Aapo Kokko

3D-mallin hyödyntäminen lääkinnällisen hoitolaitteen UIX-kehitystyössä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

02.05.2022

Tiivistelmä

Tekijä:	Aapo Kokko
Otsikko:	3D-mallin hyödyntäminen lääkinnällisen hoitolaitteen UIX-kehitystyössä
Sivumäärä:	55 sivua + 2 liitettä
Aika:	02.05.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Hyvinvointi- ja terveysteknologia
Ohjaajat:	Lehtori Juha Havukumpu R&D Manager Juho Lauronen

Insinöörityön tarkoituksena oli kehittää LymphaTouchin lääkinnällisen hoitolaitteen kosketusnäytölle uutta ohjelmistokäyttöliittymän toiminnallisuutta. Tavoitteena oli hyödyntää 3D-mallia ja luoda sille ja sitä ympäröivälle käyttöliittymälle toiminnallisuuksia. Lisäksi 3D-mallin hyödyllisyyttä käyttöliittymän toimintaan tutkittiin tutkimustaustan ja käyttöliittymälle tehdyn käytettävyysarvioinnin kautta.

Kehitystyö toteutettiin Qt Design Studio -ohjelmistolla, joka käyttää Qt:n uusinta 6. kehystä. Työssä kehitettiin hoitolaitteen hoito-ohjelman näkymä, johon tässä toteutuksessa kuului kuusi vaiheita. Vaiheissa siirtymiselle luotiin elementit, kuten myös vaiheiden ajastukselle ja koko hoidon ajastukselle.

Vaihe-elementtien viereen asetettiin ikkuna 3D-hahmolle sekä siihen Mixamosta tuotu 3D-asset. Hahmolle kehitettiin zoomauksen ja pyörityksen toiminnot sekä asennonmuutokset jokaiselle vaiheelle. Asennonmuutokset asetettiin animoitumaan vaiheiden välissä. Asennoilla ohjeistetaan potilaan asentoa hoidon aikana. Hoito-ohjelman hoitoalue luotiin sykkimään, jotta voi tunnistaa, mitä aluetta ollaan hoitamassa. Käyttöliittymään lisättiin lopuksi myös tapa vaihtaa hahmon sukupuolta. Edellä mainittuja toimintoja työn tilaaja hyödyntää soveltaen varsinaisen käyttöliittymän kehityksessä. Työn tilaajaa hyödyttää myös kehitysalustasta löydetyt ongelmat.

Käytettävyysarviointi tehtiin asiantuntija-arviona, jonka metodeina oli standarditarkastus ja heuristinen arviointi. Arvioinnissa tarkasteltiin koko käyttöliittymää, mutta työlle merkityksellisiä olivat tulokset 3D-malliin ja sen elementteihin liittyen. Mallista kerättiin sekä ongelmat että positiiviset havainnot myöhempää pohdintaa varten.

Tutkimustaustaa ja käytettävyysarviointia tarkasteltaessa voitiin todeta, että 3D-malleista voi olla hyötyä käyttöliittymän toiminnalle, mutta se riippuu sovelluksesta ja sen käyttäjistä. Malleista voi myös olla haittaa. Työssä toteutetulle käyttöliittymälle 3D-mallista on enemmän hyötyä kuin haittaa, ja siitä löydetyt ongelmat ovat korjattavissa.

Avainsanat: 3D-mallinnus, hoitolaitteet, kehittäminen, kolmiulotteisuus käytettävyys, käyttöliittymät

Abstract

Author:	Aapo Kokko
Title:	Utilizing 3D Model in UIX Development of Medical Device
Number of Pages:	55 pages + 2 appendices
Date:	02 May 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Health Technology
Supervisors:	Juha Havukumpu, Senior Lecturer Juho Lauronen, R&D Manager

---

The purpose of the study was to develop new software UI functionality to a touchscreen of a LymphaTouch medical device. The goal was to utilize a 3D model and create functionality for it and its user interface. Furthermore, the usefulness of a 3D model to the operation of a user interface was researched by way of going through existing research and conducting a usability evaluation to the developed UI.

The UI was developed using software called Qt Design Studio, that utilizes Qt's 6th framework. The UI development produced a treatment program view for the medical device, which in this implementation consisted of six phases. UI elements were created to switch between phases, and to time all the phases and the entire treatment.

A window for a 3D model was placed next to the phase elements and into it a 3D asset from Mixamo. Functionalities for pinch-to-zoom and rotation were developed for the model in addition to a varied pose to every single phase. Changes to the pose were animated between the phases. The poses are used to instruct how to position a patient. The treatment area of the current treatment program was set to pulsate to show, which area is to be treated. At the end of the development a way to change the sex of the model was also added. The commissioner of the study will apply the above-mentioned functionalities in their own UI development. They will also benefit from the study pointing out all the problems found in the development platform.

The usability evaluation was conducted as an expert review with standard inspection and heuristic evaluation as its methods. The evaluation examined the entire UI, but the results related to the 3D model and its elements, were the meaningful ones for the thesis. All the problems and positive observations about the model, were collected for the later discussion.

Considering everything found in existing research and the usability evaluation, it was established, that 3D models can be useful to the operation of a user interface, but that would depend on the application and its users. The models can also be an impediment. For the developed UI, the 3D model is more useful than a hindrance, and the problems found in it are fixable.

Keywords: 3D modeling, development, medical devices, three-dimensional, usability, user interfaces

Sisällys

Lyhenteet ja käsitteet

Asset: Digitaalisessa formaatissa oleva media, malli tai muu käytettävä asia, jonka mukana tulee sen käyttöoikeus.

C++: Ohjelmointikieli, joka on keskittynyt olio-ohjelmointiin.

Debuggaus: Prosessi, jossa etsitään tietokoneohjelmiston tai -systeemin sisällä olevia ohjelmointivirheitä.

Debuggeri: Ohjelmisto ja työkalu, joka etsii automaattisesti toisesta ohjelmistosta ohjelmointivirheitä.

FBX: *Filmbox.* Autodesk:n omistama tiedostomuoto 3D-geometrialle ja -animaatiolle.

GUI: *Graphical User Interface.* Graafinen käyttöliittymä, joka käyttää tekstiä, kuvia ja käyttöliittymäelementtejä.

Heuristiikka: Sovellettava sääntö, jolla on mahdollista ratkoa ongelma nopeammin ja käytännönläheisesti.

IEC: *International Electrotechnical Commission.* Kansainvälinen standardiorganisaatio, joka julkaisee standardeja elektronisille, sähköteknisille ja muille liittyville aloille.

ISO: *International Organization for Standardization.* Kansainvälinen standardiorganisaatio, joka julkaisee standardeja teknisille, teollisille ja kaupallisille aloille.

JavaScript: Ohjelmointikieli, jota käytetään yleisimmin verkkoympäristössä dynaamisena komentokielenä.

Kehys: Runko, jonka päälle on mahdollista rakentaa tietokoneohjelmisto. Muodostuu usein erilaisista valmiista osista, joita voi hyödyntää ohjelmiston rakennuksessa.

Kognitiivinen absorptio:
Tila, jossa käyttäjä on syvästi omistautunut ohjelmiston käyttöön. Tilaan kuuluu dissosiaatiota ajasta, syventynyttä keskittymistä, kasvanutta nautinnollisuutta, hallintaa ja uteliaisuutta.

Kognitiivinen ylikuormitus:
Tila, jossa käyttäjä saa kerralla liikaa informaatiota tai tehtäviä, mikä johtaa vajavaiseen kykyyn käsitellä informaatiota tai toimia.

Proof-of-concept:
Menetelmän tai idean todentaminen toteutuskelpoiseksi.

Python: Ohjelmointikieli, joka on korkeatasoinen ja yleiskäyttöinen.

QML: *Qt Modeling Language.* Qt Groupin omistama merkintäkieli käyttöliittymiin, jota käytetään deklaratiiviseen ohjelmointiin käyttöliittymäkeskeisten ohjelmistojen kehityksessä.

UI: *User Interface.* Käyttöliittymä eli laitteen, ohjelmiston tai muun tuotteen osa, jonka avulla käyttäjä käyttää tuotteen ominaisuuksia. Tässä työssä puhutaan pääasiassa ohjelmistokäyttöliittymistä.

UIX: *User Interaction and Experience.* Käyttövuorovaikutus ja -kokemus, mikä kattaa sekä käyttöliittymän että käyttökokemuksen.

UX: *User Experience.* Ihmisen ja tuotteen välisen vuorovaikutuksen käyttäjäkokemus.

1 Johdanto

Tietokoneiden laskentatehon jatkuva nousu on mahdollistanut, että 3D-grafiikkaa voidaan käyttää erilaisten laitteiden ohjelmistokäyttöliittymissä. Onko niistä hyötyä ohjelmistokäyttöliittymien toimintaan käytettävyyden kannalta vai onko niiden käyttö vain visuaalinen lisä? Tässä insinöörityössä selvitetään tätä kysymystä tarkastelemalla aiempaa tutkimusta 3D-grafiikan käytettävyyteen, kehittämällä toiminnallisuutta 3D-hahmolle lääkinnällisen laitteen UIX:ssä ja arvioimalla kehitystyön käytettävyyttä standardien ja Nielsenin menetelmien avulla. Työhön siis kuuluu ohjelmistokäyttöliittymän (UI) toteutus ja sen käyttökokemuksen (UX) eli tässä tapauksessa käytettävyyden arviointi.

LymphaTouch Oy tutkii tällä hetkellä uusia teknologian käyttömahdollisuuksia ohjelmistokäyttöliittymissä. LymphaTouch on vuonna 2005 perustettu terveysteknologiayritys [1]. Nykyinen heidän nimeään kantava tuote malliltaan LT01 on lääkinnällinen hoitolaite, ja sen voi nähdä toiminnassa kuvassa 1. Laite tuottaa ja käyttää alipainetta sekä mekaanista korkeataajuusvärinää suulakkeen kautta kehon kudosten liikuttamiseen, millä saadaan parannettua imunesteen kiertoa kyseisellä alueella. Tätä ominaisuutta käytetään muun muassa pre- ja postoperatiivisiin turvotuksiin, arpeumiin, lymfedeemaan, palauttaviin hoitoihin ja toiminnallisuuden parantamiseen. [2, s. 2.] Laitteen käyttäjäkuntaan kuuluu fysio-, lymfa- ja toimintaterapeutit, urheilufysioterapeutit, kiropraktikot, hierojat, hoitajat sekä lääkärit [3].

LymphaTouchin vanhan laitemallin kuvassa 1 nähtävä ohjelmistokäyttöliittymä on vanhahtava, joten käyttöliittymien kehityksessä kaivataan modernia otetta, mitä muun muassa haetaan 3D-mallin hyödyntämisellä. Työn tilaajana he haluavat, että hoitolaitteen kosketusnäytölle kehitetään hoito-ohjelman käyttöliittymä. Työssä keskitytään käyttöliittymän ja siihen lisättävän 3D-mallin toiminnallisuuteen.

Kuva 1. Lääkinnällinen hoitolaite LymphaTouch LT01 [4, s. 1].

Toiminnallisuus on se osa-alue, jota työn tilaaja haluaa työstä hyödyntää omassa kehityksessään. Hyötyä myös tuo kaikki havainnot kehitysalustasta kehityksen aikana. Kehitysalustana käytetään Qt Design Studio -ohjelmistoa, joka perustuu Qt-kehykseen. Qt Design Studio on ohjelmistokäyttöliittymien suunnitteluohjelmisto, joka tarjoaa monia työkaluja auttamaan ja nopeuttamaan työntekoa. [5.]

Tämän Qt-kehitystyön ja kirjallisen taustan avulla vastataan kysymykseen 3D-mallin hyödyistä. Kehitystyön lopputuloksesta tehdään käytettävyysarvio. Lopputulosta tarkastellaan arvioon ja kirjalliseen taustaan peilaten, mistä annetaan työn lopulliset johtopäätökset pohdinnassa.

