1 Animace paže

Animace paže je vytvářena podle naměřených dat, které určují, kde se mají trackery nacházet v každém snímku animace. Existuje tedy model paže, který se bude animovat a ke kterému jsou připojeny modely trackerů. Tímto vznikne virtuální paže s trackery, která bude dále pozicována.

Pro animaci paže dle naměřených dat byla využita inverzní kinematika (viz kapitola 2.7).

Co se týče přístupu ke generování, jednotlivé snímky lze počítat v realném čase, nebo je po vygenerování ukládat pro pozdější přehrání. Jelikož cílem je se co nejvíce přiblížit naměřeným datům, lze předpokládat, že výpočet pro jeden snímek bude trvat delší dobu.

Pokud by byly snímky vytvářeny v reálném čase, byl by program omezen rychlostí s jakou je počítač schopen provádět potřebné výpočty. Animace by měla, v ideálním případě, obsahovat více než 15 snímků za vteřinu, aby byla zachována iluze plynulého pohybu. Počítat snímky v reálném čase s dostatečnu rychlostí je v tomto případě tedy nemožné.[13]

Proto je zvolen přístup dopředného vygenerování celé animace a její uložení do souboru. V rehabilitační aplikaci se tedy pouze načte a přehraje právě potřebný soubor.

2.6.2 Animace dlaně

Animace dlaně jsou vytvářeny manuálně za použití keyframes v Blenderu. Opět je vhodné použít kostry z důvodů, které byly uvedeny výše.

Keyframe je dán svou pozicí v čase, a uchovává hodnoty vlastností objektu. [5] Které hodnoty budou uchovávány lze zvolit při ukládání daného keyframe.

Tyto keyframes je potřeba zadat manuálně nastavením rotací kostí a logika v programu Blender vytvoří pohyb, který vyplní časový prostor mezi

dvěmi pevně danými pozicemi. Mezi těmito pozicemi tedy dochází k interpolaci. Pokud je například zadán keyframe v čase t_1 a keyframe v čase t_2, animační software za pomoci interpolace automaticky určí pozici objektu pro všechny snímky mezi těmito keyframes.

2.7 Inverzní kinematika

Ve vstupních datech jsou uchované pozice trackerů upevněných na reálném člověku. Ovšem pro vytvoření animace je potřeba znát úhly, které mají kosti virtuální paže. S využitím inverzní kinematiky lze tyto úhly získat.

Složitost výpočtu závisí na počtu stupňů volnosti. V tomto případě animace paže lze rotovat kloubem lokte a ramene, což dává 6 stupňů volnosti. Dále lze pohybovat pozicí ramene (neboli celé paže), což znamená 3 další stupně volnosti. Dohromady tedy 9 stupňů volnosti. Samozřejmě platí, že čím více stupňů volnosti, tím složitější je výpočet.

2.7.1 Omezení

Stupně volnosti je možné omezit na určité povolené hodnoty. Na první pohled se zdá, že toto by mohlo být v případě animace paže značně výhodné. Omezením rozsahu úhlů, které je možné nastavit, by šlo předejít zbytečným generováním reálně neproveditelných pozic paže.

Nicméně tento přístup má i své nevýhody. Hlavní komplikací je, že by se mohlo stát, že cesta k optimální poloze by při generaci vedla přes nějakou polohu, která by byla vyhodnocena jako "neproveditelná". Generování by tedy mohlo přestat v nějakém falešném minimu a nepokračovat dále.

Jelikož je generován pohyb podle naměřených dat, která byla měřena snímáním pohybu reálného člověka, lze předpokládat, že se dá spolehnout na jejich kvalitu. Z tohoto důvodu lze dospět k závěru, že omezení úhlů nebude potřeba. Z podstaty nasnímaných dat vyplývá, že pokud se virtuální paže co nejvíce přiblíží prvnímu snímku naměřených dat, bude další její pohyb také následovat snímky dat a nebude docházet k ohybání kloubů špatným směrem.

2.8 Optimalizace na základě funkce

Při počítání nové pozice pomocí inverzní kinematiky lze využít optimalizaci podle funkce. Podmínkou je, že bude stanovena nějaká objektivní funkce, jejíž hodnota se bude minimalizovat. Tato funkce musí být schopna dostatečně postihnout a zhodnotit polohu paže v prostoru.

Samotná optimalizace je prováděna tak, že jsou měněny hodnoty stupňů volnosti a vždy vypočítána hodnota objektivní funkce. Pokud je tato hodnota menší než předchozí, pozice je přijmuta a postup opakován. Pokud ne, proběhne návrat zpět k poslední přijaté pozici. Důležitou součástí tohoto postupu je limita, kdy s úpravami skončit. Tedy musí být stanovena nějaká ukončovací podmínka.

Další komplikací může být existence lokálních minim. Cílem je nalézt globální minimum, ovšem projít celý funkční obor a hledat nejmenší ohodnocení je nemožné. Zároveň pokud se paže dostane do lokálního minima může být problémem ji z něj dostat a pokračovat do minima globálního. Jelikož z této konfigurace hodnot stupňů volnosti je její blízké okolí ohodnoceno vyšší hodnotou objektivní funkce.

3 Dekompozice

3.1 Transformace dat

Aby vytvořená animace nebyla ovlivněna tím, jak byl v prostoru orientován člověk, jehož pohyb byl zaznamenán, je třeba vstupní data transformovat do zvolené "neutrální polohy". Znalost této neutrální polohy umožní další flexibilitu při přehrávání animace v rehabilitační aplikaci.

Tato neutrální poloha byla definována tak, že hrudník bude umístěn vždy v bodě [0, 0, 0], tedy v počátku globální soustavy souřadnic. Rotace jednotlivých trackerů budou dále upravovány vzhledem ke vstupním datům tak, aby lokální osa z trackeru hrudníku vždy ležela v globální rovině yz. Efektivně to znamená, že se hrudník pouze může naklánět nahoru a dolů, či rotovat okolo lokální osy z. Pozice a rotace ostatních trackerů je třeba upravit takovým způsobem, aby se správně posunul celý snímek a nedošlo k poškození vstupních dat.

Pozice trackerů je upravena tak, že od jejich momentální pozice je odečtena pozice trackeru hrudníku. Tímto se tracker hrudníku posune do počátku soustavy souřadnic, a trackery na paži se posunou s ním o odpovídající vzdálenost tak, aby si zachovaly stále stejný odstup.

Poté jsou trackery otočeny okolo počátku globální osy souřadnic v globální ose y o úhel vypočtený ze vzorce 3.1. Kde $t_c.forward$ je směrový vektor udávající orientaci lokální osy z virtuálního trackeru hrudníku v globální soustavě souřadnic. A $t_c.forward.z$ je jeho z složka a $t_c.forward.x$ je jeho x složka.

$$rotY = atan2(t_c.forward.z, t_c.forward.x)/PI * 180 - 90 \tag{3.1}$$

Funkce $atan2$ vrací úhel mezi vektorem $[t_c.forward.x, t_c.forward.z]$ a osou x. Pro tento úhel platí, že hodnota jeho tangens je rovna $t_c.forward.z/t_c.forward.x$.[1]

Jelikož je tento úhel v radiánech, je nutné jej převést na stupně, a protože rotace má být prováděna o úhel, který vektor svírá nikoli s globální osou x, ale s globální osou z, je potřeba odečíst 90 stupňů.

3.2 Umístění trackerů

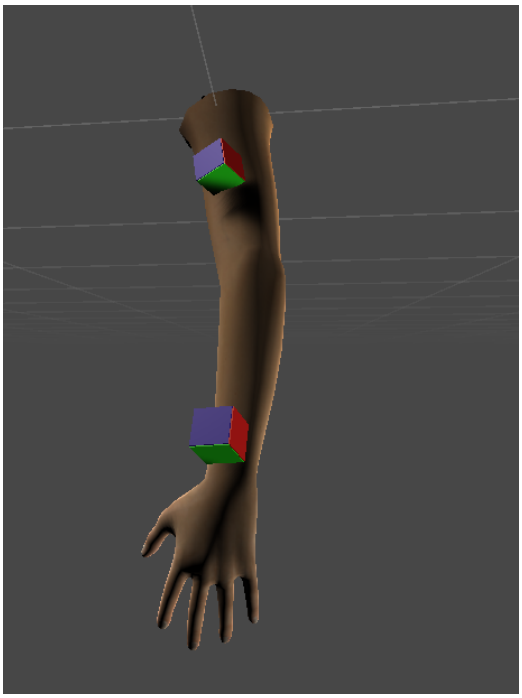
Ve specifikaci rehabilitační aplikace je určeno, jak jsou umístěny reálné trackery na paži při měření pohybů. Tyto pozice je třeba nasimulovat v aplikaci pro generování animací, aby bylo možné naměřená data použít pro generování animace a výpočet objektivní funkce.

Proto je třeba umístit nějaké objekty – virtuální trackery – na model paže tak, aby odpovídaly umístění reálných trackerů na paži člověka. Tedy jejich polohy a rotace by si měly co nejvíce odpovídat.

Tracker nadloktí je třeba umístit pod deltoid ze strany paže a tracker předloktí na horní stranu zápěstí. Lokální osa y obou trackerů musí mířit podél paže a lokální osa z musí být kolmá na paži.

Konkrétně to znamená, že tracker na nadloktí je umístěn do jedné poloviny délky kosti nadloktí a tracker na předlotí umístěn do jedné patnáctiny kosti předloktí od kosti zápěstí. Oba trackery jsou posunuty o jednu šestinu délky kosti, na které se nacházejí, směrem, který je kolmý na vektory orientací kostí nadloktí a předloktí, tak, aby byly zdánlivě umístěny "na kůži" virtuální paže. Dále je potřeba je zrotovat takovým způsobem, aby jejich lokální osy ukazovaly stejným směrem jako u reálných trackerů.

Toto umístění virtuálních trackerů lze vidět na obrázku 3.1.



Obrázek 3.1: Umístění virtuálních trackerů na paži, zelená stěna objektu určuje směr osy y, modrá směr osy z a červená směr osy x

3.3 Objektivní funkce

Pokud jsou vytvořené virtuální trackery a transformovaná vstupní data, lze přistoupit k výpočtu objektivní funkce. Zde je potřeba vzít v úvahu jak rozdíly orientací trackerů, tak odchylky v jejich poloze.

Objektivní funkci lze tedy stanovit jako funkci 3.2.

$$f = error_u + error_f \tag{3.2}$$

Kde $error_u$ je ohodnocení umístění trackeru na nadloktí a $error_f$ je ohodnocení trackeru na předloktí. Tyto hodnoty se vypočtou podle vzorce 3.3 nad příslušnými trackery.

$$error = d*dist^2 + a*angleDiff(x)^2 + b*angleDiff(y)^2 + c*angleDiff(z)^2, \tag{3.3}$$

kde $dist$ je vzdálenost virtuálního a reálného trackeru. Dále $anleDiff()$ je funkce, která vypočte úhel mezi lokálními osami trakcerů. O kterou osu se jedná je předáváno parametrem. Hodnoty a, b, c a d jsou konstanty, které je potřeba vhodně zvolit. Druhé mocniny se ve vzorci vyskytují proto, abychom vytvořili funkci, která bude vhodná pro další výpočet v rámci algoritmu gradientního sestupu, viz kapitola 3.4.2.

3.4 Generování animace paže

Animace paže je vytvářena na základě naměřených vstupních dat. Detaily o formátu a obsahu jsou popsány v kapitole 2.5. Pro každý naměřený snímek dat bude vytvářen jeden snímek animace. Tento algoritmus je založen na principu inverzní kinematiky (viz kapitola 2.7).

Každý naměřený snímek dat je potřeba nejprve transformovat podle postupu popsaného v kapitole 3.1 a až poté je možné přistoupit k vlastní generaci animačního snímku.

V průběhu generace se paže bude nacházet v různých pozicích. Každá z těchto pozic je definována aktuální polohou ramene v prostoru (tedy souřadnicemi xyz) a globálními eulerovskými úhly rotací kosti nadloktí a kosti předloktí (opět reprezentováno jako xyz). Těchto 9 hodnot se označuje jako stupně volnosti soustavy.

Každá pozice má své ohodnocení, které je dáno výpočtem objektivní funkce (viz kapitola 3.3), a počítá se vůči aktuálnímu zpracovávanému snímku dat.

Na začátku generování se paže nachází ve své neutrální pozici. Rotace virtuálních trackerů jsou stejné jako rotace kostí v modelu a poloha ramene je odvozena od polohy hrudníku prvního snímku dat.

3.4.1 První náhodná pozice

Objektivní funkce, jak již bylo zmíněno v kapitole 2.8, má více lokálních minim a jedno globální. Toto globální minimum je to, kterého se ideálně chce dosáhnout. Gradientní sestup, použitý k optimalizaci, ovšem může konvergovat k jednomu z lokálních.

Proto je důležité dostat paži co nejblíže cílové pozici ještě před zahájením algoritmu gradientního sestupu. Nejprve je tedy náhodně vygenerován stanovený počet různých pozic paže. Tyto pozice se mezi sebou liší jen rotacemi kostí. Poté se pomocí gradientního sestupu pokusí z této pozice dostat co nejblíže prvnímu snímku dat. Z těchto konečných pozic se vybere ta, která má nejmenší ohodnocení.

3.4.2 Gradientní sestup

Princip gradientního sestupu je takový, že jsou využívány pouze lokální informace pro nalezení minima funkce. Touto funkcí je objektivní funkce z kapitoly 3.3. Kterým "směrem" se po ní vydat, je určeno vytvořením gradientu postupným derivováním dané funkce podle jejích stupňů volnosti. Derivace jsou využívány, protože určí "sklon" funkce v určitém bodě. Dále pro urychlení sestupu je využíván pararmetr δ. Nová pozice při optimalizaci pomocí gradientního sestupu se tedy zjistí pomocí vzorce 3.4. Kde p_i je aktuální pozice a $\bigtriangledown F$ je gradient.[14]

$$p_{i+1} = p_i - \delta * \bigtriangledown F \tag{3.4}$$

Z toho lze odvodit, že gradient je vektor, který popisuje směr největšího růstu. Počet jeho komponent odpovídá počtu stupňů volnosti, a každá hodnota komponenty odpovídá zlepšení při posunutí v jednom z nich.

Při optimalizaci polohy ruky se počítá s 9 stupni volnosti, tím pádem počet komponent gradientu je také 9. Gradient lze obecně popsat jako vektor 3.5, kde x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2 a z_3 jsou jednotlivé stupně volnosti, a F je objektivní funkce 3.3.

$$\bigtriangledown F = [\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial y_1}, \frac{\partial F}{\partial z_1}, \frac{\partial F}{\partial x_2}, \frac{\partial F}{\partial y_2}, \frac{\partial F}{\partial z_2}, \frac{\partial F}{\partial x_3}, \frac{\partial F}{\partial y_3}, \frac{\partial F}{\partial z_3}] \tag{3.5}$$

Z obecného tvaru gradientu je vidět, že jeden jeho prvek se vypočítá jako derivace funkce podle odpovídajícího stupně volnosti. Tento výpočet lze provést podle vzorce derivace 3.6. Kde $f(x)$ je hodnota funkce v bodě x a h značí parametr, který se musí limitně blížit k nule.

$$f' = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3.6}$$

Dále parametr δ určuje velikost kroku po objektivní funkci a je využíván pro urychlení sestupu do minima. Za předpokladu, že se situace v daném směru posunutí v daném stupni volnosti zhoršila, bude se zlepšovat pokud se posuneme na stranu opačnou, a to i pokud se posuneme o větší hodnotu. Tedy dává smysl odečíst od aktuální pozice δ násobek gradientu.

Tento předpoklad samozřejmě nemusí být vždy pravdivý, proto se musí výsledná pozice opět ověřovat, a výsledky vyhodnotit. Pokud došlo ke zhoršení situace, parametr δ se zmenší, a pokus o posun bude proveden znovu.

Pro implementaci algoritmu je tedy nejprve třeba stanovit parametr δ. Ten je nastaven na zvolenou počáteční hodnotu, a zároveň je stanovena dostatečně malá hodnota ε, která bude považována za hraniční hodnotu tohoto parametru.

Následující postup je opakován dokud je δ větší než ε.

1. Vypočtení ohodnocení pozice

   Vypočteno ohodnocení aktuální pozice paže. Tato hodnota je označena jako `f0`.

