UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Upravovanie záznamu ľudského pohybu a sekvencií snímaných 3D kamerou

Bakalárska práca

2022 **Samuel Piteľ**

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Upravovanie záznamu ľudského pohybu a sekvencií snímaných 3D kamerou

Bakalárska práca

Študijný program: Aplikovaná informatika

Študijný odbor: Informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: RNDr. Martin Madaras, PhD.

Konzultant: Mgr. Dana Škorvánková

2022 **Samuel Piteľ**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Samuel Piteľ
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Postprocessing of motion capture and 3D camera sequences
Upravovanie záznamu ľudského pohybu a sekvencií snímaných 3D kamerou

Anotácia: Ľudský pohyb vieme zaznamenať pomocou technológii snímania pohybu, buď optickým spôsobom alebo pomocou inerčných oblekov. Táto téma sa zaoberá jednak optickým snímaním pomocou 3D kamier, a tiež inerčným snímaním pomocou oblekov alebo rukavíc. Naším cieľom je zachytiť dataset zložený z pohybových dát zachytených inerčným systémom a nahrať synchronizovanú sekvenciu s použitím 3D kamery. Rozhranie pre editovanie a úpravu nahraných datasetov nám umožní robiť prepojené operácie s kostrou a mračnom bodov, využiteľné pre úlohy analýzy ľudského tela. Editovací systém bude evaluovaný a testovaný na nahratých datasetoch, kde modifikované dáta môžu byť z rozhrania exportované.

Cieľ:
- Naštudovať relevantné články k nahrávaniu datasetov a technológií 3D kamier
- Vytvoriť dataset zložený zo synchronizovaných dát pohybu človeka a sekvencie snímkov z 3D kamery
- Naštudovať systémy ako Unity, Unreal Engine a pridávanie C++ modulov
- Navrhnúť a implementovať aplikáciu na úpravu nahraných datasetov, napr. škálovanie kostry, modifikácia segmentácie, alebo filtrovanie pozadia
- Evaluácia metódy, a písanie práce

Literatúra:
MoVi: A large multi-purpose human motion and video dataset, Ghorbani et al. 2021, https://doi.org/10.1371/journal.pone.0253157

Motion Capture Research: 3D Human Pose Recovery Based on RGB Video Sequences, Min 2019 et al., https://doi.org/10.3390/app9173613

Survey on Style in 3D Human Body Motion: Taxonomy, Data, Recognition and its Applications, Ribet et al. 2019, https://hal.archives-ouvertes.fr/hal-02420912/document

Berkeley MHAD: A Comprehensive Multimodal Human Action Database, Ofli et al. 2013, https://openreview.net/pdf?id=TL6wabAZ9

Kľúčové slová: Mocap, 3D kamery, úprava dát

Vedúci: RNDr. Martin Madaras, PhD.
Konzultant: Mgr. Dana Škorvánková

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 26.10.2021

Dátum schválenia: 27.10.2021

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné vyhlásenie

Týmto čestne prehlasujem, že som túto prácu vypracoval sám. Všetky informácie, ktoré som v práci využil sú označené a referencované v bibliografii.

.....

Poďakovanie

Týmto by som rád poďakoval môjmu školiteľovi RNDr. Martinovi Madarasovi, PhD., za jeho pomoc a cenné rady počas vytvárania aplikácie. Taktiež ďakujem Michalovi Šabíkovi za jeho ochotu pomôcť mi nahrať dáta nevyhnutné pre realizáciu mojej práce. V neposlednom rade som vďačný mojej rodine za podporu.

Abstrakt

Úlohou tejto bakalárskej práce bolo zaznamenať ľudský pohyb pomocou inerčného mocap obleku, ktorý zaznamenáva pohyby tela nositeľa a súčasne pomocou optickej 3D kamery, pred ktorou sa pohyb vykonáva. Pre zrealizovanie tejto úlohy sme zvolili špeciálny oblek Rokoko smartsuit pro, z ktorého sme získali transformácie kostry, ktoré tvoria digitálnu animáciu nášho pohybu. Na zachytenie pohybu z 3D kamery sme využili Microsoft Kinect v2, pomocou ktorého sme pohyb zachytili ako mračná bodov ľudského tela v rôznych časových úsekoch. Ďalším cieľom tejto práce je navrhnúť a implementovať aplikáciu na úpravu získaných datasetov. Pre splnenie tohto cieľa sme zvolili Unreal Engine, v ktorom je aplikácia vytvorená. Aplikácia obsahuje množstvo funkcionalít na úpravu mračna bodov a kostry. Taktiež umožňuje používateľovi odstrániť nežiaduce efekty ktoré vznikajú nahraním dát alebo vložením dát do aplikácie. Navyše, aplikácia obsahuje funkcionality ktoré využívajú algoritmus RANSAC, ktorý vie oddeliť a upraviť nežiaduce celky. V práci opisujem jej východiská, návrh riešenia problémov a implementáciu aplikácie pomocou blueprintov, s ktorými Unreal Engine pracuje, ako aj s programovacím jazykom C++.

Kľúčové slová: 3D kamera, mocap oblek, animácia, snímanie pohybu, úprava pohybu, kostra, mračná bodov

Abstract

The goal of this thesis is to record human movement using an inertial motion capture suit that records movements of wearer's body while simultaneously using a 3D camera in front of which the movement is performed. To carry out this task we use a special Rokoko Smartsuit Pro from which we obtain skeletal transformations that form a digital animation of our movement. To capture movement from a 3D camera we use Microsoft Kinect v2 to capture the movement as a point cloud of human body in different time frames. Another goal of this work is to design and implement an application to modify the obtained datasets. To carry out this task we use the Unreal Engine software in which the application is created. The application contains many functionalities for modifying point clouds and skeletons. It also allows the user to eliminate unwanted effects of capturing the data or importing data to application. In addition, the application contains functions that use the RANSAC algorithm, which can separate and modify unwanted units. In this work we describe the resources used, solutions of problems encountered and implementation of application using blueprints and the C++ programming language.

Keywords: 3D camera, motion capture suit, animation, motion capture, motion editing, skeleton, point cloud

Obsah

Úvod ... **1**
1. Východiská ... **2**
1.1 Motion capture ... 2
1.1.1 Performance animation ... 3
1.1.2 Súborový formát FBX ... 3
1.1.3 Kostra ... 3
1.1.3.1 KĺBy ... 5
1.1.3.2 Kinematika ... 6
1.1.3.3 Upravovanie kostry ... 6
1.1.4 Optické motion capture systémy ... 7
1.1.5 Hybridné systémy ... 8
1.1.6 Inerčné obleky ... 8
1.1.6.1 IMU senzor ... 10
1.2 3D Model ... 11
1.2.1 Mesh ... 11
1.2.2 3D skener ... 11
1.2.2.1 3D skenovanie ... 12
1.2.2.2 Kinect SDK ... 14
1.2.3 Mračno bodov (Point Cloud) ... 14
1.2.3.1 XYZ súborový formát ... 15
1.2.4 Point Cloud Library (PCL) ... 15
1.2.4.1 RANSAC ... 16
1.2.4.2 PCD súborový formát ... 16
1.3 Unreal Engine 4.24.3 ... 17
1.3.1 Blueprintový vizuálny skriptovací systém ... 18
1.3.2 C++ vs blueprint ... 18
1.3.3 Unreal obchod ... 20
1.3.3.1 Plugin ... 20
2. Návrh ... **22**
2.1 Použité technológie ... 22
2.1.1 Technológie skenovania 3D kamerou ... 22
2.1.1.1 Softvér ... 23
2.1.1.2 Hardvér ... 23

2.1.2 Technológie zaznamenania pohybu mocap oblekom ... 24
2.1.2.1 Softvér ... 24
2.1.2.2 Hardvér ... 25
2.2 Vytvorenie dát pohybu človeka a sekvencie snímok z 3D kamery ... 25
2.3 Synchronizovanie nahraných dát ... 26
2.4 Upravovanie nahraných dát ... 26
2.4.1 Upravovanie animácie (kostry) ... 26
2.4.2 Upravovanie mračna bodov ... 27
2.4.2.1 Upravovanie mračna bodov knižnicami ... 27
3. Implementácia ... 31
3.1 Scéna aplikácie ... 31
3.2 Datasety ... 31
3.2.1 3D skeny ... 32
3.2.2 Transformácie kostry ... 34
3.3 Používateľské rozhranie ... 34
3.3.1 Práca so súbormi (File) ... 35
3.3.1.1 Tlačidlá na prácu so súbormi ... 36
3.3.2 Upravovanie kostry (Edit skeleton) ... 38
3.3.2.1 Tlačidlá na úpravu kostry ... 38
3.3.2.2 Checkboxy na úpravu kostry ... 39
3.3.3 Upravovanie mračna bodov (Edit point cloud) ... 40
3.3.3.1 Tlačidlá na úpravu mračien bodov ... 41
3.3.4 Algoritmy PCL knižnice ... 41
3.3.4.1 Passthrough filter ... 42
3.3.4.2 Background filter ... 43
3.3.4.3 Plane segmentation ... 43
4. Testovanie ... 45
Záver ... 46
Použitá literatúra ... 47

Úvod

Existuje veľa metód zachytenia pohybu ľudského tela: mechanicky, opticky, magneticky a inerčne.

K výberu tejto témy ma viedla možnosť práce s inerčným oblekom a prepojenie dát získaných z dvoch rôznych hardvérových technológii vo vlastnej aplikácii, kde je pohyb ľudského tela súčasne zachytávaný pomocou optického a inerčného systému.

Technika zachytenia ľudského tela je efektívnejšia a v modernom animovaní nahradila tradičné počítačové animovanie. Má viacero využití vo vedeckých výskumoch alebo v modernej medicíne. Pohyb v našom riešení zachytávame inerčne pomocou mocap obleku. Pohyb je následne vizualizovaný na virtuálnej kostre, čo vytvára digitálnu animáciu nášho pohybu. Digitálnu animáciu následne vložíme do našej aplikácie. Počas vykonávania pohybu, nás taktiež bude snímať 3D skener, pomocou ktorého vytvoríme mračno bodov, ktoré bude reprezentovať 3D model nášho tela.

Nahrané dáta následne synchronizujeme v nami vytvorenej aplikácii, ktorá umožní zobrazenie jednotlivých dát, s možnosťou ich úpravy v 3D priestore. Aplikácia pritom umožňuje voľný pohyb po scéne, pre uľahčenie označenia podmnožín dát, ktoré chceme upravovať. Mračna bodov vieme do scény načítať, alebo uložiť do súboru. Aplikácia taktiež podporuje upravovanie mračien pomocou algoritmov segmentácie alebo odstránenia pozadia, kde môžeme nastaviť parametre algoritmov.

Práca je rozdelená do štyroch kapitol. V prvej kapitole opisujeme teoretické východiská a objasňujeme základné pojmy, s ktorými sa počas riešenia problematiky stretneme. Nasledujúca kapitola obsahuje návrh a princípy riešenia stanovených problémov zo zadania. Tretia kapitola popisuje konkrétnu implementáciu aplikácie pomocou blueprintov a C++. V kapitole sú rozpísané jednotlivé widgety používateľského rozhrania a spôsob, akým modifikujú nahrané dáta. V poslednej kapitole sa nachádzajú informácie z testovania aplikácie a jej slabé stránky.

1 Východiská

V tejto kapitole sa budeme venovať základným teoretickým východiskám, potrebným pre pochopenie a vyriešenie danej problematiky. Na úvod začneme s vysvetlením relevantných pojmov, ako napríklad 3D skenovanie alebo motion capture. Neskôr si popíšeme, ako dokážu hĺbkové kamery zachytiť 3D dáta, v akých dátových formátoch sa vyskytujú, aký softvér a jeho funkcie budeme používať na vizualizáciu a úpravu nahratých údajov.

1.1 Motion capture

Motion capture je technológia, ktorá umožňuje prevod pohybu v reálnom čase na digitálnu reprezentáciu pohybu do aplikácie, ktorá ho spracuje. Cieľom snímania pohybu je zaznamenať pohyb herca (zvyčajne, ale nie vždy, človeka) kompaktným a použiteľným spôsobom [18]. Pri snímaní pohybu sa zachytáva pozícia snímaného objektu veľa krát za sekundu, čo sa označuje pojmom FPS (frames per second), čo znamená snímky za sekundu. Rôzne motion capture systémy majú možnosť nahrávania v rôznych FPS rozmedziach.

Existuje veľa spôsobov ako nahrať pohyb. Ľudské motion capture systémy sa delia na:

- **outside-in**: systém využívajúci vonkajšie senzory umiestnené na tele postavy, snímané vizuálne (Optické motion capture systémy)
- **inside-out**: systém so senzormi na tele, ktorý sníma vonkajšie vnemy (elektromagnetické pole)
- **insinde-inside**: systém využívajúci vnútorné spojenia postavy (inerčné, elektromagnetické obleky)

Pohyby kamery môžu byť tiež zachytené tak, že virtuálna kamera sa v scéne bude nakláňať alebo otáčať okolo javiska, na ktorom herec hrá. Systém snímania pohybu vie zachytiť scénu aj s kamerou, čo umožňuje počítačovým 3D modelom rovnakú perspektívu ako záznam z kamery. Počítač spracuje údaje a zobrazuje pohyby herca, pričom poskytuje požadované polohy kamery z hľadiska objektov v súprave. Keď snímanie pohybu zahŕňa tvár a prsty alebo výraz tváre, označuje sa ako performance animation. V mnohých oblastiach sa

snímanie pohybu nazýva motion tracking, ale vo filmovej tvorbe a hrách sa skôr nazýva match moving [2].

1.1.1 Performance animation

Je namapovanie nahraného pohybu na digitálnu postavu. Performance animation sa využíva pri natáčaní filmov alebo pri namapovaní pohybu na 3D model v hernom priemysle, kde nie je snímaný len pohyb tela, ale napríklad aj rúk či tváre [15].