2 Qt Design Studio

Qt Design Studio on yksi Qt Groupin kehittämistä UI-suunnittelu- ja -kehitysympäristöistä. Se mahdollistaa sekä suunnittelijoille että kehittäjille nopean tavan kehittää monipuolisia ja skaalautuvia ohjelmistokäyttöliittymiä sekä niiden prototyyppejä. [5.] Qt Group on perustettu vuonna 1994 nimellä Trolltech Norjassa. Ensimmäinen Qt nimen omaava kehys graafisten käyttöliittymien (GUI) luomiseen julkaistiin vuonna 1995. [6; 7] Qt Group on maailmanlaajuinen ohjelmistoyritys, jonka teknologioita käytetään yli miljardissa laitteessa ja sovelluksessa. Yrityksen nykyinen keskustoimisto sijaitsee Suomessa Helsingissä, ja se toimii 12:ssa eri maassa. [6.]

Tämän insinöörityön teon aikana uusin Qt Design Studion julkaisuversio on 3.0 ja se julkaistiin 3.2.2022 [8]. Se käyttää uusinta Qt:n kehystä 6, jonka uusin versio kirjoitushetkellä 6.2.3 on julkaistu 31.1.2022 [9; 10]. Qt 6 -version kehitys keskittyy erityisesti 3D-grafiikkaan mukaillen alan nykyistä kehitystä [9]. Qt Design Studio -ohjelmistosta on olemassa avoimen lähdekoodin versio Community Edition sekä kaksi maksullista versiota Professional ja Enterprise Edition. Enterprise Edition tarjoaa maksullisista versioista enemmän ominaisuuksia, ja ei vaadi käyttöön Qt-kehityslisenssiä kuten kaksi muuta versiota. [8.] Avoimen lähdekoodin käyttäjille ei ole tarjolla yrityksen tuotetukea [11].

Qt-kehys ja sen kehitysympäristöt ovat C++-ohjelmointikieleen perustuvia työkaluja, joille on luotu laajennus QML- ja JavaScript-ohjelmointikielille. Lisäksi Qt:n käyttö on yhdistettävissä myös muille kielille kuten Pythonille. [12.] QML on Qt:n oma deklaratiivinen eli toteava ohjelmointikieli, jota käytetään UI-kehityksessä Qt Design Studiossa [5; 13]. Sen ohella voi myös käyttää JavaScriptiä imperatiivisena eli käskevänä kielenä. Käytännössä keskustelu tietokannan ja käyttöliittymän välillä tehdään C++:lla ja käyttöliittymän logiikka QML:n ja JavaScriptin yhdistelmällä. JavaScriptin käyttöä suositellaan minimoimaan. [13.]

2.1 Moodit

Qt Design Studio -sovelluksessa on kuusi eri moodia [14] projektien hallintaan ja kehitykseen:

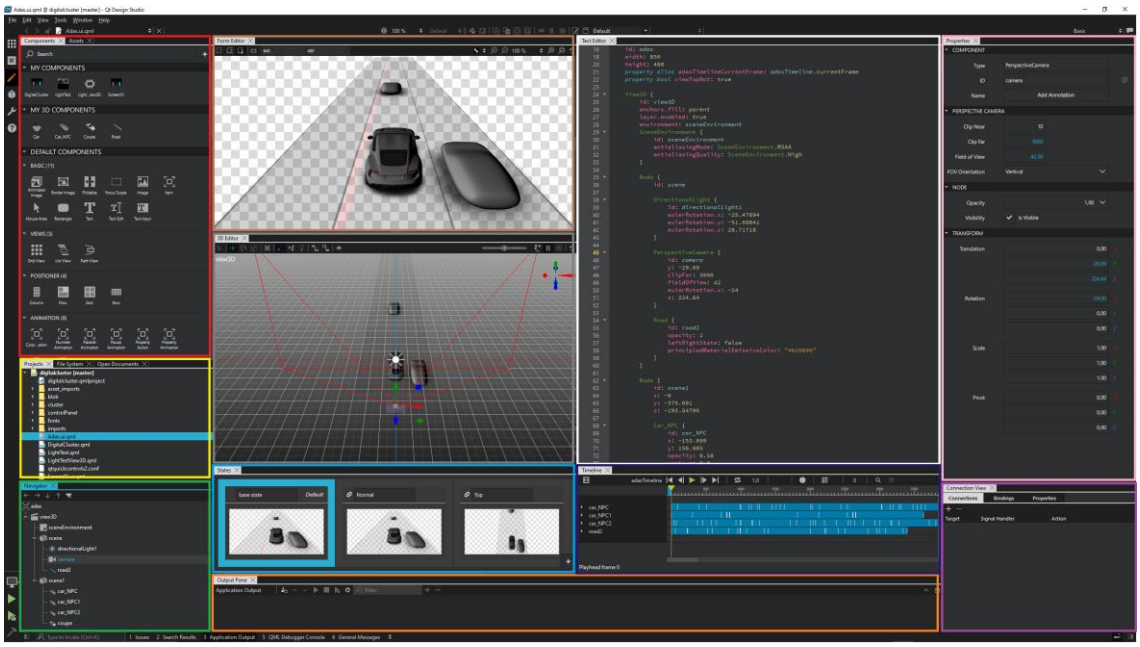
- **Welcome**-moodi, jossa luodaan uusia projekteja, avataan vanhoja projekteja, tutoriaaleja ja esimerkkiprojekteja, luetaan uutisia yhteisöltä ja Qt:n blogia sekä luodaan ja hallinnoidaan Qt-tiliä [15].
- **Edit**-moodi, jossa muokataan projektia ja sen lähdetiedostoja [14].
- **Design**-moodi, jossa tehdään käyttöliittymien suunnittelua ja kehitystä erilaisten näkymien avustuksella [14;16]. Näkymistä kerrotaan lisää kappaleessa 2.2.
- **Debug**-moodi, jolla voi debuggaamalla tarkastaa oman projektin tilan, etsiä virheitä ja käyttää erilaisia työkaluja koodin analysointiin [14].
- **Projects**-moodi, jossa valitaan työkalu projektin reaaliaikaiseen esikatseluun. Moodia voi käyttää vain, kun projekti on auki. [14]
- **Help**-moodi, jossa luetaan Qt:n dokumentaatiota [14].

2.2 Näkymät

Design-moodissa on 14 eri näkymää [16] projektin suunnittelua varten:

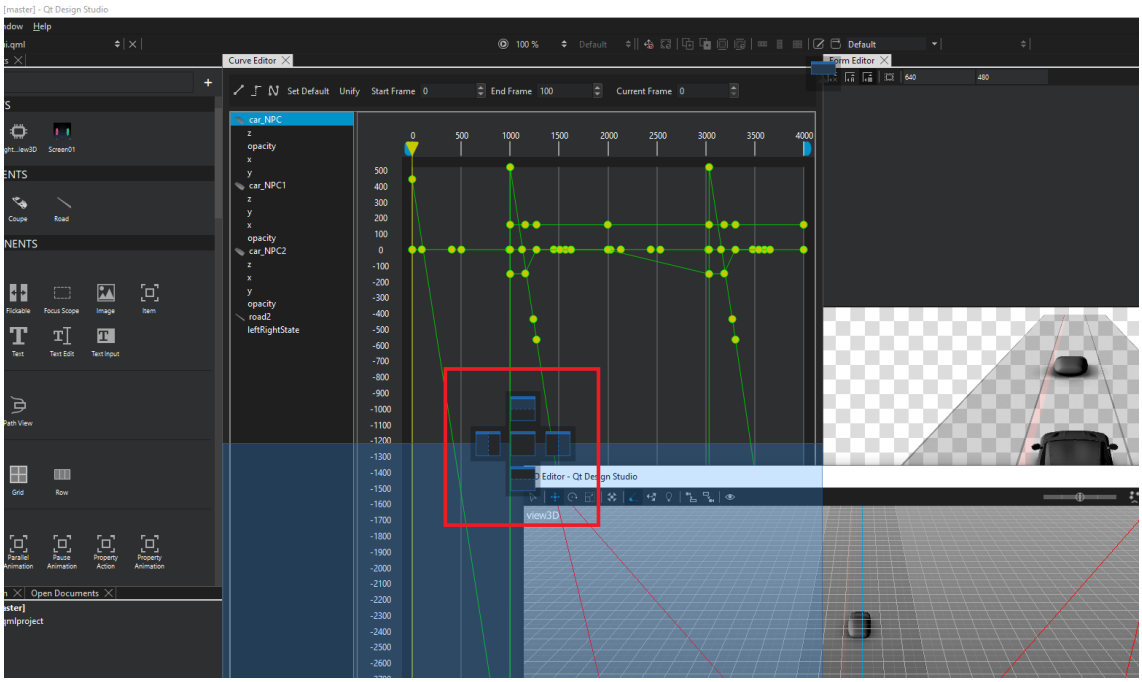
- **Form Editor** on työtila, jossa käyttöliittymän osia suunnitellaan 2D-ruudulle. Osia voi siirtää, kiertää ja niiden kokoa voi muuttaa. Mahdolliset 3D-grafiikat projisoituvat ruudulle kameran mukaisesti. [16;17] Näkymä on nähtävissä kuvassa 2 ruskean ruudun sisältä ylhäällä.
- **3D Editor** -näkymässä muokataan sovelluksen kanssa yhteensopivia 3D-malleja ja niihin liittyviä kameraa, valonlähdettä sekä materiaaleja. 3D-tilassa voi liikkua, zoomata ja kiertää 3D-mallin suhteen. [16;18] Kuvan 2 keskellä harmaassa ruudussa on 3D Editor:ssa oleva 3D-automalli.
- **Library** sisältää kaikki valmiit osat, jotka voi lisätä Form Editorin ruutuun tai Navigator-puurakenteeseen ja siten projektiin [16;17;19]. Osiin kuuluu sovelluksessa valmiina olevia komponentteja, mahdollisia kehittäjän itse tekemiä komponentteja ja sovellukseen tuotuja assetteja [16]. Library on kuvan 2 vasemmassa reunassa punaisessa ruudussa.
- **Navigator** esittää projektin tiedostossa olevat komponentit ja assetit sekä niiden suhteet toisiinsa puurakenteessa [16]. Navigator näkyy kuvassa 2 vasemman alakulman vihreässä ruudussa.

- **Properties**-näkymässä muokataan valitun komponentin tai assetin yksilöllisiä ominaisuuksia [16]. Näkymä on esitettynä kuvan 2 oikeassa reunassa vaaleanpunaisessa ruudussa.
- **Connection View** -näkymässä luodaan yhteyksiä komponenttien, assettien, signaalien sekä komponenttien ja assettien ominaisuuksien välille. Tällä voidaan luoda toiminallisuutta käyttöliittymään. [16] Näkymä on kuvan 2 oikeassa alakulmassa violetissa ruudussa.
- **States**-näkymässä luodaan tiloja, joissa projektin tiedoston komponentit ja assetit voivat muokkaantua käyttöliittymän ajon aikana tiloihin liittyvien ehtojen mukaan [16;20]. Kuvassa 2 olevassa projektissa on kolme tilaa, jotka näkyvät alaosan vaaleansinisessä ruudussa.
- **Transition Editor** -näkymällä luodaan ja tarkastellaan projektin tiedoston tilojen välisten muutosten animaatioita [16].
- **Translations**-näkymässä lisätään projektiin useamman kielen toiminnallisuus [16].
- **Timeline** näyttää aikajanan komponenttien ja assettien ominaisuuksien animaatioille [16]. Kuvan 2 alaosassa näkyy tummansinisessä ruudussa esimerkkiprojektin aikajana
- **Curve Editor** -näkymällä voi muokata projektin animaatioiden liikkeen kaarevuutta aika-paikka-funktiossa [16]. Näkymän voi nähdä kuvan 3 keskellä.
- **Text Editor** on työtila, jossa voi kirjoittaa ja muokata projektin tiedostojen koodia. Muiden näkymien tekemät muokkaukset luovat automaattisesti koodia näihin tiedostoihin. [16] Kuvan 2 valkoisessa ruudussa on auki olevan tiedoston koodia.
- **Projects**-näkymässä näkyy aukinaiset projektit ja niiden tiedostot [16]. Näkymän voi nähdä kuvan 2 vasemman reunan keltaisesta ruudusta.
- **File System** -näkymässä näkyy sovelluksessa aukinaisen kansion kaikki tiedostot [16].
- **Open Documents** -näkymässä näkyy kyseisellä hetkellä aukinaiset tiedostot [16].



Kuva 2. Qt Design Studiossa avattuna oleva esimerkkiprojekti ”Digital Cluster” Design-moodissa. Näytössä olevat kaikki eri näkymät on eroteltu värillisten ruutujen avulla.

