TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ragnar Luga 155207IAPB

VIRTUAALSE KEHA JUHTIMINE KINECTI SENSORI JA UNITY ABIL

Bakalaurusetöö

Juhendajad: Sven Nõmm
PhD
Jaagup Irve
MSc

Tallinn 2018

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Ragnar Luga

23.05.2018

Annotatsioon

Käesoleva töö eesmärk on juhtida inimeste 3D mudeleid virtuaalkeskonnas, kusjuures sisendiks on nende liigutused päris maailmas. Inimeste liigutuste mõõtmiseks kasutatakse Kinect sensorit ning virtuaalkeskkond luuakse Unity mängumootoris. Töö idee on ajendatud Von Krahli teatri soovist rakendada sensorit lavaetenduses olevate näitlejate jälgimiseks, et luua teatrietendusse virtuaalne lisakiht. Töö tulemusena on valminud reaalajas töötav süsteem, mis on võimeline jälgima ja animeerima kokku kuni kuut inimest. Müra ning häiringute jaoks on andmeid vastavalt filtreeritud ning süsteemi operaatorile on loodud kasutajaliides, mille abil on võimalik jälgida süsteemi hetkeseisu. Funktsionaalne kasutajaliides võimaldab operaatoril vahetada iga avatari 3D mudelit, jälgida nende hetke seisundit ning vastavalt soovile peita neid stseenist.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 7 peatükki, 20 joonist, 0 tabelit.

Abstract

Controlling virtual bodies using Kinect sensor and Unity

The main goal of the thesis is to control the 3D models of real persons in a virtual environment according to the movements the real persons. A Kinect sensor is used for the tracking of the movements. The idea for the use of the sensor came from the Von Krahl theatre who want to use the sensor in their theatrical performances to track actors in order to create a virtual layer. As a result a system was made in Unity game engine that is capable of tracking and animating actors in real time up to six persons. In order to cancel out noises in data, a filter has been used to smooth the data coming from the sensor. Lastly, an user interface was made for the operator of the system in order have control over the avatars and to keep track of their state. The functional user interface allows the operator to change models of the avatars, keep track of their state and when needed, hide them from the scene.

The thesis is in Estonian and contains 33 pages of text, 7 chapters, 20 figures, 0 tables.

Lühendite ja mõistete loetelu

3D mudel	Kolmedimensionaalne virtuaalne kujutis esemest või olendist.
Avatar	Päris inimese virtuaalne kasutajatunnus. Näiteks 3D mudel või pilt.
Armatuur	3D mudelis paiknev inimese skeletiga sarnanev üksus, mis sisaldab luid ning mille abil on võimalik 3D mudelit liigutada.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Käesolev töö pakub välja uue Kinecti kasutusvaldkonna teatrietendustes. Von Krahli teater on avaldanud soovi luua kahekihilist etendust, kus samal ajal on laval näha tavapärast etendust ning ekraanist on võimalik vaadata muundatud varianti, milles näitlejad on asendatud 3D mudelitega (edaspidi avatar), kuid endiselt liiguvad nagu näitlejad. Selline kasutusvaldkond on unikaalne, sest sellega tekib infotehnoloogial täiesti oma koht teatrivaldkonnas ning tulemusena sünnib uus teatrivorm.

Microsoft tõi 2010. aastal avalikkuse ette esimese Kinecti sensori, mis muutus lühikese ajaga hästi populaarseks. Sensor oli mõeldud kasutamiseks koos Xbox mängukonsooliga ning võimaldas žestide ning hääle tuvastamist. Peatselt hakkas Xboxile ilmuma mitmeid mänge, mis kasutavad inimese tehtavaid žeste või häält oma mänguloogika sisendiks. Kinecti sensori võlu peitub tema lihtsuses. Nimelt ei vaja sensori inimese tuvastamiseks eraldi markeeringut ning samuti arvutab Kinect ise välja inimese erinevate kehaosade asukohad ja orientatsioonid. Selle tõttu on Kinecti kasutatud teistes valdkondades nagu meditsiin või viipekeele tõlkimine.

Sarnaseid lahendusi on eelnevalt Kinectiga tehtud, kus sensorit kasutatakse inimese liikumise salvestamiseks ning arvutis virtuaalse keha ehk avatari liigutamiseks, ent enamasti on need loodud sensori võimaluste demonstreerimiseks ning ei ole leidnud laiemat rakendust. Samuti leidub Unity Asset Store's valmislahendus nii esimese kui teise versiooni Kinect sensorite jaoks, kusjuures esimese versiooni lahendus on tasuta ning teine versioon maksab 22 dollarit [1] [2]. Bakalaurusetöö lahendus kasutab samamoodi viimast versiooni Kinect sensorist, ent erineb sellepoolest, et võimaldab reaalajas mudelite vahetamist, kuni kuue inimese samaaegset kuvamist avataridena ning ülevaadet operaatorile, kes teatrietenduse ajal süsteemi jälgib.

Bakalaurusetöö teises peatükis räägitakse Kinecti eelnevatest rakendustest ja kolmandas konkreetse bakalaurusetöö uudsusest. Neljandas peatükis püstitatakse lahendatavad probleemid, viiendas peatükis kirjutatakse metoodikast ja kasutatud tehnoloogiast ning kuuendas peatükis räägitakse lahendusest.

2 Kinect sensori rakendused

Kinect sensorit on tänu oma omadustele kasutatud mitmetes teadustöödes. 2013. aastal ilmunud teadusartiklis pakuti välja alternatiivne lähenemine inimese motoorikas õppimise käigus toimuvate muudatuste mõõtmiseks ja kirjeldamiseks. Liigutuste analüüsi ja žestide tuvastamise jaoks kasutatavad meetodid põhinevad tänapäeval enamasti liigutustunnuste eristamisel, liigutusmustrite tuvastamisel ja klasterdamisel. Alternatiivne meetod tulemuste kirjeldamiseks jäsemete liigutuste hulka ning nende sujuvust. Andmete kogumiseks kasutati Kinect sensorit, millega mõõdeti palliga sihtmärgi pihta viskamise õppeprotsessi. Katsete tulemustest järeldati, et uudne meetod võib olla laialt rakendatav erinevate liigutuste õppimisel, sealhulgas sportlaste treenimisel ja ajukahjustuse läbi teinud inimeste motoorsete võimete rehabilitatsioonis [3].

Samasugust lähenemist kasutati 2016. ilmunud teadustöös Parkinsoni tõve tuvastamiseks. Mõõdetava näitajana kasutati eelmises artiklis tutvustatud liigutuste massi (motion mass), mis kirjeldab kindla liigutuste tsükli käigus tehtud liigutuste hulka eukleidilises ruumis. Artikli sisuks oli näidata, et alternatiivse lähenemise puhul kasutatud näitaja erineb selgelt tervete ja Parkinsoni tõvega inimeste vahel [4].

Lisaks teadustöödele on sensorit kasutatud meditsiini valdkonnas kirurgide abistamiseks. Kirurgid peavad tavaliselt operatsiooni käigus jälgima mitmesuguseid pilte või teisi visuaalseid informatsiooniallikaid, mida aitavad kindlad inimesed ekraanil kuvada. Olukorrad, kus kirurgil on vaja teha kiireid otsuseid, võivad tekitada tal stressi, sest tal puudub otsene kontroll piltide kuvamise üle. Artiklis testitud süsteem kasutas Kinect sensorit žestide tuvastamiseks ja nende abil graafiliste piltide kuvamiseks. Katsetes osalenud kirurgid olid väga rahul süsteemi tööga [5].

2.1 Töö erinevus

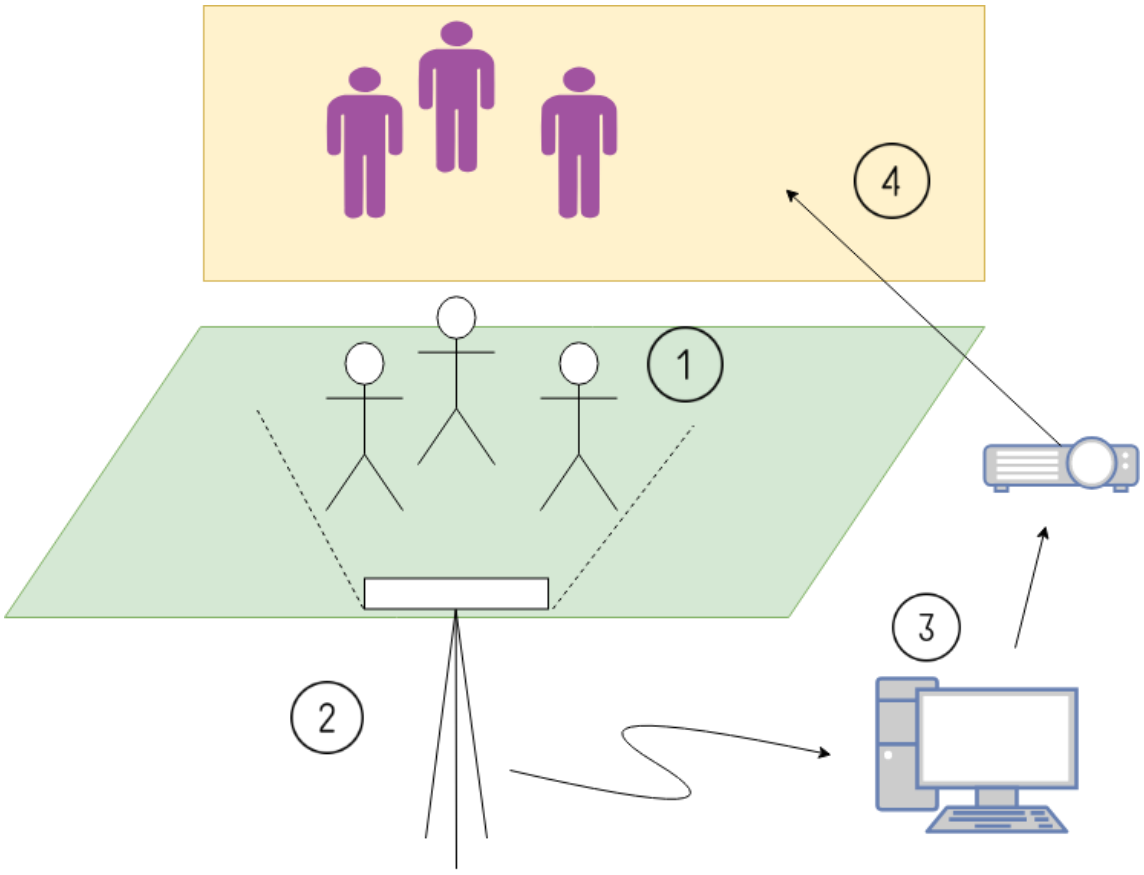
Bakalaurusetöö poolt pakutav lahendus valmib Unity mängumootoris ning kasutab Microsoft Kinect sensorit vajaliku info kättesaamiseks, mille abil animeeritakse samal ajal kokku kuni kuut avatari. Samal ajal on võimalik süsteemi operaatoril jälgida etenduse ajal töökäiku ning vajadusel muuta avatare või neid ära kaotada. Avataride ehk 3D mudelite jaoks kasutatakse tasuta avatud lähtekoodiga 3D modelleerimistarkvara Blender, mille abil luuakse mudelitele armatuurid. Armatuuri abil on võimalik anda Unity mängumootoris 3D mudeli vastavatele kehaosadel tema asukoha ja orientatsiooni.