Nastavenie digitálnej postavy je rozdelené do dvoch krokov:

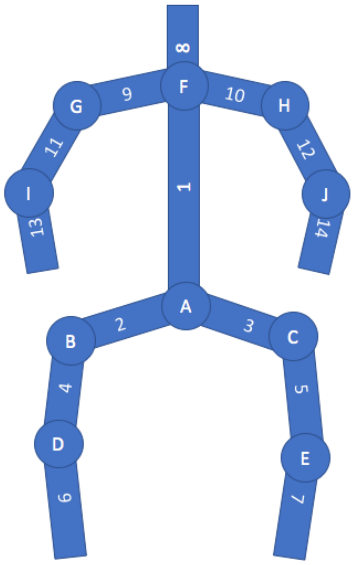
- **mechanické nastavenie** - nastavenie zahŕňa vytvorenie kostry, ktorá bude „riadiť“ postavu, ako aj slúžiť na vytvorenie rôznych typov pohybov, ktoré budú animovať postavu
- **nastavenie deformácie** - nastavenie definuje parametre každého bodu postavy na kostre

1.1.2 Súborový formát FBX

Nahrané animácie budeme exportovať vo forme fbx so snímkovou frekvenciou 100 FPS. Fbx formát je navrhnutý na opísanie animačných scén a je podporovaný väčšinou 3D animačných softvérov. Súbor obsahuje topologické vzťahy medzi segmentami v hierarchii, ktoré sú uložené v Connections sekcii, transformačné dáta v Relations sekcii, rotačné dáta sú uložené ako animačné kľúče v Takes sekcii. Dáta sú uložené v ASCII alebo v binárnej reprezentácii.

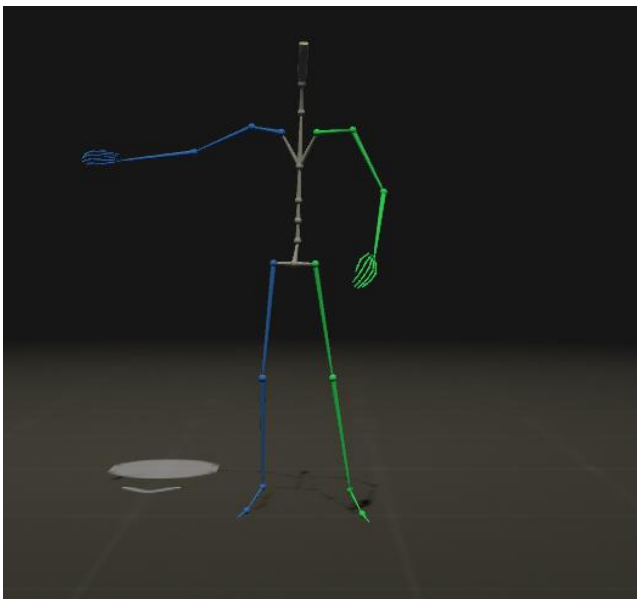
1.1.3 Kostra

Kostra je hierarchická množina vzájomne prepojených častí (kostí), ktoré spoločne tvoria kostru. Každá kostra má svoj root (koreň), ktorý je základom štruktúry, pre človeka je to väčšinou panva. Kostra je tvorená stromovou hierarchiou začínajúcou od koreňa.



Obr. 1: hierarchia kostry, prevzaté z [4]

Kľúčové body (senzory) na obleku sú rozložené tak, aby kopírovali funkčnosť kostry človeka, vytvárajú tak digitálnu kostru. Sú umiestnené na ľudských kĺboch, ktoré pohybujú tuhými časťami tela, kosti sú reprezentované ako spojenia týchto bodov. Každá kosť má trojrozmernú transformáciu z predvolenej pozície väzby, (ktorá zahŕňa jej polohu, mierku a orientáciu vzhľadom ku koreňu) a voliteľnú rodičovskú kosť. Základná kostra obsahuje miesta kĺbov a hierarchiu, ale môže mať aj orientáciu [14]. Vytvorenie kostry taktiež umožňuje animátorom previesť animáciu z jedného objektu na druhý, pokiaľ majú rovnakú kostru. Pre lepšiu presnosť mapovania sa dajú body doladiť alebo pridať softvérom.

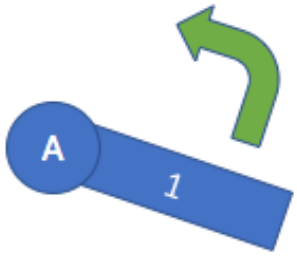


Obr. 2: kostra v softvéri Rokoko Studio

1.1.3.1 Kĺby

Kĺby sú určené polohou senzorov na obleku a rozdeľujeme ich na 2 typy:

- **Otočné kĺby** – kĺb je spojený s článkom, ktorý sa okolo neho otáča.

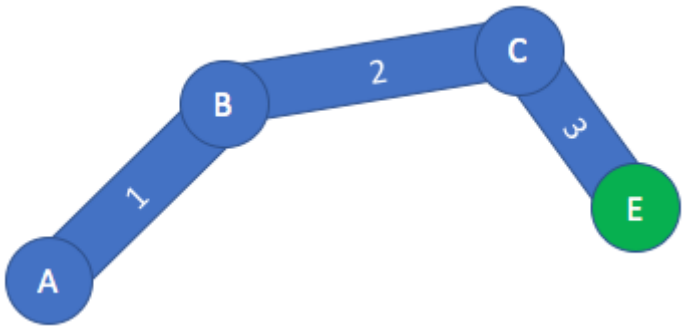


Obr. 3: otočný kĺb, prevzaté z [4]

- **Prizmatické kĺby** – kĺb funguje podobne ako vysúvanie/zasúvanie dĺžky výsuvnej rukoväte na kufri.

Obr. 4: prizmatický kĺb, prevzaté z [4]

Ak chceme zmeniť pozíciu koncových častí kostry, musíme využiť funkcie kinematiky. Koncové časti hierarchie kostry sa nazývajú koncové efektory. Koncový efektor je voľný koniec reťaze striedajúcich sa kĺbov a článkov. Nie je to kĺb, iba poloha na konci kĺbového telesa. Kĺbové teleso môže mať viacero koncových efektorov, rovnako ako binárny strom môže mať viacero listov [4].



Obr. 5: koncový efektor, prevzaté z [4]

1.1.3.2 Kinematika

Pojem kinematika zahŕňa doprednú kinematiku a inverznú kinematiku.
Funkcia doprednej kinematiky (forward kinematics) berie pozíciu všetkých kĺbov ako vstup a vypočíta polohu koncového efektora ako výstup [4]. To znamená, že pri veľkej štruktúre musíme definovať veľký počet kĺbov, čo je neefektívne.

Ak chceme zmeniť koncový efektor na konkrétnu pozíciu, teda poznáme polohu, ale nevieme polohu kĺbov, tú vieme zistiť pomocou inverznej kinematiky. Funkcia inverznej kinematiky (inverse kinematics) berie koncový efektor ako vstup do funkcie a vypočítava polohu potrebnú na to, aby koncový efektor dosiahol cieľovú polohu – poloha je výstup. Tieto prístupy slúžia na úpravu animácií.

1.1.3.3 Upravovanie kostry

Kostra opisuje hierarchiu spojených bodov. Každý bod je prepojený s predchádzajúcim, a preto sa pri zmene pozície jedného bodu zmení pozícia aj ostatných bodov k nej naviazaných, teda ak vytočíte stehno doprava, vaša noha bude smerovať doprava. Úplná transformácia podriadeného uzla je produktom jeho rodičovskej transformácie a jeho vlastnej transformácie. Ak chceme škálovať konkrétnu kosť alebo kĺb, potrebujeme iba jednu maticu na výpočet konečnej polohy vrcholu. Ale v softvéri na 3D modelovanie zvyčajne kosť nie je len transformačná matica, ale štruktúra, ktorá spája dva kĺby. Takže ak je kĺb v ramene a ďalší

kĺb v lakti, ramenná kosť je to, čo spája rameno s lakťom. Preto môžeme kosť opísať z hľadiska východiskovej polohy, dĺžky a rotácie.

1.1.4 Optické motion capture systémy

Optické motion capture systémy využívajú údaje získané z viacerých kalibrovaných kamier na trianguláciu 3D polohy objektu tak, aby poskytovali prekrývajúce sa projekcie. Údaje sa získavajú pomocou špeciálnych bodov pripojených k postave, ktoré musí kamera vidieť. Každý bod má svoje údaje ako pozícia, farba a tvar. Značky je potrebné umiestniť na miesta, kde je koža v blízkosti kostí, aby sa zabránilo skĺznutiu. Kamery musia umožňovať synchronizáciu medzi ostatnými kamerami a včas zaregistrovať viacero video streamov. Externá synchronizácia je možná cez spúšťací impulz odoslaný z paralelného portu počítača do každej kamery. Signál je prenášaný cez koaxiálne káble do externého konektora v každej kamere.

Výhody:

- ľahká konfigurácia bodov
- optické dáta sú presné
- je možné snímať veľké množstvo bodov
- veľmi vysoká vzorkovacia frekvencia, ktorá umožňuje zachytávanie rýchlych pohybov ako sú bojové umenia [17]

Nevýhody:

- nahrávanie je obmedzené na špeciálne prostredie
- nemožné nahrávať, ak sú zakryté body
- po nahratí dáta potrebujú spracovanie

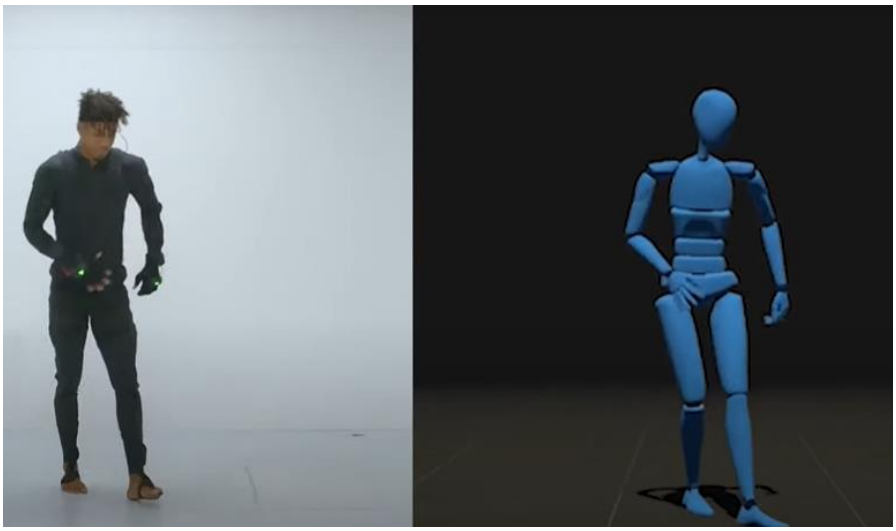
1.1.5 Hybridné systémy

Hybridné systémy kombinujú inerčné senzory s optickými senzormi na zvýšenie presnosti detekcie pohybu, bez manuálneho čistenia dát.

1.1.6 Inerčné obleky

Na nasnímanie pohybu ľudského tela v reálnom čase slúži motion capture oblek, ktorý prevedie pohyby človeka do matematických súradníc pomocou snímania kľúčových bodov na obleku, ktoré sú prevedené na 3D reprezentáciu pohybu, pomocou vytvorenia kostry a jej transformácie v čase. Animátori potom pomocou počítačových programov prekryjú informácie o pohyboch a vytvoria virtuálnu množinu, v ktorej sa pohyb uskutoční. Väčšina týchto systémov používa inerčné meracie jednotky (IMU), ktoré obsahujú gyroskop, magnetometer a akcelerometer na meranie rýchlosti otáčania. Systémy inerčného snímania pohybu zachytávajú všetkých šesť stupňov voľnosti pohybu ľudského tela v reálnom čase. Vo všeobecnosti platí, že čím viac IMU senzorov oblek obsahuje, tým presnejšie sú dáta, no drahší oblek a náročnejšia kalibrácia.

Obleky nevyužívajú žiadnu kameru pre nahratie pohybu, no v prípade potreby je možné využiť kameru na získanie absolútnej polohy herca. Tieto pohybové dáta sa používajú na živé vysielanie pohybu cez WiFi alebo asynchrónne nahratie dát do softvéru, ako je Unreal Engine 4 alebo MotionBuilder.



Obr. 6: inerčný oblek, prevzaté z https://www.rokoko.com/products/smartsuit-pro

Výhody:

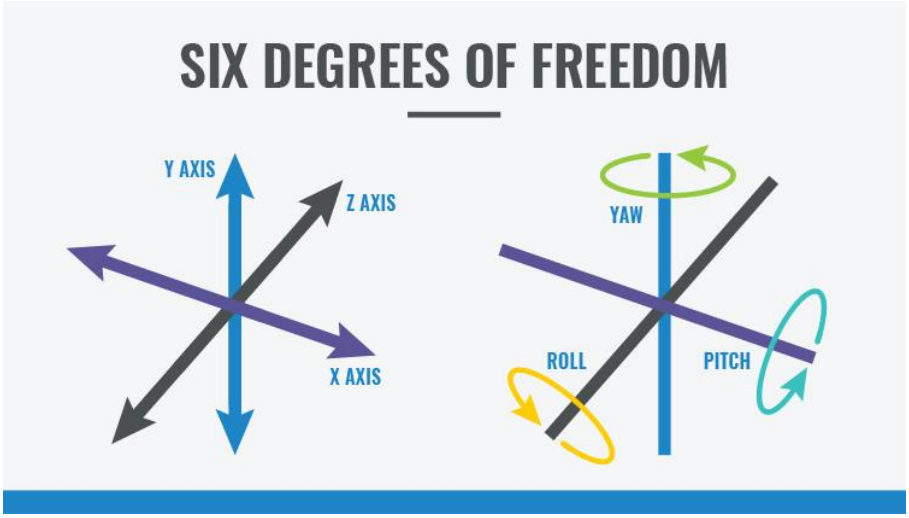
- nízka odozva blízka reálnemu času
- množstvo animácií, ktoré je možné vyprodukovať za daný čas, je extrémne veľké v porovnaní s tradičnými animačnými technikami, čo prispieva k efektívnosti a dodržiavaniu stanovených termínov
- zložitý pohyb a realistické fyzikálne vplyvy, ako napríklad výmena síl alebo vplyv váhy, sú ľahko simulovateľné

Nevýhody:

- keďže má každý človek iné proporcie, pre každého herca je potrebná osobitná kalibrácia
- potreba špecifického softvéru a hardvéru
- vysoké náklady na hardvér, čo môže odradiť malé firmy
- nemožno zachytiť pohyb, ktorý sa neriadi fyzikálnymi zákonmi
- ak má počítačový model iné proporcie ako herec, môžu sa vyskytnúť abnormality. Napríklad, ak má kreslený model nadrozmerné ruky, tieto môžu pretínať telo postavy, ak si herec nedáva pozor na ich fyzický pohyb.

1.1.6.1 IMU senzor

Senzor umožňuje zachytiť 2 až 6 DOF (degrees of freedom) stupňov slobody, čo predstavuje počet rôznych spôsobov, ktorými sa objekt môže pohybovať v 3D priestore. Maximálny počet DOF je 6, čo zahŕňa 3 stupne posuvného (plochého) pohybu cez priamu rovinu pozdĺž každej osi a 3 stupne rotačných pohybov okolo každej osi [3].