Näkymät saa avattua sovelluksen ylävalikosta kohdasta View ja sen alavalikosta Views. Näkymien sijaintia voi muokata siirtämällä niitä hiirellä hyödyntämällä Docking-systeemiä. Kuvassa 3 näkee punaisessa laatikossa grafiikan, joka auttaa sijoittamaan näkymiä. Näkymien kokoa voi myös muokata. [16; 21.]



Kuva 3. Qt Design Studiossa avattuna oleva esimerkkiprojekti ”Digital Cluster” Design-moodissa. Näkymiä on muokattu hyödyntäen sovelluksen Docking-systeemiä. Docking-systeemin grafiikka on eroteltu punaisella laatikolla.

Näkymät saa myös irrotettua erillisiksi ikkunoiksi ja tallennettua työtiloiksi. Kun ohjelmiston ottaa ensimmäisen kerran käyttöön, vain osa näkymistä on esillä, mutta edellä mainituilla tavoilla saa itselle sopivat näkymät käyttöön. [16; 21.]

3 3D-grafiikka käyttöliittymissä

Tämän insinöörityön keskeinen tutkimuskysymys liittyy vahvasti työssä tapahtuvan käytännön kehitystyön tarkoitukseen, työn tilaajan asettamiin tavoitteisiin ja työstä syntyvään hyötyyn tilaajalle. Tässä työssä vastataan kysymykseen, onko 3D-malleista hyötyä ohjelmistokäyttöliittymien toimintaan käytettävyyden kannalta vai onko niiden käyttö vain visuaalinen lisä. Edellä mainitun tutkimuskysymyksen perusteella käsitellään tämän työn taustaa seuraavissa luvuissa. Aiheita käsitellään soveltuvin osin hyödyntäen aiheeseen liittyvää kirjallisuutta sekä alan standardeja. Kirjallisen tutkimuksen tuloksia käytetään apuna, kun pohditaan ja arvioidaan työn lopputuloksia.

3.1 3D-grafiikan hyödyt ja haitat

Aina, kun siirrytään yhdenlaisesta toteutustavasta toiseen, menetetään jotain hyötyä, mitä edellisessä toteutuksessa on ollut, mutta voidaan saada jotain uutta etua vaihdossa. Se, onko tämä vaihtokauppa loppujen lopuksi etujen mukainen kehittäjälle, riippuu menetetyn ja saavutetun suhteesta sekä saavutettujen hyötyjen merkityksestä. Tietokoneiden laskentatehon kasvu on mahdollistanut ohjelmistokäyttöliittymien 2D-elementtien korvaamisen 3D-malleihin liittyvillä ratkaisuilla. Tämän hyödyistä on tehty tutkimusta monien eri alojen laitteiden, ohjelmistojen ja verkkosivujen käyttöliittymissä.

Merkittävä ero, minkä useampi tutkimus on havainnut 2D:n ja 3D:n välillä, on suurempi informaation määrä, minkä 3D tarjoaa. Tämä ero ei kuitenkaan ole yksiselitteisesti hyödyllinen tai haitallinen, vaan ratkaisun mukaan se on voinut vaikuttaa toimintaan positiivisesti tai negatiivisesti. [22–28.] Visinescu ym. [22, s. 6–7] vertasi tutkimuksessaan perinteistä verkkosivulla olevaa kirjakauppaa 3D-mallinnettuun virtuaaliseen kirjakauppaan, jossa oli mahdollista liikkua virtuaalisten hyllyjen välejä. Virtuaalinen kirjakauppa osoittautui olevan vaikeampi käyttää, ja ei suuremmasta informaatiomäärästä huolimatta johtanut suurempaan kognitiiviseen absorptioon [22, s. 7–10].

Kirjakauppaan nähden samankaltainen tulos saatiin myös Oulasvirran ym. [23, s. 309–312] tutkimuksessa, jossa verrattiin 2D- ja 3D-karttojen käyttöä mobiililaitteessa. 3D-kartan tulkinta vei käyttäjiltä enemmän aikaa, vaikka tulokset paranivatkin harjoituksen myötä [23, s. 312–317]. Toisessa Liaon ym. [24, s. 4–6, 8–12] tekemässä karttoihin liittyvässä tutkimuksessa, jossa verrattiin Google Earth -3D-maapallokarttaa ja Google Map -2D-karttaa, havaittiin myös sama tulos 3D-kartan käytön hitaudesta. Tämän epäiltiin johtuvan kognitiivisesta ylikuormituksesta ja näkymän estymisestä [24, s. 12]. Heikkouksien lisäksi tutkimuksessa löydettiin osa-alue, jossa 3D-kartta toimii paremmin. 3D-kartalla käyttäjät pystyivät paremmin löytämään itsensä ja orientoitumaan kompleksisissa tehtävien valintatilanteissa. Tässä oli apuna käyttäjien avaruudellinen muisti. [24, s. 8–12.]

Avaruudellisen muistin hyödyt huomattiin erityisesti Rottermannerin ym. tutkimuksessa, jossa tarkasteltiin lennonjohtamista 3D-visualisaatiossa verrattuna perinteiseen 2D-kartalle piirrettyihin lentoratoihin. Vaikka kokeneet lennonjohtajat kritisoivat 3D-toteutusta, tutkimuksesta kerätty data osoitti, että se vähensi käyttäjien työkuormaa, ja paransi heidän tilannetietoisuuttaan. [25, s. 889–890.]

Avaruudellisen muistin hyötyjä puoltaa myös Settapat ym. [26, s. 510–514] tekemä pilottitutkimus, jossa verrattiin 2D- ja 3D-visualisaatioita aivojen anatomian opetuksessa. Hyötyjä havaittiin myös avaruudellisessa hahmotuskyvyssä. 3D-visualisaatiot antoivat opiskelijoille paremmat oppimistulokset sekä oli heille mielekkäämpi tapa opiskella. [26, s. 512–514.] Toisessa Wangin ym. [29, s. 1945–1951] pilottitutkimuksessa tutkittiin 2D- ja 3D-median vaikutusta avaruudellisen hahmotuskyvyn harjoittamiseen. Pieni käytännön ero havaittiin 3D-median eduksi, mutta se ei ollut tilastollisesti merkittävä. Tutkijat totesivat otannan olevan liian pieni, ja että aihetta tulisi tutkia lisää suuremmalla otannalla. [29, s. 1951–1952, 1955.]

Yksi mahdollinen hyöty 3D-toteutusten suuremmassa informaatiomäärässä on rajallisen tilan tehokkaampi käyttö. Kimin ym. mobiililaitteiden 2D- ja 3D-menuja tutkivassa artikkelissa vertailtiin perinteisempiä listamaisia 2D-menuja kolmeen

erilaiseen pyörivään 3D-ratkaisuun. 3D-ratkaisut ja erityisesti niistä yksi yli muiden sai 2D-menuun verrattuna paremmat tulokset tilankäytössä ja käytön mielekkyydessä. 2D-menu toimi sen sijaan nopeammin, koska se käytti vähemmän laitteen muistia. [27, s. 2059, 2061.] Tutkimuksen tuloksia puolsi myös meta-analyysi, jossa havaittiin 3D-ratkaisujen olevan tilankäytöltään parempia [30, s. 5].

Menuihin liittyen Arkin ym. tutkimuksessa verrattiin ohjelmaikonien toteutusta 2D- ja 3D-malleilla. Niiden sijoitusta myös verrattiin geneerisen taustan ja niille luodun luonnollisen ympäristön välillä. Luonnollisessa ympäristössä ikonit asetettiin kolmiulotteiselta näyttävään toimistopöydän taustakuvaan. Tutkimuksessa käyttäjät löysivät halutut ohjelmien ikonit nopeammin 3D-mallien ja luonnollisen ympäristön avulla. [28, s. 216–218.]

3.2 Käytettävyys

Käytettävyyteen liittyen tässä työssä käsitellään standardin ISO 9241 osia 11 ja 161 sekä standardin IEC 62366 osaa 1 ja sen teknistä raporttia eli osaa 2 niiden soveltuvin osin ottaen huomioon työn vaiheen ja luonteen. Käytettävyys määritellään käyttäjän ja käytettävän tuotteen välisen vuorovaikutuksen tuloksellisuudeksi, tehokkuudeksi ja mielekkyydeksi. Tuloksellisuus ja tehokkuus liittyy tuotteen käyttötarkoituksen, ja sen perusteella määritettäviin tehtäviin ja käyttötavoitteisiin. [31, s. 11–12; 32, s. 10.]

Käytettävyyteen vaikuttaa itse tuotteen lisäksi käyttäjien kohderyhmän tietotaito, resurssit tuotteen käyttöön sekä ympäristö, missä tuotetta käytetään [31, s. 12]. Mitä tulee lääkinnällisiin laitteisiin, käytettävyys ja sen arviointi menevät käsi kädessä turvallisuuden ja riskin kanssa. Koska ollaan tekemisissä ihmisten terveyden ja hengen kanssa, turvallisuus ja riski tulee tuoda keskiöön. Jokaiseen käytettävyyden ongelmaan liittyy riski, joka pitää joko eliminoida, minimoida, tai siitä tulee viime kädessä informoida käyttäjää. [32, s. 6, 12–14.]

3.2.1 Käytettävyyden arviointi ja testaus

Lääkinnällisten laitteiden käytettävyyttä arvioidaan sekä formatiivisesti että summatiivisesti. Molempien arviointien tarkoitus on löytää mahdolliset käyttövirheet käyttöliittymästä, mutta niitä tehdään eri vaiheissa tuotteen elinkaarta, ja niiden skaala sekä keinot ovat yleensä erilaiset. Formatiivisen arvioinnin pääasiallisena tarkoituksena on ennaltaehkäisevästi havaita käyttövirheitä. Arviointi on tarkoitus aloittaa mahdollisimman varhaisessa vaiheessa tuotekehitystä. Arviointeja kannattaa tehdä useampia kehityksen elinkaaren aikana, ja ne ovat usein pienempimuotoisempia ja vähemmän formaaleja kuin summatiivisessa arvioinnissa. Tosin arvioinnit usein kasvavat ja muuttuvat formaalimmiksi kehityksen mennessä eteenpäin, joten viimeistä formatiivista arviointia voidaan pitää esisummatiivisena arviointina. [33, s. 42.]

Summatiivisessa arvioinnissa verifioidaan tuotteen käyttöliittymän turvallisuus, ja se on yleensä yksi osa tuotteen validointia. Arviointi tehdään siis kehityksen elinkaaren loppuosassa, jossa on jo tuotantovalmis tai lähes tuotantovalmis tuote. Tarkoituksena ei ole sinänsä enää löytää uusia käyttövirheitä, vaikka niitä voi mahdollisesti ilmetä, vaan verifioida, onko jäljellä olevien käyttövirheiden riskit hyväksyttäviä. Summatiivinen arviointi on yleensä viimeinen tarkistus, joka tekee tuotteesta turvallisen käytettäväksi, mutta joskus tuote voi vielä lisäksi tarvita kliinistä tutkimusta. [33, s. 42–43]

Summatiivisen arvioinnin käytettävyyden tarkastelu tehdään pääasiassa hyvin formaalin käyttäjätestauksen kautta. Siinä suunnitellaan testiprotokolla, jossa käydään muun muassa läpi käyttötarkoitus, käyttäjäryhmät ja käyttöympäristö sekä luodaan aiemmin havaittuihin käyttövirheisiin liittyvät tehtävät. Testaus tehdään hallitussa ympäristössä protokollan mukaan ja siitä kerätään sekä objektiivista tietoa että subjektiivisia havaintoja. Kerätyistä tiedoista tehdään lopulta raportti, jossa analysoidaan tulokset. [33, s. 43–44.]