2. Odhad gradientu

   Hodnotu jedné komponenty v gradientu lze vypočítat tak, že aktuální hodnota daného stupně volnosti je dočasně zvětšena o jeden krok a je vypočteno ohodnocení `f`. Rozdíl `f` a `f0` vydělen velikostí kroku udává, o kolik se situace zhorší, pokud by se aktuální pozice paže změnila v daném směru. Tento postup je opakován pro všechny stupně volnosti.

   Pokud na dané pozici vyjde záporné číslo, znamená to, že posun v daném stupni volnosti v tomto směru zlepšuje situaci.

   Pokud na dané pozici vyjde kladné číslo, znamená to, že pro zlepšení situace by bylo třeba se v daném stupni volnosti posunout ve směru opačném.

3. Odečtení gradientu

   Tento vektor je odečítán od aktuální pozici tak, že od hodnoty v každém stupni volnosti je odečten δ násobek hodnoty na odpovídající pozici gradientu.

4. Vypočtení ohodnocení nové pozice

   Nyní je potřeba ověřit, zda takto vytvořená pozice paže skutečně situaci zlepšila nebo ne. Proto je nyní opět vypočteno ohodnocení pozice. Tato hodnota je označena jako `fnew`.

5. Vyhodnocení

   Pokud `fnew` je menší než `f0`, nová pozic je přijmuta jako aktuální a hodnota `f0` je nastavena na hodnotu `fnew`. Dále se pokračuje krokem 2.

   Pokud `fnew` je větší než `f0`, aktuální pozici paže zůstává nezměněna, a δ je vyděleno dvěma. Dále se pokračuje krokem 3, pokud δ je stále větší než ε.

3.4.3 Perturbace

Postupem, který byl popsán v předchozí sekci, se stále může stát, že pozice která byla vytvořena není optimální. Nebo-li došlo k tomu, že bylo nalezeno pouze jedno z lokálních minim objektivní funkce. Abychom se pokud možno vyhnuli prohlášení takového lokálního minima za nejlepší možnou polohu paže, budou dále prováděny perturbace.

Je stanoven si počet opakování, kolikrát bude probíhat následující postup.

1. Vypočtení ohodnocení pozice

   Je vypočteno ohodnocení aktuální pozice paže. Tato hodnotu je označena jako `f0`.

2. Vytvoření náhodného posunu

   Ke každé hodnotě stupňů volnosti reprezentující rotaci kostí aktuální pozice paže je přičtena náhodně vygenerovaná hodnota. Rozsah náhodných čísel je stanoven dostatečně malý, tak abychom zcela nepřišli o vygenerovanou pozici. Předpokládá se, že funkce bude konvergovat k lokálnímu optimu jen v blízkém okolí.

3. Gradientní sestup

   Na pozici, která byla vytvořena v předchozím kroku, je opět aplikován postup gradientního sestupu. Tímto se získá nová, optimalizovaná, pozice.

4. Vyhodnocení situace

   Je vypočteno ohodnocení vzniklé pozice. Tato hodnota je označena jako `fnew`, a porovnána s `f0`.

   Pokud je `fnew` menší pozice je akceptována jako aktuální a `f0` je nastavena na hodnotu `fnew`.

   Pokud je `fnew` větší, aktuální pozice zůstává nezměněna.

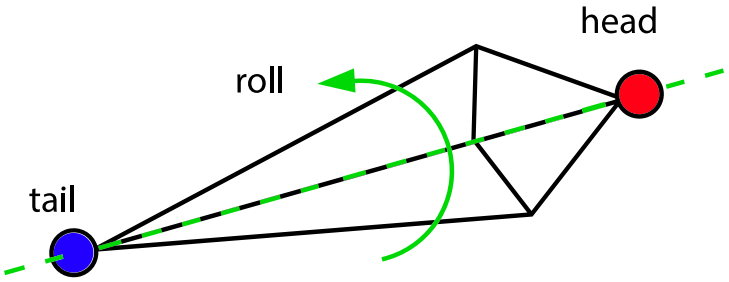
   V obou případech se vrací zpátky ke kroku 1.

Tímto postupem je získána pozice, která co nejvíce odpovídá pozici zaznamenané ve zpracovávaném snímku dat. Ta je poznamenána jakožto snímek animace a začne se zpracovávat další snímek dat.

3.5 Doplnění animace dlaně

Animace dlaně se vytváří v programu Blender. Zde může užviatel ručně nastavovat úhly kostí a posléze je ukládat jako keyframes. Lze tedy takto vytvořit animaci pomocí ručního nastavení pouze relativně malého množství snímků, a o vygenerování zbytku se postará aplikace Blender. Konkrétní postup uživatele při vytvoření animace je popsán v kapitole 7.1.2. Formát animačního souboru dlaně je popsán v 3.7.2.

Pozice každé kosti lze popsat jako pozice její head a tail a hodnota úhlu roll. Grafické znázornění těchto hodnot lze vidět na obrázku 3.2.



Obrázek 3.2: Znázornění kosti

Toto je využíváno ve skriptu, který po spuštění projde všechny animační snímky momentálně aktivní animace a zaznamená do souboru hodnoty pozice tail kostí a orientaci jejich lokální osy z. Tímto se získá jednoznačná reprezentace polohy kosti v prostoru, za předpokladu, že se při vytváření animací měnily pouze úhly kostí a nikoli pozice head v prostoru. Hodnotu úhlu roll lze nahradit orientací lokální osy z díky tomu, jak jsou osy kosti orientovány. Orientace os lze vidět na obrázku 3.3.

Exportovaný soubor je dále potřeba zpracovat v Unity tak, aby byly získány lokální úhly kostí, které jsou zaznamenány ve finálním animačním souboru.

3.6 Spojení obou animací

Pro vytvoření kompletní animace, která bude dále použitelná v rehabilitační aplikaci, je potřeba spojit jednu animaci paže s jednou animací dlaně. Snímky budou propojeny lineárním přepočtem indexů, který určí, které k sobě přísluší. Nebo-li pro každý snímek animace paže hledá odpovídající snímek dlaně. A jelikož jeden snímek animace dlaně je rozdělen na několik řádek, ke každému řádku paže se hledá 21 odpovídajících řádek dlaně.

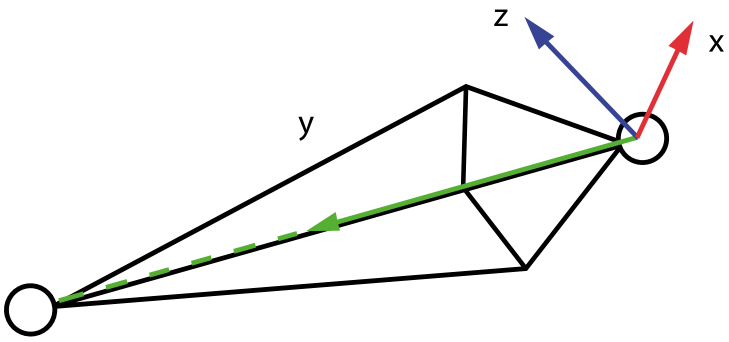
Ovšem nejprve je třeba data animace dlaně převést z hodnot, které jsou zaznamenány při exportu animace z Blenderu, na hodnoty, které jsou uchovávány ve finálním animačním souboru.

Aplikace Blender a Unity každá používají jiný souřadnicový systém. Prvním krokem tedy bude převést hodnoty ze souřadnicového systému Blenderu do souřadnicového systému Unity.

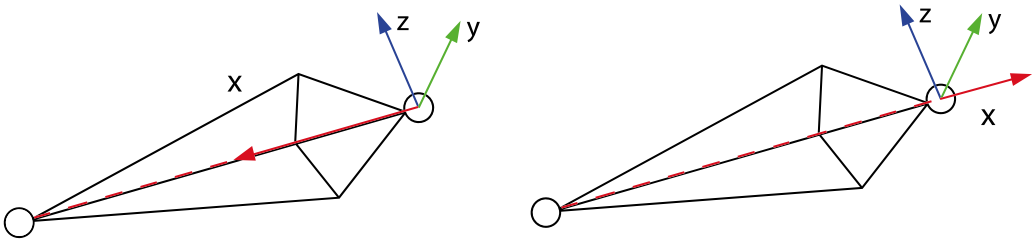
Dále je potřeba vzít v úvahu fakt, že kost v Unity a v Blenderu má jiné orientace lokálních os. Zároveň v Unity záleží na tom, zda se jedná o kost pravé či levé paže.

Zobrazení os kostí v Unity lze vidět na obrázku 3.4 a v Blenderu na obrázku 3.3. Orientace os kostí v Unity na pravé a levé ruce na objektu paže je vidět na obrázku 3.5.

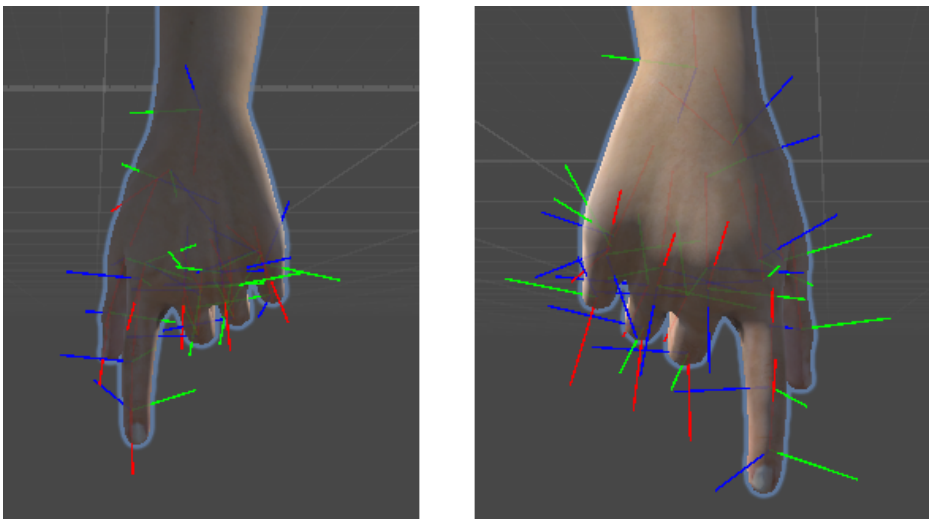
Pokud je toto správně aplikováno, stačí v Unity orientovat osy tak, aby osa x mířila na (resp. od v případě pravé paže) zaznamenané pozice tail a osa z mířila ve stejném směru jako je zaznamenáno. Dále stačí přečíst lokální úhly kosti a výsledek je zapsán do finálního souboru. Formát tohoto souboru je popsán v 3.7.3.



Obrázek 3.3: Znázornění os kostí v Blenderu



Obrázek 3.4: Znázornění os kostí levé a pravé paže v Unity



Obrázek 3.5: Znázornění os kostí na levé a pravé dlani

3.7 Výstupní data

V této sekci se nachází popis formátu jednotlivých souborů s animacemi. Dohromady jsou výstupem práce tři různé typy souborů, z nichž pouze jeden je dále použitelný v rehabilitační aplikaci. Další dva typy jsou mezivýsledky, které lze libovolně kombinovat mezi sebou, například v případě, kdy jedna animace paže bude spojována s dvěmi různými animacemi dlaně.

3.7.1 Animace paže

Pro vytvoření tohoto typu souboru je potřeba vstup v podobě složky s naměřenými daty zachycujícími pohyb. Název souboru je generován podle názvu složky, ve které se vstupní data nacházejí a jedná se o .csv soubor.

Znak středníku (';') soubor rozděluje do šesti sloupců. První tři sloupce popisují globální eulerovské rotace kosti nadloktí v pořadí os x, y a z. Druhé tři popisují globální eulerovské rotace kosti předloktí, opět v pořadí os x, y a z.

Jedna řádka souboru tedy popisuje jeden snímek animace. Jelikož je animace generována tak, aby každý její snímek odpovídal jednomu snímku vstupních dat, má stejný počet řádek jako vstupní soubory.

3.7.2 Animace dlaně

Tento typ souboru je vytvořen pomocí skriptu v programu Blender. Jeho jméno odpovídá názvu exportované animace a opět se jedná o .csv soubor jehož sloupce jsou odděleny středníky (';').

Formát tohoto souboru je následovný:
Na první řádce souboru se nachází hlavička. Jedná se o řádku s textem "//HandAnim". Další řádky již obsahují data o jednotlivých snímcích.

Jedna řádka souboru obsahuje informace o jedné kosti dlaně. Jeden snímek animace je tedy reprezentován 21 řádky.

Počet sloupců v tomto souboru je šest. První tří sloupce obsahují pozici tail kosti na osách x, y a z, v souřadnicové soustavě armature. Druhé tři obsahují směrový vektor lokální osy z stejné kosti. Tyto souřadnice jsou opět zapsány v pořadí x, y a z a jsou v souřadnicové soustavě armature.

3.7.3 Výsledný pohyb

Vstupem k vytvoření posledního typu animačního souboru je jeden soubor s animací paže a jeden soubor s animací dlaně. Jméno výstupního souboru se vytváří jako kombinace názvů vstupních souborů. Pokud se vstupní soubor s animací paže jmenuje "take1" a vstupní soubor s animací dlaně "diagonala" vznikne soubor s názvem "take1_diagonala".

Opět se jedná o .csv soubor, jehož sloupce jsou odděleny středníky (';'). Jeho formát je takový, že se za sebou vždy následují jeden snímek animace paže a jeden snímek animace dlaně. Ve výsledku jeden snímek výsledné animace bude mít 22 řádek, kdy na prvním jsou informace o kostích paže a na zbylých informace o jednotlivých kostech dlaně.

Formát řádky animace paže zůstává nezměněn. Formát jedné řádky animace dlaně je takový, že jedna řádka uchovává informace o rotaci jedné kosti, tentokrát ovšem uložené jako hodnoty lokálních rotací kosti v osách x, y a z.

4 Implementace

V této kapitole se nachází popis implementace aplikace. V první řadě je zde popsána struktura Unity projektu, a dále popsán způsob jakým mezi sebou vytvořené třídy komunikují. Konkrétní popis vytvořených třídy a jejich metod je v Příloze 2 – Programátorská dokumentace.

4.1 Unity projekt

Základem Unity projektu je scéna, do které se umisťují objekty, které se budou po spuštění vykreslovat. Na tyto objekty jsou dále napojovány různé komponenty, jako skripty, materiály, kolidery nebo další. Tyto komponenty ovlivňují chování nebo vzhled daného objektu, na který se váží.

Při vývoji se z velké míry píší a používají skripty. Unity skript je třída, oděděná od třídy MonoBehaviour. Tato třída poskytuje metody, které jsou automaticky vyvolávány během života aplikace. Metody, které budou využívány jsou `Start` a `Update`.

Metoda `Start` se provádí maximálně jednou za život aplikace. K jejímu zavolání dojde, pokud je skript aktivní, jednou při spuštění aplikace. Její kód bude vykonán před prvním voláním `Update`. Je tedy vhodná k nastavení počátečních hodnot atributů, či provedení dalších nutných akcí před vlastním výkonným kódem aplikace.

Dále metoda `Update` bude volána každý zobrazený snímek aplikace. Tato metoda je většinou využívána pro reagování na změny v aplikaci a na akce uživatele.

Další vlastností skriptu je, že pokud má public atribut, jeho hodnotu lze nastavit v editoru.

Dále je možné Unity projekt rozšiřovat za pomocí tzv. assetů, které lze získat přímo v "Asset store" v Unity. V tomto projektu byl využit asset Runtime FileBrowser[12] pro vytvoření dialogového okna pro výběr složek a souborů.

V dalších sekcích bude nejprve popsána vytvořená scéna v Unity, a poté popsáno jak mezi sebou komunikují vlastní vytvořené skripty, které jsou navázány na objekty nebo volány za běhu.

Tyto vytvořené skripty se dají rozdělit do tří skupin:

1. Ovladače scénářů
2. Ovladače objektů
3. Pomocné třídy

4.1.1 Ovladače scénářů

Tyto třídy slouží k řízení průběhu jednotlivých požadavků uživatele. Obstarávají hlavní funkcionalitu aplikace, tedy generování nového pohybu paže, spojování animace paže a animace dlaně a přehrávání vytvořených pohybů. Do této skupiny patří třídy `GenerateController`, `JoinController` a `ReplayController`.

4.1.2 Ovladače objektů

Tyto třídy slouží k manipululaci s objekty ve scéně za běhu aplikace. Do této skupiny patří třídy `AdjustController`, `AnimationController`, `ArmController`, `CameraController`, `DialogHandler`, `StateLine`, `TrackerController` a `UIHandler`.