Obr. 7: šesť stupňov slobody, prevzaté z [3]

Senzor sa skladá z troch hlavných zariadení:

Akcelerometer: meria lineárne zrýchlenie cez jednu os (napríklad pád mobilu na zem). Ak zintegrujeme zrýchlenie raz, získame tým rýchlosť, no ak opätovne zintegrujeme zrýchlenie, dostaneme odhad polohy. Pre dvojitú integráciu nie je akcelerometer odporúčaný na odhad vzdialenosti.

Gyroskop: meria uhlovú rýchlosť okolo troch osí. Po fúzii senzorov pomocou softvéru je možné použiť gyroskop na určenie orientácie objektu v 3D priestore.

Magnetometer: meria magnetické polia, čo mu umožňuje zistiť kolísanie magnetického poľa Zeme meraním hustoty magnetického indukčného toku vo vzduchu v bode umiestnenia senzora v priestore. Vďaka tomu dokáže tento senzor nájsť vektor smerom k magnetickému severu Zeme. Ak spojíme tieto dáta s dátami z akcelerometra a gyroskopu, vieme určiť absolútny kurz. Senzory, v kombinácii so softvérom na splynutie senzorov, sa dajú použiť na určenie pohybu, orientácie a smerovania [3].

1.2 3D Model

3D model je digitálna reprezentácia objektu alebo povrchu. Tento model môže vzniknúť manuálne človekom, manipuláciou siete (mesh), alebo automaticky pomocou 3D skenov. Proces manuálneho modelovania pri príprave geometrických údajov pre 3D počítačovú grafiku je podobný ako pri plastike, ako je napríklad sochárstvo. 3D model je možné fyzicky vytvoriť pomocou 3D tlačových zariadení, ktoré tvoria 2D vrstvy modelu s trojrozmerným materiálom, jednu vrstvu po druhej. Model ľudského tela, ktorý budeme upravovať sa bude skladať z mračna bodov (point cloud).

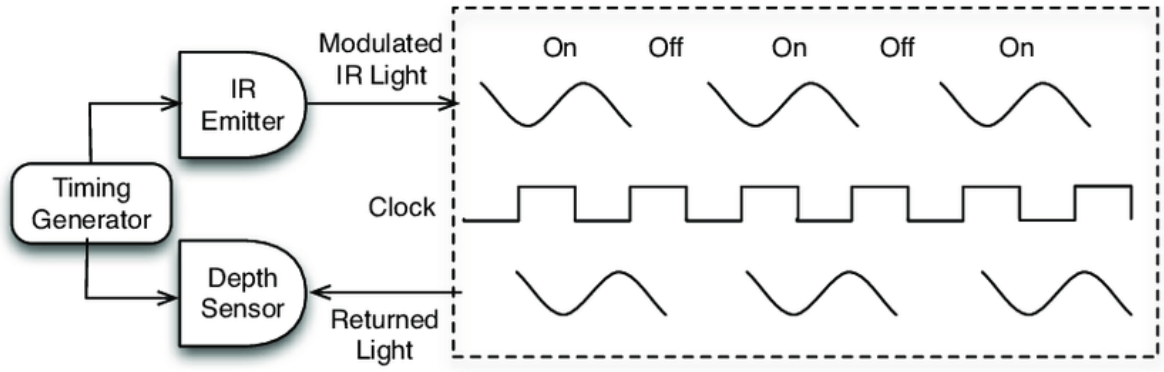
1.2.1 Mesh

Je kolekcia bodov vo virtuálnom priestore, ktoré tvoria objekt. Tieto body sú hranami pospájané do polygónovej siete. Sieť je množina vrcholov a polygonálnych prvkov (ako sú trojuholníky a štvorce), ktoré spoločne definujú trojrozmerný geometrický tvar. Najjednoduchšia sieťová reprezentácia teda pozostáva zo zoznamu vrcholov a polygónu [1]. Každý bod alebo vrchol má svoju polohu na sieti a spojením týchto bodov do tvarov sa vytvorí povrch objektu. Mesh v našej aplikácii využívame na vykonanie animácie, ktorú upravujeme, samotný mesh sa mení len ako dôsledok zmeny kostry, ktorú využíva.

1.2.2 3D skener

Je zariadenie, ktoré analyzuje objekt z reálneho sveta alebo prostredia a získava informácie o jeho tvare a vzhľade [5]. Získané dáta môžu byť použité na vytvorenie 3D modelu. Tieto zariadenia môžu využívať viaceré technológie. Každá z nich má svoje výhody a obmedzenia. Laserové skenovanie je riadené vychyľovanie laserových lúčov, viditeľných alebo neviditeľných. Väčšina laserových skenerov používa na riadenie laserového lúča pohyblivé zrkadlá [16]. 3D skener sa podobá 3D kamere, pretože sníma len predmety, ktoré vidí priamo, nie sú zakryté pred jeho snímaním. HĺbkovÉ senzory v zariadeniach zvyčajne používajú na

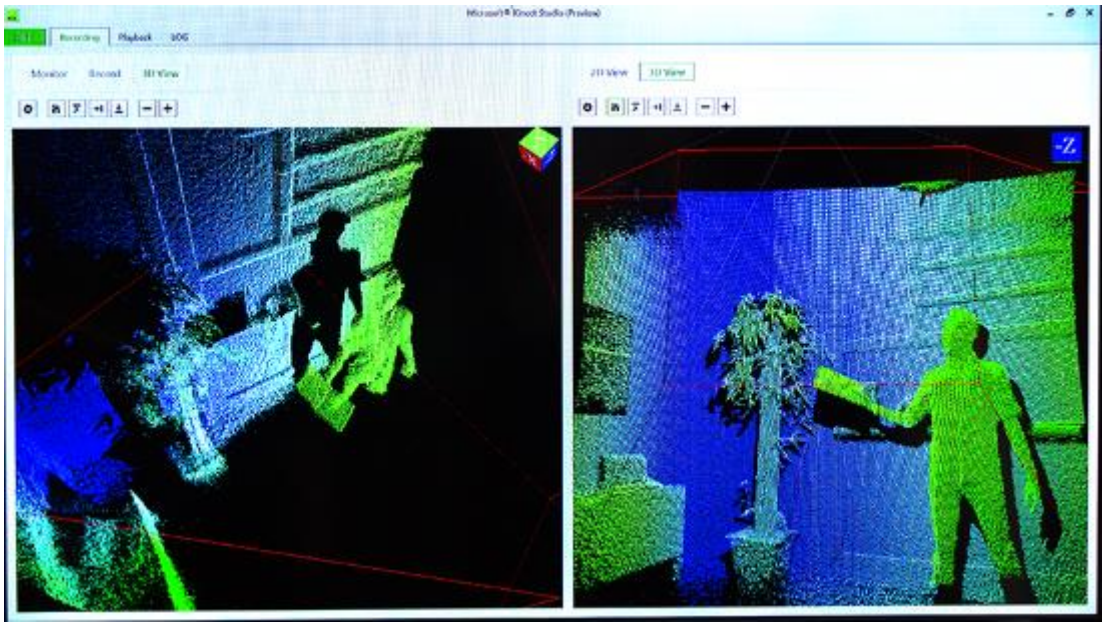
získanie údajov o hĺbke techniku nazývanú „čas letu“. Základnou myšlienkou tejto techniky je, že sa uvoľní lúč svetla a meria sa čas medzi tým, kedy svetlo opustí senzor, odrazí sa od objektu a vráti sa späť do senzora. Čas sa potom interpoluje do vzdialenosti, ktorú svetlo prešlo. Získava sa tak hĺbkový obraz snímaného objektu [6]. Každý pixel takého obrazu je opísaný vzdialenosťou od skenera. Hĺbkový obraz tak prispieva treťou dimenziou pri vytváraní mračna bodov. Vytvorené body skenovaním objektu majú teda hodnotu na x-ovej, y-ovej aj z-ovej súradnici.



Obr. 8: hĺbkové skenovanie, prevzaté z *Roanna Lun, Wenbing Zhao. A Survey of Applications and Human Motion Recognition with Microsoft Kinect*

1.2.2.1 3D skenovanie

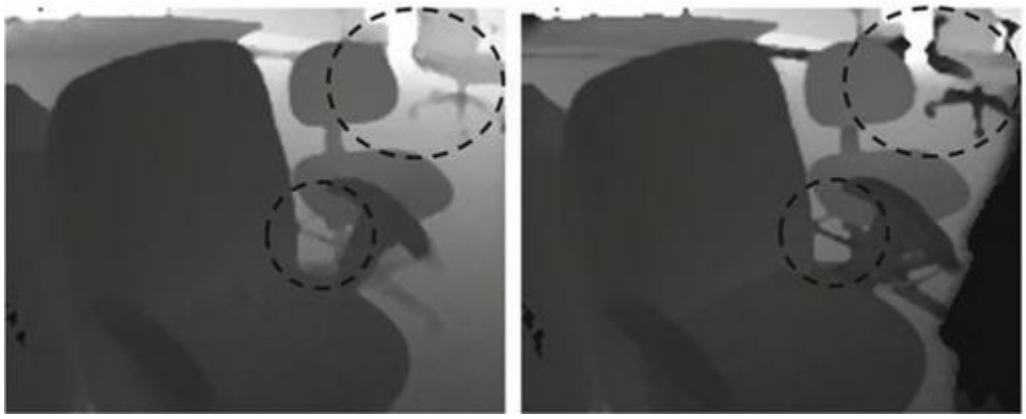
Používa sa na vytvorenie mračna bodov z geometrických vzoriek povrchu predmetu. Vo väčšine prípadov ale nestačí len jeden sken na vytvorenie digitálneho 3D modelu. Predmet je potrebné snímať z rôznych uhlov pre získanie dát o celom jeho povrchu. Tieto skeny sa musia preniesť do spoločného referenčného systému, čo je proces, ktorý sa zvyčajne nazýva zarovnanie alebo registrácia. Potom sa musia zlúčiť, aby sa vytvoril úplný model.



Obr. 9: 3D videnie, prevzaté z https://docs.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows

3D skenovanie nie je bezproblémové. Jedným z problémov je presnosť merania, ktorá je obmedzená silou emitovaného infračerveného signálu, ktorý je zvyčajne dosť nízky v porovnaní s denným svetlom, takže denné svetlo kontaminuje odrazený signál. Ďalším problémom pri 3D skenovaní je pohybový efekt (motion blur), ktorý vzniká pohybom kamery alebo skenovaného objektu počas snímania. Tento efekt vykazuje unikátne problémy pri 3D skenovaní v porovnaní s konvenčnou 3D kamerou. Presnosť hĺbky a snímková frekvencia sú kvôli požadovanej integrácii času hĺbkovej kamery obmedzené a dlhší integračný čas umožňuje vyššiu presnosť merania hĺbky [6]. Pri snímaní statických objektov je preto lepšie, znížiť snímkovú frekvenciu, aby sme získali vyššiu presnosť merania z dlhších integračných časov. Ak ale chceme nasnímať pohyb ľudského tela 3D kamerou, proces je zložitejší.

Práca s názvom Motion Capture Research: 3D Human Pose Recovery Based on RGB Video Sequences [12] navrhuje postup na získanie ľudskej pózy zo záznamu 3D kamery. Tento postup zahŕňa tri časti: komplexný 3D odhad ľudskej pozície na základe jednotlivých snímok, generovanie pozície ľudského tela na základe video streamov a overenie experimentu so zachytením pohybu na základe 3D obnovy ľudskej pozície. Zistilo sa, že lacné a pohodlné riešenie založené na RGB video sekvenciách v kombinácii s informáciami o ľudskej výške, môže získať výsledky zodpovedajúce komerčnej platforme snímania pohybu na základe RGB-D video sekvencií.



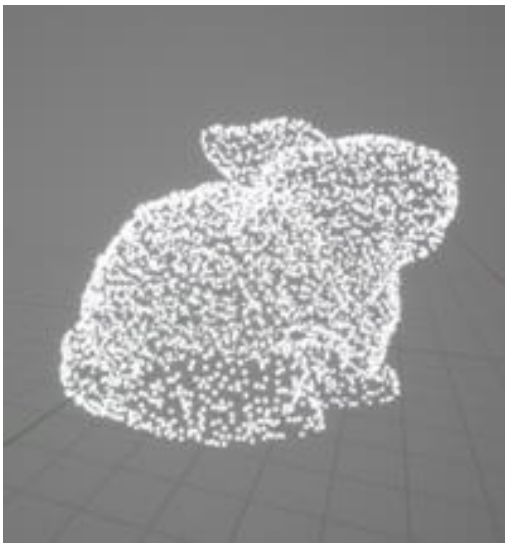
Obr. 10: pohybový efekt, prevzaté z *Miles Hansard, Seungkyu Lee, Ouk Choi, Radu Patrice Horaud. Time-of-Flight Cameras: Principles, Methods and Applications. South Korea*

1.2.2.2 Kinect SDK

SDK obsahuje ovládače na používanie snímačov Kinect v2 na počítači s operačným systémom Windows. Taktiež obsahuje aplikačné programovacie rozhranie (API) a aplikačné rozhranie so vzorkami kódu. API je pre vývoj aplikácií a prácu s Kinectom nevyhnutné.

1.2.3 Mračno bodov (Point Cloud)

Mračno bodov je súbor údajových bodov v priestore, definovaných daným súradnicovým systémom [12] x,y,z v troch rozmeroch s možnosťou informácie o farbe. Mračno môže definovať tvar nejakého skutočného alebo vytvoreného fyzického systému. Vo všeobecnosti sú mračná vytvorené 3D skenerom, pričom sa dajú priamo vykresliť alebo konvertovať na modely polygónovej alebo trojuholníkovej siete. Široké využitie znemožňuje organizáciám vlastniť celý proces nahrávania mračien, čo umožnilo vznik Point Cloud Library.



Obr. 12: mračno bodov

1.2.3.1 XYZ súborový formát

Xyz je veľmi jednoduchý súborový formát na zaznamenávanie bodov v 3D priestore. Štruktúra xyz súborov sa skladá z riadkov so súradnicami bodov a s komentárom. Tie začínajú znakom „#“. Tieto riadky pri čítaní súboru ignorujeme, rovnako ako prázdne riadky, ktoré sa môžu objaviť kdekoľvek. Súradnice bodov sú zapísané v jednotlivých riadkoch oddelene medzerou alebo čiarkou ako hodnoty na x-ovej, y-ovej, z-ovej osi.

```
1.27246809,-1.31143999,3.92800021
1.26305604,-1.31274402,3.93200016
1.25426006,-1.31471813,3.93800020
```

Obr. 11: dáta zo súboru s formátom xyz

1.2.4 Point Cloud Library (PCL)

Je voľne dostupná knižnica algoritmov pre spracovanie mračna bodov a 3D geometrie. Je napísaný v C++ a vydaný pod licenciou BSD [8].
Knižnica obsahuje algoritmy pre:

- filtrovanie

- rekonštrukciu povrchu
- 3D registráciu
- segmentáciu

Každý modul je implementovaný ako menšia knižnica, ktorú je možné skompilovať samostatne. Táto modularita je dôležitá pre fungovanie na platformách so zníženým výpočtovým výkonom. PCL taktiež obsahuje moduly pre konverziu z najbežnejších dátových formátov mračien bodov (xyz, ply) na pcd, s ktorým pracuje. Niektoré knižnice využívajú algoritmus RANSAC.