Koska formatiivinen arviointi on vähemmän formaalia, siinä voidaan käyttää useampia erilaisia tarkastus- ja testausmetodeja [33, s. 42]. Aiemmin mainittu

käyttäjätestaus on yksi niistä, mutta muita mahdollisia metodeja ovat Nielsenin [34, s. 413–414] mukaan ainakin seuraavat:

- **Heuristinen arviointi** (heuristic evaluation), jossa tarkastetaan noudattaako käyttöliittymän elementit ennalta valikoituja käytettävyysperiaatteita eli heuristiikkoja.
- **Kognitiivinen läpikäynti** (cognitive walkthrough), jossa asetutaan käyttäjän näkökulmaan, ja käydään läpi käyttöliittymän käytön vaiheita tarkastaen johtaako se haluttuihin lopputulemiin.
- **Formaali käytettävyystarkastus** (formal usability inspection, jossa käydään kuusivaiheinen tarkoin roolitettu toimenpide, missä yhdistetään heuristista arviointia yksinkertaiseen kognitiiviseen läpikäyntiin.
- **Pluralistinen läpikäynti** (pluralistic walkthrough), jossa käydään käyttäjäkeskeinen läpikäynti käyttöliittymän käytön vaiheista kuten kognitiivisessa läpikäynnissä, mutta se tehdään ryhmäkokouksessa.
- **Ominaisuustarkastus** (feature inspection), jossa listataan käyttöliittymälle tyypilliset tehtävät ja niihin liittyvät ominaisuudet järjestetysti, ja käydään ne systemaattisesti läpi tarkastaen, onko ominaisuuksien käyttö tehtävissä ymmärrettävää, saavutettavaa ja tehokasta.
- **Johdonmukaisuustarkastus** (consistency inspection), jossa useampi suunnittelija erilaisista projekteista tarkastaa, tekeekö käyttöliittymä asioita samalla tavalla kuin heidän omassa työssään.
- **Standarditarkastus** (strandards inspection), jossa asiantuntija tarkastaa, onko käyttöliittymä yhdenmukainen standardien kanssa.

Edellä mainituista metodeista heuristinen arviointi, kognitiivinen läpikäynti, ominaisuustarkastus ja standarditarkastus ovat luonteeltaan sellaisia, että niitä on tarkoitus tehdä itsenäisesti asiantuntija-arvioina. Yksittäiset asiantuntija-arviot tosin voidaan ja usein kootaan suuremmaksi kokonaisuudeksi. Pluralistinen läpikäynti ja johdonmukaisuustarkastus tehdään aina useamman asiantuntijan paneeleissa. Formaalissa käytettävyystarkastuksessa on joitain itsenäisiä ja joitain paneelissa tehtäviä vaiheita. [34, s. 413–414.]

3.2.2 Käyttöliittymän käytettävyysarvioinnin suunnittelu

Koska tämän insinöörityön ohjelmistokäyttöliittymän kehitys on niin varhaisessa vaiheessa, ei käyttöliittymälle tehdä summatiivista arviointia. Käyttöliittymälle tehdään tosin pienimuotoinen formatiivinen arviointi, joka rajataan insinöörityössä kehitettyyn ohjelmistokäyttöliittymään. Lisäksi arviointi rajataan pelkkään

käytettävyysarvioinnin vaiheeseen eli turvallisuutta eikä riskiä mahdollisten käyttövirheiden määrittämisen lisäksi tarkastella. Arviointia käytetään lähinnä työkaluna tutkimuskysymykseen vastaamiseen.

Arviointiin sovelletaan kahta metodia heuristista arviointia ja standarditarkastusta ja se tehdään itsenäisenä asiantuntija-arviona. Asiantuntijana toimii työn tekijä itse. Se, että arviossa on vain yksi asiantuntija, tarkoittaa Nielsenin ja Landauerin [35, s. 209–210] mukaan, että iso osa ongelmista voi jäädä löytämättä, mutta työn aikarajoitteet estävät järjestämästä laajempaa arviointia. Arviota myös käytetään enemmän työkaluna tutkimuskysymyksen pohdinnalle, joten kaikkien ongelmien havaitseminen ei ole tulevan kehityksen kannalta ehdotonta.

Koska asiantuntijana toimii opintojen loppuvaiheessa oleva opiskelija, arviointimetodeissa on hyvä ottaa huomioon asiantuntemuksen vajavaisuus. Ensimmäinen valittu metodi standarditarkastus antaa valmiiksi määriteltyjä vaatimuksia, jotka ovat kansainvälisesti standardoituja. Tämä antaa vahvan lähtökohdan niiden elementtien arvioinnille, joille on tarjolla valmiita vaatimuksia. Heuristinen arviointi vaatii enemmän tulkintaa, mutta antaa heuristiikkojen kautta rakennetta ja suuntaa arviolle. Se valittiin erityisesti, koska se mahdollistaa käyttöliittymän ongelmien lisäksi, sen positiivisten puolien havainnoinnin, mikä on tärkeää tämän työn tutkimuskysymyksen pohdinnalle.

Standarditarkastuksessa otetaan huomioon standardin IEC 62366 teknisen raportin yleisvaatimukset ohjelmistokäyttöliittymälle ja standardin ISO 9241 osan 161 antamat ohjeistukset visuaalisten elementtien käytölle. Standardin IEC 62366 teknisen raportin [33, s. 47] antamat yleisvaatimukset ohjelmistokäyttöliittymälle ovat seuraavat:

- Jokaisella ruudulla tulee olla tarkoitusta kuvaava otsikko, jotta käyttäjä voi ymmärtää oman sijaintinsa käyttöliittymän rakenteessa ja edistymisen sen hetkisessä tehtävässä.
- Mikäli käyttöliittymässä oleva toiminallisuus vaatii vähintään kolmen sekunnin odotuksen, tulee ruudulla olla jonkinlainen merkki odotuksen edistymisestä.

- Käyttöliittymän jokaisessa ruudussa tulee olla vähintään yksi dynaaminen elementti, jotta käyttäjä voi huomata ohjelmiston kaatumisen.
- Käyttöliittymässä olevan tekstin tulee olla kooltaan vähintään 14 pikseliä, jotta luettavuus on turvattu myös huonompinäköisille.

Standardin ISO 9241 osan 161 [36] antamia ohjeistuksia tarkastetaan sen mukaan, mitä visuaalisia elementtejä on käytetty lopullisessa käyttöliittymässä. Ohjeistukset muutetaan vaatimuksiksi, joiden täyttymistä voidaan arvioida. Mikäli yhdenmukaisuudessa on poikkeamia, arvioidaan mitä käyttövirheitä ne voivat aiheuttaa.

Heuristisessa arvioinnissa käytetään Nielsenin kymmentä heuristiikkaa, jotka ovat lueteltuna taulukossa 1. Ne valittiin, koska Nielsen on Molichin ohella arviointimetodin alkuperäinen kehittäjä, ja ne ovat edelleen laajassa käytössä [37]. Heuristiikoissa koettiin myös olevan tarpeeksi monta näkökulmaa arviointia varten.

Taulukko 1. Nielsenin [37] kymmenen heuristiikkaa.

Nro	Heuristiikka	Kuvaus
1	Järjestelmän tilan näkyvyys	Käyttäjää tulee informoida sijainnista käyttöliittymässä ja järjestelmän tilasta.
2	Järjestelmän ja todellisuuden vastaavuus	Käyttöliittymässä tulee käyttää käyttäjille tuttua ja intuitiivista kieltä ja suunnittelua.
3	Käyttäjän kontrolli ja vapaus	Käyttäjillä tulee olla vapaus kumota tekoja ja palata tilasta tai ruudusta takaisin.
4	Johdonmukaisuus	Käyttöliittymän kielen ja toimintojen tulee olla johdonmukaisia.
5	Virheiden ehkäisy	Käytössä tulevat virheet tulee minimoida suunnittelussa.
6	Muistin sijaan tunnistus	Ulkomuistin tarve tulee minimoida käyttöliittymässä.
7	Joustavuus ja tehokkuus	Käyttäjille tulee tarjota eri vaihtoehtoja tehdä toimintoja ja mahdollisia oikopolkuja.
8	Minimalismi	Kaikki tarpeeton informaatio tulee poistaa käyttöliittymän ruuduista.
9	Virheiden viestintä	Virheistä tulee viestiä hyvin, ja tarjota niille ratkaisua.
10	Apu ja dokumentaatio	Ohjeiden tarve tulee minimoida, ja ohjeiden tulee olla nopeita löytää sekä helppo ymmärtää.

Heuristiikkoja käytetään tässä työssä luvun 4.1 tavoitteisiin nähden valmiin käyttöliittymän ominaisuuksien arviointiin, ja niiden perusteella pyritään löytämään mahdollisia ongelmia ja niihin liittyviä käyttövirheitä. Arvioinnissa keskitytään erityisesti 3D-malliin ja sen ominaisuuksiin. Tutkimuskysymyksen vuoksi 3D-mallista etsitään mahdollisten käyttövirheiden lisäksi käytettävyyden meriittejä.

4 Käyttöliittymän toteutus

Tämän työn käytännön vaiheeseen kuuluu osittainen käyttöliittymän toteutus. Työssä keskitytään laitteen hoito-ohjelman näyttöön ja erityisesti siihen lisättävään 3D-malliin. Tavoitteet käydään tarkemmin läpi seuraavassa luvussa 4.1 ja toteutuksen vaiheet käydään läpi yksityiskohtaisesti luvuissa 4.2 ja 4.3.

4.1 Tavoite

Toteutukselle on työn tilaajan kanssa yhdessä sovittu tavoite, joka pyritään täyttämään työn aikarajoitteiden sisällä. Tavoitteeseen sisältyy yksinkertainen versio laitteen hoito-ohjelman näytöstä, joka on yksi osa kosketusnäytöltä ajettavaa käyttöliittymää. Osatavoitteet on eritelty taulukossa 2.

Taulukko 2. Kehitystyön tavoitteiden osa-alueet, osatavoitteet ja niitä vastaavat vaatimukset.

Toteutuksen osa-alueet	**Vaaditut osatavoitteet**	**Vaatimus**
Hoito-ohjelman vaiheiden UI-elementit	Vaiheiden määrittäminen	Jokaiselle hoidon vaiheelle tulee määrittää sille ominainen tila ja näyttää ikkunassa, missä vaiheessa ollaan.
	Vaiheissa siirtyminen	Vaiheiden välillä tulee pystyä siirtymään painikkeen painalluksella.
	Vaiheiden ajastus	Jokaiselle hoidon vaiheelle tulee luoda ajastin, ja määritettävä tavoiteltava aika, johon asti lasketaan aikaa. Lisäksi koko hoito tulee ajastaa.
3D-hahmon UI-elementit	3D-hahmon tuonti käyttöliittymään	Valittu 3D-hahmon asset tulee saada tuotua näkyviin sille luotuun ikkunaan.
	3D-hahmon zoomaus	3D-hahmoa tulee pystyä zoomaamaan sen ikkunassa, ja se tulee olla mahdollista kahden sormen nipistyksellä kosketusnäytössä (pinch-to-zoom).
	3D-hahmon pyöritys	3D-hahmoa tulee pystyä pyörittämään sekä sivu- että pystysuunnassa käyttäen kosketusnäyttöä.
	3D-hahmon asetusten palauttaminen	3D-hahmon zoomaus ja pyöritys tulee pystyä palauttamaan vaiheen alun asetuksiin.
	3D-hahmon asennon muutos	3D-hahmon asennon tulee pystyä muuttumaan vaiheesta toiseen.
	3D-hahmon asennon muutosten animointi	3D-hahmon eri vaiheissa olevat asentojen erot tulee animoitua paikoilleen vaiheiden siirron välillä.
	3D-hahmon hoitoalueen esitys	Hoito-ohjelman hoitoalue tulee esittää jollain tavalla 3D-hahmon kehossa.

Taulukon 2 ensimmäisessä sarakkeessa annetaan se osa-alue, johon osatavoitteet liittyvät. Toisessa sarakkeessa kerrotaan osatavoitteissa tavoiteltavat ominaisuudet. Kolmannessa sarakkeessa selitetään, mitä ominaisuudelta

vaaditaan. Luvussa 5 eli tuloksissa kerrotaan, kuinka taulukossa mainituissa tavoitteissa on onnistuttu.

4.2 Käyttöliittymän elementtien toteutus

Työn alussa suunniteltiin ja lisättiin Form Editorin ruutuun valmiista komponenteista nelikulmaiset ikkunat 3D-hahmolle ja vaiheiden numeroinnille sekä painikkeet vaiheiden vaihtamiseen. Kuvassa 4 on kehitystyön alkutilanne sen jälkeen, kun ensimmäiset komponentit aseteltiin näyttöön.

Kuva 4. ”Form Editor” -näkymä hoito-ohjelman näytöstä, kun ensimmäiset komponentit on lisätty.