4.1.3 Pomocné třídy

Třídy, které poskytují pomocné metody pro výpočet hodnot, načítání dat, konfiguraci aplikace nebo se jedná a přepravky dat. Do této skupiny patří třídy `Config`, `DataReader`, `DataWriter`, `FileManip`, `MathFunc` a `Position`.

4.2 Scéna

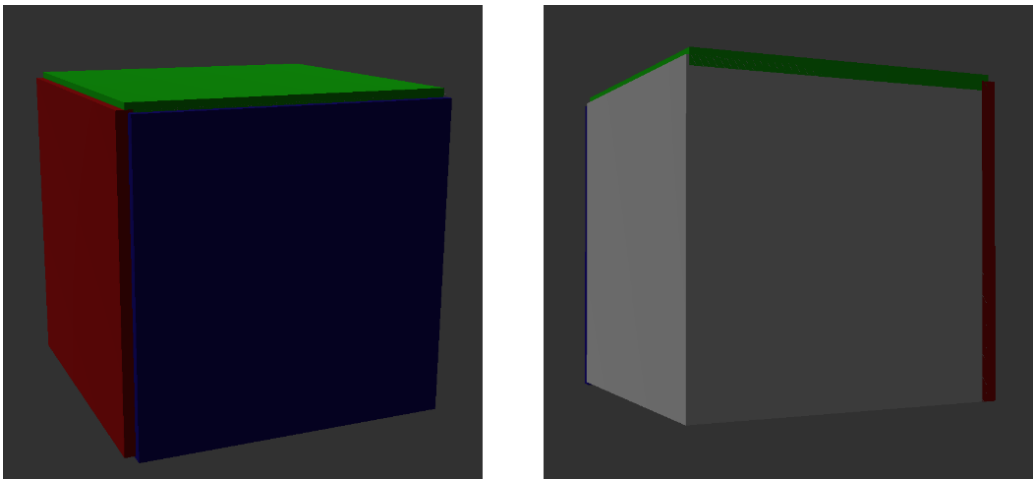
Při spuštění aplikace jsou ve scéně umístěny následující objekty:

Kamera, která slouží k určení, která část scény bude právě vykreslena a její pozici v prostoru lze ovlivňovat za běhu. K tomuto chování slouží na ní připojený skript `CameraController`.

Dále se jedná o dvě světla, která scénu osvětlují. Tato světla jsou umístěna tak, aby co nejlépe vynikl objekt paže.

Dalším objektem je sféra nacházející se v bodě [0, 0, 0]. Ta slouží jako označení místa, na kterém se bude zobrazovat objekt paže. Zároveň jsou na ni připojeny skripty, které mají na starost ovládání paže a obsluhu požadavků uživatele. Konkrétně se jedná o `GenerateController`, `ReplayController`, `ArmController`, `AdjustController` a `JoinController`.

Dále se zde nachází prázdný objekt, který bude za běhu rodičem instancovaných objektů trackerů. Na něj je připojen skript `TrackerController`. Pojem prázdný objekt označuje takový objekt, který je definován pouze svou pozicí v prostoru. Využívá se tedy jako přepravka pro své potomky. Objekty reprezentující trackery nejsou viditelné uživateli, k nahlédnutí jsou na obrázku 4.1.



Obrázek 4.1: Objekt reprezentující tracker

Dalším objektem ve scéně je `EventSystem`. Jedná se o defaultní objekt, který se stará o zpracovávání událostí, které se ve scéně odehrají.

Dále se zde nachází objekt `Canvas`, který obsahuje všechny prvky uživatelského rozhraní (UI). Na tento objekt jsou připojeny skripty `UIHandler` a `StateLine`.

Prvky UI lze rozdělit do několika větších částí: část, obsahující stavový řádek aplikace, části pro ovládání generování animace, spojování animací a přehrávání animací, a nakonec tlačítko pro ovládání, zda má být použita pravá či levá ruka a panel sloužící jako pozadí pro prvky UI.

Stavový řádek aplikace se skládá ze 4 textových polí, může tedy zobrazit jen 4 různé zprávy najednou. Nejnovější zpráva se vždy nachází v textovém poli nejblíže k hornímu okraji okna aplikace.

Části pro ovládání požadavků uživatele se skládají z tlačítek, která spouští akci, nebo otevírají prohlížeč souborů a textového pole pro zobrazení aktuálně vybrané cesty ke vstupním datům.

Posledním, při spuštění neaktivním, potomkem Canvas je panel, který bude zobrazen při vykonávání déle trvající aktivity, která nevyžaduje žádný vstup od uživatele. Jedná se o generování nové animace paže a slučování animací. Tento panel bude dále označován jako obrazovka procentuelního

postupu. Zobrazuje procento, kolik z požadovaného výkonu bylo již zpracováno.

Při běhu aplikace jsou do scény přidávány další objekty. Těmito objekty je objekt paže a objekty reprezentující trackery. Všechny tyto objekty jsou zničeny, pokud nejsou dále potřeba.

V aplikaci se nacházejí dva typy trackerů. První z nich lze označit jako virtuální trackery. Tyto objekty mají stálou polohu na objektu ruky, a představují simulaci trackerů na ruce člověka při provádění pohybu. Druhý typ se označuje jako vedoucí objekty. Tyto objekty reprezentují naměřená data, a vždy se pohybují po korektní trajektorii.

4.3 Komunikace mezi skripty

Jak jsou jednotlivé skripty mezi sebou provázány je vidět na UML diagramu tříd 4.2.

V UML jsou pro přehlednost zakresleny jen skripty ze skupiny Ovladačů scénářů a Ovladačů objektů, kromě třídy `DialogHandler`. Ta slouží jen k ovládání UI a nemá žádnou návaznost na třídy ostatní.

Třída `UIHandler` se stará o zpracování většiny žádostí uživatele. Dále volá metody, které odstartují průběh požadované akce, a metody pro výpis zprávy na obrazovku a zobrazení obrazovky procentuelního postupu poskytované třídou `StateLine`. Tato třída pří každém volání `Update` kontroluje průběh probíhajících požadavků a updatuje zprávy zobrazené v UI.

4.3.1 Generování animace paže

Po stisknutí tlačítka, které odstartuje generování animace paže `UIHandler` volá metodu `StartGeneration` třídy `GenerateController`, která spustí samotné generování.

Tato třída poté při každém volání metody `Update` volá metodu `AnimationController.NextFrame`, kde probíhá výpočet dalšího animačního snímku. Ke generování animačních snímků je potřeba metod ze tříd `AdjustController`, `ArmController` a `TrackerController`.

Po vygenerování všech animačních snímků je výsledná animace zapsána do souboru a třída `UIHandler` se postará o spuštění jejího přehrání.

4.3.2 Spojování

Po stisknutí tlačítka, které odstartuje spojování animace paže a animace dlaně `UIHandler` volá metodu `StartJoin` třídy `JoinController`, která spustí

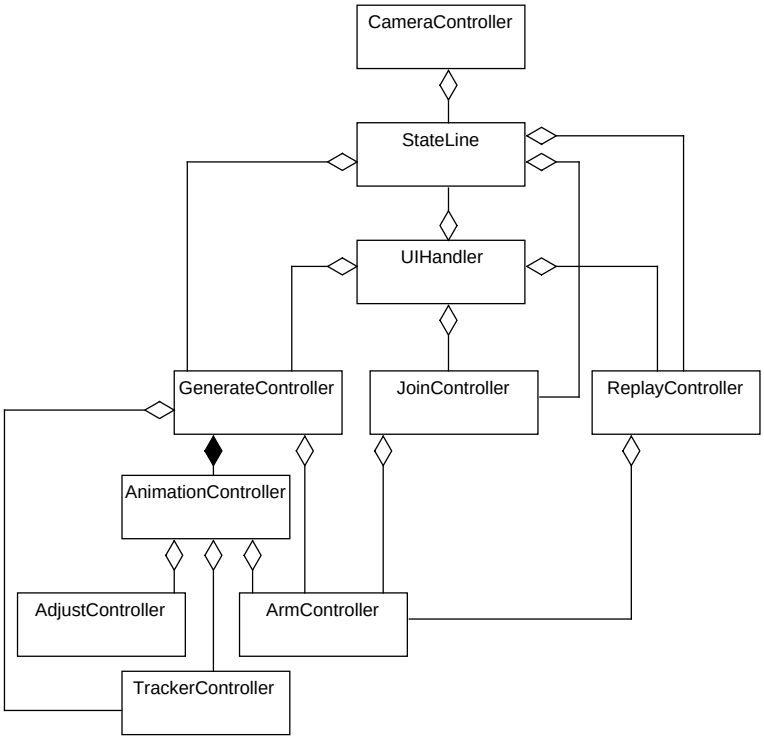
samotné spojování.

Tato třída poté při každém volání metody `Update` volá metodu `ArmController.MoveHand`, s jejíž pomocí převádí snímky animace dlaně na data, která budou zapsaná ve finální animaci. Po převedení všech snímků, sloučí animaci dlaně s animací paže a výsledek zapíše do souboru.

4.3.3 Přehrávání

Po stisknutí tlačítka, které odstartuje přehrávání animace `UIHandler` volá metodu `StartReplay` třídy `ReplayController`, která spustí samotné přehrávání.

Tato třída poté při každém volání metody `Update` nastaví objektu paže hodnoty načtené ze souboru. K tomu využívá metody poskytované třídou `ArmController`.



Obrázek 4.2: UML diagram tříd projektu

5 Testování

V této kapitole se nachází testování korektnosti výsledků aplikace podle několika kritérií.

5.1 Naměřené pohyby

Aplikace byla testována na datové sadě naměřených pohybů. Tyto pohyby byly vytvořeny záznamem cvičení pana L. Rodiny. Jedná se tedy o pohyby, které budou dále reálně využity v rehabilitační aplikaci a tedy i v praxi.

Tyto pohyby se skládaly ze tří diagonálních pohybů paže, pravotočivé a levotočivé spirály, měřených pro pravou i levou ruku. Dále byl naměřen pohyb vstávání ze židle a opětovaného sedání, tento pohyb byl naměřen jen pro pravou ruku.

Prvním kritériem ohodnocení výsledků tedy bylo, jak se animace vizuálně jevily. Toto je důležité, jelikož animace budou dále prezentovány pacientům a fyzioterapeutům, kteří budou jejich kvalitu hodnotit pouze vizuálně.

Výsledné animace vizuálně vypadaly velice podobně, jako předcvičení. Samozřejmě je nutné vzít v úvahu transformaci dat (viz. 3.1). Na přiloženém CD se nacházejí nahrávky předcvičovaných pohybů a soubory obsahující výsledné animace.

5.2 Kvalita pohybu

Dále lze vyhodnocovat kvalitu výsledných animací se zaměřením na jejich přesnost, s jakou odpovídají vstupním datům. K tomuto byly použity následující metriky:

- vzdálenost virtuálních trackerů a vedoucích objektů
- úhlové rozdíly virtuálních trackerů a vedoucích objektů
- vzdálenost virtuálních trackerů mezi dvěma po sobě jdoucími snímky

Kvalita byla testována na animačních datech přiložených na CD, která byla dále používána i v rehabilitační aplikaci. Aplikace pro generování animací dvakrát za sebou nevygeneruje stejné výsledky, jelikož část jejího algoritmu je založena na náhodě, a bude tedy záležet i na pořadí, v jakém budou

animační soubory generovány. Pro nově vygenerovanou animaci tedy mohou vycházet trochu jiné hodnoty těchto metrik.

První dvě metriky hodnotí, jaká byla celková odchylka virtuálních trackerů od vedoucích objektů. Výsledky pro vytvořené animace jsou zaznamenány v tabulkách 5.1 pro tracker předloktí a 5.2 pro tracker nadloktí.

V tabulkách vidíme, že maximální i průměrné odchylky úhlů virtuálních trackerů od vedoucích objektů při přehrávání animace se ve většině případů pohybovaly v hodnotách pod jeden stupeň. Největší hodnota maximální odchylky byla 2.29 stupně v ose x u virtuálního trackeru nadloktí u animace prvního měření pohybu "Spirala B" pravé ruky. Největší hodnota průměrné odchylky byla 2.60 stupně v ose x u virtuálního trackeru nadloktí u stejné animace. Průměr průměrných průměrných odchylek vychází na 0.33 stupně.

Bylo experimentálně ověřeno, že pokud jsou hodnoty odchylky pod pět stupňů, lze výsledky prohlásit za odpovídající.

Dále si je nutné uvědomit, že vzhledem k datům, která jsou uchovávána v animačním souboru, vzdálenost virtuálních trackerů a vedoucích objektů při přehrávání animace není nejobjektivnějším hodnocením. V animačním souboru jsou uchovávány jen úhly kostí, poloha ramene je v rehabilitační aplikaci odvíjena z polohy aktuálního uživatele. Proto ji uchovávat v souboru s animací nemá smysl. Naopak je ovšem pravděpodobné, že pokud by hodnota odchylky v poloze virtuálního trackeru od vedoucího objektu vycházela příliš velká, mohlo by se jednat o selhání animačního sofware.

Z výsledků lze vyčíst, že maximální vzdálenosti virtuálních trackerů od vedouchích objektů se pohybovaly v rozsahu 0.10 až 0.21 metru. Maximální průměrěná odchylka byla 0.03 metru. Tyto odchylky se dají odůvodnit zmíněnou absencí údajů o umístění ramene v prostoru.

Dle těchto výsledků lze soudit že aplikace generuje animace odpovídající naměřeným datům.

Třetí metrika udává jaký je největší "skok" mezi dvěma animačními snímky. Pokud zde budou vycházet vysoké hodnoty, znamená to, že daný posuv bude potenciálně zaznamenatelný lidským okem a naruší plynulost animace. Výsledky pro vytvořené animace jsou zaznamenány v tabulce 5.3.

Samozřejmě, určitý posun mezi dvěma po sobě jdoucími snímky být musí, jinak by se nejednalo o animaci. Je nutné ovšem zhodnotit, jaká velikost kroku je již nadměrná. Lze odhadnout, že hodnoty skoku pod jeden centimetr budou odpovídat posunu při pohybu v rámci animace.

Skok může vzniknout rovnou ve vstupních datech, pokud při záznamu HTC Vive nemá ideální "výhled" na trackery, nebo pokud byl pohyb vykonáván příliš rychle. Tyto odchylky se ve vstupních datech nevyskytovaly, ovšem je to něco, na co si musí uživatel aplikace dávat pozor. Dále skok

může vzniknout pokud je pozice paže vyhodnocena jako optimální, přestože existuje pozice s menším ohodnocením. Tomuto je snaha předejít perturbacemi, tento postup ovšem nemusí být úspěšný ve všech případech. Dalo by se dále zvyšovat počet perturbací, ovšem toto by se dělo na úkor času, který je potřeba k vytvoření jedné animace paže.

Největší hodnotou skoku v animacích je 0.029 metru u virtuálního trackeru předloktí u animace prvního měření pohybu "Stand Up" pravé ruky. Největší hodnotou průměrného skoku byla hodnota 0.004 metru, přičemž u trackeru předloktí vycházely větší hodnoty. Důvodem je, že předloktí se při naměřených pohybech také více pohybuje.

Ve všech pohybech se vyskytuje přinejmenším jeden zazamenatelný skok v animaci, ale jelikož všechny průměrné hodnoty skoku jsou menší než pět milimentrů, bude plynulost pohybu obecně zachována.

Vysvětlivky k tabulkám:

- n – číslo měření
- t – typ paže (zda se jendá o pravou nebo levou)
- a – vektor maximálních odchylek os vedoucího objektu a virtuálního trackeru ve stupních, pořadí os je x, y, z
- $\bar{a}$ – vektor průměrných odchylek os vedoucího objektu a virtuálního trackeru ve stupních, pořadí os je x, y, z
- d – maximální vzdálenost virtuálního trackeru od vedoucího objektu v metrech
- $\bar{d}$ – průměrná vzdálenost virutálního trackeru od vedoucího objektu v metrech
- d_f – maximální vzdálenost pozic virtuálního trackeru předloktí ve dvou po sobě jdoucích snímkách v metrech
- $\bar{d_f}$ – průměrná vzdálenost pozic virtuálního trackeru předloktí ve dvou po sobě jdoucích snímkách v metrech
- d_u – maximální vzdálenost pozic virtuálního trackeru nadloktí ve dvou po sobě jdoucích snímkách v metrech
- $\bar{d_u}$ – průměrná vzdálenost pozic virtuálního trackeru nadloktí ve dvou po sobě jdoucích snímkách v metrech

5.3 Uživatelé

Druhá verze rehabilitační aplikace, která obsahovala animace vygenerované podle posledních naměřených dat měla být testována cílovými uživateli a panem L. Rodinou na konci března. Bohužel kvůli pandemii Covid-19 se toto testování uskutečnit nemohlo.