1.2.4.1 RANSAC

Algoritmus RANSAC vykonáva dva iteratívne opakovania krokov na danom mračne bodov a to generovanie hypotézy a overenie. Hypotézy sa generujú náhodným výberom minimálnej podmnožiny n bodov a odhadom parametrov modelu [9]. Minimálna podmnožina obsahuje najmenší počet bodov potrebných na jednoznačný odhad modelu, napríklad na určenie roviny sú potrebné 3 body. Potom sa ostatné body v mračne testujú s výslednými tvarmi. Po určitom počte iterácií sa extrahuje tvar, ktorý má najväčšie percento obsahujúcich bodov. RANSAC odhaduje globálny vzťah, ktorý vyhovuje údajom, pričom súčasne klasifikuje údaje na vnútorné hodnoty (body v súlade so vzťahom) a odľahlé hodnoty (body, ktoré nie sú v súlade so vzťahom). Vďaka svojej schopnosti tolerovať veľkú časť odľahlých hodnôt je algoritmus populárnou voľbou pre rôzne robustné problémy s odhadmi [23].

1.2.4.2 PCD súborový formát

Je súborový formát, ktorý slúži na ukladanie údajov 3D mračna bodov. Bol vytvorený, pretože existujúce formáty nepodporovali niektoré funkcie knižnice PCL a taktiež nemali flexibilitu a rýchlosť, ktorú ponúka pcd. Každý súbor pcd obsahuje hlavičku, ktorá deklaruje určité vlastnosti údajov mračna bodov v súbore, pričom je zakódovaná v ASCII.