3D-hahmon ikkuna on vasemmalla ja vaiheen ikkuna sekä painikkeet oikealla. Qt Design Studion toimintaan tutustuttiin työstämällä ensin näitä komponentteja. Ensimmäisenä näistä kehitettiin vaiheiden numerointia ja vaiheissa siirtymisen toimintaa.

4.2.1 Hoito-ohjelman vaiheet

Vaiheen numeron esitys toteutettiin ensimmäisessä iteraatiossa tekstinä eli text-komponentilla, jota pyrittiin muokkaamaan kahdella painikkeella. Painikkeilla oli tarkoitus muuttaa tekstin numeroarvoa ylös (Forward) ja alas (Back). Toimintoja lisättiin Connection View -näkymän kautta hyödyntäen painikkeiden vakiosignaalia onClicked. Signaali käynnistyy, kun vastaavaa painiketta painetaan. Koska projektin tiedostojen kielen QML:n ohella voi käyttää JavaScriptiä [13], painikkeiden käskyissä käytettiin funktiota parseInt esimerkkikoodin 1 esittämällä tavalla. Funktio muuttaa tekstimuotoisen numeron kokonaisluvuksi. Sen avulla pystyttiin muokkaamaan tekstimuodossa olevia numeroita matemaattisilla operaatioilla.

```
buttonForward.onClicked: {
     textInput.text = parseInt(textInput.text) + 1
}
buttonBack.onClicked: {
     textInput.text = parseInt(textInput.text) – 1
}
```

Esimerkkikoodi 1. Teksikentässä textInput olevan numeron kasvattaminen ja pienentäminen yhdellä JavaScript-operaatioilla. Operaatioita kutsutaan painikkeiden signaalista.

Ongelma tässä toteutuksessa oli se, että vaiheen liikkumista ei ollut mahdollista rajoittaa tietylle alueelle. Tämän korjaamiseksi vaiheiden numerointi ja niissä siirtyminen toteutettiin toisessa iteraatiossa hyödyntäen liukusäädintä ja käyttöliittymän tiloja.

4.2.2 Käyttöliittymän tilat

Jokaiselle vaiheelle luotiin oma tila (State). Koska jokaiselle tilalle voi tehdä sille ominaiset muutokset näytön komponentteihin, vaiheiden numerot tehtiin pysyviksi tilojen mukaan [38]. Käyttöliittymään lisättiin slider-komponentti eli liukusäädin, jonka rajoiksi asetettiin vaiheiden ylä- ja alaraja. Jokaiselle tilalle annettiin oma liukusäätimen arvo, kuten tehtiin vaihenumeron kohdalla. Painikkeiden

toiminnot muutettiin muokkaamaan liukusäätimen arvoa signaalin onClicked seurauksena esimerkkikoodin 2 mukaisesti.

```
buttonForward.onClicked: {
     sliderPhase.value += 1
}
buttonBack.onClicked: {
     sliderPhase.value -= 1
}
```

Esimerkkikoodi 2. JavaScript-operaatiot, joilla kasvatetaan tai pienennetään liukusäätimen sliderPhase arvoa yhdellä, mikä vaihtaa käyttöliittymän tilan. Operaatioita suoritetaan painikkeiden signaalien kutsusta.

Liukusäätimen arvoa pystyy myös muokkaamaan käyttämällä hiirtä tai koskетusnäyttöä. Arvo liitettiin yhtäläisyysoperaattorilla ”==” myös vastaavan numeron omaaviin tiloihin tekemällä siitä ehto tilasta toiseen siirtymiseen. Tästä Text Editor -näkymä antoi tosin seuraavan varoitusviestin.

> == and != may perform type coercion, use === or !== to avoid it. (M126)

Korjaavana toimenpiteenä korvattiin yhtäläisyysoperaattori ”==” varoitusviestin ehdottamalla ”===” yhtäläisyysoperaattorilla. Tämä antoi uuden varoitusviestin.

> Logical value does not depend on actual values (M325)

Tälle varoitukselle ei kuitenkaan ollut sopivaa ratkaisua. Käyttöliittymän toimintaan se ei vaikuttanut millään tavalla. Viestissä varoitetaan, että numerot voidaan edellä mainitussa vertailussa tulkita loogisiksi arvoiksi. Loogisilla arvoilla määritetään, onko jokin asia tosi tai epätosi. Varoituksesta olisi mahdollista päästä eroon pakottamalla numeroarvot integer-tyyppisiksi eli kokonaisluvuiksi. Sen voisi tehdä JavaScriptin parseInt-funktiolla, mutta funktioita ei voi käyttää tiedostotyypissä, jossa tilat ovat. Tilojen siirtoa harkittiin toiseen logiikkatiedostoon, mutta siinä oleva Window-tyyppi ei tue tiloja. Varoitus päätettiin jättää huomioimatta, koska käyttöliittymä toimii siitä huolimatta.

Käyttöliittymän ajossa oli ongelmia tilojen välillä siirtymisessä ja debuggeri eli ohjelmiston osa, joka jäljittää virheitä, antoi kaksi virheviestiä.

> QML StateGroup: Can't apply a state change as part of a state definition.
>
> QML State: Binding loop detected for property "when"

Kun tilojen komponenttimuutoksista poistettiin tiloille ominainen liukusäätimen arvo, poistuivat sekä ongelmat että virheviestit. Vaiheiden ajan mittaus oli projektin seuraava työvaihe käyttöliittymän elementteihin liittyen.

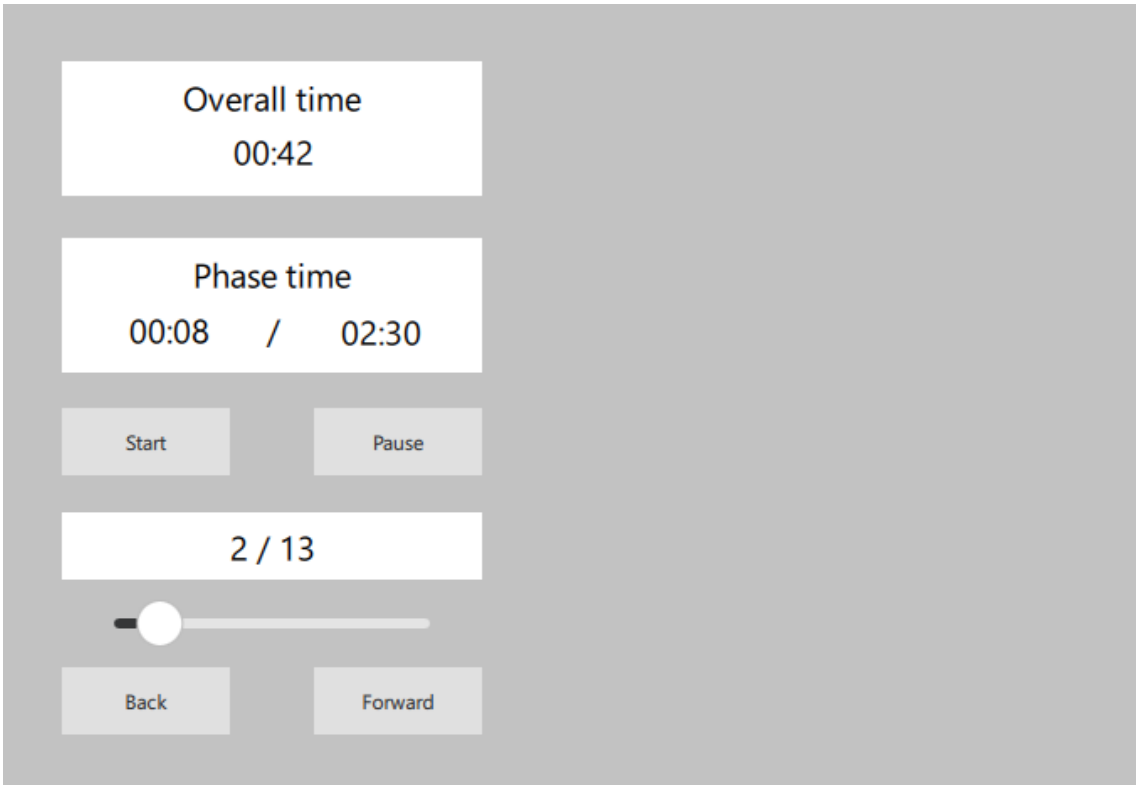
4.2.3 Vaiheiden ajastus

Ajan mittaamista varten tarvitsee käyttää Timer-tyyppiä eli ajastinta, ja luoda funktio ajan päivitystä varten [39]. Kun sen koodia kirjoitettiin projektin tiedostoon, antoi Text Editor -näkymä seuraavan virheviestin.

> This type (Timer) is not supported in a Qt Quick UI form. (M221)

Tiedosto, jolle oli toistaiseksi tehty projektin kehitystä, oli lomake (form) -muotoinen, joten ajastustoiminnot tuli tehdä erilliselle tiedostolle. Erilliseen tiedostoon tuli myös tehdä kaikki muut funktioita vaativat ominaisuudet. [40.]

Timer-tyypin ajastimen intervalliksi asetettiin 1000, mikä vastaa 1000 millisekuntia eli yhtä sekuntia, ja se asetettiin toistamaan. Jokaisen intervallin päätteeksi ajastin asetettiin käynnistämään funktio signaalin onTriggered seurauksena. Signaali herää ajastimen ollessa käynnissä jokaisen intervallin alussa. Funktio kasvattaa sekuntimuuttujan arvoa, joka ehdollisesti kasvattaa minuuttimuuttujan arvoa, ja nollautuu, kun se saavuttaa 60:n. Erillinen ajastin tehtiin sekä kokonaisajalle ja vaiheajalle. Vaiheiden maksimiajoille tehtiin vielä funktio, joka valikoi if-lauseiden perusteella kyseiseen tilaan eli vaiheeseen kuuluvan maksimiajan, joka tallennetaan uusiin sekunti- ja minuuttimuuttujiin. Tämä vaati vaiheen arvon eli sen liukusäädön arvon tallentamisen muuttujaan. Vaiheajan ajastimen funktioon lisättiin vertailu vaiheen maksimiaikaan, mikä pysäyttää vaiheajastimen käynnin. Toteutetut ajastimet näyttävät kuvan 5 mukaisilta.



Kuva 5. Hoito-ohjelman vaiheiden ja koko hoidon ajastus toteutettuna kahteen ikkunaan.

Vaiheajan aloittamiselle ja pysäyttämiselle haluttiin vielä lisätä painikkeet (Start ja Pause), jotta käyttäjä voi aloittaa hoitovaiheen silloin, kun on valmis, ja pysäyttää sen tauolle, jos sille on tarvetta. Vaiheajastimen käynnissä olemisen ominaisuus eli running asetettiin vakiona false-arvoon eli kiinni. Start-painikkeelle annettiin yhteys muuttamaan running-ominaisuus true-arvoon eli käyntiin ja Pause-painikkeelle false-arvoon. Painikkeet ovat kuvassa 5 ajastinikkunoiden alapuolella.

Vaiheen päättymisestä haluttiin vielä antaa käyttäjälle ilmoitusviesti, vaikka kyseistä ominaisuutta ei ole tavoitteissa. Toteutukseen vaadittua moduulia Qt Quick Dialogs ei tosin saatu tuotua tiedostoon. Vararatkaisuna vaiheajastimen funktiossa maksimiajan saavuttamiselle lisättiin käsky, jolla vaiheajan tekstin väri muuttuu punaiseksi ja lihavoituu.

Myöhemmin kehityksessä selvisi, että Qt Quick Dialogs -moduuli on lisätty uusimpaan Qt 6.2 -kehykseen ja tarvittava MessageDialog-tyyppi on tulossa kehykseen tulevassa 6.3 päivityksessä. Ilmoitusviestin toteuttaminen olisi siis voinut olla mahdollista, mutta päivitys viivästyi sen vertaa, että ominaisuutta ei keretty enää kehittää [41;42]. Vararatkaisu siis jätettiin toteutukseen.