Töö unikaalsus avaldub oma lihtsuse, kasutusvaldkonna eripära ning edasiarendusvõimaluste tõttu. Kinecti sensorit on senini rohkem kasutatud žestide kasutamiseks kui tervete avataride animeerimiseks. Samuti pole sensoril olnud teatrilavastustes suuremat kohta, kuid arvestades rakendatavat kasutusvaldkonda muudab Kinecti eriliseks ka asjaolu, et ta ei vaja inimese kehal täiendavaid spetsiaalseid riietusesemeid. Teatri seisukohast on see tähtis, sest nii ei piirata nende rõivaste valikuid. Praegune lahendus ei keskendu hetkel ümbruskonna peale ehk puuduvad igasugused lisakaunistused nagu puud, põõsad, lauad või seinad. Samuti ei eksisteeri hetkel avataridel võimalust virtuaalse maailmaga suhelda nagu näiteks esemeid tõsta või midagi lüüa. Eelnimetatud puudused on tegelikkuses realiseeritavad, sest neid kõike on võimalik Unity mängumootoris hõlpsalt lahendada. Täiendavalt on võimalik tulevikus arendada süsteemi häälkäsklustega, sest Kinecti sensor sisaldab mikrofone, millega saab salvestada inimese häält.

3 Probleemi püstitus

Käesoleva bakalaurusetöö idee on kasutada Kinect sensorit lavaetendustes ja sellega luua etendustesse virtuaalne lisakiht, võimendades vaatamiskogemust ning lisades näitemängudesse tänapäevast infotehnoloogiat. Täiendavalt kujutatakse tulevikus ette, et näitemängudes sensoriga saadud andmete abil tehakse näidenditest valmis eraldi filmid, mida on võimalik külastajatel vaadata siis, kui aktiivne etenduste hooaeg on lõppenud.

Lisatav kiht kujutaks endast teleekraanis või lava tagaseinal kuvatavat videopilti, kus näitlejad oleksid asendatud avataridega. Sensori abiga suudaksid avatarid järgida inimeste liigutusi ja teostada žestide ning häälkäskluste abil erinevaid animatsioone või tegevusi. Bakalaurusetöös kuulub lahendamisele näitlejate kuvamine videopildis avataridena.



Joonis 1 Kinecti sensori kasutamine lavaetenduses

Joonis 1 kirjeldab, kuidas kujutatakse ette Kinecti sensori kasutamist lavaetenduses. Sensoriga (2) filmitakse laval toimuvat näitemängu (1) ning Kinectist saaduid andmeid töödeldakse reaalajas temaga ühendatud arvutis (3). Töödeldud pilt kuvatakse prožektori abiga (4) lava tagumisele ekraanile. Kõige efektiivsemaks töötamiseks peab Kinect olema asetatud horisontaalselt maapinnaga ning olema lava ees keskel. Lava peab olema hästi valgustatud, et kaamera suudaks võimalikult täpselt tuvastada näitlejate kehaosasi ning nende liigutusi. Sensori vaatenurk seab samuti piirangud näitlejate poolt kasutatavale ruumile. Kinecti spetsifikatsioonis on kirjas, et sensor on võimeline tuvastama inimesi vahemikus 0.8 kuni 4 meetrit [6]. Sensori horistonaalseks vaatenurgaks on 70 kraadi. Sõltuvalt oludest võib sensori poolt jälgitavad ala muutuda.

Eesmärgist lähtuvalt on lahendatava ülesande põhiprobleemiks Kinecti sensorist saadava info abil Unity keskkonnas avataride animeerimine. Ülesande täitmiseks tervikuna jaotatakse põhiprobleem alamprobleemideks ning selgitatakse lühidalt, kuidas neid tuleks lahendada. Valdaval osa ülesannetest lahendatakse koodiskriptide kirjutamisega, ent mõningad ülesanded vajavad ka programmeerimise välist lahendust. Järgnevalt selgitatakse alamülesandeid.

1) Unity keskkonnas ligipääsu tagamine Kinecti sensori infole (programmeerimine).
    a) Kirjutada valmis vajalikud klassid sensoriga suhtlemiseks.
    b) Edastada vajalikest allikatest kätte saadud andmed edasi järgmistele klassidele.
2) 3D mudeli armatuuri valmistamine (modelleerimine).
    a) Mudelile tuleb valmistada armatuur, mis vastaks Kinecti poolt jälgitavate kehaosadega.
3) Avatari animeerimine (programmeerimine).
    a) Andmetes olevad kehaliigeste asukohaandmed peab üle kandma avatarile.
    b) Andmetes olevad kehaliigeste orientatsioonid peab üle kandma avatarile.
4) Liigeste liikumiste silumine (programmeerimine).

a) Liikumiste silumiseks tuleb kirjutada koodijupp, mis kasutab kindlat metoodikat liikumiste silumiseks.

5) Süsteemi mõistliku käitumise tagamine, kui näitleja kaob kaamera näiteväljast või liigub teise näitleja tagant läbi (programmeerimine).

   a) Avatari peitmine ekraanilt kui näitleja kaob sensori vaateväljast või liigub teise näitleja taha.

   b) Avatari näitamine ekraanil kui näitleja ilmub tagasi sensori vaatevälja.

6) Kasutajaliidese loomine süsteemi operaatori jaoks (programmeerimine).

   a) Kasutajaliides peab võimaldama jälgida, milliseid avatare hetkel animeeritakse.

   b) Kasutajaliides võimaldab operaatoril muuta avatari poolt kasutatavat 3D mudelit.

   c) Kasutajaliidese abil on operaatoril võimalik vajadusel peita avatare.

   d) Kasutajaliidese abil saab operaator kuvada ja peita lihtsustatud vaadet avataridest.

4 Metoodika ja tehnoloogia

Kolmandas peatükis nimetati lühidalt käesolevas töös kasutatuid vahendid: Kinect sensor, Unity mängumootor ja Blenderi 3D modelleerimistarkvara. Järgnevalt kirjeldatakse kasutatuid tehnoloogiaid ja tööriiste ning selgitatakse nende eripärasi.

4.1 Kinect sensor

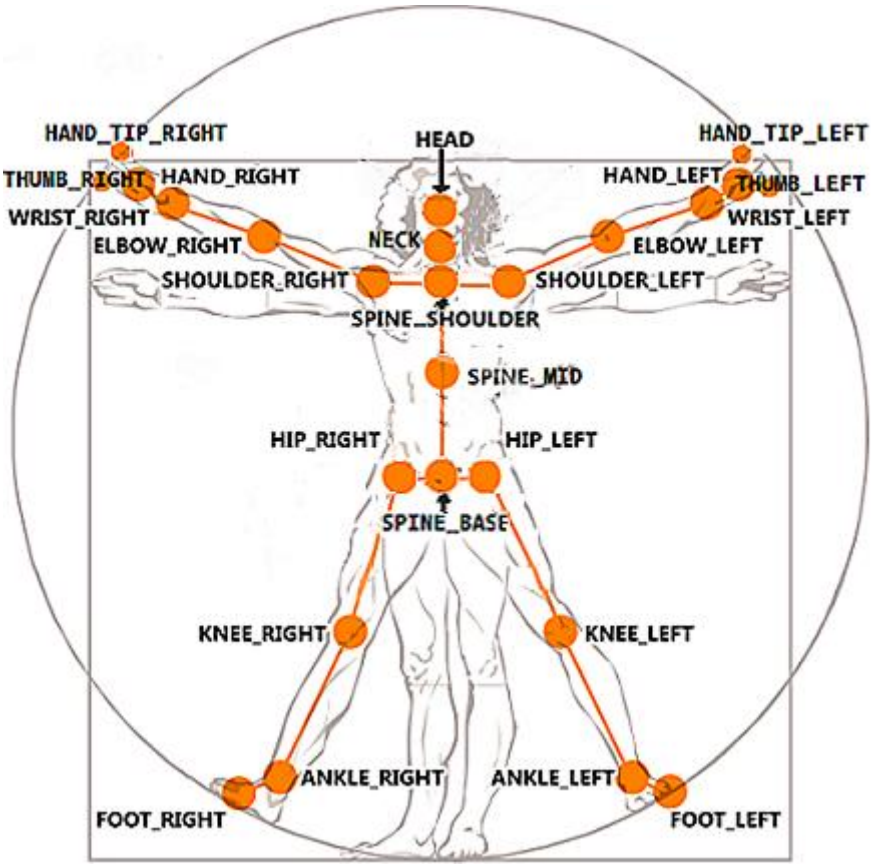
Joonis 2 Kinecti teise versiooni sensor

Kinect on Microsofti poolt toodetud sensor, mille esimene versioon tuli välja 2010. aastal ja oli mõeldud kasutamiseks koos Xbox mängukonsooliga. 2013. aastal arendati välja teine versioon, mille puhul parandati sensori üksikosasi ning lisati täiendavaid funktsioone. Sensoril on kolm komponenti: RGB kaamera, sügavussensor ja mikrofon. Sensori viimases versioonis on RGB kaamera resolutsioon 1920x1080 pikslit. Sügavussensori resolutsioon on 512x424 pikslit ning kasutab sügavuse tuvastamiseks

aega, mis kulub sensorist väljuval valgusel peale peegeldumist tagasi jõudmiseks sensorini. RGB kaamera ja sügavussensor tagastavad ametlikult videopilti 30 kaadrit sekundis, kuid olenevalt töödeldavast infost võib kaadrisagedus olla väiksem. Kinecti mikrofonisüsteem koosneb 4 mikrofonist, mis tagastavad heli sagedusega 48 kHz [7].

4.1.1 Kinecti võimalused keha jälgimise puhul

Kinecti uusim sensor on võimeline jälgima nii inimese keha kui nägu, kuid kuna selles töös ei keskenduta näole, siis järgnevalt kirjeldatakse ainult keha jälgimise võimalusi. Võrreldes sensori esimese versiooniga on uusima variandiga võimalik korraga jälgida kuni kuut inimest ning iga inimese puhul 25 kindlaks määratud kehaosa.