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z rgb
SIZE 4 4 4 4
TYPE F F F U
COUNT 1 1 1 1
WIDTH 1188
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 1188
DATA ascii
0.373256 0.275094 4.364 4278190080
0.36134 0.275104 4.364 4278190080
0.373074 0.286884 4.362 4278190080
0.361164 0.286894 4.362 4278190080
```

Obr. 13: dáta zo súboru s formátom pcd

1.3 Unreal Engine 4.24.3

Je herný engine, ktorý bol pôvodne vyvinutý pre video hry z pohľadu prvej osoby. Odvtedy sa používa v rôznych oblastiach, kde sa využíva priestorová (3D) grafika kvôli jeho masívnemu systému nástrojov a editorov. Pomocou ktorých umožňuje organizovať vaše položky (assets) a manipulovať s nimi. Medzi komponenty Unreal Engine patrí zvukový, fyzikálny a grafický engine [19].

Unreal Engine je vytvorený v C++ a podporuje širokú škálu desktopových, mobilných, konzolových platforiem a platforiem virtuálnej reality [16]. Zložité scény sa v ňom vytvárajú ľahko, pretože Unreal má zabudované materiály a animačné nástroje. C++ sa v Unreal Engine používa na vytvorenie vlastných skriptov. Ak ale chceme použiť už naprogramované funkcionality, vieme využiť špecialitu Unreal Enginu, ktorou sú blueprinty.

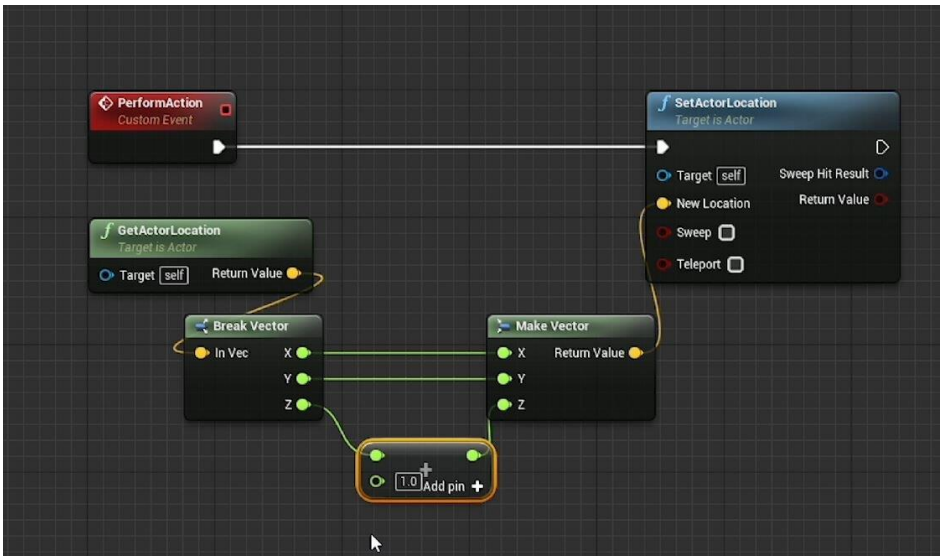
Medzi jeho hlavné výhody patrí:

- grafická úroveň
- práca s programom je žiadaná
- používateľské rozhranie sa neustále aktualizuje pomocou najnovších nástrojov a možností
- podpora

- obchod
- C++, ktorý samotný robí z programu program prvej voľby vývojára

1.3.1 Blueprintový vizuálny skriptovací systém

Je herný skriptovací systém, ktorý využíva rozhranie založené na uzloch na vytváranie herných prvkov z Unreal Editora [17], inými slovami ide o programovanie vo vizuálnej forme. Rovnako ako u bežných skriptovacích jazykov slúži na definovanie objektovo orientovaných objektov a tried. Blueprinty využívajú uzly spojené navzájom. Tieto uzly môžu byť udalosti, akcie, podmienky a podobne. Uzly sú v podstate prefabrikované bloky kódu, ktoré môžete pridať k svojim objektom a vytvoriť tak interakcie, ktoré pomáhajú používateľom vytvárať funkcionality bez písania skriptov a kódov [11]. Uzly môžu mať vstupy a výstupy. Ak dáte uzlu vstupné hodnoty, vypočíta čo má, a na výstup vráti výsledok.



Obr. 14: blueprinty

1.3.2 C++ vs blueprint

V Unreal Engine máme na výber medzi programovaním s blueprintami alebo v C++. Oba vynikajú v rôznych oblastiach. Vo všeobecnosti platí, že na vytvorenie zložitých výpočtových metód by mal byt použitý C++ kód, na všetko ostatne sú tu blueprinty [22].

Výhody C++:

- väčšia flexibilita pomocou prístupu k celému kódu v projekte
- mechanika C++ sa vykonáva rýchlejšie ako blueprintová verzia, hoci to nemusí byť viditeľné pri tvorbe malých projektov, no pri vývoji veľkých hier je rozdiel výraznejší
- mechanika, ktorá využíva určité funkcie ako Tick alebo BeginPlay, môže by v C++ ľahko organizovaná, ale v blueprintoch môže by náročné pripojiť niekoľko rôznych mechaník k jednému uzlu Tick [20]
- nízka programovacia úroveň
- možnosť implementovať knižnice tretej strany [21]

Nevýhody C++:

- pri vytváraní herných mechanizmov je C++ náchylnejšie na robenie chýb, ktoré je ťažšie vystopovať a v niektorých prípadoch môže dokonca Unreal Engine prestať pracovať, ak je chyba kritická
- práca s C++ je komplikovanejšia

Výhody blueprintov:

- umožňujú veľmi rýchlo testovať a iterovať
- vytvorenie uzlov a ich zapojenie do hierarchie kódu je rýchlejšie ako písanie kódu a jeho kompilácia
- pokiaľ ide o vizuálne a 3D aspekty herných mechanizmov, je oveľa jednoduchšie vytvoriť napríklad kolízny box a nastaviť správnu veľkosť vo vnútri plánu, ako hádať pomocou pokusov a omylov v C++

Nevýhody blueprintov:

- ak nie ste opatrní, vytváranie komplexných herných mechanizmov pomocou blueprintov môže vytvoriť obrovskú spleť uzlov a spojovacích vláken, čo môže sťažiť zisťovanie, kde sa nachádzajú určité časti hernej mechaniky, alebo odstraňovanie chýb, ak mechanika nefunguje správne

My sme používali vo väčšej miere blueprinty, nakoľko využívame funkcionality rôznych pluginov, ktoré sú implementované pomocou vlastných blueprintov. Pri využívaní blueprintov

vieme pomocou C++ vytvoriť blueprinty s vlastnou funkcionalitou a tie potom vieme napojiť do existujúcej štruktúry blueprintov.

1.3.3 Unreal obchod

V obchode sa nachádza veľké množstvo obsahu od materiálov, cez modely až po pluginy, ktoré môžeme pridať do svojho projektu. Časť tohto obsahu je zadarmo, no väčšina je za cenu určenú pre vývojára. Ak chceme predávať vlastný obsah, stretneme sa s daňou 12% z predaja. [18] My sme využili obchod na získanie voľne dostupných pluginov na importovanie a renderovanie, ako aj pre rozšírenie ponuky dostupných blueprintov.

1.3.3.1 Plugin

Je kolekcia kódu a údajov, ktoré môžu vývojári jednoducho povoliť alebo zakázať v editore v jednotlivých projektoch podľa potreby. Pluginy môžu pridávať herné funkcionality, upravovať vstavané funkcionality enginu, umožniť načítanie nových súborových typov a podobne. Väčšina existujúcich podsystémov enginu bolo navrhnutých tak, aby boli rozšíriteľné pomocou pluginov [13]. Unreal Engine priamo neumožňuje importovať súbory s mračnom bodov, preto potrebujeme doplnok, ktorý nám umožní importovať tieto dáta. Nato sme využili funkcionality LIDAR Point Cloud Plugin. Pomocou tohto doplnku môžeme importovať, vizualizovať a upravovať mračná bodov, uložené v najbežnejších typoch súborov na prácu s mračnami bodov. Taktiež implementuje rôzne funkcionality:

- podpora pre súbory ASCII (txt, xyz, pts, las)
- runtime import a export dát
- asynchrónny import údajov
- podporuje dynamické tiene, a to ako vysielač, tak aj prijímač
- dynamický systém úrovne detailov a streamovanie GPU umožňujú načítanie veľkých prostriedkov
- viaceré techniky farbenia (ako RGB, intenzita, nadmorská výška, klasifikácia)
- vykresľovanie údajov ako bodov alebo ikon

- umožňuje asynchrónnu modifikáciu údajov za behu
- podpora blueprintov
- implementuje techniku osvetlenia Eye-Dome na zlepšenie zvýraznenia tvaru
- rozsiahle farebné úpravy

2 Návrh

V tejto kapitole sa budeme zaoberať návrhom práce. Každé riešenie problému vychádzajúceho zo zadania práce bude detailne rozpísané vo vlastnej podkapitole. Kapitola bude logicky rozdelená na 4 podkapitoly, ktoré sú ešte rozdelené podľa potreby na ďalšie podkapitoly. Podkapitoly začínajú od použitých technológií 3D skenovania a mocap obleku, následne si jednotlivo rozoberieme problémy a ich riešenia, vychádzajúce zo zadania práce: Nahrávanie dát, synchronizácia nahraných dát, upravovanie kostry a mračná bodov.

2.1 Použité technológie

Aplikáciu som sa rozhodol implementovať v hernom Unreal Engine vo verzii 4.24.3, nakoľko mám s prácou v ňom skúsenosti. Importovanie animácií a objektov do Unreal Enginu je pomerne jednoduché a má množstvo preddefinovaných funkcií na prácu s 3D objektmi. Funkcionality aplikácie sú vytvorené pomocou blueprintov a programovacieho jazyka C++ vo vývojovom prostredí Microsoft Visual Studio 2019. Verziu 4.24.3 som si vybral preto, lebo je to posledná podporovaná verzia pluginu Lidar Point Cloud, ktorý využívam a taktiež novšie verzie Unreal Enginu neponúkajú žiadne nevyhnutné funkcionality, ktoré som potreboval využiť v práci.

2.1.1 Technológie skenovania 3D kamerou

Ľudské telo sme pred 3D skenerom zachytili vo forme mračna bodov. Mračno bodov sme si vybrali preto, lebo sa dá relatívne ľahko upravovať, zobrazovať a existuje veľa bezplatných riešení na prácu s ním. V nami zvolenom riešení snímame pohybujúci sa objekt (ľudské telo), pričom vzniká pohybový efekt (kapitola 1.2.2.1). Keďže nemusíme nahrávať rýchle zmeny polôh ľudského tela, ako napríklad pri športovaní, pričom 3D skener ostáva nehybný, pohybový efekt vieme minimalizovať. Zvyšný šum pri nahrávaní vieme doladiť softvérom.

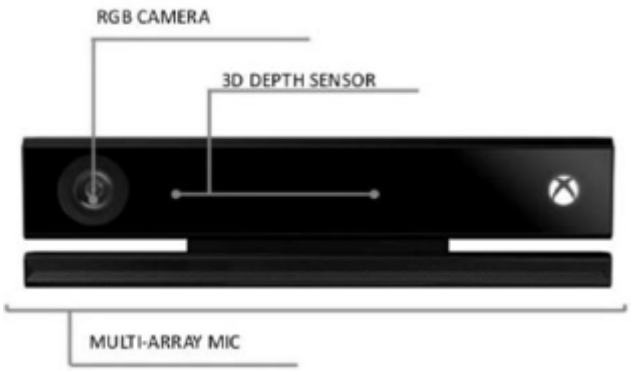
Keďže sme objekt skenovali len spredu, vyhli sme sa zlučovaniu snímok zachytených z rôznych uhlov okolo celého objektu ako pri bežnom 3D skenovaní.

2.1.1.1 Softvér

Na úpravu snímok sme použili Cloud Compare. Do programu sme importovali celú sekvenciu snímok pre účel ich zmenšenia a odstránenia nežiaduceho pohybového efektu.

2.1.1.2 Hardvér

Na skenovanie človeka vykonávajúceho pohyb sme použili Microsoft Kinect v2. Kinect používa karteziánsky súradnicový systém sústredený na Kinect. Kladná os Y smeruje nahor, kladná os Z ukazuje, kam smeruje Kinect a kladná os X je vľavo. Výstup z 3D kamery dostaneme ako tok video snímok daný ako 4 bajty na pixel vo formáte BGRA (modrá-zelená-červená-alfa) a pixely sa skenujú po riadkoch horizontálne v doméne obrazu. Úplná nekomprimovaná snímka s veľkosťou 640 x 480 má 640 x 480 x 4 bajty. Tok hĺbkových snímok je daný ako 2 bajty na pixel hĺbky v krátkom formáte bez znamienka. Kinect má infračervený senzor a zdroj infračerveného svetla. Používa techniku, ktorá poskytuje rovnaký druh údajov ako hĺbková kamera, ale za výrazne zníženú cenu. Kinect priamo nepodporuje skenovanie okolia vo forme mračna bodov. Pomocou Kinect SDK (kapitola 1.2.2.2) a C++ dokážeme dostať informácie o snímanej scéne vo forme mračna bodov.



Obr. 15: Microsoft Kinect v2, prevzaté z https://www.researchgate.net/figure/Kinect-V2-depth-sensor-and-its-specifications_fig3_328697418

2.1.2 Technológie zaznamenania pohybu mocap oblekom

Ľudský pohyb sme zaznamenali pomocou mocap obleku a vytvorili tak animáciu do softvéru. Po nasadení obleku je potrebné skontrolovať všetky senzory a ich umiestnenie, aby ich poloha zodpovedala umiestneniu kĺbov na ľudskom tele. Ďalším krokom je kalibrácia. Tá sa dá nastaviť manuálne pomocou fyzického pomerania človeka a jeho vzdialeností kĺbov pre zdokonalenie animácie. Ak nepotrebujeme dokonale presnú animáciu, kalibráciu môžeme nastaviť aj automaticky, a to pomocou zadania výšky človeka. Oblek je pripojený pomocou vlastného prijímača na rovnakú sieť ako počítač, na ktorom je softvér spustený. Oblek posiela informácie cez sieť priamo do softvéru v reálom čase, pritom softvér spracováva údaje a vytvára animáciu. Odtiaľ ju exportujeme vo forme súboru, ktorý importujeme cez editor Unreal Enginu do našej aplikácie.

2.1.2.1 Softvér

Pre uloženie nasnímaného pohybu budeme využívať Rokoko Studio, kde nahráme pomocou obleku a jeho senzorov kostru, ktorá sa bude transformovať v čase a tým pádom bude vykonávať rôzne animácie. Následne výslednú animáciu pomocou funkcií programu exportujeme vo forme súboru s formátom fbx.

2.1.2.2 Hardvér

Pre nahratie pohybového datasetu budeme využívať inerčný oblek Smartsuit Pro od dánskej spoločnosti Rokoko, ktorý využíva 19 zabudovaných 9-stupňových IMU senzorov (kapitola 1.1.4.1).

Obr. 16: Rokoko smartsuit pro, prevzaté z https://nofilmschool.com/rokoko-review-high-quality-mocap-fraction-cost

2.2 Vytvorenie dát pohybu človeka a sekvencie snímok z 3D kamery

Výstupné údaje z 3D skeneru sme pomocou C++ uložili do súborov s formátom xyz, pričom každý súbor bude predstavovať jednu snímku nahrávaného ľudského tela vo forme point cloudu v čase. Sekvencia snímok mračien bodov je rozlíšená názvom súborov (Take04_00000, Take04_00001, Take04_00002...). Čo sa týka výstupnej sekvencie transformácií kostry z obleku, tie uložíme do samostatného súboru s formátom fbx. Súbory sa budú nachádzať v súborovej štruktúre s našou aplikáciou pre neskoršiu prácu s nimi.

2.3 Synchronizovanie nahraných dát

Po nahratí dát hardvérovými technológiami do digitálnej podoby, sme sa stretli s ďalším problémom a to synchronizáciou nahraných dát. Tá sa dala vyriešiť spôsobom vytvorenia nového dátového formátu. Ten by obsahoval údaje o sekvencii snímky nahranej 3D skenerom a animácie pohybu nahranej oblekom vo forme transformácie kostry v čase. Dátový súbor s vlastným formátom by sme nahrali do aplikácie a získali tak v určitom časovom úseku sken a transformáciu, čiže synchronizované dáta pohybu. Tento prístup bol ale veľmi náročný, pretože by sme museli načítať zložitú štruktúru súborového formátu fbx, či už na spracovanie údajov do vlastného súborového formátu, alebo priame načítanie transformačných dát kostry do aplikácie. Keďže Unreal Engine už obsahuje funkcionality na importovanie takýchto súborových formátov v editore, dokumentácia k tomuto prístupu je nedostačujúca.

Preto sme sa rozhodli pre riešenie problému pomocou načítania oboch súborových štruktúr do aplikácie jednotlivo. Tie synchronizujeme pomocou rozdelenia oboch sekvencií na jednotlivé snímky, ktoré budeme zobrazovať v určitom pomere podľa zaznamenania snímok za sekundu (FPS) jednotlivých hardvérových zariadení.

2.4 Upravovanie nahraných dát

Dáta budeme upravovať priamo v aplikácii, kde budú zobrazené vedľa seba. Upravovanie zahŕňa označenie bodov, ktoré chceme odstrániť, odstránenie rovinou podľa určenia osi, odstránenie pozadia, zarovnanie, segmentácia.

2.4.1 Upravovanie animácie (kostry)

V našom riešení je potrebné upravovať kostru animácie. Na to, aby sme ju dokázali upravovať, musíme si animáciu rozložiť na snímky (frames). Jednotlivé snímky potom môžeme upravovať jednotlivo. Deformované snímky vieme zaznamenať a vytvoriť tak spätne animáciu, čím docielime upravovanie nahranej animácie. Aby sme mohli nejakú časť kostry upraviť, musíme si označiť celok, ktorý chceme upravovať. Aplikácia taktiež podporuje

označenie viacerých celkov naraz (hlava, ľavá ruka, pravá noha). Označené celky vieme škálovať pomocou predlžovania a skracovania prizmatických kĺbov (kapitola 1.1.4.3.1), rotovať pomocou otáčania otočných kĺbov (kapitola 1.1.4.3.1) okolo ľubovoľnej osi, alebo posúvať pozdĺž všetkých osí (x,y,z). Ak máme označenú časť, ktorú chceme upraviť, musíme si ešte vyznačiť konkrétnu snímku v čase, v ktorom ju chceme upraviť. Týmto prístupom vieme upravovať kostru v konkrétnej nami vyznačenej snímke animačnej sekvencie alebo si uložiť viacero upravených snímok naraz do súboru, z ktorého ich vieme následne spätne načítať do animácie. Taktiež vieme nahrávať zmenenú sekvenciu snímok, tu následne uložiť do súboru, z ktorého ju vieme podľa potreby opäť načítať do aplikácie. Aplikácia ďalej podporuje aj klasické funkcionality spojené s nahrávaním sekvencie, ako odstránenie zle nahranej sekvencie alebo obnovenie počiatočnej animácie po nahratí zle upravenej sekvencie.

2.4.2 Upravovanie mračna bodov

Funkcionality upravovania mračna bodov sú implementované ako označenie bodov kockou, ktorú vykreslíme po kliknutí do priestoru aplikácie. Označené body sa farebne odlíšia, následne ich môžeme odstrániť alebo zneviditeľniť vo všetkých snímkach sekvencie alebo jednotlivo. Mračno bodov bez odstránených bodov sa dá následne uložiť do súboru. Zneviditeľnené body sa dajú opätovne zviditeľniť a po vyzvaní na uloženie sa uložia s celým mračnom, na rozdiel od tých odstránených, ktoré sa po vyzvaní na uloženie do súboru neuložia. Ďalej sa dá nastaviť aj **point budget** mračna bodov, čo znamená, že mračno sa po vykreslení zadaného počtu bodov už nebude ďalej vykresľovať. Medzi funkcionality aplikácie patrí aj výber tvaru a veľkosti bodov, ktoré sa vykresľujú. Posunutie celého mračna pozdĺž určitej osi a výber farebného odlíšenia bodov podľa tejto zmeny je tiež na výber.

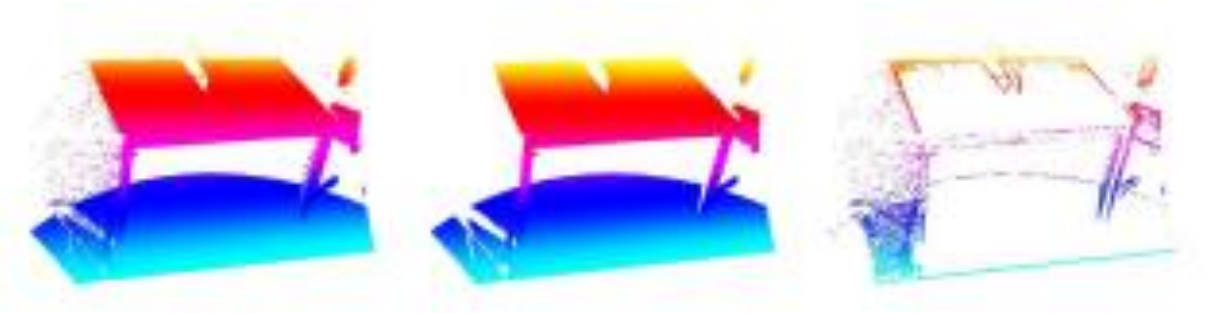
2.4.2.1 Upravovanie mračna bodov knižnicami

Na upravovanie mračna bodov taktiež využívame rôzne PCL knižnice, ktoré konvertujú vybranú snímku v súborovej forme xyz a konvertujú ju na pcd, s ktorými PCL knižnice pracujú. Na pcd snímkach vykonajú úpravu a uložia snímky späť vo forme xyz, aby sme s nimi vedeli ďalej pracovať.

2.4.2.1.1 Filtrovanie pozadia

Takéto úpravy mračien bodov zahŕňajú štatistické odstránenie bodov v pozadí na označených snímkach. V dôsledku chýb merania niektoré mračná bodov obsahujú veľký počet tieňových bodov. Tieto nežiaduce body vieme odstrániť pomocou použitia filtrov založených na štatistickom odstránení bodov, ktoré nemajú dostatočný počet susedných bodov. Tým pádom sa nachádzajú mimo mračna bodov.

Algoritmus prejde vstupom dvakrát: Prvý krát sa vypočíta priemerná vzdialenosť, ktorú má každý bod k svojím najbližším K susedom. Hodnota K môže byť nastavená pomocou setMeanK() v aplikácii. Ďalej sa vypočíta priemer a štandardná odchýlka všetkých týchto vzdialeností, aby sa určil prah vzdialenosti. Taktiež možno v aplikácii nastaviť násobiteľ pre štandardnú odchýlku pomocou setStddevMulThresh(). Počas ďalšej iterácie budú body klasifikované ako vnútorne alebo vonkajšie, podľa ich priemernej susednej vzdialenosti.



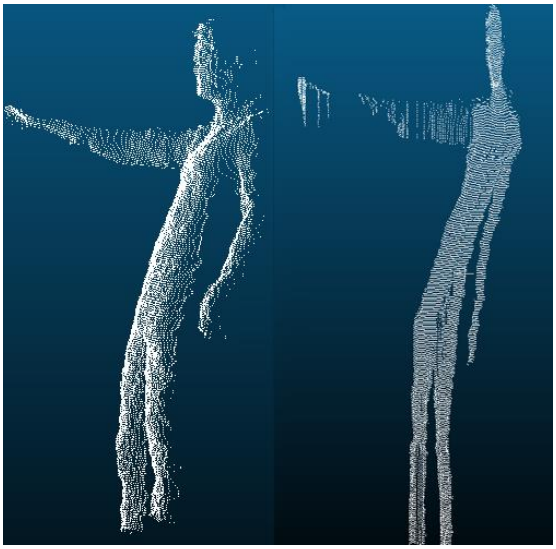
Obr. 17: Vľavo: surové mračno bodov získané skenerom,
v strede: výsledné mračno bodov po použití filtra na odstránenie pozadia
vpravo: body, ktoré sa odstránili, teda majú vzdialené susedné body, prevzaté z
https://pointclouds.org/assets/pdf/pcl_icra2011.pdf

2.4.2.1.2 Zarovnanie mračna bodov

Táto úprava mračna bodov zarovná konvexnosť mračna bodov do roviny pomocou morfológie, vytvorí zemnú (ground) reprezentáciu bodov v mračne. Matematická morfológia skladá operácie na základe teórie množín na extrahovanie prvkov z obrázka. Väčšinou sa

využívajú dve základne operácie: zväčšenie (dilatácia) alebo zmenšenie (erodovanie) veľkosti prvkov v binárnych obrazoch.

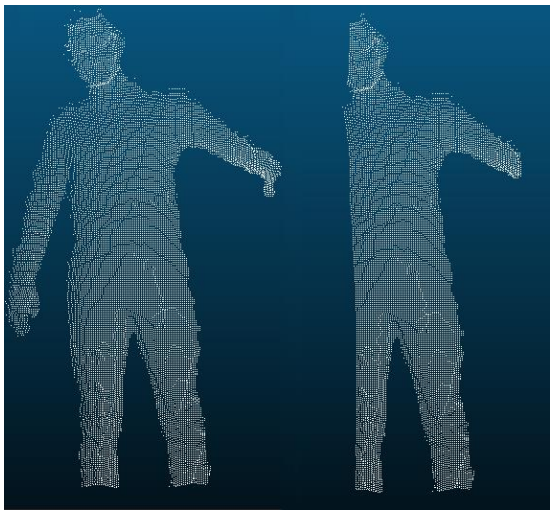
Práca Filtering airborne laser scanning data with morphological methods [24] navrhuje spôsob filtrovania mračna bodov zachytených z pohľadu z výšky na zem. Myšlienkou tejto morfologickej metódy je priblíženie povrchu terénu pomocou morfologických operácií. Prvým krokom je vytvorenie 2D rastra mračna bodov tým, že sa vezme hodnota nadmorskej výšky posledného výstupu každého impulzu v bunke 1m x 1m. Potom sa morfologický vytvorí približná holá zem, stromy a vegetácia je odstránená. Body zachytené vo veľkej nadmorskej výške možno posunúť na nižšiu hodnotu nadmorskej výšky pomocou iných bodov vo vnútri mračna [25].



Obr. 18: vľavo: surové mračno bodov získané skenerom
vpravo: mračno po aplikovaní úpravy pomocou morfológie

2.4.2.1.3 Orezanie mračna rovinou na osi

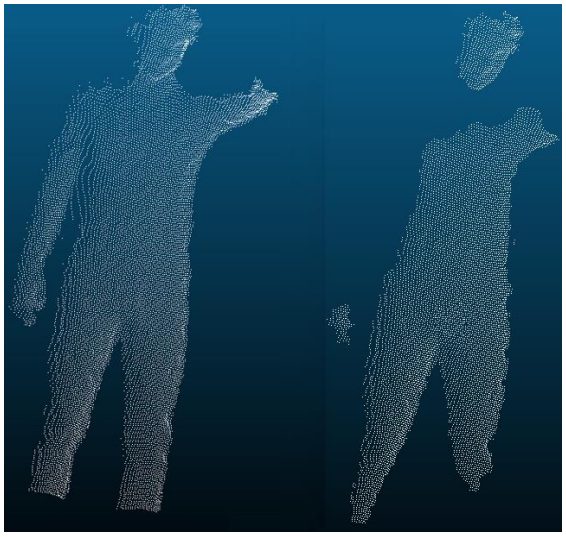
Táto funkcia nám umožňuje vybrať os, podľa ktorej aplikujeme filter na orezanie bodov, pričom zadávame minimálnu a maximálnu hodnotu súradníc bodov na danej osi. Body s hodnotou súradnice na danej osi v zadanom intervale ostanú súčasťou mračna, pričom body mimo daný interval sa vymažú.



Obr. 19: orezanie na osi x s minimálnou hodnotou 0 a maximálnou 2

2.4.2.1.4 Rovinná segmentácia mračna bodov

Takáto segmentácia mračna bodov, nám na základe vytvorenia roviny rozdelí mračno na viacero mračien, ktoré sa nachádzajú v rovnakej rovine. Funkcia na detekciu roviny používa štandardný konsenzus náhodných vzoriek (RANSAC) (kapitola 1.2.4.1).



Obr. 20: rovinná segmentácia

3 Implementácia

V tejto kapitole budeme popisovať implementáciu upravovania mračna bodov, teda odstraňovanie vybraných bodov, filtrovanie pozadia, segmentáciu a exportovanie mračna z aplikácie v jednotlivých snímkach alebo celej sekvencie. Ďalej sa budeme zaoberať implementáciou upravovania kostry a jej ukladaním v jednotlivých snímkach alebo celej sekvencie. Rozoberieme aj aplikačnú architektúru, používateľské rozhranie a súborové formáty, s ktorými aplikácia pracuje.

3.1 Scéna aplikácie

V aplikácii sa používateľ môže pohybovať voľne do všetkých smerov pomocou WASD alebo šípok na klávesnici. Taktiež sa dá kamera používateľa otáčať okolo všetkých osí pomocou stlačenia pravého tlačidla na myši. Pomocou ľavého tlačidla sa dá obsluhovať používateľské rozhranie aplikácie. V scéne sa nachádzajú dva objekty na rozdielnych pozíciách ANIMblueprint a PCblueprint. Jeden objekt slúži na vykonávanie animácie, pričom druhý slúži na vykreslenie mračna bodov. Mračno bodov je možné pomocou používateľského rozhrania posunúť na pozíciu, kde sa vykonáva animácia, pre účel zlepšenia viditeľnosti synchronizácie nahraných dát alebo uľahčenia úpravy animácie podľa nahraného mračna.

3.2 Datasety

V tejto podkapitole si prejdeme spôsoby, akými sme získali požadované datasety.

3.2.1 3D skeny

Ako sme spomínali v návrhu aplikácie (kapitola 2.2), sken ľudského tela vo forme mračna bodov potrebujeme získať v súborovej forme xyz. Pomocou Kinect SDK (kapitola 1.2.2.2) získame skeny v požadovanej forme.

Kinect SDK poskytuje funkciu, ktorá nám povie, ktorý pixel na obrázku RGB zodpovedá konkrétnemu bodu na obrázku hĺbky. Tieto informácie si uložíme v globálnom poli *depthToRgbMap*. Konkrétne ukladáme stĺpec a riadok (t. j. súradnicu x a y) farebného pixelu v poradí pre každý pixel hĺbky.

Teraz, keď pracujeme s 3D údajmi, chceme si predstaviť hĺbkový frame (snímku) ako zhluk bodov v priestore a nie ako obrázok s rozmermi 640 x 480. Takže v našej funkcii *getDepthData* vyplníme dočasnú vyrovnávaciu pamäť (buffer) súradnicami každého bodu (namiesto hĺbky každého pixelu). To znamená, že vyrovnávacia pamäť, do ktorej prechádzame, musí mať veľkosť šírka*výška*3*sizeof(float) pre súradnice typu float.