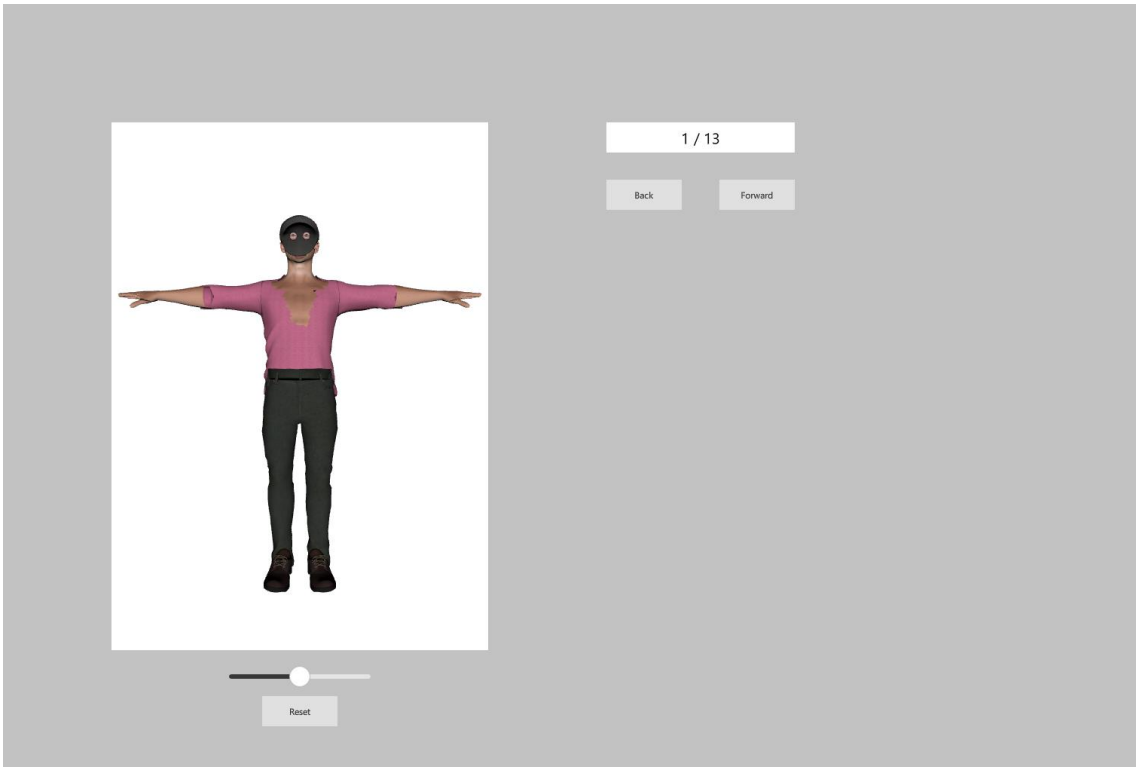
4.3 3D-hahmon toteutus

3D-hahmon kanssa työskenneltiin vaihenumeroinnin ensimmäisen iteraation jälkeen. Malli tätä varten haettiin Mixamosta, ja hahmoista valikoitui Malcolm-niminen asset [43]. Hahmon lisääminen projektiin tuotti aluksi ongelmia, kun sitä ei saatu Form Editor -näkymässä näkyviin. Tämä korjaantui lisäämällä View3D-komponentti hahmolle tarkoitettuun ikkunaan. Hahmo lisättiin Navigator-näkymän puurakenteeseen View3D-komponentin sisälle. Jokainen 3D-malli vaatii sitä seuraavan kameran ja valonlähteen, jotka edellä mainittu komponentti tuo automaattisesti mukanaan [44]. Onnistuneesti lisätty 3D-hahmo projektissa näkyy kuvassa 6.

Käyttöliittymän ajo antoi 3D-hahmon lisäyksen jälkeen debuggerissa seuraavat virheviestit.

> The number of morph attributes exceeds 8. This morph target will be ignored.
>
> Unknown vertex input attr_unsupported in mesh

Viestien tarkkaa tarkoitusta ja tapaa niitä korjata ei ole vielä löydetty. Hahmon poikkeava ulkonäkö Mixamon sivuilla näkyvään vaikuttaa joka tapauksessa olevan yhteydessä edellä mainittuihin virheisiin. Koska hahmon ulkonäkö ei ole tässä kehitysvaiheessa tärkeä, kehitystä jatkettiin suunnitelman mukaan eteenpäin.



Kuva 6. Hoito-ohjelman näyttö 3D-hahmon lisäämisen jälkeen.

Ensimmäisinä ominaisuuksina hahmolle kehitettiin zoomauksen, Reset-painikkeen eli hahmon asetuksen palautuksen ja pyörityksen ensimmäiset iteraatiot. Näiden ominaisuuksien kehitys tapahtui ensimmäisen vaihenumeroinnin ja tilojen kehityksen välissä. Tilojen toteutuksen jälkeen kehitettiin pyörityksen, zoomauksen ja Reset-painikkeen seuraavat iteraatiot, hahmon animointi ja hoitoalueen esitys.

4.3.1 3D-hahmon zoomaus

Zoomauksen lopullisena tavoitteena on pystyä toteuttamaan se kosketusnäytölle tyypillisellä sormien nipistyksellä ja avaamisella (pinch-to-zoom). Kehitys aloitettiin sen sijaan hyödyntämällä liukusäädintä. Tämä lähestymistapa valittiin, koska haluttiin testata zoomauksen toteutumista yksinkertaisemmalla tavalla. Nipistyszoomauksen testaus vaatii kosketusnäyttöä, mikä hidastaa kehitystä sen alkuvaiheessa. Liukusäätimen arvojen yhteydeksi asetettiin ensiksi 3D-hahmon skaalan (Scale) arvon muuttaminen, mutta koska se ei toiminut siirryttiin

muokkaamaan sen sijaan hahmoa katsovan kameran sijaintia. Liukusäädin asetettiin muokkaamaan kameran sijaintia syvyyssuunnassa eli z-akselilla arvojen 150–750 välille. Tässä käytettiin liukusäätimen onMoved-signaalia, joka käynnistyy liukusäädintä liikutettaessa. Signaalissa asetettiin kameran etäisyys olemaan sama kuin liukusäätimen uusi arvo. Kuvassa 7 on edellä mainitulla tavalla toteutettu zoomaus 3D-hahmosta.

Kuva 7. 3D-hahmon zoomaus liukusäätimellä hoito-ohjelmassa.

Koska tavoitteissa oli myös hahmon palauttaminen alkuperäiseen asentoon, tuli näyttöön myös lisätä painike (Reset) sitä varten. Painikkeen painallusta ei yhdistetty pelkästään kameran alkuperäiseen z-akselin arvoon vaan myös liukusäätimen arvoalueen keskelle. Tämä tehtiin, koska liukusäätimen arvon ja kameran z-akselin arvon halutaan olevan koko ajan yhteydessä toisiinsa. Reset-painike ja zoomauksen liukusäätö ovat 3D-hahmon ikkunan alla kuten kuvassa 7 nähdään.

Nipistyszoomauksen (pinch-to-zoom) kehitykseen tarvittiin siihen erityisesti tarkoitettua komponenttia PinchArea ja kosketusnäytön omaavaa kannettavaa

tietokonetta [45]. PinchArea asetettiin samankokoiseksi kuin 3D-hahmon ikkuna. Komponentin ominaisuus, joka mittaa sormien etäisyyttä toisiinsa on pinch.scale, mutta sen lisäksi oli tarve tietää, kasvaako vai pieneneekö kyseinen arvo. Sen takia etäisyys otettiin ensin muistiin muuttujaan signaalissa onPinchStarted, joka käynnistyy, kun PinchArea havaitsee kaksi kosketuspistettä. Signaalissa onPinchUpdated verrataan muuttujaan tallennettua etäisyyttä ja uutta etäisyyttä joka kerta, kun kosketuspisteiden etäisyys muuttuu. Sen perusteella joko suurennetaan tai pienennetään zoomauksen liukusäädintä, mikä zoomaa hahmoa suuremmaksi tai pienemmäksi. Liukusäätimen signaaliksi tuli vaihtaa onValueChanged, joka käynnistyy aina, kun liukusäätimen arvo muuttuu.

Edellä mainitulla tavalla zoomaus ei kuitenkaan ollut responsiivinen, vaan esimerkiksi suurentamisesta pienentämiseen vaihtaessa oli viive. Tämän ongelman korjaamiseksi sormien etäisyyttä tulisi verrata tiheämmässä tahdissa. Signaaliin onPinchUpdated lisättiin muuttujan ja nykyisen etäisyyden vertailun jälkeen operaatio, joka tallentaa nykyisen etäisyyden muuttujaan. Tämä tosin aiheutti tilanteen, jossa tapahtui eksponentiaalinen suurennus, jonka lisäksi pienennys ei toiminut ollenkaan. Tämän epäiltiin johtuvan liian tiheään tapahtuvan vertailun takia.

Vertailutiheyden pienentämiseksi käytettiin ajastinta Timer-komponentin avulla, jolla voitaisiin hallita tiheyttä, jolla vertailu tapahtuu. Aiemmin onPinchUpdated-signaalissa tapahtunut vertailu siirrettiin ajastimen funktioon, joka käynnistyy ajastimen intervallin välein onTriggered-signaalin kautta. Ajastin asetettiin käynnistymään nipistyksen alussa onPinchStarted-signaalissa ja sulkeutumaan onPinchFinished-signaalissa. Signaali onPinchFinished käynnistyy, kun toinen tai molemmat kosketuspisteistä poistuvat PinchArea-komponentin alueelta. Sormien väliseen etäisyyteen eli pinch.scale-ominaisuuteen ei päässyt käsiksi ajastimen funktiossa. Etäisyydelle tehtiin yksi muuttuja lisää. Ensimmäinen muuttuja asetettiin seuraamaan pinch.scale-ominaisuutta reaaliaikaisesti onPinchUpdated-signaalissa ja toinen päivittymään funktiossa intervallin välein vertailun jälkeen. Funktiossa käytettiin esimerkkikoodin 3 mukaisia operaatioita.

```
function pinchToZoom() {
     screen01.sliderZoom.value += (itemPinchToZoom.pinchScaleOld -
     itemPinchToZoom.pinchScaleNew) * 150
     itemPinchToZoom.pinchScaleOld = itemPinchToZoom.pinchScaleNew
}
```

Esimerkkikoodi 3. JavaScript-funktio, jossa kasvatetaan zoomauksen liukusäätimen sliderZoom-arvoa sormien välisen etäisyyden kasvun tai pienentymisen perusteella. Funktiota suoritetaan ajastimen kutsumana.

Etäisyysmuuttujien vertailu kerrottiin 150:llä, jotta zoomauksesta saatiin riittävän nopea. Intervalliksi valikoitui 25 millisekuntia, millä zoomaus tuntui tarpeeksi responsiiviselta muutoksiin.

Pyöritysominaisuuden kehitys sen myöhemmässä vaiheessa aiheutti tarpeen muokata zoomausta uudella tavalla. Koska kameraa tarvitsi liikuttaa pyörityksessä, pelkkä z-koordinaatin muokkaus ei riittänyt suoraviivaiseen zoomaukseen. Tämän ongelman korjauksesta kerrotaan seuraavan luvun 4.3.2 loppuosassa kyseisen pyöritysominaisuuden ohella. Muutos ei kuitenkaan vaikuttanut nipistyszoomauksen toteutukseen, koska se on yhteydessä zoomauksen liukusäätimeen.

4.3.2 3D-hahmon pyöritys

Hahmon pyörittäminen toteutettiin suoraan hiiren painalluksella ja liikkeellä sen sijaan, että olisi aloitettu liukusäätimellä kuten zoomauksen kohdalla. Tämä ei ollut niin monimutkaista toteuttaa, ja voitiin kehittää ilman kosketusnäyttöä. Samat käskyt toimivat sekä hiirellä että kosketusnäytön kosketuksella. Kehityksessä aloitettiin sivuttaisesta pyörityksestä. Muokkauskohteeksi valittiin tällä kertaa mieluummin hahmo kuin kamera. Kameran pyörittäminen hahmon ympärillä olisi vaatinut jatkuvaa kameran rotaation ja kahden eri akselin arvojen muokkausta. Sen sijaan hahmoa voi yksinkertaisesti pyörittää sen akselin ympäri.