Joonis 3 Kinectiga jälgitavad kehaliigesed [8]

Nendeks on pea, kael, rangluu keskmine punkt, õlad, küünarnukid, randmed, käed, käeotsad, pöidlad, selgroo keskmine punkt, vaagna keskpunkt, puusad, põlved, kannad ja jala otsad. Iga kehaosa puhul tagastab sensor tema positsiooni kolmekohalise vektorina [9], orientatsiooni kvaternioonina ning jälgimisoleku seisundi tõeväärustüübina. Viimane on oluline seetõttu, et tihti võib kehaosa olla mingi eseme või objekti taga peidus, mille pärast ei ole teda võimalik jälgida. Orientatsioonide puhul on tähtis teada, et sensor töötab

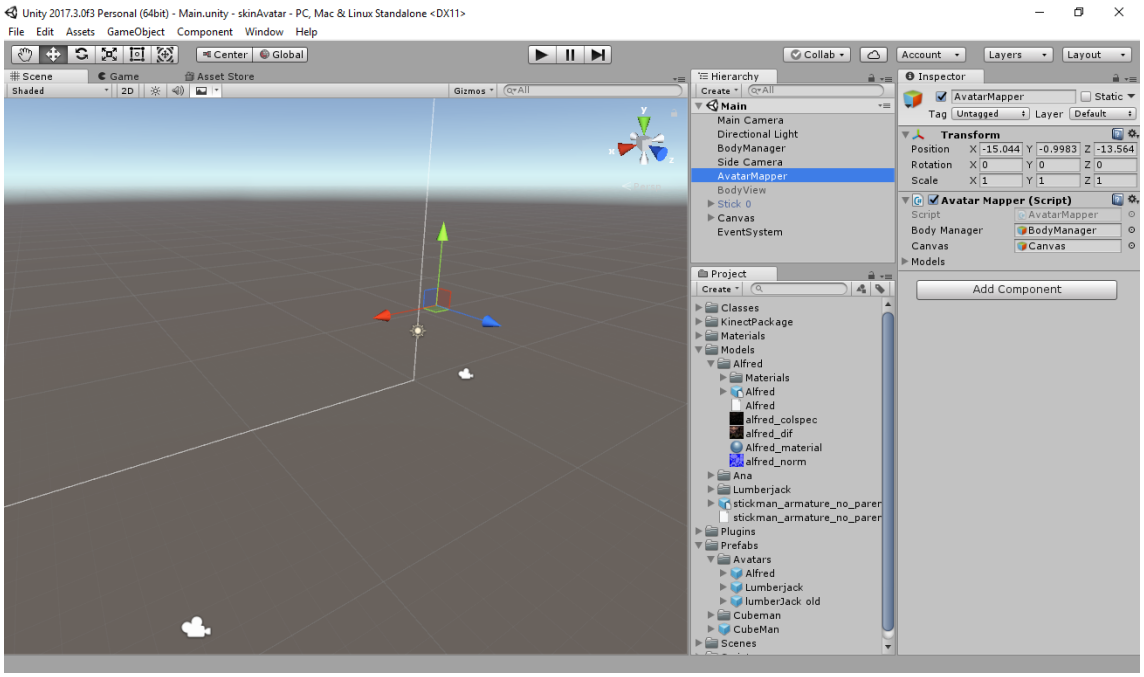
parema käe koordinaatsüsteemiga, kus positiivne z-telg on suunatud sensorist otse välja, y-telg on suunaga sensorist üles ja ja x-telg suunaga sensorist vasakule.

4.1.2 Kinectist info pärimine

Kinecti jaotab töödeldud info eraldi allikatesse ning arendaja peab käsitsi ise valima, millisest allikast ta andmeid tahab. Sõltuvalt soovist on võimalik kätte saada näiteks värvikaamera, sügavussensori kui ka infrapunasensori kaadreid, kuid töö ülesandest lähtuvalt on tähtsaimad kehade infot sisaldavad kaadrid [10]. Sensor tagastab iga kaadri kohta kuuekohalise Body tüüpi massiivi, kus paiknevad kuni kuue keha andmed. Kui massiivis puudub mingi indeksi puhul väärtus, siis on see väärtusega NULL [11].

Igal kehal on olemas unikaalne 64-bitine jälgimisnumber, mille abil saab teda andmemassiivist üles leida. Arendajal on tähtis meeles hoida, et iga kord kui uus keha ilmub sensori vaatevälja, määrab sensor talle suvalise jälgimisnumbri [12]. See tähendab, et kui ühele inimesele oli süsteem määranud indeksiks 1, siis peale tema kadumist ja uuesti ilmumist võib tema indeksiks olla hoopis 2 või mingi muu number. Olukorras, kus sensori vaateväljas on mitu inimest ning igale ühele neist on määratud erinev avatar, on selline käitumine ebasoodne ning vajab eraldi tähelepanu. Sellest tulenevalt on probleemi püstituses välja toodud punkt selliste olukordade käsitlemiseks.

4.2 Unity mängumootor



Joonis 4 Kuvatõmmis Unity mängumootori kasutajaliidesest

Unity on mitmeplatvormne mängumootor, mida arendab firma Unity Technologies SF. Kasutades Unityt on arendajatel võimalik luua kahe- ning kolmemõõtmelisi mänge arvutitele, konsoolidele kui ka teistele nutiseadmetele. [13]. Mängumootori keskkonnas saab mugavalt luua mängutasemeid, menüüsi, koodiskripte kui ka mänguprojekte [14]. Enamus ressursse nagu helid, 3D mudelid, tekstuurid, materjalid ja fondid ei ole võimalik Unitys valmistada ning tuleb eraldi luua, ent neid on võimalik hõlpsalt mängumootorisse importida. Unitys on võimalik kasutada ka koodijuppe, mida saab kirjutada C# keeles. Kuni 2017 aastani eksisteeris lisavõimalusena programmeerida Javascripti keeles, kuid selle toetus on nüüdseks lõpetatud. Prototüübi tegemisel on Unity mängumootor keskmiseks elemendiks, kuhu imporditud 3D mudelid pannakse C# skriptide abil liikuma, kasutades Kinecti kaamerast saadud kehaosade andmeid.

4.2.1 Unity mänguobjektid

Unity puhul on kõige tähtsam aru saada kontseptist, et iga objekt, olgu see valgus, 3D tegelane, kaamera või eriefektid, on mänguobjekt (Unitys kasutatakse sõna GameObject). Mänguobjektist üksinda ei ole mingit kasu ning sellele tuleb määrata omadusi, et sellest saaks lõpuks tegelane, keskkond või eriefekt. Vastavaid omadusi antakse mänguobjektile

kasutades komponente ning sõltuvalt soovitud lõpptulemusest tuleb mõnikord kasutada mitmeid komponente [15].

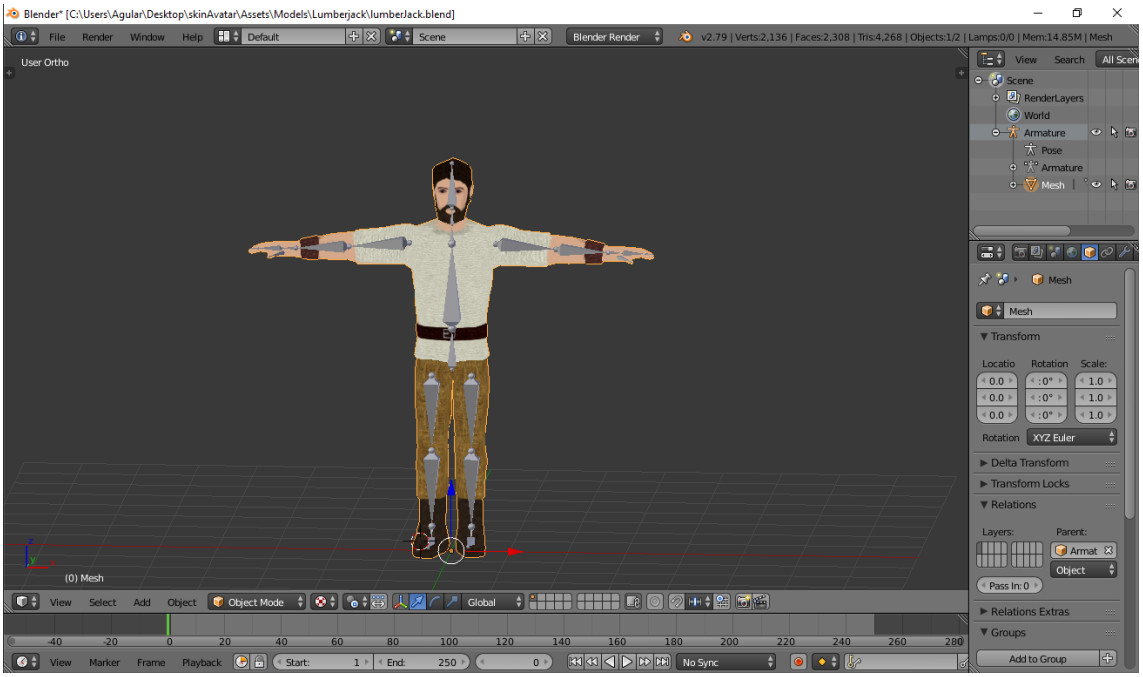
4.2.2 Unitys programmeerimine

Unitys on iga skript algselt komponent, mida on võimalik mänguobjekti külge panna. Sedasi saab mänguarendaja programmeeritaval viisil kontrollida mänguobjekti käitumist mängu jooksul [16]. Selle muudab võimalikuks fakt, et iga uue skripti loomise puhul pannakse ta algul laiendama MonoBehaviour baasklassi. MonoBehaviour baasklass toetab skriptide lisamist mänguobjektidele kui ka koondab kokku mitmed baasfunktsioonid, mida kasutatakse skripti elutsükli erinevatel etappidel. Kõige tähtsamad neist on funktsioonid Awake() ja Update() [17].

Awake() funktsiooni kasutatakse muutujate ja mängu seisu algväärtustamiseks enne kui mäng hakkab. Igas skriptis käivitatakse seda ainult ühe korra tema elutsükli alguses. Erinevalt tavalisest programmeerimisest soovitatakse MonoBehaviour klassi laiendavate klasside puhul kasutada konstruktorite asemel Awake() funktsiooni, sest mänguobjekt, mille külge on skript lisatud, ei pruugi konstruktori väljakutsumisel olla algväärtustatud ning võib põhjustada vea [18].

Update() on ehk kõige tähtsaim funktsioon baasfunktsioonide hulgast, sest seda käivitatakse iga kuvatava kaadri eel vajalike arvutuste tegemiseks. Sõltuvalt skriptide otstarbest ei pruugi iga klass seda kasutada, kuid on enim kasutatud funktsioon MonoBehaviour klassi laiendavate skriptide puhul [19].

4.3 Blender 3D modelleerimistarkvara



Joonis 5 Töös kasutatava 3D mudeli armatuuri valmistamine Blenderis

Blender on tasuta avatud lähtekoodiga 3D modelleerimistarkvara, mis toetab tervet 3D modelleerimise süsteemi: modelleerimist, armatuuri loomist, animeerimist, simuleerimist, renderdamist, kompositsioneerimist ning isegi videotöötlust ja mängude tegemist. Tarkvara on mitmeplatvormne ning töötab operatsioonisüsteemides nagu Windows, Linux ja macOS. Tänu avatud lähtekoodile on kasutajatel võimalik parandada tarkvaras ilmnevaid koodivigu kui ka lisade uusi funktsioone, tõstes sellega Blenderi kvaliteeti ja kasutatavust. [20]. Prototüübi koostamisel otsustati, et 3D mudeleid eraldi selle töö jaoks ei looda ning kasutatakse internetis leiduvaid tasuta mudeleid, sest mudeli modelleerimine ise on väga ajamahukas ja raske tegevus. Ent modelleerimise asemel on Blender tarvilik mudelitele vajaliku armatuuri loomiseks, mille abil on võimalik hiljem mudelit Unity keskkonnas juhtida. Kokku kasutati töö jooksul kolme mudelit: üks autori poolt enda poolt loodud mudel ning kaks mudelit, mis võeti tasuta TurboSquidi veebilehelt.