Pohyb	t	n	a	$\bar{a}$	d	$\bar{d}$
Diagonala 1	L	1	[0.80 0.81 0.88]	[0.33 0.40 0.39]	0.13	0.01
Diagonala 1	L	2	[0.79 0.81 0.79]	[0.31 0.29 0.28]	0.12	0.01
Diagonala 1	L	3	[0.83 0.83 0.90]	[0.31 0.35 0.35]	0.13	0.01
Diagonala 2	L	1	[0.83 0.71 0.76]	[0.35 0.34 0.37]	0.19	0.02
Diagonala 2	L	2	[0.84 0.83 0.83]	[0.33 0.28 0.30]	0.20	0.02
Diagonala 2	L	3	[0.78 0.78 0.84]	[0.35 0.38 0.40]	0.20	0.02
Diagonala 3	L	1	[0.93 0.88 0.91]	[0.43 0.43 0.46]	0.20	0.03
Diagonala 3	L	2	[0.75 0.87 0.95]	[0.31 0.37 0.36]	0.20	0.03
Diagonala 3	L	3	[0.83 0.85 0.90]	[0.40 0.36 0.37]	0.21	0.02
Spirala A	L	1	[0.91 0.89 0.89]	[0.26 0.30 0.30]	0.12	0.01
Spirala A	L	2	[0.58 0.83 0.83]	[0.25 0.27 0.33]	0.12	0.01
Spirala A	L	3	[0.63 0.86 0.90]	[0.23 0.27 0.30]	0.12	0.01
Spirala B	L	1	[0.68 0.85 0.90]	[0.22 0.31 0.30]	0.12	0.01
Spirala B	L	2	[0.71 0.82 0.82]	[0.28 0.35 0.34]	0.11	0.01
Spirala B	L	3	[0.66 0.82 0.86]	[0.28 0.33 0.33]	0.12	0.01
Diagonala 1	R	1	[0.90 0.73 0.90]	[0.36 0.33 0.33]	0.20	0.02
Diagonala 1	R	2	[0.72 0.84 0.89]	[0.35 0.40 0.39]	0.21	0.03
Diagonala 1	R	3	[0.94 0.81 0.94]	[0.43 0.39 0.51]	0.20	0.02
Diagonala 2	R	1	[1.48 1.21 1.38]	[0.40 0.41 0.37]	0.19	0.02
Diagonala 2	R	2	[0.87 0.87 0.82]	[0.35 0.41 0.41]	0.19	0.02
Diagonala 2	R	3	[0.84 0.91 0.82]	[0.38 0.43 0.41]	0.18	0.02
Diagonala 3	R	1	[0.81 0.90 0.79]	[0.39 0.42 0.40]	0.18	0.02
Diagonala 3	R	2	[0.86 0.84 0.79]	[0.36 0.30 0.32]	0.19	0.01
Diagonala 3	R	3	[0.91 0.88 0.90]	[0.42 0.33 0.37]	0.18	0.01
Spirala A	R	1	[1.21 1.02 1.21]	[0.51 0.49 0.58]	0.17	0.03
Spirala A	R	2	[1.13 1.03 1.05]	[0.53 0.51 0.59]	0.17	0.02
Spirala A	R	3	[1.15 1.15 1.18]	[0.53 0.43 0.57]	0.17	0.03
Spirala B	R	1	[1.89 0.96 1.84]	[0.55 0.51 0.65]	0.18	0.03
Spirala B	R	2	[1.15 0.95 1.18]	[0.55 0.50 0.64]	0.17	0.03
Spirala B	R	3	[1.17 0.92 1.19]	[0.55 0.44 0.62]	0.19	0.03
Stand up	R	1	[0.74 0.86 0.88]	[0.34 0.37 0.36]	0.19	0.03
Stand up	R	2	[0.80 0.72 0.80]	[0.31 0.33 0.32]	0.21	0.03
Stand up	R	3	[0.75 0.84 0.86]	[0.33 0.37 0.36]	0.18	0.02

Tabulka 5.1: Kvalita animace trackeru na předloktí podle odchylky od dat

Pohyb	t	n	a	$\bar{a}$	d	$\bar{d}$
Diagonala 1	L	1	[0.72 0.70 0.68]	[0.25 0.25 0.22]	0.10	0.01
Diagonala 1	L	2	[0.77 0.77 0.66]	[0.29 0.30 0.28]	0.10	0.01
Diagonala 1	L	3	[0.79 0.70 0.72]	[0.31 0.33 0.33]	0.11	0.01
Diagonala 2	L	1	[0.76 0.64 0.71]	[0.26 0.28 0.25]	0.13	0.01
Diagonala 2	L	2	[0.75 0.77 0.74]	[0.34 0.33 0.26]	0.14	0.01
Diagonala 2	L	3	[0.67 0.67 0.60]	[0.23 0.26 0.26]	0.13	0.01
Diagonala 3	L	1	[0.74 0.76 0.75]	[0.27 0.30 0.30]	0.17	0.02
Diagonala 3	L	2	[0.66 0.59 0.68]	[0.23 0.28 0.28]	0.18	0.02
Diagonala 3	L	3	[0.68 0.63 0.67]	[0.27 0.31 0.32]	0.19	0.02
Spirala A	L	1	[0.69 0.69 0.71]	[0.34 0.28 0.28]	0.12	0.01
Spirala A	L	2	[0.73 0.78 0.73]	[0.26 0.26 0.23]	0.12	0.01
Spirala A	L	3	[0.60 0.65 0.63]	[0.30 0.27 0.28]	0.10	0.01
Spirala B	L	1	[0.73 0.75 0.77]	[0.27 0.27 0.24]	0.11	0.01
Spirala B	L	2	[0.79 0.79 0.79]	[0.28 0.29 0.26]	0.10	0.01
Spirala B	L	3	[0.63 0.70 0.74]	[0.28 0.28 0.24]	0.10	0.01
Diagonala 1	R	1	[0.75 0.78 0.66]	[0.24 0.29 0.31]	0.15	0.02
Diagonala 1	R	2	[0.64 0.78 0.71]	[0.28 0.31 0.31]	0.15	0.02
Diagonala 1	R	3	[0.70 0.70 0.73]	[0.24 0.27 0.28]	0.14	0.02
Diagonala 2	R	1	[0.97 0.77 0.97]	[0.25 0.26 0.29]	0.16	0.02
Diagonala 2	R	2	[0.71 0.78 0.70]	[0.25 0.26 0.28]	0.16	0.02
Diagonala 2	R	3	[0.73 0.71 0.75]	[0.24 0.26 0.25]	0.15	0.02
Diagonala 3	R	1	[0.77 0.76 0.77]	[0.26 0.27 0.28]	0.16	0.02
Diagonala 3	R	2	[0.77 0.81 0.75]	[0.32 0.30 0.27]	0.16	0.02
Diagonala 3	R	3	[0.71 0.71 0.70]	[0.29 0.30 0.27]	0.16	0.02
Spirala A	R	1	[2.10 2.06 1.76]	[1.38 1.34 0.87]	0.16	0.02
Spirala A	R	2	[2.13 2.10 1.73]	[1.39 1.31 0.87]	0.14	0.02
Spirala A	R	3	[2.07 1.39 2.00]	[1.05 0.81 1.16]	0.15	0.02
Spirala B	R	1	[2.29 2.22 1.98]	[1.60 1.46 1.17]	0.16	0.02
Spirala B	R	2	[2.23 2.24 1.93]	[1.60 1.47 1.20]	0.16	0.02
Spirala B	R	3	[2.09 1.42 2.25]	[1.25 0.80 1.43]	0.17	0.02
Stand up	R	1	[0.81 0.74 0.78]	[0.31 0.28 0.33]	0.17	0.02
Stand up	R	2	[0.85 0.67 0.86]	[0.24 0.25 0.25]	0.17	0.02
Stand up	R	3	[0.72 0.84 0.74]	[0.26 0.27 0.27]	0.17	0.02

Tabulka 5.2: Kvalita animace trackeru na nadloktí podle odchylky od dat

Pohyb	t	n	d_f	$\bar{d}_f$	d_u	$\bar{d}_u$
Diagonala 1	L	1	0.020	0.003	0.008	0.001
Diagonala 1	L	2	0.019	0.004	0.007	0.001
Diagonala 1	L	3	0.017	0.003	0.006	0.001
Diagonala 2	L	1	0.018	0.003	0.005	0.001
Diagonala 2	L	2	0.016	0.003	0.005	0.001
Diagonala 2	L	3	0.016	0.003	0.005	0.001
Diagonala 3	L	1	0.022	0.004	0.007	0.001
Diagonala 3	L	2	0.023	0.004	0.007	0.001
Diagonala 3	L	3	0.019	0.003	0.006	0.001
Spirala A	L	1	0.020	0.003	0.005	0.001
Spirala A	L	2	0.016	0.003	0.004	0.001
Spirala A	L	3	0.020	0.003	0.007	0.001
Spirala B	L	1	0.018	0.003	0.006	0.001
Spirala B	L	2	0.018	0.004	0.005	0.001
Spirala B	L	3	0.023	0.003	0.007	0.001
Diagonala 1	R	1	0.020	0.003	0.006	0.001
Diagonala 1	R	2	0.025	0.004	0.006	0.001
Diagonala 1	R	3	0.024	0.004	0.006	0.001
Diagonala 2	R	1	0.032	0.004	0.007	0.001
Diagonala 2	R	2	0.026	0.004	0.006	0.001
Diagonala 2	R	3	0.020	0.004	0.005	0.001
Diagonala 3	R	1	0.022	0.004	0.005	0.001
Diagonala 3	R	2	0.019	0.004	0.005	0.001
Diagonala 3	R	3	0.027	0.004	0.006	0.001
Spirala A	R	1	0.020	0.003	0.006	0.001
Spirala A	R	2	0.021	0.003	0.006	0.001
Spirala A	R	3	0.013	0.003	0.005	0.001
Spirala B	R	1	0.018	0.003	0.005	0.001
Spirala B	R	2	0.018	0.004	0.006	0.001
Spirala B	R	3	0.021	0.004	0.008	0.001
Stand up	R	1	0.029	0.002	0.007	0.001
Stand up	R	2	0.061	0.003	0.014	0.001
Stand up	R	3	0.051	0.003	0.013	0.001

Tabulka 5.3: Kvalita animace podle vzdálenosti dvou po sobě jdoucích snímků

6 Závěr

Cílem bylo naprogramování aplikace pro vytváření animací paže. Tato aplikace vznikla, a byla v ní generována animační data, která byla dále využívána rehabilitační aplikací.

Vlastní export animace dlaně nebylo možné zahrnout přímo do aplikace v Unity. Důvodem je fakt, že k importování .blend souboru Unity potřebuje fbx converter, který nemá k dispozici při spuštění aplikace mimo Unity editor. Proto by nebylo možné zajistit, aby uživatel měl přístup k objektu paže a upravoval jeho animační data, která by aplikace po spuštění využívala.

Jako další rozšíření by tedy bylo možné vytvořit aplikaci, která by umožňovala přímo vytváření a export animací dlaně. Jelikož cílovou skupinou jsou fyzioterapeuti, kterým by mohly rozsáhlé možnosti a schopnosti aplikace Blender způsobovat problémy.

Výsledné animace byly zhodnoceny podle několika kritérií a byla ověřena přesnost generování.

Samotné generování animace paže na testovacím stroji trvalo průměrně hodinu, dalo by se tedy dále zaměřit na další urychlení algoritmu. Je ovšem otázkou do jaké míry je toto možné. Nejrychlejšího generování bylo v rámci testování dosaženo obětováním responzivity UI, a na testovacím stroji jeden pohyb poté trval průměrně 40 minut.

Výsledné animace odpovídají naměřeným pohybům s tolerovatelnými odchylkami a vizuálně vypadají jako předcvičované pohyby.

Zároveň aplikace vytvořená v Unity poskytuje přehledné UI pro použití uživateli. Postup při jejím použití je popsán v příloze č.1 Uživatelská dokumentace. Zde se nachází i návod k exportování animace dlaně v programu Blender.

7 Přílohy

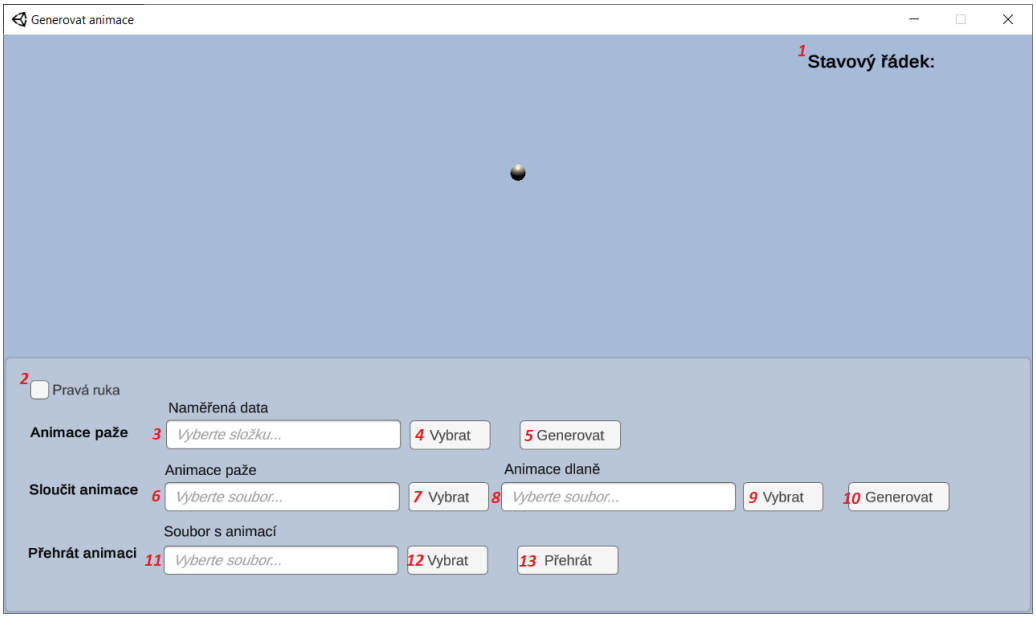
7.1 Příloha A – Uživatelská dokumentace

Zde se nachází postup jak zacházet s aplikací pro generování pohybů a jak vytvořit a exportovat novou animaci dlaně v programu Blender.

7.1.1 Aplikace v Unity

Aplikace je spuštěna dvojklikem na soubor "Generovat animace.exe". Otevře se okno aplikace, které je nakonfigurované na velikost 2/3 výšky a 2/3 šířky obrazovky.

Nejprve se zobrazí logo Unity a poté se zobrazí interface aplikace pro generování animací. Toto interface je vidět na obrázku 7.1.



Obrázek 7.1: Okno aplikace pro generování animací, červená čísla slouží pro snažší identifikaci jednohlivých prvků v dokumentaci

Terminologie

Naměřená data – tento termín označuje data, která byla vytvořena záznamem pohybu.

Animace paže – tento termín označuje animaci, která vznikla generováním z naměřených dat.

Animace dlaně – tento termín označuje animaci, která vznikla exportem z programu Blender podle postupu v kapitole 7.1.2.

Finální animace – tento termín označuje animaci, která vznikla sloučením animace paže a animce dlaně. Tuto animaci lze využít v rehabilitačním software.

Popis uživatelského rozhraní

Na obrázku 7.1 je vidět okno aplikace, ve kterém jsou čísly zvýrazněny prvky, které jsou v dalším textu detailněji popisovány.

1. Stavová řádka

   Zde budou zobrazovány zprávy od aplikace během vykonávaných úkolů. Vypisuje se například, že bylo odstartováno přehrávání, která animace se přehrává a poté, že přehrávání skončilo či bylo přerušeno. Příklad výpisů lze vidět na obrázku 7.2.

2. Toggle

   Zde lze zvolit s jakou paží se nyní bude pracovat. Pokud tato možnost není zaškrtnuta, pracuje se s levou paží. Pokud zaškrtnuta je, tak s pravou.