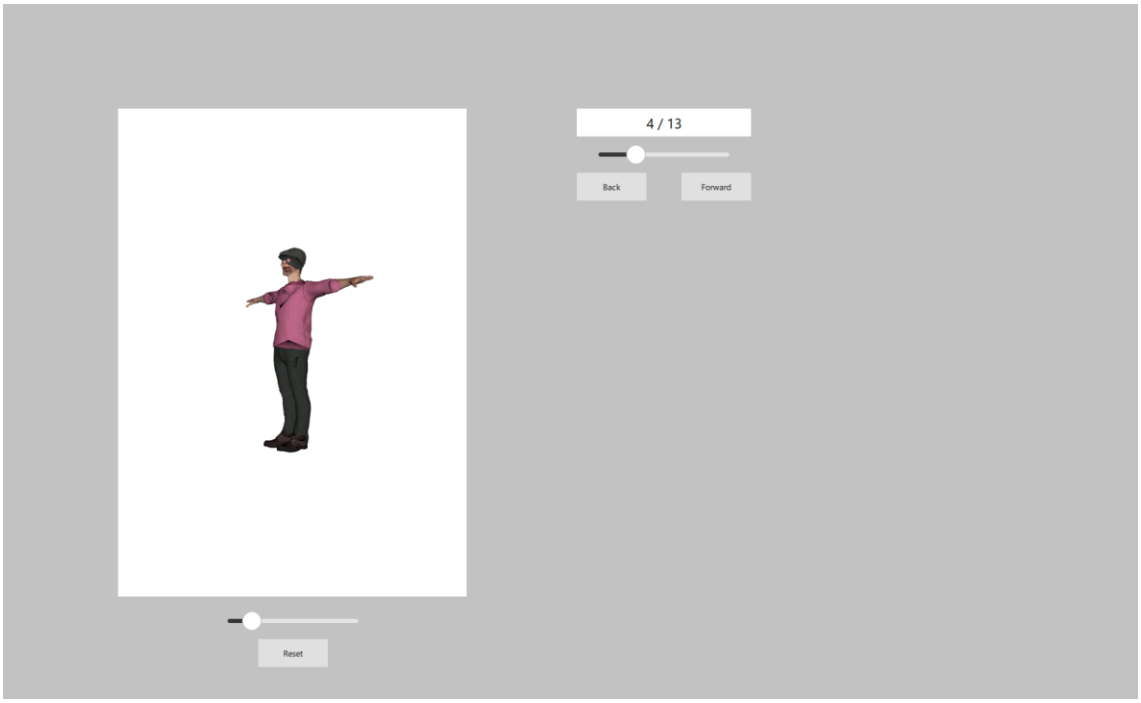
Lopullisessa tavoitteessa on pyörittää hahmoa kaikkiin suuntiin, mutta ensiksi tehtiin se vain yhden akselin ympäri. Koska ensimmäisenä haluttiin pyörittää sivusuunnassa, akseliksi valikoitui y-akseli. Ensiksi 3D-hahmon ikkunaan lisätiin

MouseArea-komponentti, jonka avulla alue ottaa huomioon hiiren tai kosketuksen toiminnan huomioon, ja tarjoaa niiden perusteella erilaisia signaaleja käyttöliittymän toimintojen luomiseksi [46]. Signaali, joka herää hiiren sivuttaisesta liikkeestä, on onMouseXChanged, jolle luotiin esimerkkikoodin 4 määrittelemä yhteys. Tämä oli ensimmäinen kokeilu saada pyöritys toimimaan.

```
mouseArea.onMouseXChanged: {
     malcolm.rotate((mouseArea.mouseX - 250) / 500, Qt.vector3d(0, 1,
     0), mouseArea.Center)
}
```

Esimerkkikoodi 4. QML-deklaraatio, joka pyörittää 3D-hahmoa malcolm hiiren tai kosketuksen x-koordinaatin määrittämän määrän hahmon y-akselin ympäri. Operaatiota suoritetaan painetun hiiren tai kosketuksen sivuttaisen liikkeen kutsumana.

Tällä 3D-hahmoa (malcolm) pyöritetään hiiren x-koordinaatin arvon mukaisesti, jaettuna 500:lla ja y-akselin ympäri ikkunan keskipisteen mukaan. Arvo jaettiin, jotta pyörimisnopeus saatiin hidastettua sopivammaksi. Arvosta myös ensin poistettiin 250, jotta hahmon ikkunan keskipisteen vasemmalla puolella hiirtä liikuttaessa hahmo pyörii myötäpäivään, ja oikealla puolella hiirtä liikuttaessa hahmo pyörii vastapäivään. Kuvassa 8 näytetään pyörimisen toiminta toteutuksen jälkeen.



Kuva 8. 3D-hahmo pyöritettynä ja zoomattuna etäämmälle hoito-ohjelmassa.

Edellä mainitussa toteutuksessa on useampi ongelma. Pyöritys myötä- tai vastapäivään toimii vain keskipisteen molemmin puolin, mikä ei ole tavallinen tapa tehdä pyöritystä käyttöliittymissä. Pyörityksen nopeus myös kiihtyy, mitä kauemmaksi liikutaan ikkunan keskipisteestä. Järkevämpään toteutukseen olisi tarvinnut x-koordinaatin muutoksen positiiviseen tai negatiiviseen suuntaan, mutta MouseArea-komponentti ei anna tätä informaatiota suoraan. Ongelmalle ei heti löytynyt ratkaisua, mutta Reset-painikkeelle annettiin joka tapauksessa uusi yhteys palauttamaan myös hahmon rotaatio takaisin alkuperäiseen arvoon eli 0:ksi.

Vaiheiden ajastuksen toteuttamisen jälkeen huomattiin, että ajastuksessa käytettyä Timer-tyyppiä voisi käyttää myös pyörityksessä. Koska MouseArea-komponentti antaa hiiren tai kosketuksen x-koordinaatin arvon, sen muutosta voi seurata ajastuksen kautta painalluksen tai kosketuksen ajan. Ensimmäiseksi x-koordinaatti tulee ottaa muuttujaan muistiin painallus- tai kosketushetkellä, jotta vertailu voidaan aloittaa jostain pisteestä. Ajastin asetettiin vakiona suljetuksi eli running-ominaisuus false-arvoon. Se myös asetettiin toistumaan 25

millisekunnin intervallein, kun se on käynnissä. Intervallin välein ajastin käynnistää funktion signaalin onTriggered seurauksena, jossa tapahtuu esimerkkikoodin 5 mukaiset operaatiot.

```
function rotationDirection() {
     itemRotation.rotationX = (screen01.mouseArea.mouseX - itemRota-
     tion.mouseX) / 2
     itemRotation.mouseX = screen01.mouseArea.mouseX
     screen01.malcolm.rotate(itemRotation.rotationX, Qt.vector3d(0, 1,
     0), screen01.malcolm.pivot)
{
```

Esimerkkikoodi 5. JavaScript-funktio, jossa pyöritetään 3D-hahmoa malcolm sivusuuntaisesti sen y-akselin ympäri riippuen hiiren tai sormen kulkusuunnasta oikealle tai vasemmalle. Funktiota suoritetaan ajastimen kutsumana.

Pyörimisen määrä asetettiin olemaan sen hetkisen x-koordinaatin erotus edellisessä intervallissa muuttujaan tallennetusta x-koordinaatista. Tällöin vasemmalle liikuttaminen aiheuttaa negatiivisen arvon ja siten myötäpäivään pyörimisen. Oikealla liikuttaminen vastoin aiheuttaa positiivisen arvon ja pyörimisen vastapäivään. Intervalli asetettiin 25 millisekuntiin, koska matalammat arvot aiheuttivat viivettä ja pyörimisen toimimattomuutta. Koordinaattien erotus jaettiin kahdella, jotta pyörimisnopeus saatiin sopivammaksi. Pyörimispiste (Pivot) asetettiin aiemmasta poiketen hahmon omaan pyörimispisteeseen.

Kun sivuttainen pyöritys saatiin toteutettua, jatkettiin pystysuuntaiseen pyörittämiseen. Siinä y-koordinaatin muutos yhdistettiin x-akselin ympäri tapahtuvaan pyörimiseen, millä alettiin kokeilemaan yhdistettyä pyöritystä. Vastaavanlaisesti sivuttaisen pyörittämisen kanssa y-koordinaatti tallennettiin painalluksen tai kosketuksen hetkellä muuttujaan ja esimerkkikoodin 5 ajastimen funktioon lisättiin esimerkkikoodin 6 operaatiot.

```
itemRotation.rotationY = (screen01.mouseArea.mouseY - itemRota-
tion.mouseY) / 2
itemRotation.mouseY = screen01.mouseArea.mouseY
screen01.malcolm.rotate(itemRotation.rotationY, Qt.vector3d(1, 0, 0),
screen01.malcolm.pivot)
```

Esimerkkikoodi 6. Pyörityksen ajastimen sisällä olevaan funktioon lisätyt JavaScript-operaatiot, joilla pyöritetään 3D-hahmoa ”malcolm” pystysuuntaisesti

sen x-akselin ympäri riippuen hiiren tai sormen kulkusuunnasta ylös tai alas. Funktiota suoritetaan ajastimen kutsumana.

Pyöritys toimi muuten hyvin, mutta jostain syystä 3D-hahmon pyörimispiste oli sen jaloissa, kun sen haluttaisiin olevan hahmon keskipisteessä. Kun hahmo pyörii jalkojen tason ympäri, se ei pysy kameran keskellä, vaan liikkuu sen rajauksen ulkopuolelle. Pisteen sai siirrettyä muokkaamalla hahmon y-akselin Pivot-ominaisuuden arvoa Properties-näkymässä, jolloin pyöritys toimi halutusti. Reset-painikkeelle luotiin vielä uuden pyörityssuunnan vuoksi uusi yhteys, joka nollaa rotaation x-akselin ympäri. Toiminnan testauksen jälkeen havaittiin, että tämä ei riitä, koska y- ja x-akseleiden ympäripyörityksen yhdistelmä aiheuttaa myös z-akselin ympäri tapahtuvaa pyörimistä. Reset-painikkeelle lisättiin myös yhteys z-akselin ympäri olevan rotaation nollaamiseen.

Reset-painikkeeseen liittyvistä havainnoista huomattiin myös, että pystysuuntainen pyöritys ei toimi täysin kuten sen pitäisi, kun pyöritystä tehdään vain x-akselin ympäri. Hahmo ei pysy ikkunan näkökulmassa pystyasennossa, vaan hahmon ollessa sivuttain se kaatuu. Pystysuuntainen pyöritys tulisi olla x- ja z-akselin ympäri tapahtuvien pyörimisten yhdistelmä riippuen y-akselin rotaatiosta. Tätä toteutettiin ensin luomalla astealueet, joissa x- ja z-akselin rotaatio muuttuu 0–100 % toisiinsa nähden vastakkain. Hetken kehittämisen jälkeen tämä koettiin liian monimutkaiseksi toteuttaa ja sen sijaan haettiin yksinkertaisempaa ratkaisua.

Y-akselin rotaatiota ei tarvitse ottaa huomioon, jos sivusuuntaisen eli y-akselin rotaation tekee hahmon pyörityksellä niin kuin aiemmin, mutta pystysuuntaisen eli x-z-tason pyörimisen kameran liikkeellä ja rotaatiolla. Kamera tulee siis saada puoliympyrän muotoiseen rataan hahmon ympärille pystysuunnassa niin, että kameran katse säilyy hahmon suunnassa. Kehitys aloitettiin yksinkertaisesta, ja siitä laajennettiin haasteellisempiin toimintoihin. Puoliympyrän asteet oli helppo lukita tekemällä liukusäätimen, jonka arvot ovat välillä –90...90. Kameran x-akselin rotaatio liitettiin liukusäätimen arvon negaatioon. Näin kamera katsoo yläasennossa alas ja ala-asennossa ylös. Seuraavaksi tuli määrittää y- ja z-koordinaattien muutos liukusäätimen arvojen muuttuessa niin, että ne

muodostavat ympyrän kaaren. Tässä käytettiin hyödyksi yksikköympyrän trigonometrian määritelmää eli käytettiin siniä ja kosinia Maknunin ym. artikkelin [47, s. 3] mukaisin kaavoin (kaavat 1 ja 2).

$$sin\alpha = \frac{y}{r} \quad (1)$$

$$cos\alpha = \frac{x}{r} \quad (2)$$

α on (x,y)-pisteen ja origon välisen suoran kulma x-akseliin
r on (x,y)-pisteen ja origon etäisyys toisistaan.