4.3.1 Armatuuri komponendid ja selle loomine

Mudeli armatuur koosneb 25 luust, mis on vastavad peatükis 5.1.1. Kinecti poolt jälgitavate kehaosadega. Sõltuvalt mudeli kujust võivad osade luude asupaigad ja suurused kohati erineda, kuid on üleüldiselt samades kohtades. Armatuuri ühendamisel

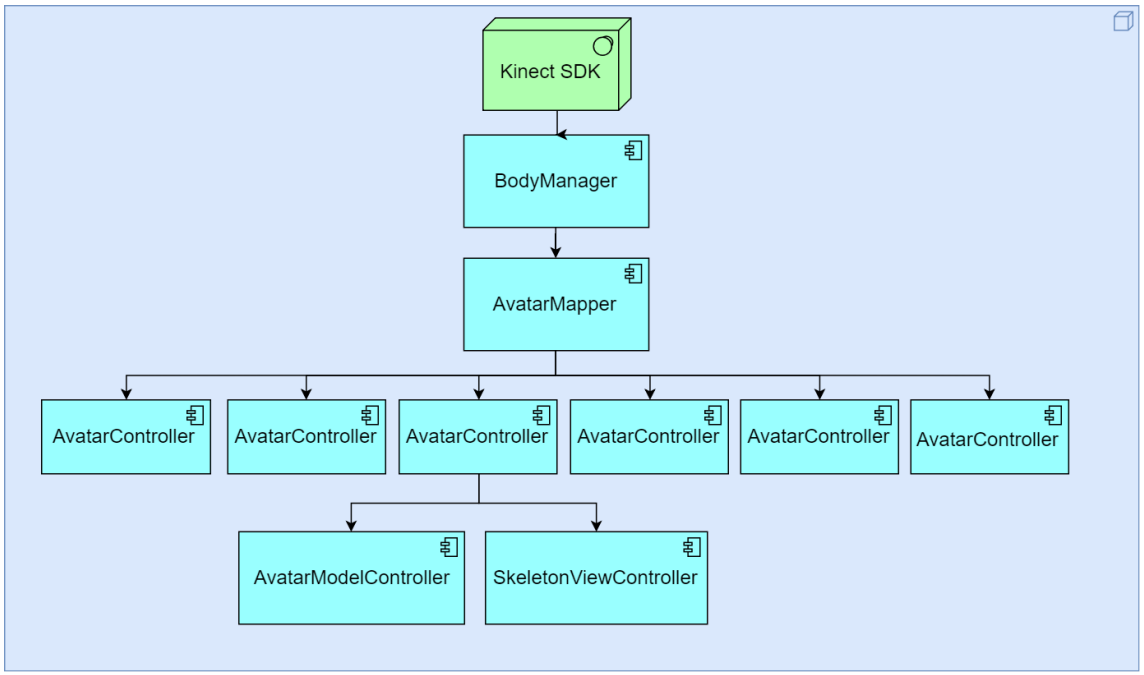
mudeliga kasutati automaatset raskusjaotust, millega seoti mudeli osad liikuma koos vastavate luudega. Kuna automaatse raskusjaotuse puhul ei ole alati tagatud kõige optimaalsem lahendus, tuleb kunstnikul enamasti raskusjaotus manuaalselt ümber teha, et oleks tagatud mudeli optimaalne käitumine luude liikumisel. Vastasel juhul võib tekkida ebameeldivaid olukordi, kus näiteks ainult käe liigutamisel võib liikuda kaasa ka mudeli selg või kael.

Lahenduses ei kirjeldata konkreetse armatuuri tegemist, vaid piirdutakse ülaloleva kirjeldusega. Sellega lõpeb probleemi püstituses kirjutatud alamülesande 2 lahenduse kirjeldus.

5 Lahendus

Järgnevalt kirjeldatakse, kuidas lahendati virtuaalsete kehade liigutamine. Töö käigus lahendati prototüüp iteratsioonide kaupa, kuid siin kirjeldatakse töö lõplikku lahendust . Esmalt kirjutatakse lahti süsteemi hierarhia ning kuidas suuremad klassid omavahel suhtlevad. Seejärel tuleb juttu avataride animeerimisest Kinectist saadava info järgi. Viimastes alampeatükkides räägitakse kasutajaliidese loomisest ning andmete filtreerimisest, et avataride liikumine oleks võimalikult sujuv ja häiringuvaba. Kirjelduse käigus viidatakse probleemi püstituses koostatud alamülesannetele. Lahenduse kirjeldusest on jäetud välja Unity seadistamine vajaliku Kinect koodipakiga, kuid mille komponentidele vihjatakse ning vajadusel selgitatakse lahti.

5.1 Süsteemi hierarhia

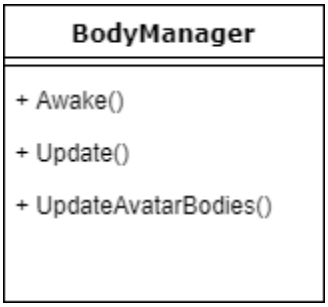


Joonis 6 Loodud süsteemi hierarhia andmete liikumise kaudu

Joonisel 6 on kujutatud süsteemi klasside hierarhiat, mis annab kiire ülevaate alamülesandele 1.b, mis on seotud andmete vahetamisega. Kõige ülemises otsas on klass BodyManager, mille ainus ülesanne on Kinecti sensorist lugeda andmed ning neid edasi anda AvatarMapper klassile. AvatarMapper klassi ülesandeks on saadud info jaotada

avataride vahel laiali, edastades vastavad andmed iga avatari juhtivale klassile AvatarController. Kuna AvatarMapperi klass on seotud andmete laiali jagamisega, on sellel ka täiendavaid ülesandeid, mida hiljem lähedamalt kirjeldatakse. Iga AvatarController skriptikomponent on seotud vastava Avatari mänguobjektiga. AvatarController klassi kontrollida on veel kaks klassi: AvatarModelController ja SkeletonViewController. Esimene neist vastutab ainuüksi selle eest, et saadud kehaandmed saaksid ülekantud juhitavale 3D mudelile. Teine klass on loodud põhiliselt testimise eesmärgiga ning pakub lihtsustatud võimalust inimese kujutamiseks.

5.1.1 BodyManager klass



Joonis 7 Klassi BodyManager tähtsamad funktsioonid

BodyManager klassi ainsaks ülesandeks pärida Kinecti sensorilt andmeid ning neid edasi anda ning lahendab alamülesannet 1.a. See klass pärineb algselt Unity jaoks mõeldud Kinecti koodipakist, mis aitab arendajal kiiremini Kinect sensoriga tööle hakata. Antud klass laiendab ka Unity baasklassi MonoBehaviour, sest klass peab sensorilt pidevaid uusi andmeid küsima ning seda võimalust pakub baasklassi Update() funktsioon. Antud klassis on kolm funktsiooni, mis on kõige tähtsamad: Awake(), Update(), ja GetData(). Et oleks tagatud lugeja mugav lugemine, näidatakse siin ja edaspidi klassidest ainult vajalikke lõike.

```
void Awake()
    {
        _Sensor = KinectSensor.GetDefault();

        if (_Sensor != null)
        {
            _Reader = _Sensor.BodyFrameSource.OpenReader();
            if (!_Sensor.IsOpen)
            {
                _Sensor.Open();
            }
        }

        bodyCount = _Sensor.BodyFrameSource.BodyCount;
        FaceFrameFeatures faceFrameFeatures =
FaceFrameFeatures.RotationOrientation;

        faceFrameSources = new FaceFrameSource[bodyCount];
        faceFrameReaders = new FaceFrameReader[bodyCount];

        avatarBodies = new Avatar.Body[bodyCount];
        for (int i = 0; i < bodyCount; i++)
        {
            faceFrameSources[i] = FaceFrameSource.Create(_Sensor, 0,
faceFrameFeatures);
            faceFrameReaders[i] = faceFrameSources[i].OpenReader();
        }

        for (int i = 0; i < bodyCount; i++)
        {
            avatarBodies[i] = new Avatar.Body();
            for (JointType jt = JointType.SpineBase; jt <= JointType.ThumbRight;
jt++)
            {
                avatarBodies[i].Joints[jt] = new Avatar.Joint();
                avatarBodies[i].Joints[jt].JointType = jt;
            }
        }

    }
```

Joonis 8 Klassi BodyManager funktsioon Awake()

Start() funktsiooni ülesandeks on algväärtustada sensori lugeja ning käivitada sensor ise. Sensorit tähistava muutuja _Sensor väärtuseks seatakse staatilise funktsiooni abil tavaline klass ning järgnevalt algväärtustatakse lugeja muutuja _Reader, mis on mõeldud lugema kehade kohta käivaid andmeid. Järgnevalt seadistatakse koodis vajalikud muutujad näo andmete lugemiseks. Peale seda luuakse kehade sõnastikku tühjad kehade objektid, kuhu hakatakse tulevikus andmeid salvestama ning mille abil antakse andmed edasi teistele klassidele.

```
void Update()
{
    if (_Reader != null)
    {
        var frame = _Reader.AcquireLatestFrame();
        if (frame != null)
        {
            if (_Data == null)
            {
                _Data = new Body[_Sensor.BodyFrameSource.BodyCount];
            }
            frame.GetAndRefreshBodyData(_Data);
            UpdateAvatarBodies();
            frame.Dispose();
            frame = null;
        }


    }
}
```

Joonis 9 Klassi BodyManager funktsioon Update()

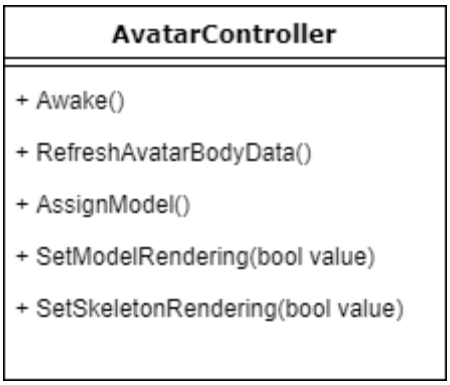
Update() funktsiooni kehas küsitakse igas tsüklis sensori lugejalt viimast saadavat kaadrit. Esimesel käivitamisel luuakse kehade andmete kogumiseks massiiv, kuid neid kasutatakse ainult keha kohta käivate andmete salvestamiseks. Järgnevalt värskendatakse massiivis paiknevaid andmeid kõige uuematega. Peale seda uuendatakse UpdateAvatarBodies() funktsioonis loodud sõnastikus olevaid kehade objekte massiivis olevate andmetega ning samal ajal salvestatakse samas funktsioonis ka näo andmetest saadav pea orientatsioon. Lõplikult kõrvaldatakse keha kaader turvaliselt ära. Keha objektidesse salvestatud andmete küsimiseks on loodud funktsioon GetAvatarBodies().

5.1.2 AvatarController klass

```
using UnityEngine;

public class AvatarController : MonoBehaviour {

    public int Index { get; set; }
    public ulong TrackingId { get; set; }
    public int MaxFilterSamples { get; set; }

    private GameObject sign;
    private AvatarModelController ModelCont;
    private SkeletonViewController SkeletonCont;
    private Filter filter;
    private bool skeletonIsRendered = false;
    private bool modelIsRendered = false;
```

Joonis 10 AvatarController klassi muutujad

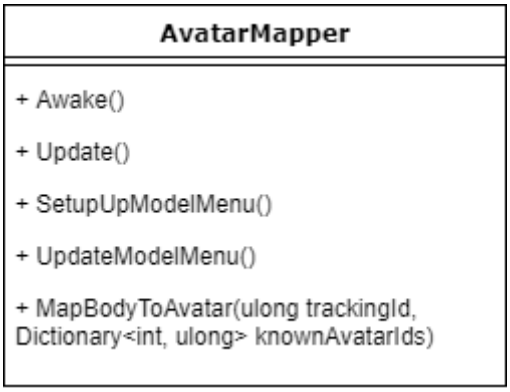
AvatarControlleri klass on avatari mänguobjektile lisatud komponent, mille kaudu saab juhtida ühe konkreetse avatari käitumist.. Klassis on võimalik leida avatari indeks ja jälgimisnumber, mis on tähtsad andmete jagamisel. Samuti on olemas liigutuste silumiseks kasutatavate kaadrite arvu muutuja.