Řádka Animace paže

3. Textové pole

   Do tohoto textového pole lze zadat absolutní cestu ke složce, která obsahuje naměřená data pohybu, který se bude generovat jako animace paže.

   Cestu lze zadat přímo z klávesnice nebo lze využít tlačítko 4.

4. Tlačítko Vybrat

   Toto tlačítko zobrazí dialog, který umožňuje procházet soubory na disku a vybrat vstupní složku, která obsahuje naměřená data pohybu. Tato cesta bude zobrazena v textovém poli 3.

5. Tlačítko Generovat

   Tlačítko, které odstartuje generování animace paže. Po jeho stisknutí se načtou data z cesty, zapsané v textovém poli 3. Dále se zatmaví obrazovka a aplikace začne zobrazovat postup generování. Tuto obrazovku lze vidět na obrázku 7.3.

Řádka Sloučit animace

6. Textové pole

   Do tohoto textového pole lze zadat absolutní cestu k animačnímu souboru paže. Cestu lze zadat přímo z klávesnice nebo lze využít tlačítko 7.

7. Tlačítko Vybrat

   Toto tlačítko zobrazí dialog, který umožňuje procházet soubory na disku a vybrat vstupní animační soubor paže. Tato cesta bude zobrazena v textovém poli 6.

8. Textové pole

   Do tohoto textového pole lze zadat absolutní cestu k animačnímu souboru dlaně. Cestu lze zadat přímo z klávesnice nebo lze využít tlačítko 9.

9. Tlačítko Vybrat

   Toto tlačítko zobrazí dialog, který umožňuje procházet soubory na disku a vybrat vstupní animační soubor dlaně. Tato cesta bude zobrazena v textovém poli 8.

10. Tlačítko Generovat

    Tlačítko, které odstartuje slučování animace paže a animace dlaně do finálního animačního souboru. Po jeho stisknutí se načtou data z cest, zapsaných v textových polích 8 a 6. Dále se zatmaví obrazovka a aplikace začne zobrazovat postup slučování. Tuto obrazovku lze vidět na obrázku 7.3.

Řádka Přehrát animaci

11. Textové pole

    Do tohoto textového pole lze zadat absolutní cestu k animačnímu souboru, který bude následně přehrán. Cestu lze zadat přímo z klávesnice nebo lze využít tlačítko 12.

12. Tlačítko Vybrat

    Toto tlačítko zobrazí dialog, který umožňuje procházet soubory na disku a vybrat vstupní animační soubor. Tato cesta bude zobrazena v textovém poli 11.

13. Tlačítko Přehrát

    Tlačítko, které odstartuje přehrávání animace. Po jeho stisknutí se načte animační soubor z cesty, zapsané v textovém poli 11, zobrazí se paže a začne se přehrávat animace. Aplikaci během přehrávání lze vidět na obrázku 7.2.

Popis práce s aplikací – Generování animace paže

Předpokladem pro zahájení práce, je mít připravenou složku s naměřenými daty o pohybu, který bude třeba animovat. V této složce jsou potřeba soubory Chest.csv, Head.csv, Arm.csv a Forearm.csv.

Poté je třeba cestu k této složce zadat do textového pole 3, popřípadě tuto složku vybrat v dialogu, který je zobrazen stisknutím na tlačítko 4 Vybrat. Poté je třeba zmáčknout tlačítko 5 Generovat. Stisknutím na toto tlačítko se odstartuje generování animace paže.

Nyní se začne generovat animační soubor. Tento proces bude trvat řádově hodinu. Během této doby, je na obrazovce zobrazován postup generování v procentech.

Po vygenerování animace, je tato animace automaticky přehrána a animační soubor je uložen do složky "output". Tato složka se nachází ve stejném adresáři jako soubor "Generovat animace.exe". Jméno tohoto souboru je stejné, jako bylo jméno složky, která obsahovala naměřená data.

Popis práce s aplikací – Generování finální animace

Předpokladem pro zahájení práce je existence souboru s animací paže, která byla vygenerována touto aplikací, a souboru s animací dlaně, která byla vytvořena exportem animace z aplikace Blender postupem popsaném v kapitole 7.1.2.

Cestu k animačnímu souboru paže je třeba zadat do textového pole 6 a cestu k animačnímu souboru dlaně je třeba zadat do textového pole 8. Popřípadě je možné použít dialog, který bude spuštěn tlačítkem 7 resp. 9 a dané soubory vybrat tímto způsobem.

Dále je třeba zmáčknout tlačítko 10 Generovat. Tímto se spustí slučování vybraných souborů. Tento proces trvá řádově vteřiny. Aplikace bude opět

zobrazovat postup v procentech, jako při generování animace paže. Po ukončení slučování se vytvoří ve složce "output" soubor s finální animací. Název tohoto souboru je vytvořen zřetězením názvů animačních souborů paže a dlaně.

Soubor, který vznikne tímto postupem, lze dále využít v rehabilitační aplikaci.

Popis práce s aplikací – Přehrání animace

Dále lze také přehrávat animační soubory, které byly v minulosti touto aplikací vygenerovány. Stačí zadat cestu k danému souboru do textového pole 11, či vybrat daný soubor v dialogu, který je vytvořen po stisknutí na tlačítko 12. A poté stisknout tlačítko 13 Přehrát.

Popis práce s aplikací – Klávesové zkratky

Jakoukoli probíhající akci lze zastavit stisknutím kláves Ctrl+Q.

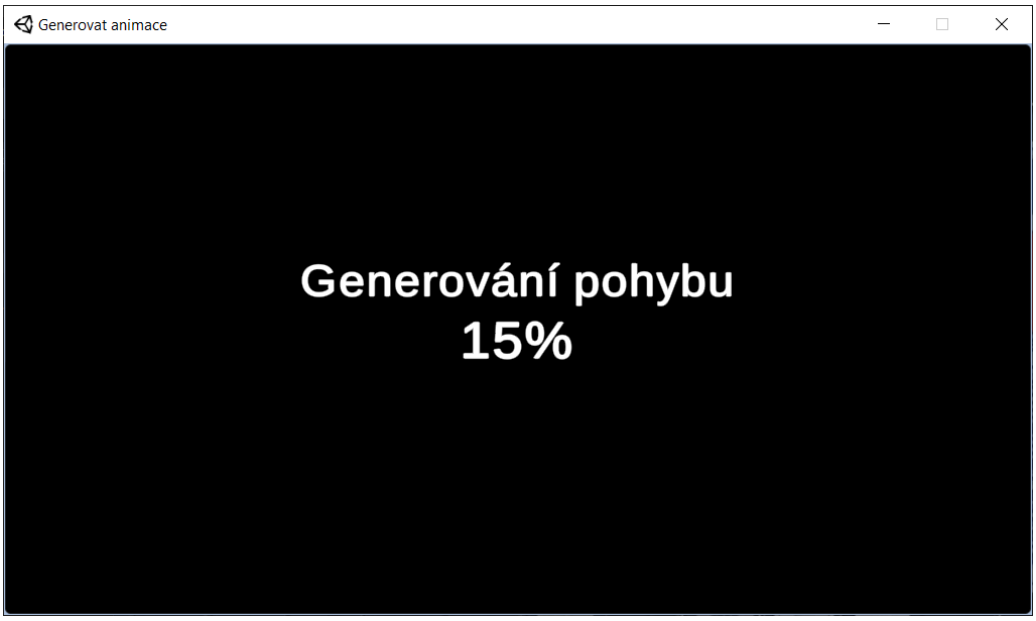
Dále je možné pohybovat kamerou okolo objektu paže a prohlížet si ji z jiných úhlů pohledu. Nejprve je potřeba zajistit, aby nebyl vybraný jiný prvek interface. Stačí kliknout kamkoli do prázdného prostoru nad šedým panelem s prvky, které slouží k ovládání aplikace.

Dále kamera reaguje na následující klávesy:

- W – pohyb dopředu
- A – pohyb doleva
- S – pohyb dozadu
- D – pohyb doprava
- Q – pohyb vzhůru
- E – pohyb dolů
- šipka nahoru – otáčení se nahoru
- šipka dolů – otáčení se dolů
- šipka doleva – otáčení si doleva
- šipka doprava – otáčení se doprava
- R – resetuje pozici kamery, tak aby bylo zobrazeno to, co při spuštění aplikace

Dále lze stisknout klávesu levý Shift, která pohyb po dobu jejího držení urychlí.

Obrázek 7.2: Okno aplikace s momentálně přehrávaným pohybem



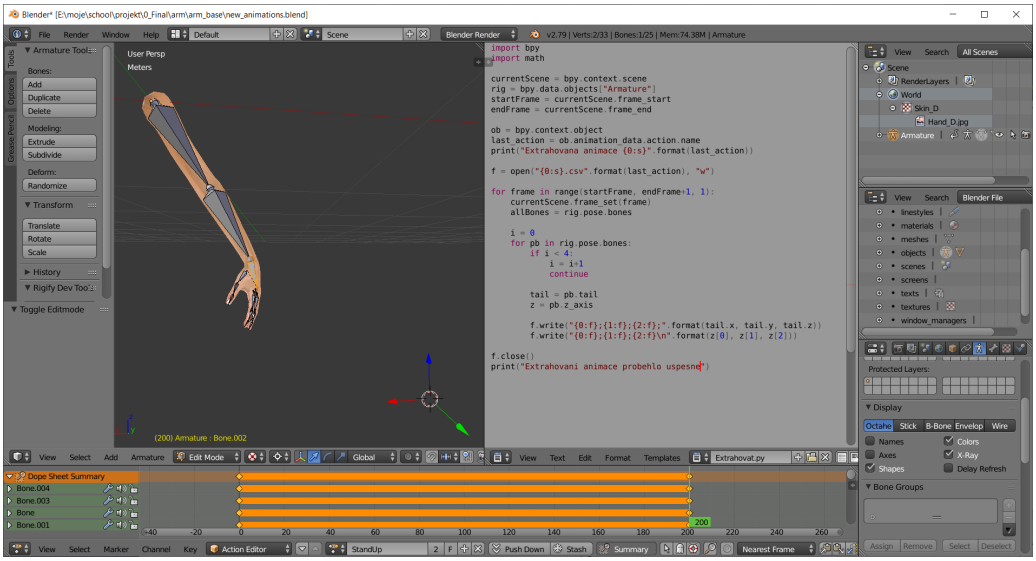
Obrázek 7.3: Okno s procentuálním výpisem postupu generování animace

7.1.2 Doplnění animace v blenderu

Předpokladem pro tuto kapitolu je instalovaný Blender v. 2.79. Novou animaci stačí vytvořit pouze na jedné paži, exportovanou animaci bude možné použít pro slučování s animacemi levé i pravé paže.

Prvním krokem je otevřít v aplikaci soubor s levou nebo pravou paží. Toto lze provést dvěmi způsoby. Prvním z nich je dvouklikem na daný .blend soubor. Druhou je z menu nabídky aplikace Blender zadat "File -> Open" a zvolit požadovaný soubor v otevřeném prohlížeči.

Nyní se na obrazovce objeví několik oddílů, pomocí kterých je možné soubor upravovat. Toto okno by mělo vypadat jako na obrázku 7.4, pokud některý z důležitých oddílů chybí, bude dále popsáno, jakým způsobem jej otevřít.

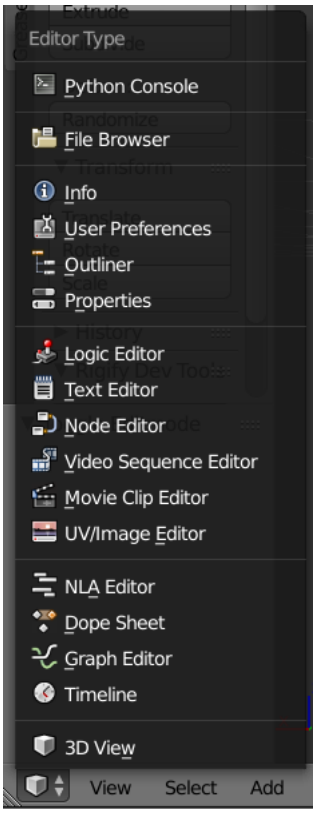


Obrázek 7.4: Okno aplikace Blender po spuštění

Pro vytvoření nového oddílu stačí potáhnout za šrafovaný pravý horní nebo levý dolní roh některého s existujících oddílů směrem do tohoto oddílu. Vytvoří se jeho kopie, ve které lze posléze zobrazit jiný editor.

Naopak oddíl lze zavřít tak, že se potáhne za jeden ze šrafovaných rohů sousedního oddílu směrem dovnitř toho oddílu, který chceme zrušit. Zobrazí se šipka, která znázorňuje, jak oddíl bude zrušen.

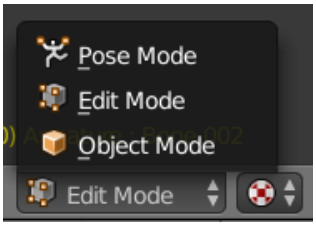
Hlavním oddílem, který bude potřeba je oddíl se scénou, ve které je zobrazen objekt ruky. Pokud tento oddíl není zobrazen, stačí vytvořit nový oddíl (nebo si vybrat, který nahradit) a kliknout na ikonku v levém dolním rohu daného oddílu. Zobrazí se menu "Editor Type" ze kterého vybrat "3D View". Toto menu je vidět na obrázku 7.5.



Obrázek 7.5: Nabídka editorů

V prostoru scény se lze pohybovat pomocí myši. Stisknutím pravého tlačítka myši se přemisťuje 3D kurzor v prostoru. Stisknutím levého tlačítka myši se vybírají objekty. Stisknutím prostředního tlačítka a potažením myši dochází k rotování. Pokud je zároveň ještě stisknutá klávesa Shift, bude docházet místo rotování k posouvání v prostoru. Dále scrollováním lze přibližovat a k oddalovat.

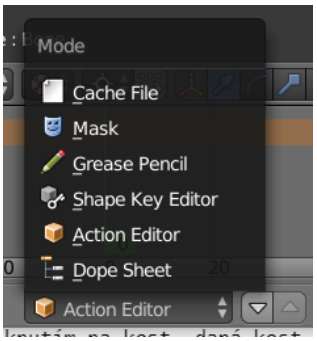
Dále je třeba přepnout se do "Pose mode". K tomu je třeba nejprve vybrat kostru, kliknutím na kteroukoli z jejich kostí, a stisknout tlačítko, na kterém je napsaný momentální zvolený mód. Otevře se nabídka s možnostmi (viz obr. 7.6). Zde vybrat "Pose mode".



Obrázek 7.6: 3D View editor

Vytvoření animace

Nyní lze přistoupit k vytváření animace. K tomu bude potřeba ještě jeden oddíl, a to "Action Editor". Opět, pokud není již otevřen, stačí vytvořit nový oddíl, nebo nahradit existující, a tentokrát z nabídky vybrat "Dope sheet". Dále z nabídky právě zobrazovaného módu (obr. 7.7 zvolit "Action Editor". Tento editor zobrazuje osu animačních snímků.



Obrázek 7.7: Mód 3D View

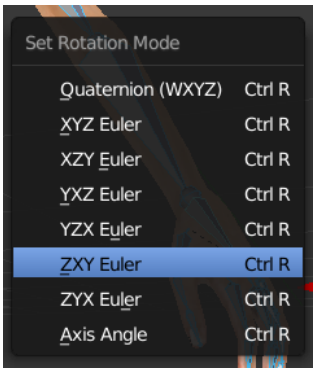
Dále lze zvolit možnost upravovat již existující animaci, popřípadě vytvořit novou. Pro zvolení existující animace je třeba zmáčknout ikonku v levém rohu nabídky na obrázku 7.8 a poté vybrat jednu z vypsaných animací. Pro vytvoření nové kliknout na plus ve stejné nabídce.

Nově vytvořenou animaci lze poté přejmenovat, klikutím na název ve stejné nabídce a zapsáním nového názvu a stisknutím klávesy Enter.

Obrázek 7.8: Action editor

Pro vytvoření vlastního pohybu dlaně, není třeba vytvářet manuálně zvlášť všechny snímky animace. Stačí vytvořit pár klíčových (tzv. keyframe), o zbytek se postará aplikace Blender. Tyto keyframes musí být umístěny mezi snímkem 1 a 200 (včetně), jelikož bude exportována pouze část animace nacházející se v tomto úseku.