```
// Global Variables
long depthToRgbMap[width*height*2];
// ...
void getDepthData(GLubyte* dest) {
// ...
        const USHORT* curr = (const USHORT*) LockedRect.pBits;
        float* fdest = (float*) dest;
        long* depth2rgb = (long*) depthToRgbMap;
        for (int j = 0; j < height; ++j) {
            for (int i = 0; i < width; ++i) {
                // Get depth of pixel in millimeters
                USHORT depth = NuiDepthPixelToDepth(*curr);
                // Store coordinates of the point corresponding to this pixel
                Vector4 pos = NuiTransformDepthImageToSkeleton(i,j,*curr);
                *fdest++ = pos.x/pos.w;
                *fdest++ = pos.y/pos.w;
                *fdest++ = pos.z/pos.w;
                // Store the index into the color array corresponding to this pixel
                NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution(
                    NUI_IMAGE_RESOLUTION_640x480, // color frame resolution
                    NUI_IMAGE_RESOLUTION_640x480, // depth frame resolution
                    NULL,
                    i, j, *curr,                  // Column, row, and depth in depth image
                    depth2rgb, depth2rgb+1        // Output: column and row (x,y) in the color image
                );
                depth2rgb += 2;
                *curr++;
            }
        }
// ...
```

Obr. 38: *getDepthData* kód

Vector4 je typ 3D bodu od spoločnosti Microsoft v homogénnych súradniciach (3D bod so súradnicami x, y, z.). *NuiTransformDepthImageToSkeleton* nám poskytuje 3D súradnice konkrétneho pixelu hĺbky. *NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution*

berie pixel hĺbky (riadok, stĺpec a hĺbku v hĺbkovom obrázku) a dáva riadok a stĺpec pixelu vo farebnom obrázku.

Teraz, keď používame body namiesto mriežok chceme, aby bol náš farebný výstup spojený s konkrétnym bodom hĺbky. Konkrétne, vstup do našej funkcie *getRgbData,* analogicky k funkcii *getDepthData*, vyžaduje vyrovnávaciu pamäť veľkosti šírka*výška*3*veľkosť(float) na uchovávanie hodnôt červenej, zelenej a modrej pre každý bod v našom point cloude [26].

```
void getRgbData(GLubyte* dest) {
// ...
        const BYTE* start = (const BYTE*) LockedRect.pBits;
        float* fdest = (float*) dest;
        long* depth2rgb = (long*) depthToRgbMap;
        for (int j = 0; j < height; ++j) {
            for (int i = 0; i < width; ++i) {
                // Determine color pixel for depth pixel i,j
                long x = *depth2rgb++;
                long y = *depth2rgb++;
                // If out of bounds, then do not color this pixel
                if (x < 0 || y < 0 || x > width || y > height) {
                    for (int n = 0; n < 3; ++n) *fdest++ = 0.f;
                }
                else {
                    // Determine rgb color for depth pixel (i,j) from color pixel (x,y)
                    const BYTE* color = start + (x+width*y)*4;
                    for (int n = 0; n < 3; ++n) *fdest++ = color[2-n]/255.f;
                }
            }
        }
// ...
```

Obr. 37: *getRgbData* kód

Snímka s farebným obrázkom je vo formáte BGRA, jeden bajt na kanál, usporiadaný po riadkoch. Takže lineárny index pre pixel (x, y) je x + šírka * y. Potom je požadovaný 4-bajtový blok na začiatku + lineárny index*4. Nakoniec chceme konvertovať z BGRA s hodnotou bajtov (0-255) na RGB s pohyblivou hodnotou (0,0-1,0) RGB, takže otočíme poradie bajtov a vydelíme číslom 255: farba[2-n]/255.f.

Súradnice všetkých bodov spolu s RGB hodnotami z jednotlivých snímok teraz môžeme uložiť do súborov. Sekvenciu snímok skenu postupne ukladáme ako 1 súbor = 1 snímka do súborovej štruktúry s našou aplikáciou. Dokopy sme nahrali 4 sekvencie po 50 snímok, teda 200 snímok skenov ľudského pohybu. Snímky sú uložené v súborovej štruktúre s našou aplikáciou v priečinku „Scans/Origin/“.

3.2.2 Transformácie kostry

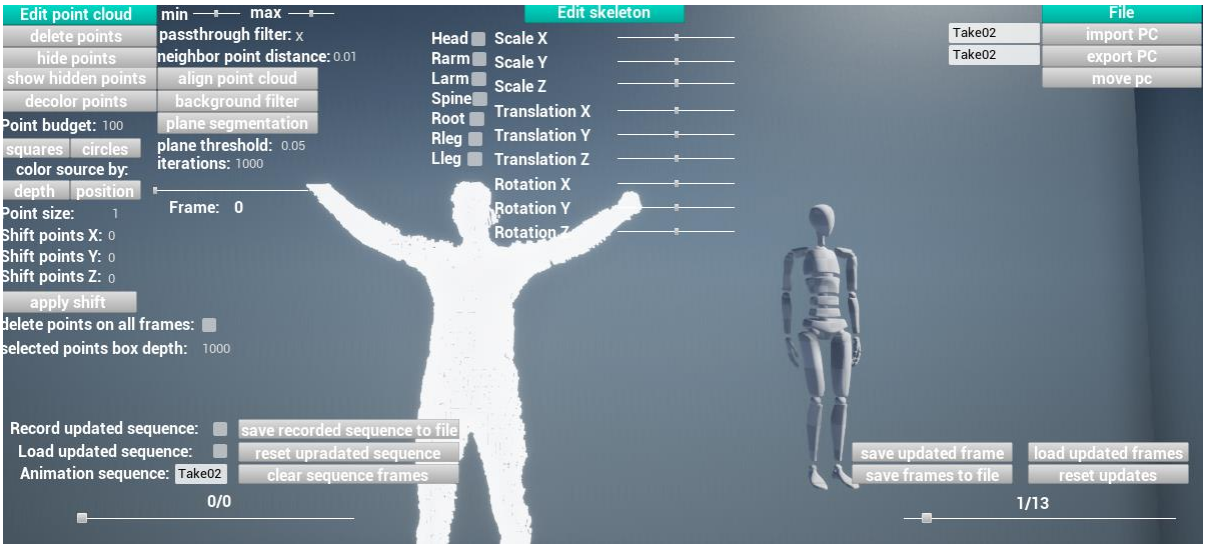
Ako sme spomínali v návrhu (kapitola 2.1.2) pohyb ľudského tela budeme nahrávať pomocou mocap obleku. Ten som si obliekol a vykonal rôzne pohyby, ktoré sa pomocou senzorov na obleku previedli do digitálnej reprezentácie kostry ktorá pohyby vykonáva transformovaním digitálnych bodov umiestnených v miestach, kde oblek obsahuje senzory. Tým vzniká digitálna animácia môjho pohybu v softvéri Rokoko Studio, z ktorého sekvenciu transformácií kostry (animáciu) exportujeme ako jeden súbor s formátom fbx. Dohromady sme nahrali štyri 50-sekundové sekvencie pohybov ľudského tela, tie sme následne uložili do súborovej štruktúry s našou aplikáciou v priečinku „RokokoDATA“.

Obr. 21: nahrávanie animácie

3.3 Používateľské rozhranie

Používateľské rozhranie je rozdelené do troch logických celkov: **File, Edit skeleton** a **Edit point cloud**. Každá časť po kliknutí ponúka úpravu rozličných dát a súborov. Časť **File** obsahuje tlačidla s funkcionalitami spojenými s ukladaním a načítavaním súborov. Po stlačení

dielca **Edit skeleton** sa zobrazia widgety, ktoré majú na starosti upravovanie kostry animácie. Zložka **Edit point cloud** obsahuje fragmenty na modifikáciu načítaného mračna bodov. Ak chceme upraviť mračno bodov, musíme ho najprv načítať do našej aplikácie cez komponent **File**. Všetky tri časti sa nachádzajú v hornej časti scény kvôli spriehľadneniu aplikácie.



Obr. 22: používateľské rozhranie

3.3.1 Práca so súbormi (File)

Po kliknutí na tlačidlo **File** na rozhraní aplikácie sa zobrazia tlačidla na prácu so súbormi, ktoré obsahujú zložky: **Import PC** a **Export PC**. Súbor fbx s animáciou do aplikácie importujeme pomocou Unreal editora, nakoľko voľne dostupné prostriedky Unreal Enginu nepodporujú načítanie fbx súborov počas behu aplikácie, preto načítavame fbx súbory pomocou editora. Existuje ale plugin ktorý umožňuje takéto načítanie súborov fbx, je ale spoplatnený v Unreal obchode sumou 50€.

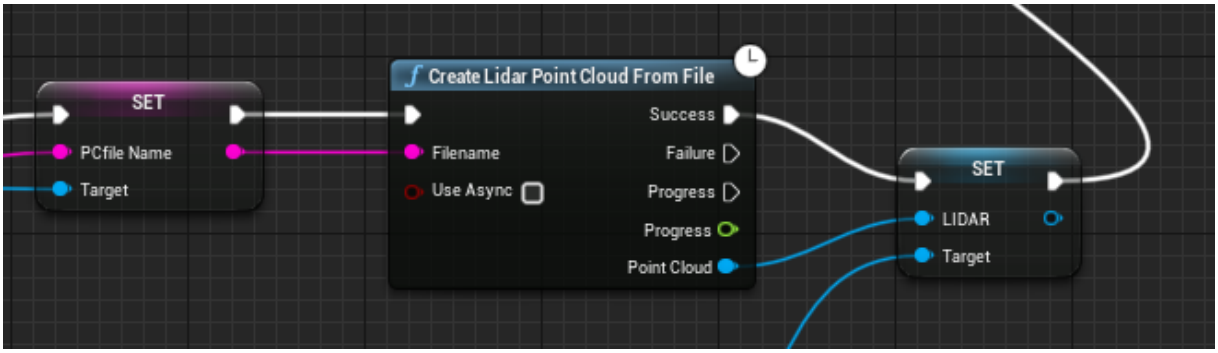
Po vložení súboru fbx do našej aplikácie sa vytvorí objekt s kostrou, fyzikou, animáciou a mesh-om (kapitola 1.2.1) ktorý animáciu vykonáva. Snímky jednotlivých frekvencií si môžeme prezerať pomocou slidera, pričom po vyvolaní na zmenu hodnoty slidera sa načíta súbor s informáciami o snímke s daným poradovým číslom v sekvencii. Oblek s ktorým sme pohyb človeka zaznamenávali, má frekvenciu nahrávania 100 snímok za sekundu (100 FPS). Skener, s ktorým skenujeme ľudské telo človeka vykonávajúceho pohyb, dokáže vytvoriť 10 snímok mračna bodov za sekundu (10 FPS). Na to, aby sme obe

sekvencie dokázali zosynchronizovať sme museli pre každých 10 snímok zaznamenaných oblekom zobraziť 1 snímku zo skenera.

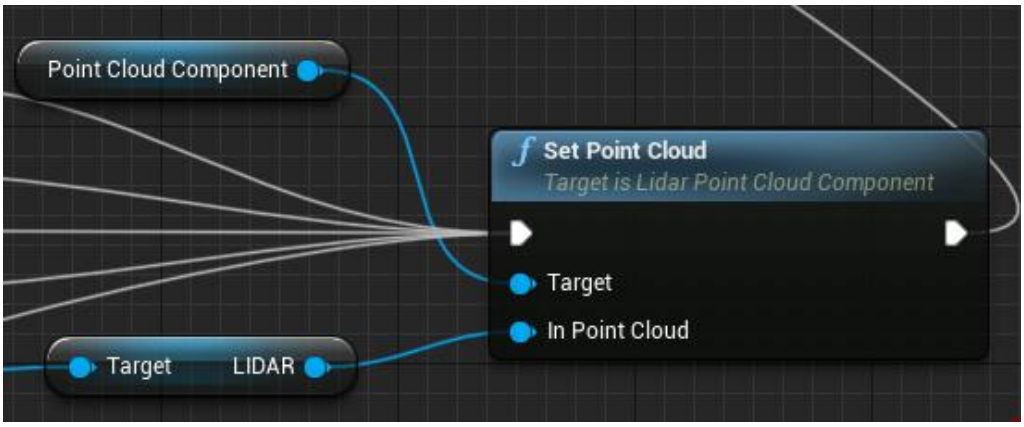
Obr. 23: používateľské rozhranie
(práca zo súbormi)

3.3.1.1 Tlačidlá na prácu so súbormi

Dáta budeme importovať do aplikácie zo súborového formátu xyz, no Unreal Engine samotný nepodporuje importovanie takéhoto formátu mračien bodov. Na importovanie takýchto súborových formátov si do aplikácie implementujeme LIDAR point cloud plugin (kapitola 1.3.2.1), ktorý nám importovanie xyz súborov umožní. Tento plugin obsahuje blueprinty na prácu s mračnami bodov. Taktiež budeme využívať VictoryBPlibrary plugin, ktorý rozširuje ponuku blueprintov o užitočné uzly. Súbor do komponentu objektu importujeme pomocou funkcie *CreatePointCloudFromFile*.



Obr. 24: blueprint *CreatePointCloudFromFile*



Obr. 25: blueprint nastavenia komponentu objektu PCblueprint