Joonis 11 Klassi AvatarController tähtsamad funktsioonid

Komponendi abil on võimalik muuta avatari kuvatavat mudelit, sisse ja välja lülitada nii mudeli kui ka lihtsustatud vaate kuvamist ning uuendada AvatarModelController ja SkeletonView klasside andmeid, sest need klassid vastutavad mudeli ja lihtsustatud vaate korrektse kuvamise eest. Enne kui AvatarController edastab kehaandmed järgnevatele klassidele, silub ta neid mürade vähendamiseks ning annab nad muudetud kujul edasi. Andmete silumisest tuleb lähemalt juttu eraldi peatükis.

5.1.3 AvatarMapper klass



Joonis 12 Klassi AvatarMapper tähtsamad funktsioonid

AvatarMapper klassis paikneb süsteemi juhtiv osa. Kõige tähtsama ülesandena on sellel klassil jagada BodyManager klassist saadavad andmed juhitavate avataride vahel laiali. Täiendavateks ülesanneteks on AvatarMapperil avataride mänguobjektide loomine ning nende seadistamine vajalike komponentidega kui ka kasutajaliidese uuendamine vajaliku infoga.

```
void Awake()
{
    modelMenu = canvas.transform.Find("Model Menu").gameObject;
    for (int i = 1; i <= bodyCount; i++)
    {
        Avatars[i] = new GameObject();
        Avatars[i].name = "Avatar " + i;
        Avatars[i].AddComponent<AvatarController>();
        Avatars[i].GetComponent<AvatarController>().SetIndex(i);
        Avatars[i].GetComponent<AvatarController>().AssignModel(models[0]);
    }
    SetupModelMenu();
}
```

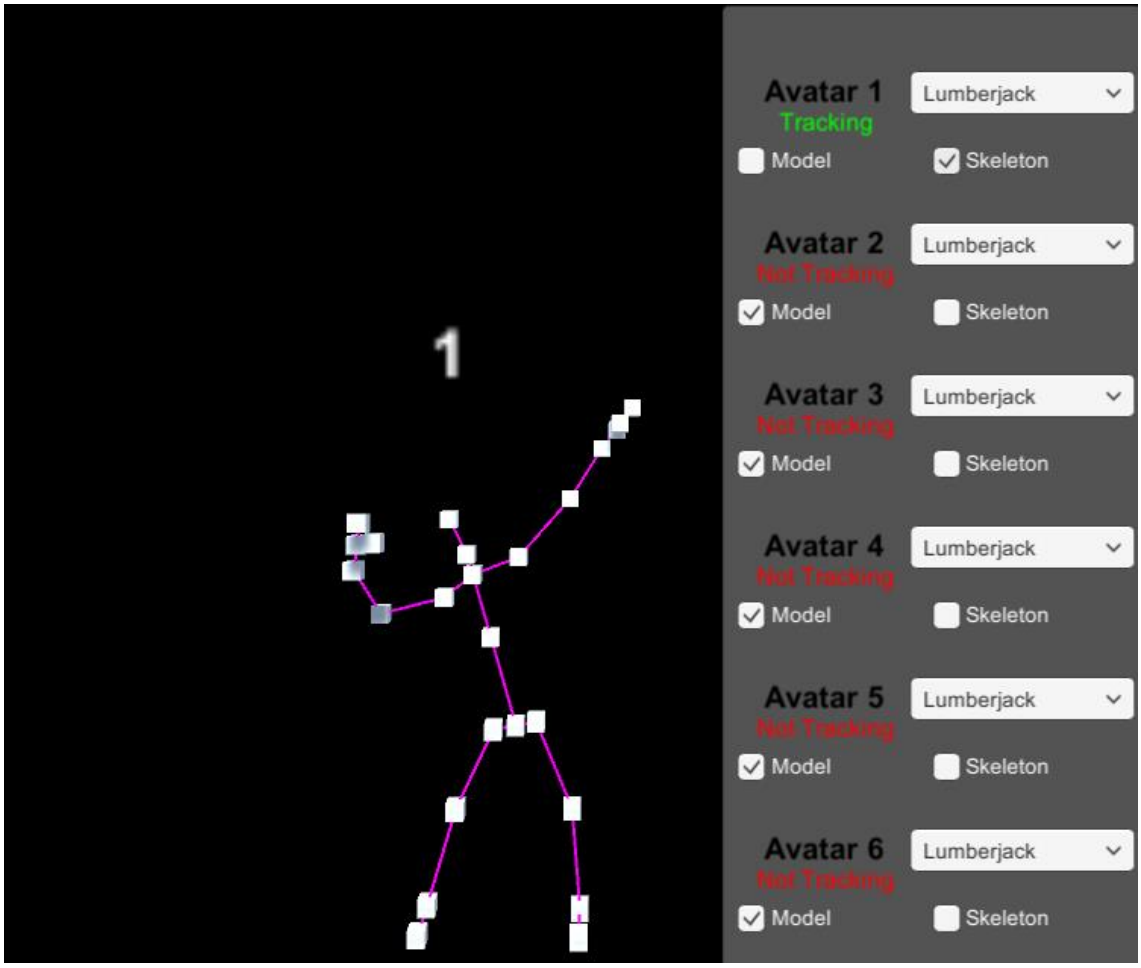
Joonis 13 Klassi AvatarMapper funktsioon Awake()

AvatarMapperi klass, mis laiendab Unity baasklassi, loob oma elutsükli alguses funktsioonis Awake() avatari mänguobjektid ning seadistab need AvatarControlleri komponentidega. Samuti määratakse avataridele esialgsed mudelid. Seejärel salvestatakse AvatarControllerite viited eraldi sõnastikku, et hiljem saaks mugavalt avatare kontrollida. Täiendavalt algseadistatakse kasutajaliideses kuvatav menüü, mis on mõeldud avataride haldamiseks.

Update() funktsiooni alguses tehakse ära vajalik eeltöö, et andmete jagamine oleks üldse võimalik. See eeltöö on vajalik alamülesande 5. lahendamiseks. Iga kord kui inimene

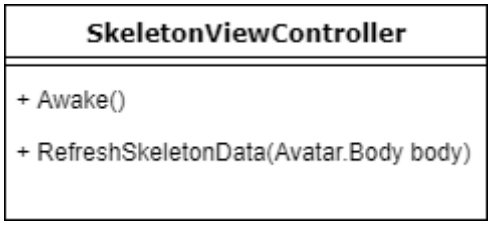
kaob kaamera vaateväljast ja tuleb tagasi, määratakse tema avatarile erinev jälgimisnumber. Selle probleemi lahendamiseks kogutakse kokku sensori poolt andmetes leiduvad jälgimisnumbrid ning süsteemis arvutiekraanil kuvatavate avataride jälgimisnumbrid. Avataride andmekogum moodustatakse sõnastikuna, kus on võtmena avatari indeks ning väärtusena avatari jälgimisnumber. Järgnevalt kontrollitakse üle, kas avataride hulgas leidub jälgimisnumbreid, mida ei sisaldu sensorist saadud viimaste andmete hulgas. Kui selline olukord tekib, siis nullitakse avatari jälgimisnumber ning lülitatakse välja mudeli ning armatuuri kuvamine. Järgnevalt jagatakse laiali kehaandmed vastavalt jälgimisnumbrile. Kui andmete hulgas olev keha identifikaator on olemas ka avataride hulgas, siis antakse keha andmed edasi vastavale avatarile. Kui mitte, siis tuleb avataride hulgast leida sobiv kandidaat, mis tehakse funktsioon MapBodyToAvatar(). Seal kasutatakse varem loodud sõnastikku, mille abil määratakse andmed esimesele väiksema indeksiga avatarile, mida hetkel ei kasutata ühegi keha näitamiseks. Järgnevalt lülitatakse sisse ka avatari 3D mudeli kuvamine.

5.1.4 SkeletonViewController



Joonis 14 Lihtsustatud vaade inimesest

Antud klass kasutab AvatarControlleri klassist filtreeritud andmeid, et kuvada ekraanil lihtsustatud variant inimesest. Lihtsustatud variandi puhul kasutatakse kõigi 25 kehaosa või liigese kuvamiseks kuupe ning nende vaheliste luude jaoks Unity komponenti LineRenderer, mis pannakse kuubi külge. LineRenderer kuvab kahe punkti vahel joont, mille omadusi nagu värv või paksus saab muuta.