Potažením po zobrazené ose se snímky se stisknutým pravým tlačítkem myši lze vybrat, který snímek nyní bude zobrazován a popřípadě manuálně vytvářen. Aktuální pozici na ose značí zobrazovaný zelený kurzor. Po tomto výběru snímku lze přistoupit k vytvoření pozice ruky.



Obrázek 7.9: Nabídka módů aplikace rotací

Pozice kostí dlaně se dá ovlivnit následujícím způsobem:
Nejprve je potřeba přemístit kurzor myši nad "3D View" a poté dvakrát za sebou zmáčknout klávesu A pro vybrání všech kostí a poté klávesovou zkratku Ctrl+R. Ze zobrazené nabídky (viz. obrázek 7.9) zvolit mód v jakém budou rotace aplikovány jako "ZXY Euler".

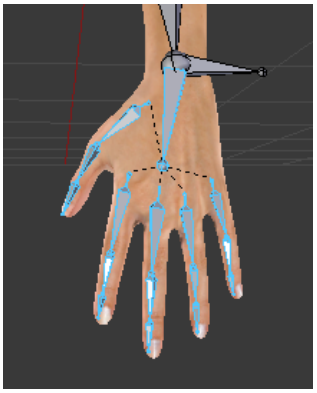
Dále kliknutím levým tlačítkem myši na kost daná kost zmodrá (pokud byly v tuto chvíli vybrány všechny kosti, je třeba kliknout dvakrát za sebou, pro vybrání pouze jedné). To znamená, že byla vybrána pro další manipulaci. Nyní lze upravovat její rotaci v prostoru.

Pravděpodobně nejefektivnější cestou jak rotaci upravovat, je nyní kliknout levým tlačítkem myši kamkoli do volného prostoru okolo objektu a potáhnout. Kost začne rotovat v závislosti na pohybu myši dokud nebude znovu zmáčknuto levé tlačítko myši, nebo klávesa Esc pro zrušení akce.
Během upravování rotací kostí je vhodné vždy jednou za čas otočit kamerou a upravovat z více úhlů, aby pozice ze všech stran vypadala tak, jak má.

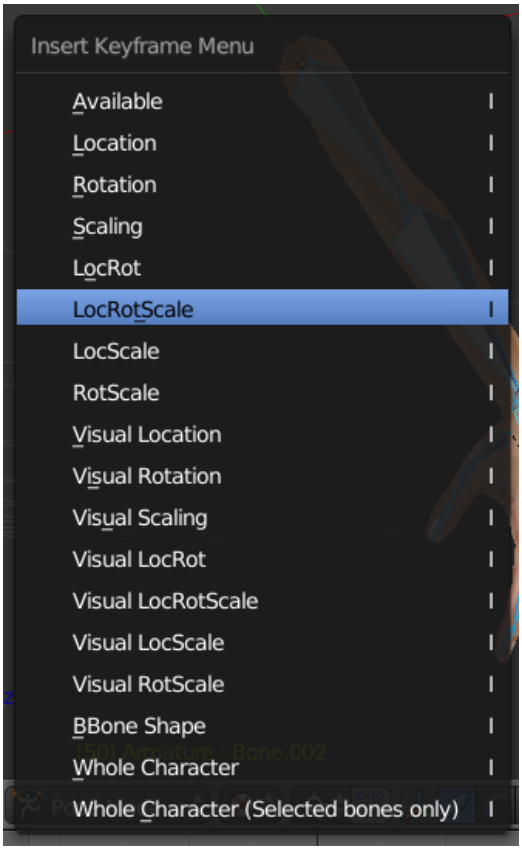
Další užitečnou klávesovou zkratkou je kobinace Alt+R, která resetuje rotaci momentálně vybrané kosti nebo kostí.

Tímto postupem lze upravit rotace kterékoli z kostí v modelu. Pro korektní exportování animace je ovšem třeba upravovat pouze rotace kostí zvýrazněných modře v obrázku 7.10.

Dále je potřeba vytvořenou pozici uložit do animace jako keyframe.
K tomu stačí zmáčknout dvakrát klávesu A, která vybere všechny kosti objektu. Dále stisknout klávesu I, která otevře menu pro vložení keyframe (viz. obrázek 7.11). Z této nabídky vybrat "LocRotScale". Tímto se na ose snímků vytvoří keyframe, uchovávající nyní vytvořenou pozici kostí.



Obrázek 7.10: Kosti, jejichž rotaci lze editovat



Obrázek 7.11: Nabídka možností uložení keyframe

Pro manipulaci s již existujícím keyframe jsou tyto možnosti:
Pro smazání existujícího keyframe jej potřebujeme nejprve vybrat. To učiníme kliknutím na diamant na první (oranžově zvýrazněné) řádce osy snímků. To, že máme vybraný daný keyframe, uvidíme tak, že celý keyframe bude oranžově zvýrazněn. Poté stačí zmáčknout klávesu Delete a vybrat možnost "Delete keyframes" ze zobrazeného menu.

Pokud chceme keyframe pouze přesunout, opět jej vybereme kliknutím na diamant v první řádce, a poté potažením lze s ním pohybovat po ose, dokud není znovu stisknuto levé tlačítko myši, nebo klávesa Esc pro zrušení akce.

Pro úpravu daného keyframe, na něj stačí najet na ose snímků jako při výběru snímku pro vytvoření nového keyframe. Poté upravit kosti a opět snímek uložit stisknutím klávesy I a vybráním z příslušné nabídky.

Pro vytvoření kopie keyframe, je potřeba jej opět vybrat, poté stisknout klávesovou zkratku Ctrl+C, a stisknutím pravého tlačítka myši a potažením nad osou snímků posunout zelený ukazatel aktuálně zobrazovaného snímku na místo, kam chceme keyframe vložit. Nakonec stačí stisknout klávesovou zkratku Ctrl+V. Tímto se vloží kopie keyframe na aktuální pozici zeleného ukazatele.

Pokud chceme přehrát vytvořenou animaci, stačí najet kurzorem nad časovou osu a zmáčknout klávesovou zkratku Alt+A. Pokud chceme přehrávání opět zastavit stačí tuto zkratku stisknout znovu.

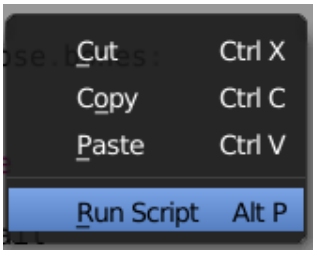
Export animace

Pro export animace je potřeba mít otevřený oddíl s textovým editorem
a v něm otevřený skript "Exportovat.py", který je dodáván spolu .blend souborem.

Pokud textový editor není otevřen, stačí opět vytvořit nový oddíl, nebo se rozhodnout, ve kterém z existujících bude otevřen, a z nabídky editorů vybrat "Text editor". Dále pro otevření požadovaného skriptu, zmáčknout tlačítko "Open" (viz. obrázek 7.12), popřípadě tlačítko se stejnou ikonkou, pokud je otevřený jiný textový dokument, a v prohlížeči vybrat požadovaný skript "Exportovat.py".

Obrázek 7.12: Textový editor

“3D VIew” se musí nacházet v módu Pose Mode a v oddílu, který zobrazuje nabídku animací, je třeba mít aktivní animaci, kterou chceme exportovat. Dále kdekoli nad textovým editorem zmáčknout pravé tlačítko myši a z nabídky (viz. obrázek 7.13) vybrat možnost “Run script”.



Obrázek 7.13: Nabídka nad textovým editorem

Úspěšnost skriptu lze ověřit otevřením konzole aplikace Blender (“Window -> Toggle System Console”). Do této konzole je při úspěšném exportu vypsána hláška “Exportovani animace dokonceno”. Konzoli lze opět zavřít stejným postupem jako byla otevřena. Pozor, stisknutím křížku této konzole se zavře celá aplikace Blender. Ve složce, ve které se daný .blend soubor nachází se vytvoří .csv soubor s názvem exportované animace.

7.2 Příloha B – Programátorská dokumentace

7.2.1 GenerateController

Tato třída slouží k řízení průběhu generování animace paže z naměřených dat. Postará se o spuštění generování animace a po jeho ukončení se postará o zapsání animace do souboru. Dále dává signál k přehrání vytvořeného pohybu.

Za zmínku stojí následující atributy.

- `TrackerController trackers` – skript pro manipulaci s trackery
- `ArmController arm` – skript pro manipulaci s objektem paže
- `AdjustController adjustCont` – skript pro manipulaci s objekty paže a trackerů při vytváření nového snímku animace
- `AnimationController animCont` – třída, která řidí průběh generování animačního snímku
- `bool generate` – určuje, zda je momentálně aktivní generování nové animace
- `bool toReplay` – určuje, zda bylo generování ukončeno a vzniklá animace má být přehrána
- `bool finished` – určuje, zda bylo generování ukončeno

bool StartGeneration()

Metoda, která se postará o vykonání všech akcí, potřebných před spuštěním samotného generování, které posléze odstartuje.

Vytvoří novou instanci třídy `AnimationController`, jejíž metody jsou volány pro obsluhu vlastních akcí potřebných pro vygenerování jednoho animačního snímku. Dále se postará o umožnění zápisu do výstupního souboru.

void Stop()

Zastaví právě probíhající generování.

void Update()

Pokud probíhá generování (atribut `generate` nastaven na true) volá metodu `AnimationController.NextFrame` pro vygenerování dalšího snímku. Pokud bylo celé generování již dokončeno (atribut `finished` je nastaven na true), získá z instance třídy `AnimationController` výslednou animaci a zapíše ji do souboru. Poté nastaví atribut `toReplay` na true signalizujíc, že se má nyní začít přehrávat vytvořený pohyb.

7.2.2 JoinController

Tato třída slouží ke spojení animačaního souboru paže, který vznikl generováním z naměřených dat, s animačním souborem dlaně, který vznikl exportem animace z blenderu. Cílem je vytvoření souboru s finální animací, který bude možné použít v rehabilitační aplikaci.

Za zmínku stojí následující atributy třídy.

- `ArmController arm` – skript pro manipulaci s objektem paže
- `bool joining` – určuje, zda právě probíhá slučování animací
- `string[] handLines` – obsahuje animační data dlaně, převedené do formátu, v jakém jsou ve finálním animačním souboru
- `string[] armFiles` – obsahuje animační data paže
- `string[] result` – obsahuje animační data výsledné finální animace
- `Queue<Position> tails` – obsahuje animační data dlaně načtená z animačního souboru dlaně

bool StartJoin(string armFile, string handFile, string outputFile)

Tato metoda se postará o vykonání všech úkonů, potřebných před spuštěním slučování. Mimo jiné se volá metoda `ReadData` pro načtení vstupních dat. Dále odstartuje samotné slučování nastavením atributu `joining` na true.

bool ReadData(string armFile, string handFile)

Metoda, která se stará o načtení vstupních dat. Parametr `armFile` udává cestu k souboru animace paže. Tento soubor bude načten do atributu `armLines` pomocí metody `DataReader.ReadAllLines`. Parametr `handFile` udává cestu k animačnímu souboru dlaně. Tento soubor bude načten pomocí metody

`DataReader.ReadBlenderAnimation` a načtená data budou dále upravena před zápisem do finálního animačního souboru.

Metoda vrací true pokud načítání proběhlo úspěšně, false pokud ne.

void JoinFiles()

Metoda, která pro každý záznam v atributu `armLines` najde odpovídající sadu záznamů v atributu `handLines`. Tímto vytvoří výslednou animaci, která je uchovávána v atributu `result`.

void WriteToHandLines()

Metoda, která zaznamená momentální lokální úhly kostí dlaně na první volné místo v atributu `handLines`.

void Stop()

Metoda, která slouží k zastavení slučování animačních souborů.

void Update()

Pokud právě probíhá slučování, volá se metoda `ArmController.MoveHand`, která se postará o nastavení hodnot z prvního nezpracovaného animačního snímku uloženého v atributu `tails`. Poté pro uložení hodnot, které bude možné zapsat do finálního animačního souboru, se volá metoda `WriteToHandFiles`.

Pokud již byly zpracovány všechny animační snímky, zavolá se metoda `JoinFiles` pro vykonání vlastního sloučení těchto animací, a metoda `DataWriter.WriteAllToFile` pro zapsání do výstupního souboru. Tímto je slučování ukončeno.

7.2.3 ReplayController

Třída, která obstarává přehrávání dříve vytvořené animace ze souboru. Umí přehrávat všechny tři typy vzniklých souborů: animaci paže vytvořenou z naměřených dat, animaci dlaně vyexportovanou z programu Blender a finální animaci vzniklou spojením animace paže a dlaně.

Za zmínku stojí následující atributy třídy.

- `ArmController arm` – skript pro manipulaci s objektem paže
- `bool replay` – určuje, zda je momentálně aktivní přehrávání

- `Queue<Position> animationData` – fornta s načtenými daty animace
- `int type` – typ přehrávané animace, pro přehrání animace paže má hodnotu 1, animace dlaně hodnotu 2, a pro přehrání finální animace má hodnotu 3

bool StartReplay(string path)

Metoda, která provede všechny potřebné akce před odstartováním přehrávání. Načte vstupní soubor z umístění `path`. Pomocí metody `FileManip.GetAnimationType` zjistí o jaký animační typ se jedná a pomocí příslušné metody ze třídy `DataReader` jej načte. Dále odstartuje vlastní přehrávání nastavením atributu `replay` na true.

void Stop()

Zastaví přehrávání.

bool ReplayHandling()

Z fronty `animationData` obsahující načtená animační data vždy vybere jeden snímek animace a tyto hodnoty nastaví kostem paže. Počet záznamů pro jeden snímek ve frontě se liší podle typu přehrávané animace.

void Update()

Pokud je nyní přehrávána animace, volá metodu `ReplayHanding`, která se postará o zobrazení dalšího animačního snímku.

7.2.4 AdjustController

`AdjustController` slouží k úpravám hodnot kostí paže a výpočtům probíhajícím při generování jednoho animačního snímku. Nachází se zde metody pro nastavení náhodné polohy paže, vytváření gradientu, změny polohy paže, či její vyhodnocení apod.

Za zmínku stojí následující atributy třídy:

- `delta` – určuje kolikanásobek gradientu se bude přičítat k pozici ruky v gradientním sestupu, počátační hodnota nastavena na 5
- `epsilon` – hraniční hodnota atributu `delta`, je nastavena na 0.0001
- `float[] upperAngles` – lokální úhly kosti nadloktí poslední přijaté pozice ruky

- `float[] forAngles` – lokální úhly kosti předloktí poslední přijaté pozice ruky
- `Vector3 pastShoulder` – poloha ramene poslední přijaté pozice ruky
- `float[] newUpper` – lokální úhly kosti nadloktí nové ještě nezhodnocené pozice ruky
- `float[] newFor` – lokální úhly kosti předloktí nové ještě nezhodnocené pozice ruky
- `float[] f` – gradient

void StartAngles(ArmController arm)

Nastaví aktuální reprezentaci rotací kostí nadloktí a předloktí uchovávanou v atributech `upperAngles` a `forAngles` na úhly, které jsou momentálně nastaveny v instanci paže ve třídě `arm`. Používá se před odstartováním gradientního sestupu. Zmíněné atributy dále při generování obsahují hodnoty úhlů poslední akceptované pozice ruky.

void RandFirstPos(ArmController arm, TrackerController trackers)

Vygeneruje náhodnou první pozici paže. Postará se o to, aby z vygenerované pozice byla vybrána ta, která vede k nejlepší pozici po gradientním sestupu. Při náhodném nastavování hodnot stupňů volnosti se mění pouze hodnoty úhlů kostí nadloktí a předloktí.

Generuje se 100 náhodných pozic, ze kterých je vybrána pozice, která je nejblíže snímku dat po provedení gradientního sestupu pomocí funkce `AdjustController.AdjustWholeArm`.

void NewShift(ArmController arm, TrackerController trackers)

Uloží aktuální pozici paže v prostoru. Dále vypočte její hodnotu objektivní funkce. Tyto hodnoty budou využívány při vyhodnocování nově vygenerované pozice. Nakonec zavolá funkci `CalculateGradient` pro vytvoření nového gradientu.