```
282  void ULidarPointCloudFileIO_ASCII::CreatePointCloudFromFile(UObject* WorldContextObject,
283      const FString& Filename, bool bUseAsync, FVector2D RGBRange,
284      FLidarPointCloudImportSettings_ASCII_Columns Columns,
285      FLatentActionInfo LatentInfo, ELidarPointCloudAsyncMode& AsyncMode,
286      float& Progress, ULidarPointCloud*& PointCloud)
287  {
288      TSharedPtr<FLidarPointCloudImportSettings_ASCII> ImportSettings =
289          MakeShared<FLidarPointCloudImportSettings_ASCII>(Filename);
290      ImportSettings->RGBRange = RGBRange;
291      ImportSettings->SelectedColumns = { FMath::Max(-1, Columns.LocationX),
292          FMath::Max(-1, Columns.LocationY), FMath::Max(-1, Columns.LocationZ),
293          FMath::Max(-1, Columns.Red),FMath::Max(-1, Columns.Green),
294          FMath::Max(-1, Columns.Blue), FMath::Max(-1, Columns.Intensity) };
295
296      ULidarPointCloudBlueprintLibrary::CreatePointCloudFromFile(WorldContextObject, Filename,
297          bUseAsync, LatentInfo, ImportSettings, AsyncMode, Progress, PointCloud);
298  }
```

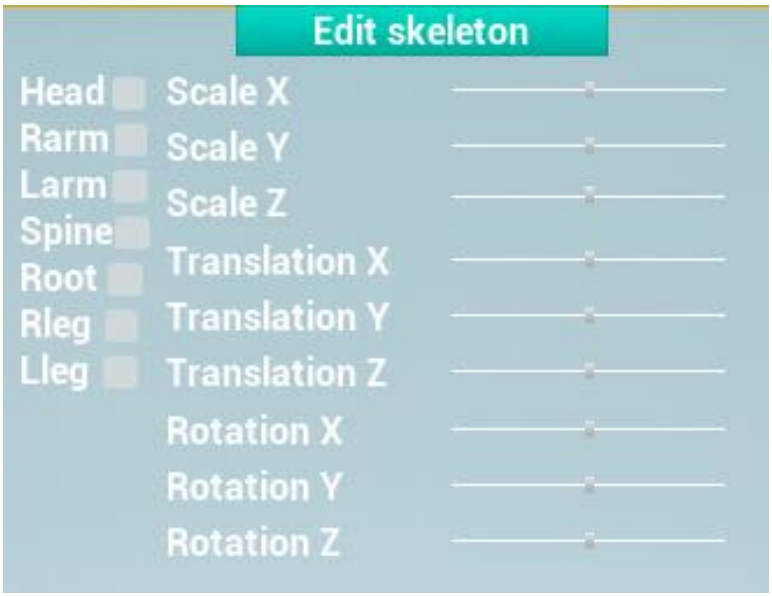
Obr. 37: CreatePointCloudFromFile funkcia blueprintu

Po vyplnení názvu súboru do textového poľa a stlačení klávesy Enter, za predpokladu, že je názov a formát súboru zadaný správne, sa načíta do objektu PCblueprint komponent s mračnom bodov zo súboru, objekt je v scéne načítaný. Súbor, ktorý chceme načítať sa musí nachádzať v súborovej štruktúre s aplikáciou v priečinku Scans. Po vyplnení názvu sekvencie do textového poľa a stlačení tlačidla **export PC**, sa upravené mračno so zadaným názvom uloží vo formáte xyz do súborovej štruktúry s našou aplikáciou do priečinka Scans.

Po prvom stlačení tlačidla **move PC** sa presunie objekt PCblueprint s namapovaným komponentom mračná bodov na pozíciu s kostrou, po druhom stlačení sa vráti objekt PCblueprint na pôvodnú pozíciu.

3.3.2 Upravovanie kostry (Edit skeleton)

V komponente ANIMblueprint je vložená animácia, fyzika, kostra a mesh zo súboru fbx, ktorý sme nahrali. Upravovanie kostry je implementované formou zaškrtnutia jedného alebo viacerých checkboxov, pre výber častí tela a výberu snímky pomocou slidera snímok kostry, v ktorej chceme kostru zmeniť. Ak máme vybratú časť alebo časti tela a snímku, kostru upravujeme pomocou sliderov v rozhraní. Aplikácia nám umožňuje meniť veľkosť častí tela pomocou posúvania kĺbov a rozťahovania kostí. Týmto spôsobom sa menia kĺby a kosti v celej hierarchii daného celku. Ďalej môžeme posúvať časti tela v smere osi pomocou zmeny polohy daného celku ku koreňu (root) kostry (kapitola 1.1.4.3). Rovnako môžeme upravovať aj rotáciu časti tela. Jednotlivé úpravy môžeme robiť pozdĺž všetkých osí.

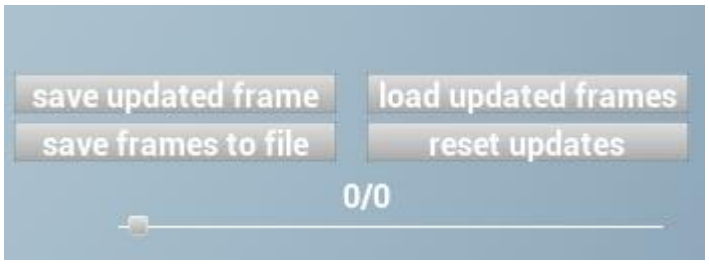


Obr. 26: používateľské rozhranie
(upravovanie kostry)

3.3.2.1 Tlačidlá na úpravu kostry

Po stlačení tlačidla **Save updated frame** sa uloží nastavenie úpravy časti kostry v snímke, ktorú vyberieme pomocou nastavenia hodnoty na slideri snímok kostry, do poľa. To nám umožňuje uložiť viacero snímok s rôznou úpravou. Stlačením tlačidla **Reset updated frame** sa vymažú údaje uložené v poli. Po vykonaní akcie na tlačidlo **Save updated frames to file**

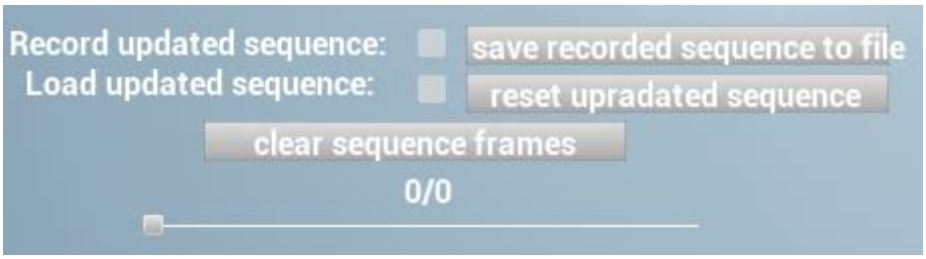
sa uložia údaje o úprave kostry v daných snímkach do súboru AnimationData v súborovej štruktúre našej aplikácie. Súbor má štruktúru: hodnota slidera, predĺženie X, predĺženie Y, predĺženie Z, posun X, posun Y, posun Z, rotácia X, rotácia Y, rotácia Z. Zo súboru AnimationData v štruktúre našej aplikácie sa po vyvolaní akcie na tlačidlo **Load updated frames** načítajú zvolené snímky častí tela do poľa v našej aplikácii. Z ktorých slider snímok kostry podľa svojej hodnoty načítava snímky animácie do ANIMblueprint objektu, ktorý vykoná animáciu s upravenou snímkou. Po vykonaní akcie na tlačidlo **Save recorded sequence to file** sa uložia informácie z poľa funkcie **Record updated sequence** do súboru AnimationData/Sequence v súborovej štruktúre našej aplikácie. Súbor má štruktúru: hodnota slidera, predĺženie X, predĺženie Y, predĺženie Z, posun X, posun Y, posun Z, rotácia X, rotácia Y, rotácia Z.



Obr. 27: používateľské rozhranie
(upravovanie snímok sekvencie kostry)

3.3.2.2 Checkboxy na úpravu kostry

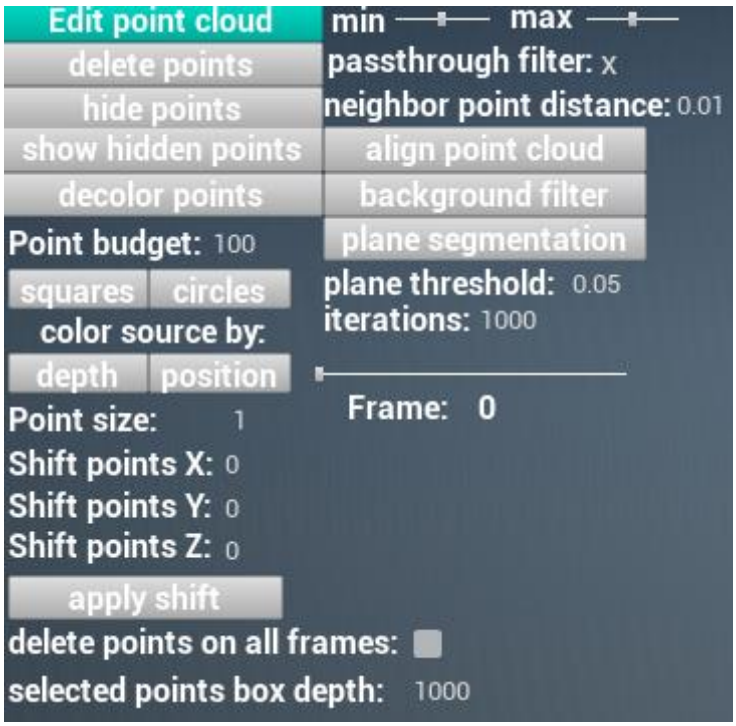
Po zaškrtnutí checkboxu a upravení časti kostry, sa po zmene hodnoty na slideri snímok kostry, uloží zmena kostry do poľa. To nám umožňuje rýchlo uložiť zmenenú kostru vo viacerých snímkach animačnej sekvencie. Nahranú zmenu kostry vieme tlačidlom **Clear recorded sequence** vymazať. Po zaškrtnutí checkboxu **Load updated sequence** a výbere časti tela, sa akciou na zmenu hodnoty slidera snímok kostry, načíta daná úprava časti kostry v danej snímke. To nám umožňuje rýchle načítanie uložených upravených snímok do aplikácie. Načítanú zmenu kostry vieme tlačidlom **Reset updated sequence** vymazať.



Obr. 28: používateľské rozhranie
(upravovanie sekvencie kostry)

3.3.3 Upravovanie mračna bodov (Edit point cloud)

Ak máme v aplikácii importované mračno bodov, vybraním konkrétnej snímky na slideri snímok PC alebo importovaním PC, teda objekt PCblueprint obsahuje neprázdny komponent s mračnom, sa v aplikácii pomocou pohybu kamery dostaneme do tesnej blízkosti daného bodu, pomocou klávesy Shift označíme stred, okolo ktorého chceme vyznačiť rovinu a pomocou ľavého tlačidla na myši označíme veľkosť roviny v ktorej chceme odstrániť body. Hĺbku v dane rovine vieme podľa potreby zmeniť vpísaním hodnoty do textového poľa.
Body, ktoré sa nachádzajú v daných rozmeroch, okolo zadaného stredu sa zvýraznia farbou, tým sú označené. Označené body sa dajú podľa potreby ďalej upravovať.



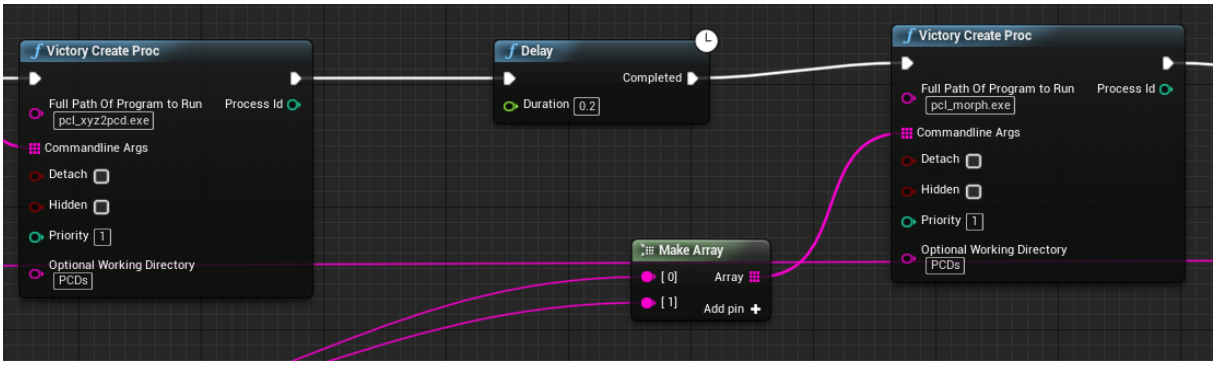
Obr. 29: používateľské rozhranie (upravovanie mračna bodov)

3.3.3.1 Tlačidlá na úpravu mračien bodov

Po stlačení tlačidla **Delete points** sa odstránia označené body v mračne bodov v určitej snímke. Odstránené body sa po vyvolaní uloženia neuložia do súboru. Ak je zaškrtnutý checkbox **delete points on all frames**, za predpokladu že sa v snímkach nachádzajú dané body, odstránia sa zo všetkých snímok sekvencie. Tlačidlo **Hide points** zneviditeľní vyznačené body vo vyznačenej snímke, **Show hidden points** zviditeľní všetky zneviditeľnené body. **Color source by: Depth/Position** nastaví hodnotu farieb jednotlivých bodov mračna podľa posunutia bodov od začiatočnej pozície. Posun mračna vieme aplikovať pomocou **Apply shift**, kde sa mračno posunie podľa hodnôt zadaných do textových polí.

3.3.4 Algoritmy PCL knižnice

Po získaní kódu z repozitáru github a následnom skompilovaní, som vytvoril spúšťateľné .exe súbory a dynamicky linkované .dll knižnice. Tie som následne pomocou funkcií poskytnutých pluginom VictoryBP, spustil cez príkazový riadok spolu so správnymi vstupnými argumentmi.



Obr. 30: blueprint z plugin VictoryBP