Joonis 15 Klassi SkeletonViewController tähtsamad funktsioonid

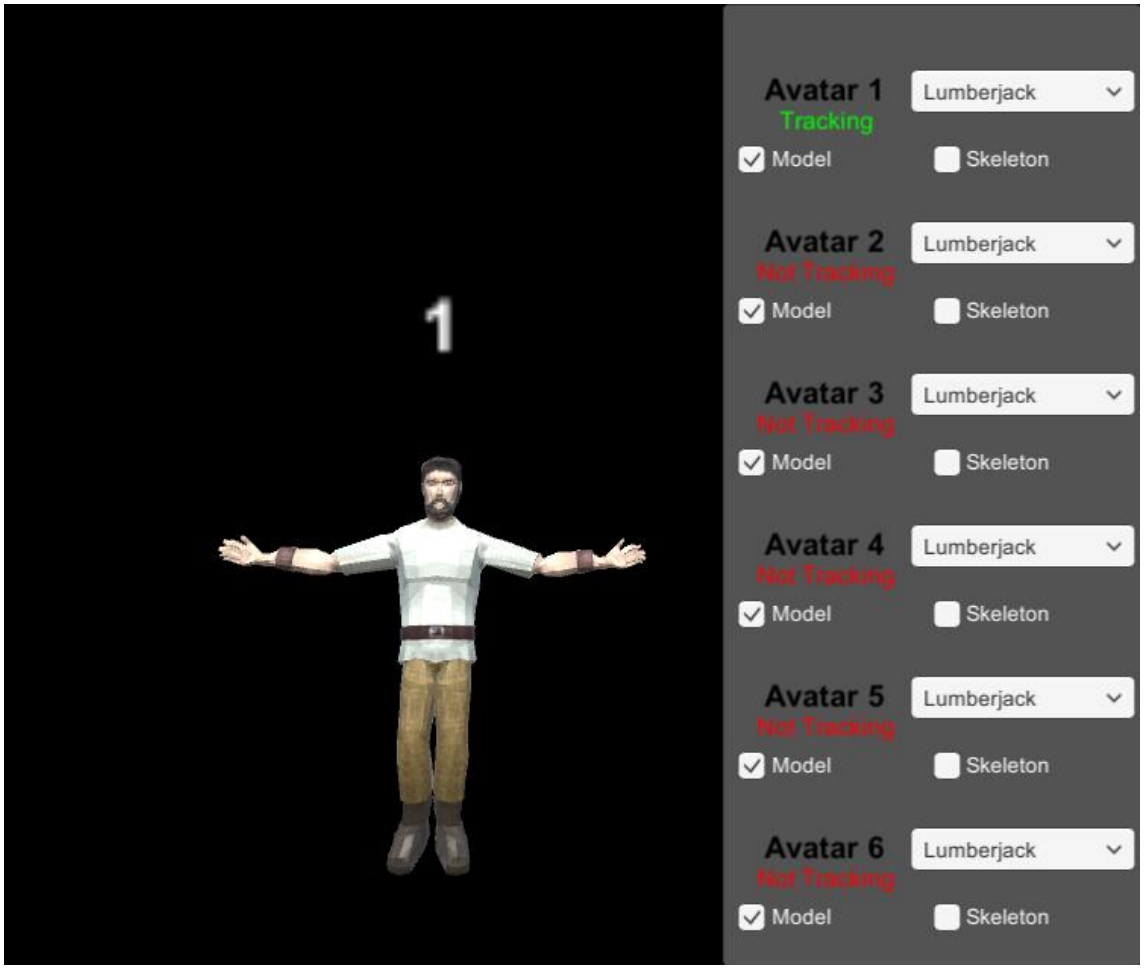
SkeletonViewController klassil on kaks tähtsamat funktsiooni. Baasklassist pärineva Awake() funktsiooni abil luuakse kuubid ning nende vahelised kuvatavad jooned, mille järel lülitatakse algselt välja mõlema kuvamine.

```
public void RefreshBodyData(Avatar.Body body)
    {
        for (JointType jt = JointType.SpineBase; jt <= JointType.ThumbRight; jt++)
        {
            Avatar.Joint srcJoint = body.Joints[jt];
            Avatar.Joint tgtJoint = null;

            if (_BoneMap.ContainsKey(jt))
            {
                tgtJoint = body.Joints[_BoneMap[jt]];
            }
            Transform jointObj = this.transform.Find(jt.ToString());
            jointObj.localPosition = ScalePosition(srcJoint.Position);

            LineRenderer lr = jointObj.GetComponent<LineRenderer>();
            if (tgtJoint != null)
            {
                lr.SetPosition(0, jointObj.localPosition);
                lr.SetPosition(1, ScalePosition(tgtJoint.Position));
                lr.startColor = GetColorForState(srcJoint.TrackingState);
                lr.endColor = GetColorForState(tgtJoint.TrackingState);
            }
            else
            {
                lr.enabled = false;
            }
        }
    }
```

Joonis 16 Klassi SkeletonViewController funktsioon RefreshBodyData()

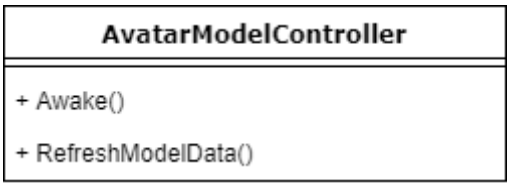
RefreshBodyData() funktsioonis kasutatakse AvatarControllerist pärinevat filtreeritud infot kuupide asukoha määramiseks. Selle jaoks on loodud abistav sõnastik, kus on määratletud ära iga luu vanem, sest joonte kuvamiseks on vaja kahte otspunkti. Järgnevalt kasutatakse for-tsüklit, kus käiakse läbi kõik kehaosad ning leitakse neile vastavad kuubid. Kuupidele määratakse asukoht ning samuti LineRenderer komponentidele, mis on kuupide küljes.

5.1.5 AvatarModelController



Joonis 17 Juhitav 3D mudel

AvatarModelController vastutab otseselt mudeli animeerimise eest nagu oli SkeletonViewController ning temaga sarnaseid funktsioone. AvatarModelController kasutab baasklassi funktsiooni Awake(), et alguväärtustada oma muutujad ning leida üles mudeli küljest juhitavad kehaosad. Mudeli kehaosade automaatseks ülesleidmiseks eeldab kontroller mudelilt kindlat struktuuri ning kehaosade nimetust. Mudeli hierarhias peab paiknema Armature nimeline objekt, mille all paiknevad mudeli armatuuri kõik luud. Luud peavad samuti olema nimetatud täpselt nagu kehaosad, mida Kinect on võimeline jälgima.



Joonis 18 Klassi AvatarModelController tähtsamad funktsioonid

Mudeli asukohtade ning orientatsioonide värskendamine toimub eraldi funktsioonis RefreshModelData, mille sisendiks on AvatarControllerilt filtreeritud ning modifitseeritud kujul andmed. Nii asukohtade kui orientatsioonide värskendamisest räägitakse omaette peatükkides lähemalt, sest nende korrektseks kasutamiseks tuleb teha lisatoiminguid, et tagada mudelite korrektne käitumine.

5.1.6 Abiklassid

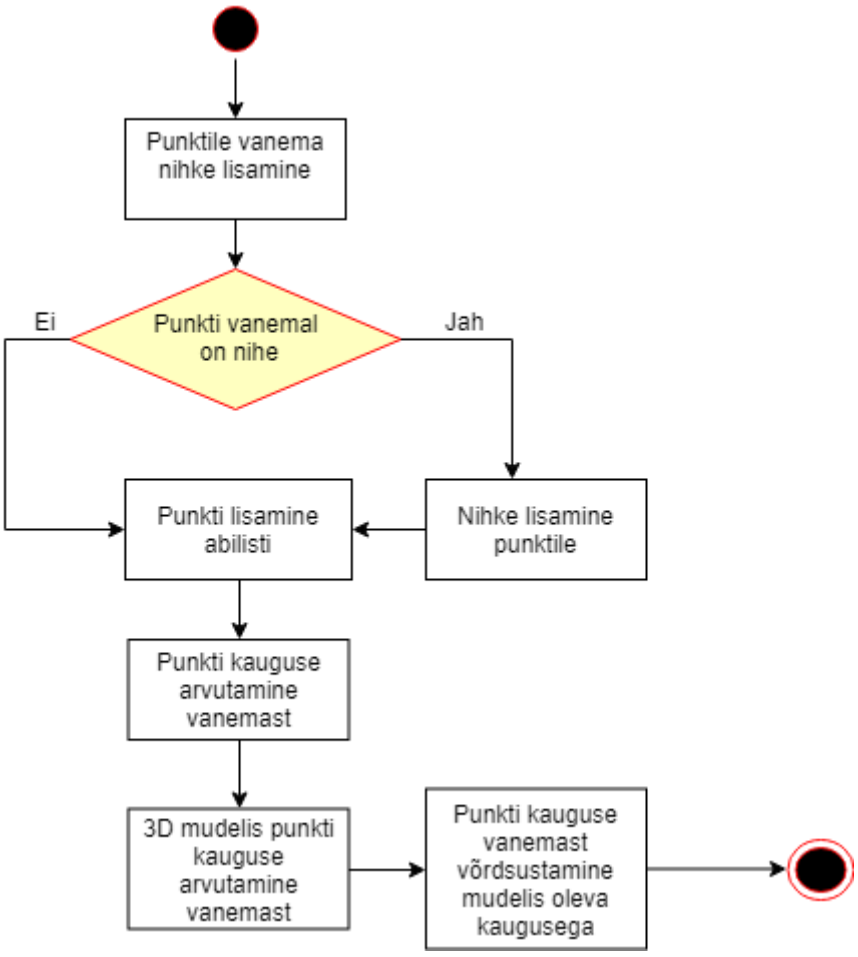
Filtreerimise käigus tekkis vajadus uute abiklasside järgi, mis hoiaksid endas ühe kehaosa positsiooni, orientatsiooni, nime ja jälgimsolekut. Vajadus tekkis sellest, et filtreerides on vaja leida mitme väärtuse keskmine ning see samas andmevormis tagastada. Ent Kinecti SDK loodud klassid ei võimalda väärtuste ülesalvestamist. Samas ei soovitud ka tekitada mitmeid andmekogumeid väärtuste hoidmiseks ja saatmiseks ning seega otsustati luua abiklassid Body ja Joint, mis oma struktuuri poolest kopeerivaid Kinecti klasse. Loodud klassid salvestati eraldi nimeruumi nimega Avatar. Joint klassi puhul säilitati Kinecti nimeruumis oleva klassi struktuur, kuid lisati juurde võimalus väärtusi ülesalvestada. Samuti lisati Jointi klassi ka kehaosa orientatsioon, mis paiknes eelnevalt eraldi klassis JointOrientation, mis hoidis koos keha kõikide kehaosade orientatsioone. Body abiklassi puhul otsest koopiat ei tehtud, kuid see sisaldab sõnastikku, kus on olemas keha kõik kehaosad. Täiendavatest omadustest, mis eksisteerivad Kinecti nimeruumis oleval Body klassil, lisati juurde jälgimisnumber ja tõeväärtustüüp IsTracked.

5.2 Skaleerimine ja asukohapunktide määramine

Kinect sensor tagastab kehaosade asukohapunkte kolmekohaliste vektoritena, kusjuures kõik vektori väärtused on meetrites. Kahjuks ei saa neid väärtusi otse rakendada mudelile, sest ebaproportsionaalsete väärtuste puhul hakkab 3D mudel kokku tõmbuma või ebainimlikult venima ning ei sobi tulemuseks. Selle tõttu peab kaamerast saadud punkte vastavalt 3D mudelile skaleerima ehk nende üksteise vahelised kaugused panema

vastavusse 3D mudelis olevate kehaosade kaugustega. Nimetatud meetod on aluseks alamülesande 3.a lahendamiseks.

Skaleerimine toimub üldiselt selliselt, et võetakse üks kehapunkt ning esmalt liidetakse temale juurde tema vanema puhul toimunud skaleerimine ehk nihe. Seejärel arvutatakse välja suunavektor vanemast tema endani. Seejärel liidetakse vanema punktile korrutis, milles üheks korrutajaks on suunavektor ja teiseks väärtus, mis näitab, kui palju seda suunavektorit peab suurendama või vähendama, et saada 3D mudeliga proportsioonis olev väärtus. Väärtuse leidmise eelduseks on vaja teada mudeli kehaosade esialgseid asukohapunkte. Need salvestatakse AvatarModelControlleri Awake() funktsioonis eraldi abilisti. Väärtus leitakse selliselt, et leitakse esialgses mudelis punkti vanema ja tema enda vaheline kaugus ning jagatakse läbi praeguse suunavektori pikkusega.



Joonis 19 Ühe punkti skaleerimine lihtsustatud kujul

Joonisel 7 on kirjeldatud ühe punkti skaleerimist lihtsustatud kujul. AvatarModelControlleri funktsioonis RefreshModelPositions() toimuv skaleerimine

toimub analoogselt, kuid näiteks joonisel kujutatud viimased kaks etappi toimuvad koodis ühekorraga.

Skaleerimise puhul on ainsateks eranditeks SpineShoulder, mis on skaleerimise lähtepunktiks ning mille väärtust kunagi ei muudeta. Samuti ei arvestata punktide Neck, SpineMid, ShoulderRight ja ShoulderLeft puhul vanema nihet, sest SpineShoulder punkti asukohta ei muudeta. Seda on võimalik näha ka joonisele lisatud otsustuspunkti kaudu.