Tato metoda se volá při začátku generace nového animačního snímku, nebo pokud se akceptuje během generace snímku nová pozice paže.

bool TryPosition(ArmController arm, TrackerController trackers)

Pokud je hodnota atributu `delta` menší než `epsilon`, rovnou vrací true. Tato hodnota signalizuje ukončení optimalizace. Jinak vytvoří novou pozici paže tak, že od aktuální pozice paže odečte `delta` násobek gradientu. Poté vrací false.

bool EvaluatePosition(ArmController arm, TrackerController trackers)

Vypočte ohodnocení aktuální pozice paže a vyhodnotí jej. Pokud se jedná o lepší konfiguraci, než byla naposledy přijata, akceptuje ji a vrací true. Pokud ne, nastaví zpět hodnoty poslední akceptované pozice a zmenší atrubut `delta` na polovinu. Poté vrací false.

void ReturnToLastPosition(ArmController arm)

Nastaví hodnoty úhlů kostí nadloktí a předloktí a pozici ramene zpátky na poslední akceptovanou pozici paže. Tyto hodnoty jsou uchovávány v atributech `pastShoulder`, `upperAngles` a `forAngles`.

void NewPerturbation(ArmController arm, TrackerController trackers)

Aktuální pozici paže posune v úhlech kosti nadloktí a předloktí o náhodně vygenerované hodnoty. Tyto hodnoty jsou z intervalu <-15, 15> stupňů.

bool EvaluatePerturbation(ArmController arm, TrackerController trackers)

Vypočte ohodnocení aktuální pozice paže, vzniklé z perturbace, a vyhodnotí ji. Pokud se jedná o lepší konfiguraci, než byla nastavena dosud, akceptuje ji a vrací true. Pokud ne, nastaví zpět hodnoty poslední akceptované pozice. Poté vrací false.

void AdjustWholeArm(ArmController arm, TrackerController trackers)

Celý proces gradientního sestupu v jedné metodě. Do té doby než je `delta` menší než `epsilon` počítá nový gradient, upravuje pozici ruky a novou pozici vyhodnocuje. K tomu využívá metody `NewShift`, `TryPosition` a `EvaluatePosition`.

void CalculateGradient(float f0, ArmController arm, TrackerController trackers)

Vytvoří nový gradient a uloží jej do atributu `f`. Jedná se o vektor o délce 9. Na indexech 0, 1 a 2 se nachází zlepšení při upravení rotací kosti nadloktí na osách x, y a z. Na indexech 3, 4 a 5 se nachází zlepšení při upravení rotací kosti předloktí na osách x, y a z. Nakonec na indexech 6, 7 a 8 se nachází zlepšení při posunutí pozice ramene paže po osách x, y a z.

Všechny tyto hodnoty se měří při úpravě nynějších pozice paže o jeden odpovídající krok. Tato posunutí provedou metody `ArmController.ShiftOneDegree` a `ArmController.ShiftOnAxis`. Poté se vždy vypočítá ohodnocení upravené pozice paže pomocí metody `TrackerControoler.CalculateObjFunction`.

Dále se využívá metoda `MathFunc.CreateGradient`, která z vektoru ohodnocení nových pozic vytvoří gradient.

7.2.5 AnimationController

Třída, která se stará o korektní průběh vlastního generování animačních snímků.

Generování jednoho snímku musí být rozděleno do několika částí. Vytvoření jednoho snímu trvá průměrně 4 vteřiny a pokud by celý proces byl spuštěn v jednom bloku, docházelo by k zamrzání okna aplikace.

Tento problém nelze vyřešit vytvořením nového vlákna, ve kterém budou tyto výpočty probíhat, jelikož se při nich hojně využívá přemisťování objektů ve scéně. Tyto akce Unity nepovoluje provádět z jiného než z hlavního vlákna aplikace.

Atributy této třídy je několik bool proměnných, které určují, v jaké fázi generování snímku se v minulé iteraci skončilo a kterou se má pokračovat. Dalšími atributy jsou:

- `TrackerController trackers` – skript pro manipulaci s objekty trackerů
- `ArmController arm` – skript pro manipulaci s objektem paže
- `AdjustController adjustCont` – skript pro upravování pozice paže v rámci generování animačních snímků
- `string[] result` – uchovává výsledné snímky animace
- `<Position> chestData` – uchovává vstupní data z trackeru na hrudi

- `Queue<Position> forData` – uchovává vstupní data z trackeru na předloktí
- `Queue<Position> upperData` – uchovává vstupní data z trackeru na nadloktí

AnimationController(TrackerController trackers, AarmController arm, AdjustController adjustCont)

Konstruktor. Instancuje objekt paže a trackerů voláním metod `ArmController.InstantiateArm` a `TrackerController.InstantiateArm`. Dále volá metodu `AdjustController.StartAngles`.

bool NextFrame()

Metoda, která na základě bool atributů třídy pokračuje s generováním snímku animace. Rozhodování o dalších prováděných krocích je prováděno podobným způsobem jako v případě konečného automatu. Metoda vrací false, pokud nebyla vygenerována celá animace a true pokud ano.

Jednotlivé kroky, které se provádějí, se dají rozdělit následovně:

1. Nový snímek dat

   Zavolá metody z tříd `TrackerController` a `ArmController` pro načtení nového snímku dat. Dále se bude pokračovat krokem 2.

2. Upravování pozice

   Pokud byla přijata nová pozice, volá se metoda `AdjustController.NewShift` pro vytvoření nového gradientu.

   Dále se volá metoda `AdjustController.TryPosition`, ve které je nastavena nová pozice přičtením gradientu. Dále bude vyhodnocena nově vzniklá pozice metodou `AdjustController.EvaluatePosition`.

   Pokud mají skončit pokusy o vylepšení aktuálně přijaté pozice paže (metoda `TryPosition` vrátí true), bude následovat přechod do dalšího kroku. Pokud byl tento krok vyvolán z kroku 1, pokračuje se krokem 3. Pokud byl vyvolán krokem 3, pokračuje se krokem 4.

3. Nová perturbace

   V případě, že již byl proveden požadovaný počet perturbací (ten je nastaven na 100), je aktuální pozice paže prohlášena za optimální a je uložena jako vygenerovaný snímek voláním metody `Write`. Dále se přechází do kroku 1.

Jinak se vytváří pomocí metody `AdjustController.NewPerturbation` nová perturbace. Dále se přechází na krok 2.

4. Vyhodnocení perturbace

   Proběhne vyhodnocení výsledku perturbace pomocí metody `AdjustController.EvaluatePerturbation`. Pokračuje se krokem 3.

void Write()

Zapíše aktuální polohu do proměnné `result`, která uchovává již vygenerované animační snímky.

void Stop()

Metoda, která je volána po ukončení generování. Zajistí, aby nebyl dále vizualizován objekt paže a objekty trackerů, které byly využívány při generování voláním metod `Stop` odpovídajících skriptů.

bool SetUp()

Metoda, která zavolá metody `ReadData` a `SetUpArmPosition` a nastaví další potřebné hodnoty před startem generování.

void SetUpArmPosition()

Metoda, která se stará o umístění paže, virtuálních trackerů a vedoucích objektů před spuštěním generování. Pro správné vykonání těchto úkonů volá metody `PositionChest`, `MoveVirtual`, `MoveLeading` ze třídy `TrackerController` a metodu `ArmController.PositionArm`.

Dále se postará o zavolání metody `AdjustController.FirstRandPos`, která nastaví počáteční pozici paže vzniklou randomizací.

void ReadData()

Postará se o načtení vstupních dat pomocí metody `DataReader.ReadDataFile`. Načtená data se uchovávají v atributech `chestData`, `forData` a `upperData`.

7.2.6 ArmController

Třída, která se stará o vytvoření objektu paže. Stará se o to, aby byla vytvořena pravá či levá, korektně podle požadavků uživatele. Dále obsahuje metody, které umožňují manipulaci s tímto objektem.

Z atributů třídy budou zmíněny následující.

- `float shiftAxis` – určuje velikost kroku při posunu ruky po ose, tato hodnota je nastavena na 0.001, tedy jeden milimetr
- `float shift` – určuje velikost kroku při úpravách úhlů kostí ruky, tato hodnota je nastavena na 1, tedy jeden stupeň

int InstantiateArm(bool right)

Vytvoří ve scéně objekt paže. Podle parametru right je rozhodnuto zda se bude jednat o pravou nebo levou paži. Pokud je hodnota right true, vytvořena paže pravá, pokud false, levá.

void Stop()

Metoda, která se postará o to, aby dále nebyl ve scéně objekt paže.

void PositionArm(Queue<Position> chestData, TrackerController trackerScript, bool remove)

Umístí objekt paže do prostoru podle první uchovávané pozice ve frontě `chestData`. Tato poloha se vypočte tak, že se poloha hrudníku posune o jednu polovinu délky kosti nadloktí vzhůru, o jednu třetinu délky kosti nadloktí do strany (zda do prava, či do leva záleží na vytvořené ruce) a jednu pětinu délky kosti nadloktí dozadu. Tyto poměry byly stanoveny experimentálně tak, aby co nejvíce odpovídaly pozici lidského ramene vůči středu hrudníku.

void ShiftOnAxis(char axis)

Metoda, která posune polohu objektu paže o jeden krok po ose axis. Velikost tohoto kroku je uchovávána v atributu `shiftAxis`.

void ShiftOneDegree(string boneName, char axis, float[] angles)

Metoda, která změní hodnotu rotace kosti `boneName` v ose `axis` o jeden krok, velikost tohoto kroku je uchovávána v atributu `shift`.

void MoveHand(Queue<Position> tails)

Metoda, která změní rotace kostí dlaně tak, aby reprezentovaly další animační snímek z dat, která jsou předávána v parametru `tails`.

V této frontě jsou uchovávány instance třídy `Position`. Jelikož se jedná o data načtená z animačního souboru vytvořeným exportem z aplikace Blender, jeden snímek dat je reprezentován 21 záznamy. Každý tento záznam obsahuje informace o jedné kosti. V atributu `position` třídy `Position` je uchovávána pozice tail kosti v souřadné soustavě armature. V atributu `rotation` třídy `Position` je uchováván směrový vektor lokální osy z kosti. Ještě je třeba podotknout, že tyto hodnoty jsou v souřadnicovém systému aplikace Blender.

Tato data jsou dále zpracována tak, že se nejprve převede `position` a `rotation` do souřadnicového systému Unity pomocí metody `MathFunc.BlendToUnity`. Dále se `position` převede do globální soustavy souřadnic a `rotation` do soustavy souřadnic kosti, ke které se tato hodnota váže.

Pomocí metody `Transform.LookAt` se nasměruje kost tak, že její lokální osa z ukazuje směrem na bod v prostoru `position`. Dále je zrotována okolo své osy y tak, aby tímto směrem ukazoval tail kosti. Konkrétní hodnota rotace je -90 u levé paže a 90 u paže pravé.

Dále je potřeba zařídit to, aby lokální osa z kosti směřovala správným směrem. Velikost úhlu, o který je třeba kost zrotovat lze vypočíst pomocí vzorce 7.1.

$$rotX = atan2(rotation.y, rotation.z)/PI * 180; \tag{7.1}$$

O hodnotu `rotX` je kost zrotována okolo lokální kladné osy x v případě levé paže a okolo záporné v případě paže pravé metodou `Transform.Rotate`.

7.2.7 CameraController

Třída, která umožňuje ovládání kamery uživatelem za běhu aplikace. Metoda načtítá vstup z klávesnice a zpracuje jej. Nastaví novou odpovídající rotaci a polohu kamery v prostoru.

Lze zmínit následující atributy třídy:

- `float movementSpeed` – rychlost pohybu kamery, nastavena na 2
- `float rotationSpeed` – rychlost rotace kamery, nastavena na 50
- `bool blocked` – určuje, zda je pohyb s kamerou zakázán či ne

void RotationListener()

Naslouchá, zda byl uživatelem zadán požadavek na změnu rotace kamery. Reaguje na stisknutí směrových šipek. Pokud byl zároveň stisknuta klávesa levý Shift, rychlost rotace je dočasně zvýšena na dvojnásobek.

void MovementListener()

Naslouchá, zda byl uživatelem zadán požadavek na změnu polohy kamery. Reaguje na klávesy W (pohyb dopředu), A (pohyb doleva), S (pohyb dozadu), D (pohyb doprava), Q (pohyb vzhůru) a E (pohyb dolů). Pokud byl zároveň stisknuta klávesa levý Shift, rychlost pohybu je dočasně zvýšena na dvojnásobek.

void ResetListener()

Naslouchá, zda byla stisknuta klávesa R. Pokud ano, resetuje polohu a rotaci kamery zpět na počáteční hodnoty.

void ToggleBlock()

Znemožní nebo znovu povolí ovládání pozice a rotace kamery uživatelem. Ovlivňuje hodnotu atributu `blocked`.

void Update()

Pokud není aktuálně vybrán jiný objekt scény, volá metody obsluhy `RotationListener`, `MovementListener` a `ResetListener`.

7.2.8 DialogHandler

Třída, která má na starost vytvoření dialogu pro výběr vsutpního souboru či složky. Samotný zobrazovaný dialog je součástí assetu Runtime File Browser[12], který byl stažen do aplikace z Asset store.

void OpenDialog(int i)

Metoda, která se postará o spuštění korutiny zobrazující dialog voláním metody `ShowLoadDialogCoroutine`. Paremetr `i` uvádí, s jakým účelem byl dialog spuštěn. Podle jeho hodnoty je nastavována hodnota vstupní cesty pro přehrávání, generování nebo slučování souborů. Stejnou hodnotu předává v parametru metodě `ShowLoadDialogCorutine`.

IEnumerator ShowLoadDialogCoroutine(int choice)

Korutina, která se stará o zobrazení dialogu a nastavení správné vstupní cesty. Pokud je parametr `choice` nastaven na 0, vybírá se nová cesta k souboru animace pro přehrání. Pokud na 1, vybírá se soubor animace paže.

Hodnota 2 znamená výběr animace dlaně a hodnota 3 znamená vybírání složky s naměřenými daty o pohybu.

7.2.9 StateLine

Třída, která obstarává správný výpis zpráv do stavové řádky aplikace a nastavuje zobrazení obrazovky procentuelního postupu při generování a slučování animací.

Za zmínku stojí následující atributy:

- `ReplayController replay` – skript řídící přehrávání animace paže
- `GenerateController generate` – skript řídící generování animace paže
- `JoinController join` – skript řídící slučování animací paže a dlaně
- `CameraController cameraCont` – skritp řídící pohyb kamery

void Update()

Kontroluje, zda neskončila probíhající akce generování, slučování nebo přehrávání animací. Pokud ano, reaguje na to výpisem do stavové řádky a případným ukončení zobrazení obrazovky procentuelního postupu.

Dále se stará o vypsání správného procenta postupu voláním metody `SetPercentage`, pokud je zobrazena obrazovka procentuelního postupu.

void Stop()

Odblokuje pohyb a rotaci kamery voláním metody `CameraController.ToggleBlock` a zastaví zobrazování obrazovky s procentuelním postupem.

void SetText(string text)

Zapíše zprávu `text` do stavové řádky programu.

void SetPercentage(double val)

Změní zobrazované procento v obrazovce procentuelního postupu na hodnotu `val`.

void DisplayPercentageScreen(int loadedLenght)

Zobrazí obrazovku procentuelního postupu probíhající akce a zablokuje pohyb kamery pomocí metody `CameraController.ToggleBlock`. Parametr `loadedLength` představuje počet řádek ke zpracování, tato hodnota bude použita pro výpočet aktuálního procenta, které bude zobrazeno uživateli.

7.2.10 TrackerController

Třída, která se stará o zobrazování a manipulaci s objekty, reprezentujícími trackery. V aplikaci existují dva druhy těchto trackerů: virtuální trackery a vedoucí objekty.

Za zmínku stojí tyto atributy:

- `Vector3 translate` – hodnota, o kterou byl naposled posunut vedoucí objekt reprezentující hrudník
- `float rotY` – hodnota, o kterou byl naposled zrotován vedoucí objekt reprezentující hrudník

void InstantiateTrackers()

Postará se o zobrazení všech trackerů. Všechny budou umístěny do bodu [0,0,0].

float CalculateObjFunction()

Zavolá metodu `MathFunc.CalculateObjFunc` pro výpočet hodnoty objektivní funkce nad nynější konfigurací trackerů. Vrací hodnotu objektivní funkce.

void MoveLeading(Queue<Position> forData, Queue<Position> upperData, bool remove = true)

Posune vedoucí objekty předloktí a nadloktí tak, aby reprezentovaly další snímek naměřených dat. Tento snímek je získán jako první hodnoty ve frontách `forData` (data trackeru předloktí) a `upperData` (data trackeru nadloktí). Dále jsou tyto hodnoty upraveny podle postupu popsaném v kapitole 3.1. Hodnoty, o které budou data upravována, jsou uchovávány v atributech `rotY` a `translate`.