```
void UVictoryBPFunctionLibrary::VictoryCreateProc(int32& ProcessId, FString FullPathOfProgramToRun,TArray<FString>
CommandlineArgs,bool Detach,bool Hidden, int32 Priority, FString OptionalWorkingDirectory)
{

    FString Args = "";
    if(CommandlineArgs.Num() > 1)
    {
        Args = CommandlineArgs[0];
        for(int32 v = 1; v < CommandlineArgs.Num(); v++)
        {
            Args += " " + CommandlineArgs[v];
        }
    }
    else if(CommandlineArgs.Num() > 0)
    {
        Args = CommandlineArgs[0];
    }

    uint32 NeedBPUINT32 = 0;
    FProcHandle ProcHandle = FPlatformProcess::CreateProc(
        *FullPathOfProgramToRun,
        *Args,
        Detach,//* @param bLaunchDetached       if true, the new process will have its own window
        false,//* @param bLaunchHidded          if true, the new process will be minimized in the task bar
        Hidden,//* @param bLaunchReallyHidden   if true, the new process will not have a window or be in the task bar
        &NeedBPUINT32,
        Priority,
        (OptionalWorkingDirectory != "") ? *OptionalWorkingDirectory : nullptr,//const TCHAR* OptionalWorkingDirectory,
        nullptr
    );


    ProcessId = NeedBPUINT32;
}
```

3.3.4.1 Passthrough filter

Po označení snímky, ktorú chceme funkciou upraviť sa v aplikácii načíta súbor s formátom xyz z priečinka Scans. Ten pomocou PCL funkcie prekonvertujeme na pcd formát a uložíme do priečinka PCDs. Pcd súbor načítame do objektu a na ten aplikujeme funkciu filtra.

```
// Create the filtering object
pcl::PassThrough<pcl::PointXYZ> pass;
pass.setInputCloud({ vybraná PC snímka });
pass.setFilterFieldName({ vstupná hodnota z text boxu });
pass.setFilterLimits({ vstupná hodnota zo slidera minimum }, { vstupná hodnota zo slidera maximum });
//pass.setFilterLimitsNegative (true);
pass.filter(*cloud_filtered);
```

Následne upravený súbor s mračnom cloud_filtered prekonvertujeme späť na xyz formát do priečinka Scans pod pôvodným názvom, aby sme ho mohli opäť sliderom v sekvencii načítať.

3.3.4.2 Background filter

Po vybratí snímky a vyvolaní akcie na tlačidlo sa načíta snímka vo forme pcd do objektu.

```
pcl::PCDReader reader;
reader.read<pcl::PointXYZ>("{vybraná snímka}", *cloud);
Na vybranú snímku aplikujeme funkciu odstránenia pozadia.
pcl::StatisticalOutlierRemoval<pcl::PointXYZ> sor;
```

Počet susedov na analýzu pre každý bod je nastaviteľný v textovom poli aplikácie.
To znamená, že všetky body, ktorých vzdialenosť je väčšia ako zadaná hodnota odchýlky strednej vzdialenosti od bodu, budú označené ako odľahlé hodnoty a odstránené.

```
sor.setStddevMulThresh({ zadaná hodnota });
sor.filter(*cloud_filtered);
pcl::PCDWriter writer;
writer.write<pcl::PointXYZ>("{vybraná snímka}", *cloud_filtered, false);
```

Výsledné mračno potom vložíme do pôvodného súboru s formátom xyz.

3.3.4.3 Plane segmentation

Po zvolení snímky na úpravu vytvoríme objekt.

```
pcl::SACSegmentation<pcl::PointXYZ> seg;
reader.read<pcl::PointXYZ>("{vybraná snímka}", *cloud);
seg.setInputCloud(cloud);
```

Do textového poľa vpíšeme a potvrdíme hodnotu prahu vzdialenosti, ktorý určuje, ako blízko musí byť bod k modelu roviny, aby bol považovaný za vnútorný bod. Následne sa pomocou metódy RANSAC (kapitola 1.2.4.1), vytvorí robustný výberový odhad. Naše rozhodnutie je motivované jednoduchosťou RANSAC (iné robustné odhady ho používajú ako základ a pridávajú ďalšie, komplikovanejšie koncepty).

```
seg.setDistanceThreshold({ zadaná hodnota prahu });
seg.setModelType(pcl::SACMODEL_PLANE);
seg.setMethodType(pcl::SAC_RANSAC);
seg.setInputCloud(cloud);
seg.setIterationLimit({ zadaná hodnota iterácií });
```

Do druhého textového poľa vložíme hodnotu, ktorá nám udáva počet iterácií metódy RANSAC, ak vložíme príliš veľké číslo iterácií, program nestihne vykonať všetky iterácie a zamrzne. Po vysegmentovaní mračna sa body, ktoré neboli v blízkosti roviny vymažú a do pôvodného súboru sa uloží zmenené mračno.

4 Testovanie

Zariadenia, na ktorých som vykonával testy, majú parametre:

Zariadenie 1 (notebook):

- Procesor Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz
- RAM 16.0 GB (15.9 GB usable)
- Operačný systém Windows 10 Home
- Úložisko SSD
- Grafická karta Nvidia Geforce GTX 1060

Zariadenie 2 (stolný počítač):

- Procesor Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz 3.19 GHz
- RAM 16.0 GB
- Operačný systém Windows 10 Pro
- Úložisko SSD
- Grafická karta Nvidia Geforce GTX 1070 Ti

Nahrávanie a úprava datasetov je pomerne rýchle. Problémy s rýchlosťou načítania mračien bodov nastávajú, až keď mračno obsahuje veľké množstvo bodov (and 100000). Tejto situácii je ale možné predísť použitím funkcie point budget v našej aplikácii. Pri načítavaní upravenej sekvencie s kostrou, za zvolenia viacerých častí tela, nastáva problém s načítaním pri zariadeniach s nižšou výpočtovou silou. V tomto prípade je odporúčané načítanie častí tela vo frekvencii jednotlivo. Veľké zaťaženie na výpočtovú silu môže nastať aj pri zvolení vysokého počtu iterácií pri funkciách využívajúcich algoritmus RANSAC. Funkcia sa však po dlhšom čase vykoná. Ostatné funkcionality sa vykonali na testovacích zariadeniach bez problémov.

Záver

Vypracovanie tejto práce nám prinieslo užitočné znalosti v oblasti počítačovej grafiky.
Vo vytvorenej aplikácii sa nám podarilo využiť veľa nástrojov na úpravu 3D objektov vo forme mračna bodov. Existujú ale aj ďalšie prístupy, ktorých implementácia nebola tak podstatná pre splnenie cieľov našej práce. Ciele tejto práce boli rozdelené na hardvérovú a softvérovú časť.

Hardvérová časť práce zahŕňala získanie dát z nahratého pohybového vystúpenia pred 3D skenerom s oblečeným inerčným oblekom. To nám umožnilo bližšie spoznať technológie, s ktorými sme pracovali, ale aj princípy ich fungovania. Inerčný oblek, s ktorým sme pracovali, patrí medzi vlajkové lode v technológii zachytávania pohybu ľudského tela, teda v odvetví, ktoré ma rýchly rozvoj a veľký potenciál.

Softvérová časť pozostávala z vytvorenia aplikácie na úpravu a synchronizáciu nahraných dát v softvéri určeného na prácu s 3D objektmi. Pomocou nástrojov dostupných v hernom engine Unreal sme vytvorili spôsob vizualizácie nahraných dát. Výsledná aplikácia dokáže počas behu načítať synchronizovanú sekvenciu nahraných mračien bodov a animácie, taktiež poskytuje užívateľovi plynulý pohyb okolo nahraných objektov. Používateľ tak môže naplno využiť nástroje aplikačného rozhrania na úpravu objektov, či už pomocou využitia predvolených hodnôt alebo pomocou zadania hodnoty pre rôzne algoritmické úpravy.
Verím, že oba ciele sa nám podarilo úspešne splniť a poradili sme si s nástrahami, s ktorými sme sa vo vývojovom prostredí stretli. Vypracovaním tejto práce som získal veľa nových skúseností s prácou v Unreal Engine za použitia blueprintov a C++. Taktiež som získal veľa poznatkov o 3D objektoch a počítačovej animácii. Prínos tejto aplikácie vidím v jej možnostiach úpravy dát, pretože získané dáta z oboch hardvérov potrebujú upraviť pred použitím v praxi, čo náš softvér umožňuje.

Aplikácia má ale priestor na zlepšenie. Implementovanie systému na importovanie súborov s formátom fbx počas behu aplikácie, by bolo veľkým prínosom pre jej funkcionalitu. Taktiež vytvorenie vizuálnej reprezentácie kostry s možnosťou označenia jej konkrétnej časti, ktorú chceme upraviť, by zlepšilo vizuálnu stránku aplikácie. Pridávanie nových bodov do mračna na požadovanú pozíciu, by tiež mohlo prispieť k zlepšeniu používateľského dojmu z aplikácie.

Použitá literatúra

[1] Ramakrishnan Mukundan. 3D Mesh Processing and Character Animation, New Zealand

[2] Motion Capture. https://assistiverobotcenter.github.io/projects/2019/06/19/sensors-paper5 [Online, 30.5.2021]

[3] Charles Pao. What is an IMU Sensor?: https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/ [Online, 30.5.2021]

[4] Luis Bermudez. Overview of Inverse Kinematics: https://medium.com/unity3danimation/overview-of-inverse-kinematics-9769a43ba956 [Online, 30.5.2021]

[5] Faxin Yu, Zheming Lu, Hao Luo, Pinghui Wang. Three-Dimensional Model Analysis and Processing, China

[6] Miles Hansard, Seungkyu Lee, Ouk Choi, Radu Patrice Horaud. Time-of-Flight Cameras: Principles, Methods and Applications. South Korea

[7] Tech27 Systems. What are point clouds?: https://tech27.com/resources/point-clouds/ [Online, 30.5.2021]

[8] Point Clouds group. Point cloud library: https://pointclouds.org/ [Online, 30.5.2021]

[9] Jixian Zhang, Xiangguo Lin, Guoqing Zhou, Prasad S. Thenkabail. An Improved RANSAC for 3D Point Cloud Plane Segmentation Based on Normal Distribution Transformation Cells. 2017

[10] Unreal engine documentation. Introduction to Blueprints: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/GettingStarted/ [Online, 30.5.2021]

[11] Sufyan bin Uzayr. Mastering Unreal Engine: A Beginner's Guide. 2022

[12] Xin Min, Shouqian Sun, Honglie Wang, Xurui Zhang, Chao Li, Xianfu Zhang. Motion Capture Research: 3D Human Pose Recovery Based on RGB Video Sequences. 2019

[13] Unreal engine Documentation. Plugins: https://docs.unrealengine.com/4.27/en-US/ProductionPipelines/Plugins/ [Online, 30.5.2021]

[15] [14] Alberto Menache. Understanding Motion Capture for Computer Animation. 2011

[16] Gerald F. Marshall, Glenn E. Stutz, Handbook of Optical and Laser Scanning.

[17] Pedro Nogueira. Motion Capture Fundamentals. 2011

[18] Michael Lee Gleicher, Nicola Ferrier. Evaluating video-based motion capture. 2002

[19] Joanna Lee. Learning Unreal Engine Game Development. 2016

[20] iD Tech. C++ vs. Blueprints: pros and cons, which should be used, and when?: https://www.idtech.com/blog/c-vs-blueprints-differences

[21] Alex Forsythe. Blueprints vs. C++: http://awforsythe.com/unreal/blueprints_vs_cpp/

[22] Pascal Mueller. High-End 3D Visualization with CityEngine, Unity and Unreal. 2018

[23] Rahul Raguram, Jan-Michael Frahm, Marc Pollefeys. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. 2008

[24] Qi Chen, Peng Gong, Dennis Baldocchi, and Gengxin Xie. Filtering airborne laser scanning data with morphological methods. 2007

[25] Simon Griffioen. 3D mathematical morphology on voxelized point clouds for outlier detection. 2017

[26] Kinect SDK C++ - 3. Kinect Point Clouds: https://ed.ilogues.com/Tutorials/kinect/kinect3.html

Point Clouds group. The PCD (Point Cloud Data) file format: https://pcl.readthedocs.io/projects/tutorials/en/latest/pcd_file_format.html

Josh Petty. What is 3D modeling: https://conceptartempire.com/what-is-3d-modeling/ [Online, 30.5.2021]

Keqi Zhang, Shu-Ching Chen, Dean Whitman, Mei-Ling Shyu, Member, Jianhua Yan, and Chengcui Zhang. A Progressive Morphological Filter for Removing Nonground Measurements From Airborne LIDAR Data. 2003