Asukohapunktide puhul on tähtis tähele panna, et Kinect tagastab asukohapunkte peegeldatuna. See tähendab, et kui inimene liigutab näiteks jalga kaamera poole, siis Unity keskkonnas liiguks see jalg tahapoole. Selle jaoks on vajalik inverteerida iga asukohapunkti puhul z-telje suund ehk asukohapunkti z väärtus korrutada läbi konstandiga -1.

5.3 Orientatsioonid

Asukohapunktide määramise ning skaleerimise kõrval on sama tähtsad kehaosade orientatsioonid ruumi suhtes. Orientatsioone saab mitmet viisi kujutada, näiteks Euleri nurkade teel, mis on lihtsam ja loetavam viis, ent Kinect on otsustanud tagastada neid kvaternioonidena. Selles peatükis selgitatakse alamülesande 3.b lahendust.

5.3.1 Kvaternioonid

Kvaternioonid on arvude süsteem, mis avaldub kujul $q = s + ix + jy + kz$, kus q on kvaternioon, s, x, y, z on reaalarvud ning i, j ja k on imaginaarühikud. Kvaternioonide peamine eelis Euleri nurkade suhtes on Gimbali luku puudumine. Näiteks kolmemõõtmelises süsteemis tekib Gimbali lukk siis, kui üks telgedest saab teise teljega paralleelseks, millega kaob üks vabadusaste. Antud olukorda ei teki kvaternioonide puhul ning on selle tõttu populaarsemaid viise orientatsioonide kujutamiseks.

5.3.2 Sensori tagastatavate orientatsioonide hulk

Kahjuks ei arvuta Kinect kõikide kehaosade jaoks välja orientatsioonid. Nendeks on kael, sõrmeotsad, pöidlad ning jalad. Pea on mõnes mõttes erandlik, sest seda ei ole võimalik saada kätte samast andmeteallikast, kust teised kehaosad, vaid tuleb omaette allikast lugdeda. Samuti on huvitav, kuidas on otsustatud iga kehaosa orientatsioone hoida. Nimelt loomulikult võiks oletada, et õla orientatsioon paikneb koos õla andmetega ning

küünarnuki orientatsiooniga. Kinecti puhul asub õla orientatsioon koos küünarnuki andmetega ning küünarnuki orientatsioon koos randme andmetega. Analoogselt on paigutatud ka ülejäänud orientatsioonid. Olukorrast võib arvata, et Kinecti arendajad soovisid rõhutada kehaosade vahelist struktuuri ning paigutasid vastavalt ka pöördeid.

5.3.3 Unitys kvaternioonide töötlemine ja 3D mudeli armatuur

Unitys on põhjalikult lahendatud kvaternioonidega arvutamine, sest kvaternioone kasutatakse oma omaduste tõttu samamoodi mängudes. See tähendab, et iga kehaosa orientatsioon on arvutatud parema käe koordinaatsüsteemi suhtes. Samuti tuleb arvestada, et eelnevalt skaleerimises peegeldati asukohtade z-koordinaati, sest Kinect tagastab inimese kehaandmed peegeldatuna. Täiendavalt on vajalik ka x-telje inverteerimine, kuid see on tingitud Unity mängumootorist, mis töötab vasakukäelises koordinaatsüsteemis. Tingimuste tulemusena on soovitud, et lõplikus kehaosa orientatsioonis oleks x-telje ja y-telje orientatsioonid võrreldes algsega 180 kraadi nihkes.

Peamine raskus seisneb imporditavatest 3D mudelitest. Nimelt kasutati selle töö jooksul kahte mudelit. Mõlemasse mudelisse tehti tehti samasuguse struktuuriga armatuur samal ajal säilitades, et loodud luud oleksid sarnaste orientatsioonidega. Unity mängumootorisse importides selgus, et mõlema mudeli puhul on armatuuri luude orientatsioonid väga suurte erinevustega, kuigi nende asendid ise on sarnased. Selle tõttu otsustati, et keha armatuuride seadmiseks orientatsioonidega tuleb iga Kinectist saadud absoluutne orientatsioon viia samasse teljestikku nagu on tema vastav kehaosa. Alternatiivse lahendusena võiks mudeli armatuuris iga luu nurga viia maailma suhtes nulli ehk iga luu oleks suunaga ülesse. Kahjuks selline lähenemine ka ei toimi, kuna eelnevalt selgus, et erinevate mudelite sarnased armatuuris võivad olla suurte erinevustega ning nende puhul nurkade nullimine võib toimida erinevalt.

Avatari 3D mudel on algselt T-positsioonis, mis on standardne positsioon 3D mudelite jaoks. T-positsioonis seisab mudel sirgelt ning mõlemad käed on kõrvale välja sirutatud. Iga kehaosa on mudeli seisukohast oma loomulikus asendis. Et sensorist saadud absoluutsed pöörded hakkaksid kehtima, tuleb igal orientatsioonil suurendada x-teljes, y-teljes või z-teljes olevat nurka vastavalt oma koordinaatteljestikule. Seda on võimalik teha korrutades orientatsiooni läbi soovitud lisapöörde kvaterniooniga. Kvaternioonide puhul tähendab korrutamine liitmist.

Eelnevalt kirjutati, et orientatsioonide y- ja x-teljed peavad olema nihkes. Kahjuks ei toiminud arenduse käigus orientatsiooni korrutamine vastava nurgaga ning seetõttu otsustati, et orientatsioonid konverteeritakse Euleri nurkadeks, mille järel inverteeriti vastavaid telgi. Lõplikult muudeti Euleri nurgad tagasi kvaternioonideks.

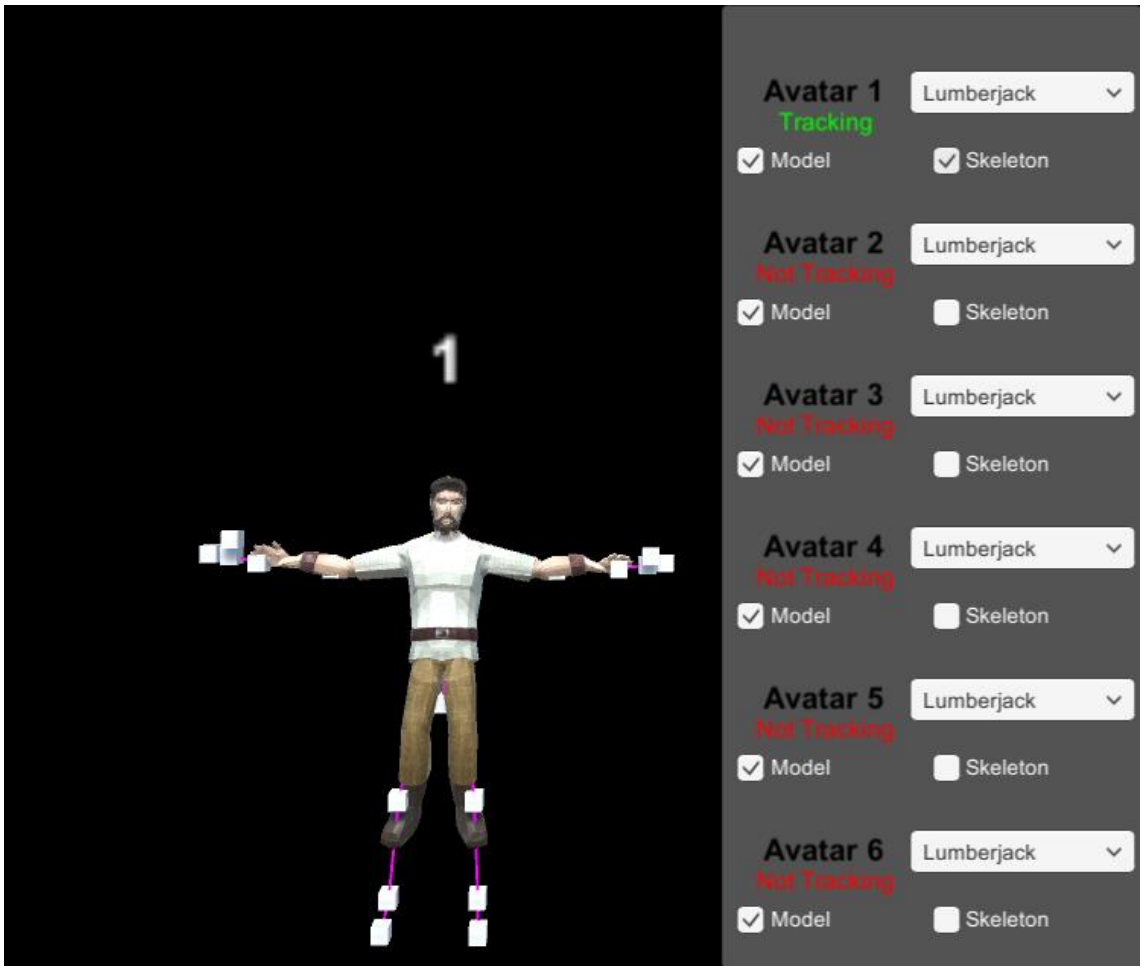
5.3.4 Pea orientatsioonid ja orientatsioonideta kehaosad

Pea orientatsiooni ei sisaldu kahjuks kehandmete hulgas, kuid on olemas näoandmete hulgas. Näoandmetele ligipääs toimub sarnaselt kehaandmetega ning on teostatud BodyManager klassis. Pea orientatsioon salvestatakse koos kehaandmetega ümber Avatari nimeruumis olevase Body klassi, et süsteemis ei peaks mõlemaid andmeid eraldi klasside vahel vahetama.

Kui pea orientatsioon on kättesaadav teisest allikast, siis on osad kehaosad, mille puhul Kinect orientatsiooni ei tagasta. Nendeks on kael, sõrmede otsad, pöidlad, kannad ja jalalabad. Orientatsioonideta kehaosade pööramine on lahendatud AvatarModelControlleri klassis selliselt, et nad pannakse Unity mängumootoris pärima oma vanema luu orientatsioone. Näiteks sõrmede otste puhul pärivad nad käe orientatsiooni või kael pärib SpineShoulder punkti orientatsiooni.

5.4 Kasutajaliides

Töö käigus programmeeriti süsteemi operaatori jaoks lihtne kasutajaliides , mille abil on võimalik jälgida süsteemi seisu ning vajadusel muuta avataride 3D mudeleid (alamülesanne 6). Kasutajaliidest on võimalik avada ning peita kasutades klaviatuuril „M“ klahvi. Menüüs on ülevalt üksteise järel kuus avatari ning iga avatari puhul kuvatakse nende seisundit. Kui avatar parasjagu on kasutuses ning on ekraanil kuvatud, on menüüs tema kõrval kirjutatud roheliselt „Tracking“. Samuti on ekraanil oleva avatari pea kohal tema indeksi number, et operaatoril oleks võimalik aru saada, millise avatariga tegemist on. Kui avatari ei kasutata hetkel ühegi mudel kuvamiseks, on punasega seisundiks kirjutatud „Not Tracking“. See võimaldab operaatoril lihtsalt aru saada, kas süsteem jälgib parasjagu kindlat näitlejat või näiteks, kas süsteem on saavutanud maksimaalse jälgitavate inimeste arvu. Menüü seadistamine ning avataride jälgimisseisundi uuendamine toimub iga kord AvatarMapperi klassi Update() funktsiooni tsüklis.