Pokud je parametr `remove` nastaven na true, hodnoty jsou z fronty odstraněny.

void PositionChest(Queue<Position> chestData)

Umístí vedoucí objekt hrudníku tak, aby reprezentoval další smínek naměřených dat. Tento snímek je získán jako první hodnota z fronty `chestData`. Ta je dále je upravena podle postupu popsaném v kapitole 3.1.

Vybraná hodnota není z fronty odstraňována.

void MoveVirtual(ArmController armScript)

Metoda, která posune trackery tak, aby byly korektně umístěny na objektu virtuální ruky.

Konkrétní umístění trackerů je popsáno v kapitole 3.2.

void Stop()

Postará se o ukončení zobrazování virtuálních trackerů a vedoucích objektů.

7.2.11 UIHandler

Třída, která má na starost ovládání UI. Její metody se starají o správné obsloužení požadavku uživatele.

Z atributů stojí za zmínku:

- `StateLine stateLine` – skript ovládající stavovou řádku aplikace
- `JoinController joinController` – skript řídící slučování animací paže a dlaně
- `ReplayController replayController` – skript řídící přehrávání animace paže
- `GenerateController generateController` – skript řídící generování animacepaže

Start()

Nastaví velikost okna aplikace na 2/3 šířky a 2/3 výšky monitoru. Dále nastaví výstupní a defaultní vstupní složku.

void Update()

Naslouchá, zda byla uživatelem stisknuta kombinace kláves Ctrl+Q. Pokud ano, ukončí právě probíhající akci.

Dále kontroluje, zda bylo ukončeno generování nové animace z načtených dat. Pokud ano, spustí její přehrání.

void Replay()

Metoda, která se postará o spuštění přehrávání animace. Načte cestu ke vstupnímu souboru a zavolá metodu `StateLine.SetText` pro vypsání příslušné hlášky do stavové řádky aplikace.

Dále zavolá metodu `ReplayController.StartReplay`, která se postará o vlastní odstartování přehrávání. Obsluha přehrávání se dále odehrává ve třídě `ReplayController`.

void Join()

Metoda, která se postará o spuštění spojování animace paže a animace dlaně. Načte cesty ke vstupním souborům, vytvoří název výstupního souboru pomocí metody `DataWriter.CreateFinalFile` a zavolá metodu `StateLine.SetText` pro vypsání příslušné hlášky do stavové řádky aplikace.

Dále zavolá metodu `JoinController.StartJoin`, která odstartuje vlastní spojování animací. Obsluha spojování se dále odehrává ve třídě `JoinController`.

void Generate()

Metoda, která se postará o spuštění generování nové animace paže. Načte cestu ke vstupní složce, a vytvoří název výstupního souboru pomocí metody `DataWriter.CreateAnimationFile`. Volá metodu `StateLine.DisplayPercScreen` pro zobrazení okna procentuálního průběhu generování.

Dále zavolá metodu `GenerateController.StartGeneration`, která odstartuje vlastní generování animace paže. Obsluha generování se dále odehrává ve třídě `GenerateController`.

void ToggleRight()

Metoda, která nastaví, zda byla uživatelem vybrána levá či pravá paže. Tato hodnota dále ovlivňuje na jakém objektu bude animace generována či přehrávána.

7.2.12 Config

Třída, která obsahuje aktuální konfiguraci aplikace. Nachází se v ní atributy, které uchovávají cesty do aktuálně zvolených složek vstupních souborů a cestu do výstupní složky. Je zde uchováváno, zda uživatel chce pracovat s levou nebo pravou paží. Dále jsou v této třídě zapsány konstanty, mimo jiné

se jedná o ty, které definují název souborů, ve kterých se hledají naměřená data a jejich formát.

7.2.13 DataReader

Třída, která poskytuje metody pro načítání dat ze souborů.

static Queue<Position> ReadDataFile(string path)

Načte soubor obsahující data naměřená při předcvičování pohybu. Tato data se dále využívají pro generování animace paže. Vrací frontu instancí třídy `Position`. Tyto instance reprezentují polohu jednoho trackeru v globální soustavě souřadnic. Jeden záznam reprezentuje polohu v jednom animačním snímku.

static Queue<Position> ReadFinalAnimation(string path)

Metoda, která načte soubor obsahující data finální animace. Tato data jsou používána při přehrávání. Vrací frontu instancí třídy `Position`. Jeden animační snímek obsahuje informace o rotacích všech kostí objektu paže a skládá se z 23 záznamů. Informace o kostech nadloktí a předloktí jsou v globální soustavě souřadnic a informace o kostech dlaně v lokální.

static Queue<Position> ReadArmAnimation(string path)

Metoda, která načte soubor obsahující data animace paže. Tato data jsou používána při přehrávání. Jsou vracena jako fronta instancí třídy `Position`. Jeden animační snímek obsahuje informace o rotacích kostí předloktí a nadloktí objektu paže a se skládá ze 2 záznamů. Informace o rotacích jsou v globální soustavě souřadnic.

static Queue<Position> ReadBlenderAnimation(string path)

Metoda, která načte soubor obsahující data animace dlaně. Tato data jsou používána při přehrávání. Jsou vracena jako fronta instancí třídy `Position`. Jeden animační snímek obsahuje informace o rotacích kostí dlaně objektu paže a skládá se ze 21 záznamů. Informace o rotacích jsou v soustavě souřadnic armature.

static string[] ReadAllLines(string path)

Metoda, která do pole načte všechny řádky ze souboru na cestě path. Toto pole posléze vrací.

7.2.14 DataWriter

Třída, která se stará o vytváření názvů výstupních souborů a o zapisování do nich.

static string CreateAnimationFile(string path)

Metoda, která vrací vytvořené jméno výstupního souboru animace paže. Parametr `path` je cesta do složky se vstupními daty. Jméno výstupního souboru bude stejné jako název této složky.

static string CreateFinalFile(string armAnimationFile, string handAnimationFile)

Metoda, která vrací vytvořené jméno výstupního finálního animačního souboru. Parametry je cesta k souboru animace paže (`armAnimationFile`) a cesta k souboru animace dlaně (`handAnimationFile`). Výsledným názvem bude spojení názvů těchto souborů.

static void WriteAllToFile(string[] lines, string path)

Zapíše všechny řetězce z pole `lines` do souboru na cestě `path`.

7.2.15 FileManip

Třída, která umožňuje manipulaci se soubory. Poskytuje metody pro ověřování správných formátů souborů a další manipulaci s nimi.

static bool IsArmFile(string[] lines)

Metoda, která ověřuje, zda soubor, který obsahuje řádky `lines`, vyhovuje formátu animačního souboru paže.

Ověřuje se, zda všechny řádky mají požadovaný počet sloupců.

static bool IsHandFile(string[] lines)

Metoda, která ověřuje, zda soubor, který obsahuje řádky `lines`, vyhovuje formátu animačního souboru dlaně.

Ověřuje se, zda soubor má požadovanou hlavičku, zda všechny řádky mají požadovaný počet sloupců a zda je počet řádků dělitelný počtem kostí dlaně.

static bool IsFinalFile(string[] lines)

Metoda, která ověřuje, zda soubor, který obsahuje řádky `lines`, vyhovuje formátu finálního animačního souboru.

Ověřuje se, zda je počet řádků dělitelný počtem kostí dlaně plus jedna.

static int GetAnimationType(string path)

Metoda, která pomocí metod `IsHandFile`, `IsFinalFile` a `IsArmFile` zjišťuje, jakým typem souboru je soubor na cestě `path`. Návratová hodnota je 1 pokud se jedná o soubor animace paže, 2 pro soubor animace dlaně a 3 pro finální animační soubor. Pokud vrací -1, jedná se o soubor v nevyhovujícím formátu.

static bool Exists(string path)

Metoda, která ověřuje, zda soubor na cestě `path` existuje a zda se jedná o .csv soubor.

static string GetName(string path, bool ext = false)

Metoda, která vrací název souboru na cestě `path`. Pokud je parametr `ext` nastaven na true, název je vrácen i s příponou.

static string[] RemoveFile(string path)

Metoda, která odstraní soubor na cestě `path`.

7.2.16 MathFunc

V této třídě se nacházejí metody provádějící matematické výpočty. Mimo jiné se zde nachází metoda pro výpočet objektivní funkce.

static float Adjust(float val)

Vrací upravenou hodnotu úhlu `val` tak, aby se pohyboval v rozsahu <-180,180>.

static Vector3 BlendToUnity(Vector3 vect)

Převede vektor kosti `vect` ze soustav souřadnic Blenderu do Unity. Prakticky to znamená, že vytvoří nový vektor [-vect.x, vect.y, vect.z]. Tento převod byl stanoven experimentálně.

static float Distance(Transform t1, Transform t2)

Vypočte druhou mocninu vzdálenosti transformací `t1` a `t2`.

static double AngleDiff(Transform t1, Transform t2, char axis)

Vypočte úhel, o který se odchyluje transformace `t1` od `t2` v ose `axis`.

Využívá vzorec 7.2 pro výpočet úhlu mezi vektory v prostoru, kde $\vec{u}$ a $\vec{v}$ jsou vektory.

$$cos\alpha = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|} \tag{7.2}$$

static float ObjFunc(Transform armVirtual, Transform armLeading, Transform forearmVirtual, Transform forearmLeading)

Vypočte hodnotu objektivní funkce pro aktuální konfiguraci trackerů. Vzorec pro výpočet se nachází v kapitole 3.3. Parametry `a`, `b`, `c` a `d` jsou zvoleny jako hodnota 0.1. Tyto hodnoty byly zvoleny experimentálně.

static float[] CreateGradient(float f0, float[] f)

Ode všech komponent odečte hodnotu `f0`, která představuje hodnotu objektivní funkce pro poslední přijmutou pozici, a komponenty reprezentující posun polohy objektu paže (tedy index 6, 7 a 8) vynásobí 1000.

7.2.17 Position

Tato třída je přepravkou pro určité umístění objektu v prostoru. Má dva atributy `Vector3 position`, který popisuje polohu v soustavě souřadnic v pořadí os x, y a z, a `Vector3 rotation`, který obsahuje eulerovské úhly rotací okolo jednotlivých os v pořadí x, y a z.

7.3 Příloha C – Obsah CD

Na přiloženém CD se nacházejí následující složky a jejich obsah:

1. animation

   Složka obsahující vygenerovaná animační data. Nachází se zde složka hand s .csv soubory s animacemi dlaně, a složka arm se složkami s názvy pohybů, které obsahují odpovídající animace paže a finální animace.

2. arm_models

   Modely paže, které byly využívány v rehabilitační aplikaci.

3. build

   V této složce se nachází build aplikace v Unity.

4. data

   Složka, ve které se nacházejí vstupní data pro generování animací paže. Data jsou opět roztříděna do složek podle názvu pohybu, který obsahují. Od každého pohybu jsou zaznamenané 3 verze.

5. extraction

   Model paže, na kterém jsou připravené animace dlaně k vygenerovaným pohybům a skript “Exportovat.py”.

6. src

   Složka, ve které se nachází Unity projekt aplikace pro generování animací.

7. video

   Složka, ve které se nacházejí videa předcvičovaných pohybů a videozáznamy vzniklých finálních animací.

Literatura

[1] *Math.Atan2(Double, Double) Method* [online]. Microsoft, 2019. [cit. 11/04/20]. .NET Dokumentace. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.math.atan2?view=netframework-4.8.

[2] Baran, I. – Popovic, J. Automatic rigging and animation of 3D characters. *ACM Trans. Graph.* 2007, 26, 3, s. 72. doi: 10.1145/1276377.1276467. Dostupné z: https://doi.org/10.1145/1276377.1276467.

[3] *Limit Distance Constraint* [online]. Blender, 2020. [cit. 10/04/20]. Blender Dokumentace. Dostupné z: https://docs.blender.org/manual/en/latest/animation/constraints/transform/limit_distance.html.

[4] *IK Solver Contstraint* [online]. Blender, 2020. [cit. 10/04/20]. Blender Dokumentace. Dostupné z: https://docs.blender.org/manual/en/latest/animation/constraints/tracking/ik_solver.html.

[5] *Keyframes* [online]. Blender, 2020. [cit. 02/04/20]. Blender Dokumentace. Dostupné z: https://docs.blender.org/manual/en/latest/animation/keyframes/introduction.html.

[6] Flores, P. *Concepts and Formulations for Spatial Multibody Dynamics.* Springer, 2015. ISBN 978-3-319-16190-7.

[7] *Game engines — how do they work?* [online]. Unity, 2020. [cit. 11/04/20]. Dostupné z: https://unity3d.com/what-is-a-game-engine.

[8] Hocking, J. *Unity in Action: Multiplatform Game Development in C# (2nd Ed.).* Manning Publications, 2018. ISBN 9781617294969.

[9] *Vive tracker* [online]. HTC Corporation, 2018. [cit. 31/02/20]. Introducing Vive Tracker for Developers. Dostupné z: https://developer.vive.com/eu/vive-tracker-for-developer/.

[10] *HTC VIVE Tracker (2018) Developer Guidelines Ver. 1.0* [online]. HTC, 2018. [cit. 02/04/20]. Vive Tracker Dokumentace. Dostupné z: https://dl.vive.com/Tracker/Guideline/HTC_Vive_Tracker(2018)_Developer+Guidelines_v1.0.pdf.

[11] Kavan, L. et al. Skinning with dual quaternions. In Gooch, B. – Sloan, P. J. (Ed.) *Proceedings of the 2007 Symposium on Interactive 3D Graphics,*

SI3D 2007, April 30 - May 2, 2007, Seattle, Washington, USA, s. 39–46. ACM, 2007. doi: 10.1145/1230100.1230107. Dostupné z: https://doi.org/10.1145/1230100.1230107.

[12] Kula, S. Y. *Runtime File Browser* [online]. 2020. [cit. 24/04/20]. Unity Asset. Dostupné z: https://assetstore.unity.com/packages/tools/gui/runtime-file-browser-113006.

[13] Magnenat-Thalmann, N. – Thalmann, D. *Computer Animation: Theory and Practice (2nd Rev. Ed.)*. Springer, 1990. ISBN 978-4-431-68105-2.

[14] Pandey, P. *Understanding the Mathematics behind Gradient Descent* [online]. Medium, 2019. [cit. 02/05/20]. Towards Data Science. Dostupné z: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e.

[15] Pokorný, P. *Blender Naučte se 3D Grafiku*. BEN - technická literatura, 2006. ISBN 80-7300-203-5.

[16] richardsee. *Female Arm* [online]. 2013. [cit. 05/04/20]. 3D model. Dostupné z: https://www.turbosquid.com/FullPreview/Index.cfm/ID/719753.

[17] Smutný, V. *Robotika - Popis Polohy Tělesa* [online]. 2019. [cit. 11/04/20]. Dostupné z: http://cmp.felk.cvut.cz/cmp/courses/ROB/roblec/geometry-notecz.pdf.

[18] Vaillant, R. *Blender tutorial to animate a cylinder* [online]. 2015. [cit. 14/04/20]. Dostupné z: http://rodolphe-vaillant.fr/?e=65.

[19] Vaillant, R. *Dual Quaternions skinning tutorial and C++ codes* [online]. 2013. [cit. 14/04/20]. Dostupné z: http://rodolphe-vaillant.fr/?e=29.

[20] Wareham, R. – Lasenby, J. Bone Glow: An Improved Method for the Assignment of Weights for Mesh Deformation. In López, F. J. P. – Fisher, R. B. (Ed.) *Articulated Motion and Deformable Objects, 5th International Conference, AMDO 2008, Port d'Andratx, Mallorca, Spain, July 9-11, 2008, Proceedings*, 5098 / *Lecture Notes in Computer Science*, s. 63–71. Springer, 2008. doi: 10.1007/978-3-540-70517-8_7. Dostupné z: https://doi.org/10.1007/978-3-540-70517-8_7.

[21] Řasová, K. *Fyzioterapie u Neurologicky Nemocných*. CEROS, 2007. ISBN 978-80-239-9300-4.

[22] Řasová, K. – kolektiv. *Možnosti Fyzioterapie v Léčbě Roztroušené Sklerózy*. Garmedis, 2017. ISBN 978-80-906747-0-7.