Joonis 20 Süsteemi operaatori jaoks loodud kasutajaliides

Lisaks jälgimisele on kasutajaliidese abil võimalik muuta ka avatari kuvatavat 3D mudelit. Selle jaoks peab operaator avama vastava avatari kõrval oleva rippmenüü, kust ta võib valida sobiva mudeli. Mudelite arv ei ole piiratud, kuid need tuleb enne süsteemi käivitamist lisada AvatarMapperis asuvate 3D mudelite listi. Täiendavalt saab süsteemi operaator valida, kas peita või näidata avatari puhul mudelit või lihtsustatud vaadet.

5.5 Mudeli liigutuste silumine

Kinecti sensor tagastab andmeid pidevalt, kuid neid ei ole alati täpsed. Täpsust mõjutavad ruumi valgustatus, kaamera asend, inimese riietus või asend kui ka töödeldav andmete hulk. Seetõttu Kinecti tagastatavad andmed võivad sisaldada müra või väikseid nihkeid, mis teevad 3D mudeli liikumise väga teravaks ja konarlikuks. Sellest tulenevalt on vajalik alamülesande 4. lahendamine. Müra kõrvaldamiseks või ühtlustamiseks otsustati kasutada liikuva keskmist, milles kasutatakse iga kehaosa jaoks n tema kõige uuemat

väärtust keskmise arvutamiseks. Hetkel on n-i väärtuseks 5, sest see arv tagab piisab juba piisava sujuvuse, kuid sõltuvalt soovist võib selle väärtus muutuda.

Töö käigus prooviti ka eksponentsiaalse keskmise kasutamist, kuid selle filtri puhul ei olnud märgata olulist paranemist, vaid isegi natuke halvenemist, sest eksponentsiaalne filter puhul on kõige uuemad andmed suurema kaaluga kui vanemad andmed. Seega kui Kinect tagastab andmed, kus ühe või mitme kehapunkti andmed on suure veaga, annab eksponentsiaalne keskmine veale suurema tähtsuse, mille tulemusena on viga rohkem näha ka 3D mudelil.

Andmete filtreerimiseks on olemas ka täiendavaid filtreid, kui neid ei proovitud seetõttu, et süsteemi keerukus oleks selle eest muutunud oluliselt raskemaks ning samal ajal peab säilima süsteemi võimekus töötada reaalajas.

6 Arutelu

Töö ülesande idee pärineb algselt Von Krahli teatri soovist ning seetõttu hakatakse esmalt kasutama näitemängude puhul. Tegelikkuses on loodud süsteem oma loomu poolest universaalne ning seda on võimalik rakendada ka mujal.

Arenduse käigus tekkis palju raskusi seoses Kinect sensori dokumentatsiooniga. Sensori kõige uuem versioon on mõeldud kasutamiseks koos vastava Kinect for Windows 2.0 arendaja tarkvarakomplektiga, kuid Microsofti lehekülgedel puudus selle jaoks vajalik dokumentatsioon. See tähendas, et arenduse käiguse tuli vaadata vanemaid dokumentatsioone, kuid kahjuks alati neist ei piisanud, sest nad ei sisaldanud vajalikke andmeid. Kõige rohkem häiris ehk asjaolu, et Microsoft ise ei maini kunagi dokumentatsoonis, et milliste kehaosade puhul puuduvad orientatsioonid, mis tähendab, et arendaja peab sellest ise testimise käigus aru saama. Kokkuvõtlikult öeldes on kahju, et Microsoft ei ole teinud piisavalt tööd dokumentatsiooniga kui samal ajal on tootnud sensorist versioone arvutitele, et tarkvaraarendajad saaksid nendega rakendusi luua.

Lõputöö suurimaks raskuseks osutus orientatsioonide rakendamine armatuuri luudele. Selgus, et lihtsalt imporditud mudelite armatuuride luud, mis on tehtud eelnevalt Blenderi rakenduses, võivad nurkade poolest kõvasti erineda, kuigi on näiliselt samades asendites. Osaliselt on see tingitud, kuidas toimub Unity mängumootori ja modelleerimistarkvarade vaheline andmevahetus importimisel. Tulevikus oleks kindlasti üheks eesmärgiks leida standardne viis, millega mängumootorisse imporditud mudelite armatuurid oleksid samasugused.

7 Kokkuvõte

Arenduse tulemusena valmis süsteem, millega on võimalik reaalajas jälgida kokku kuni kuut inimest ja animeerida nende avatare. Programmeerimise käigus leiti lahendused kõikidele probleemi püstituses seatud probleemidele:

1) Sensori info kättesaamiseks kasutati klassi BodyManager. AvatarMapper klassi kaudu jagatakse info avataride vahel laiali neid kontrollivatele kontrollerklassidele ja avataridest mudelit ning lihtsustatud vaadet kontrollivatele klassidele.

2) Blenderi modelleerimistarkvara abil valmistati mudelitele vajalikud armatuurid, millega sai Unity mängumootoris vastavaid mudeleid juhtida.

3) Avataride animeerimise puhul skaleeriti asukohapunkte sedasi, et avatarid säilitaksid oma õige kuju. Orientatsioonide puhul muudeti nende väärtusi nõnda, et need töötaksid õigesti mudeli armatuuris olevate kehaosadega.

4) Liikumiste silumiseks kasutati liikuvat keskmist, mis arvutab keskmise viimase 5 kaadri andmete põhjal. Sõltuvalt soovist võib kaadrite arvu alati muuta.

5) Näitleja kadumise puhul peidetakse tema avatar ekraanilt ning ilmumise puhul kuvatakse avatar uuesti. Näitleja ilmumise puhul seatakse temale avataride hulgast selline, mida hetkel ei kasutata ning mille indeks on neist kõige väikseim.

6) Süsteemi operaatori jaoks ülevaatlik kasutajaliides, mis võimaldab tal näha kõikide avataride jälgimisseisundit, nende kuvatavat mudelit ning vajadusel peita mudelit või kuvada neist lihtsustatud vaade.

Tulevaste arengusuundadena oleks antud süsteemil kindlasti pöördkinemaatika kasutamine. Olukordades, kus kasutaja kaob sensori vaateväljast või liigub teise inimese tagant läbi, kaob mudel sujuvalt ekraanilt ära ja tuleb tagasi, kui inimene satub uuesti sensori ette. Pöördkinemaatika abil oleks võimalik sellised kaotamised võimalik asendada kindlate animatsioonidega, näiteks pannes mudeli liikuma kindlas suunas või lihtsalt seisma. Samuti saaks ära kasutada sensori puhul tema heli jälgimise omadusi, et kindla häälesignaali puhul teeks avatar erilise animatsiooni. Lahendada tuleks kindlasti ka mudelite armatuuride probleem ehk tuleks leida standardne viis, millega saaks Unity mängumootorisse importida identsete armatuuridega mudel.

Kasutatud kirjandus

[1] R. Solutions, „Kinect with MS-SDK - Asset Store,“ 2018. [Võrgumaterjal]. Available: https://assetstore.unity.com/packages/tools/kinect-with-ms-sdk-7747. [Kasutatud 14. 05. 2018].

[2] R. Solutions, „Kinect v2 Examples with MS-SDK and Nuitrack SDK - Asset Store,“ 2018. [Võrgumaterjal]. Available: https://assetstore.unity.com/packages/3d/characters/kinect-v2-examples-with-ms-sdk-and-nuitrack-sdk-18708. [Kasutatud 14. 05. 2018].

[3] S. Nõmm ja A. Toomela, „An alternative approach to measure quantity and smoothness of the human limb motions,“ *Estonian Journal of Engineering,* kd. 19, nr 4, pp. 298-308, 2013.

[4] S. Nõmm, A. Toomela, M. Vaske, D. Uvarov ja P. Taba, „An Alternative Approach to Distinguish Movements of Parkinson Disease Patients,“ *IFAC-PapersOnLine,* kd. 49, nr 19, pp. 272-276, 2016.

[5] K. Yoshimitsu, Y. Muragaki, T. Maruyama, M. Yamato ja H. Iseki, „Development and Initial Clinical Testing of "OPECT": An Innovative Device for Fully Intangible Control of the Intraoperative Image-Displaying Monitor by the Surgeon.,“ *Neurosurgery,* kd. 10, nr 1, pp. 46-50, 2014.

[6] Microsoft, „Skeletal Tracking,“ 2018. [Võrgumaterjal]. Available: https://msdn.microsoft.com/en-us/library/hh973074.aspx. [Kasutatud 14. 05. 2018].

[7] J. Ashley, „QUICK REFERENCE: KINECT 1 VS KINECT 2,“ The Imaginative Universal, 2018. [Võrgumaterjal]. Available: http://www.imaginativeuniversal.com/blog/2014/03/05/Quick-Reference-Kinect-1-vs-Kinect-2/. [Kasutatud 14. 05. 2018].

[8] Microsoft, „JointType Enumeration,“ 2018. [Võrgumaterjal]. Available: https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx. [Kasutatud 14. 05. 2018].

[9] Microsoft, „Joint.Position Property,“ 2018. [Võrgumaterjal]. Available: https://msdn.microsoft.com/en-us/library/microsoft.kinect.joint.position.aspx. [Kasutatud 14. 05. 2018].

[10] Microsoft, „KinectSensor Class,“ 2018. [Võrgumaterjal]. Available: https://msdn.microsoft.com/en-us/library/microsoft.kinect.kinectsensor.aspx. [Kasutatud 14. 05. 2018].

[11] Microsoft, „BodyFrame Class,“ 2018. [Võrgumaterjal]. Available: https://msdn.microsoft.com/en-us/library/microsoft.kinect.bodyframe.aspx. [Kasutatud 14. 05. 2018].

[12] Microsoft, „Tracking Users with Kinect Skeletal Tracking,“ 2018. [Võrgumaterjal]. Available: https://msdn.microsoft.com/en-us/library/jj131025.aspx. [Kasutatud 14. 05. 2018].

[13] Unity Technologies SF, „Unity,“ [Võrgumaterjal]. Available: https://unity3d.com/. [Kasutatud 14. 05. 2018].

[14] I. Zamojc, „Introduction to Unity3D,“ ENVATO TUTS+, 2012. [Võrgumaterjal]. Available: https://code.tutsplus.com/tutorials/introduction-to-unity3d--mobile-10752. [Kasutatud 14. 05. 2018].

[15] Unity, „Unity - Manual: GameObjects,“ 2017. [Võrgumaterjal]. Available: https://docs.unity3d.com/Manual/GameObjects.html. [Kasutatud 14. 05. 2018].

[16] Unity, „Unity - Manual: Creating and Using Scripts,“ 2018. [Võrgumaterjal]. Available: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html. [Kasutatud 14. 05. 2018].

[17] Unity, „Unity - Scripting API: MonoBehaviour,“ 2018. [Võrgumaterjal]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html. [Kasutatud 14. 05. 2018].

[18] Unity, „Unity - Scripting API: MonoBehaviour.Awake(),“ 2018. [Võrgumaterjal]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html. [Kasutatud 14. 05. 2018].

[19] Unity, „Unity - Scripting API: MonoBehaviour.Update(),“ 2018. [Võrgumaterjal]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html. [Kasutatud 14. 05. 2018].

[20] Blender, „About — blender.org,“ 2018. [Võrgumaterjal]. Available: https://www.blender.org/about/. [Kasutatud 14. 05. 2018].