Kyseisiä kaavoja sovellettiin pyörityksen liukusäätimen arvonmuutoksesta käynnistyvän signaalin onValueChanged operaatioina esimerkkikoodin 7 mukaisesti. X-koordinaatin sijaan operaatioissa käsitellään z-koordinaattia.

```
sliderRotation.onValueChanged: {
     camera.eulerRotation.x = -sliderRotation.value
     camera.z = sliderZoom.zoomValue * Math.cos(sliderRotation.value *
     (Math.PI/180))
     camera.y = sliderZoom.zoomValue * Math.sin(sliderRotation.value *
     (Math.PI/180))
}
```

Esimerkkikoodi 7. JavaScript-operaatiot, joilla liikutetaan kameraa pystysuunnassa puoliympyrän muodossa 3D-hahmon edessä. Kameran katsesuuntaa pyöritetään liikkeessä kohti hahmoa. Operaatioita suoritetaan pyörityksen liukusäädön arvon muutoksen kutsumana.

Operaatioissa otetaan huomioon sen hetken zoomausarvo, joka toimii ympyrän säteenä. Säde kerrotaan z-koordinaatin kohdalla liukusäätimen asteiden kosinilla ja y-koordinaatin kohdalla asteiden sinillä. Asteet tuli vielä muuttaa radiaaneiksi.

Viimeisenä vaiheena pystysuuntainen pyöritys haluttiin vielä toteuttaa hiiren avulla. Jo aiemmin käytetyt esimerkkikoodin 6 operaatiot muokattiin esimerkkikoodin 8 mukaisiksi.

```
itemRotation.rotationY = (screen01.mouseArea.mouseY - itemRota-
tion.mouseY) / 2
itemRotation.mouseY = screen01.mouseArea.mouseY
screen01.sliderRotation.value += itemRotation.rotationY
```

Esimerkkikoodi 8. Pyörityksen ajastimen sisällä olevaan funktioon lisätyt JavaScript-operaatiot, joilla liikutetaan pyörityksen liukusäädintä riippuen hiiren tai sormen kulkusuunnasta ylös tai alas, mikä pyörittää kameraa hahmon edessä pystysuunnassa.

Hiiren y-koordinaatin muutos asetettiin siirtämään liukusäädintä ylös tai alas riippuen hiiren liikutuksen suunnasta. Arvo jaettiin kahdella kuten sivuttaisen pyörityksen kohdalla hidastamaan pyöritystä, jotta se tuntuu sopivalta.

Uudenlainen tapa tehdä pyöritystä aiheutti myös tarpeen suunnitella zoomaus uudella tavalla, ja lisätä Reset-painikkeelle sen mukaiset ominaisuudet. Suunnittelu ei kuitenkaan vaatinut suuria ponnisteluja. Zoomauksen käskyissä oli mahdollista käyttää täysin samoja kaavoihin 1 ja 2 perustuvia operaatioita kuin pystysuuntaisessa pyörityksessä aiemmin. Zoomauksen liukusäätimen signaalin onValueChanged-operaatiot muokattiin esimerkkikoodin 9 mukaisiksi.

```
sliderZoom.onValueChanged: {
     sliderZoom.zoomValue = sliderZoom.value
     camera.z = sliderZoom.zoomValue * Math.cos(sliderRotation.value *
     (Math.PI/180))
     camera.y = sliderZoom.zoomValue * Math.sin(sliderRotation.value *
     (Math.PI/180))
}
```

Esimerkkikoodi 9. JavaScript-operaatiot, joilla määritetään kameran etäisyys 3D-hahmosta, riippuen sen asennosta puoliympyrän muotoisessa liikeradassa. Etäisyys otetaan myös muistiin muuttujaan pyöritystä varten. Operaatioita suoritetaan zoomauksen liukusäädön arvonmuutoksen kutsumana.

Tällä zoomauksen liukusäädin muokkaa ympyräoperaatioiden sädettä, ja siten muuttaa kameran etäisyyttä hahmosta. Reset-painike käyttää samoja operaatioita hyödykseen muuttamalla zoomauksen liukusäädön vakioarvoon, mikä käynnistää kyseiset operaatiot.

Zoomauksessa syntyi myös ongelma nipistyksen ja pystysuuntaisen pyörityksen välille. Nipistyszoomaus sai myös pyörityksen käyntiin. Tätä pyrittiin korjaamaan sulkemalla MouseArea-komponentin toiminta muuttamalla sen enabled-

ominaisuus false-arvoon signaalissa onPinchStarted eli nipistyksen alussa. MouseArea asetettiin jälleen käynnistymään signaalissa onPinchFinished eli nipistyksen lopussa. Tämä korjasi ongelman, ja molemmat ominaisuudet toimivat toisistaan erillään.

4.3.3 3D-hahmon asennon muutokset ja animaatiot

3D-hahmon asennon muokkaus eri vaiheita varten olisi ollut helpointa toteuttaa hyödyntäen 3D Editor -näkymää ja sen monia työkaluja. Näistä työkaluista erityisen hyödyllinen tähän tarkoitukseen olisi ollut rotaatiotyökalu. [18.] Ongelmaksi sen käytössä muodostui se, että Mixamosta haetun 3D-hahmon luurangon nivelistä suurin osa oli harmaana rotaatiotyökalulle, eli niitä ei voinut muokata. Toista Mixamon hahmoa kokeiltiin ja kolmatta eri sivustolta, mutta kaikkien kohdalla oli sama ongelma. Tuonti (import) -asetuksissa oli vaihtoehto ”FBX: Preserve Pivot Points”. Sen valinta tosin joko heitti hahmon luurangon sekaisin tai ei tehnyt mitään. Tämän jälkeen kokeiltiin nivelten rotaatiota koodista käsin hyödyntämällä Node-tyypin ominaisuutta eulerRotation, jolla on mahdollista muokata rotaatiota kaikkien kolmen akselin ympäri [48]. Rotaatio saatiin aikaiseksi onnistuneesti, ja se näkyi 3D Editor -näkymässä reaaliaikaisesti.

Koska haluttiin eri asennot eri vaiheissa, asennon muutokset tuli tehdä vaiheita vastaavien tilojen mukaan. Hahmon nivelien muokkauksia ei kuitenkaan voi tehdä siitä tiedostosta käsin, jossa vaiheita varten luodut tilat ovat, vaan niitä voi tehdä vain hahmon omassa tiedostossa. Tämän vuoksi hahmon tiedostoon tehtiin vastaavat tilat, ja niille annettiin samat ehdot eli yhteys vaiheiden liukusäätöön. Tilojen muutoksiin annettiin nivelien rotaatioarvot, millä saatiin haluttu asento.

Tilojen välisten muutosten animoinnille on Qt-kehyksessä täysin oma tyyppi Transitions [49]. Itse animaatiot tehtiin Animation-tyypin alatyypillä NumberAnimation ja niiden ominaisuuksilla [50;51;52]. Jokaiselle nivelen rotaatiosuunnalle (x, y tai z) tuli tehdä oma animaatio, jolle määritettiin kesto millisekunneissa, arvo, johon ne animoituvat alkuperäisestä, ja easing-kaari, mikä

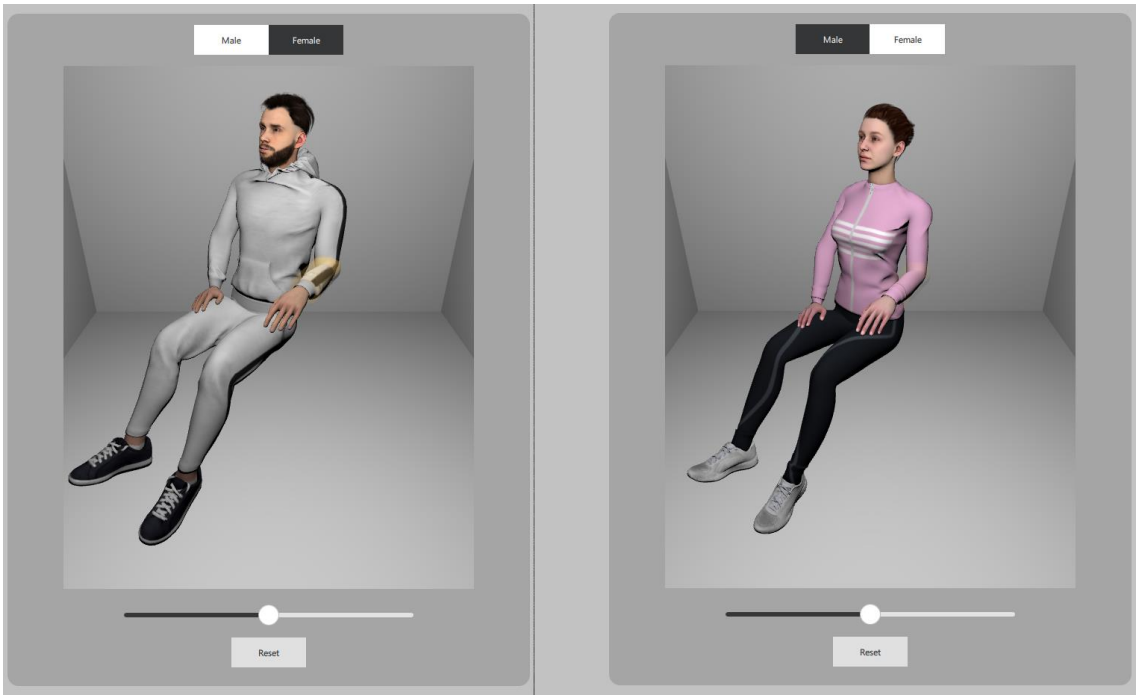
määrittää animaation kulun. Siirtymä (Transition) määritettiin tarvittaessa myös kameralle, joka joissain vaiheissa muutti zoomausta ja siirtyi hoidettavan alueen kohdalle kuten on tapahtunut kuvassa 9. Jälleen tässä tilanteessa piti ottaa huomioon Reset-painike ja muuttaa sen zoomauksen vakioarvo vastaamaan tilojen omaa arvoa. Myös tuli muuttaa zoomauksen liukusäädön vakioarvo.

Kuva 9. 3D-hahmon muuttunut asento ja kameran zoomaus käteen vaiheesta 1 vaiheeseen 2 siirryttäessä hoito-ohjelmassa. Käsi on siirtynyt sivulta reiden viereen, ja kamera siirtänyt käden ikkunan keskelle sekä zoomatuksi.

Yksi ongelma ilmeni ajon aikana. Siirtymä aiemmasta tilasta myöhempään toimi odotetusti, mutta siirtymä takaisin aiempaan ei aloittanut haluttua animaatiota. Molemmille oli määritetty siirtymä ja niiden animaatiot erikseen. Ongelma saatiin korjattua poistamalla toinen siirtymistä ja korvaamalla lähtötila merkillä ”*”. Tämä merkki tarkoittaa sitä, että siirtymässä oleva animaatio tapahtuu mistä tahansa lähtötilasta siirryttäessä määritettyyn lopputilaan.

Edellä mainituilla asennonmuokkaus- ja animointitavoilla oli tarkoitus luoda proof-of-concept vaiheet tietyn hoidettavan alueen ohjelmalle. Tätä ennen tuli korjata hahmossa olevat visuaaliset ongelmat ja virheviestit, joita on jo läpikäyty luvun 4.3 alussa. Mixamosta [42] löydettiin useampi hahmo, jossa ei ilmennyt

aiemman Malcolm-assetin visuaalisia ongelmia eikä niihin todennäköisesti liittyneitä virheviestejä. Hahmoista valikoitui yksi mieshahmo (Adam) ja yksi naishahmo (Jody). Hahmoja valikoitiin kaksi, koska päätettiin tehdä tavoitteisiin nähden ylimääräinen ominaisuus valita tarkasteltava hahmo. Hahmot voi nähdä kuvassa 10.



Kuva 10. Kaksi valittavaa hahmoa hoito-ohjelman näytön vaiheessa 2. Vasemmalla mieshahmo (Adam) ja oikealla naishahmo (Jody). Hahmot on haettu Mixamosta.

Hoidettavaksi alueeksi valikoitui vasen kyynärvarsi, ja sen ympärille luotiin yksinkertaiset vaiheet. Vaiheisiin kuuluuvat seuraavat vaiheet.

- Vaihe 1, jossa hahmo seisoo.
- Vaihe 2, jossa hahmo istuutuu kuviteltavaan hoitopenkkiin, ja asettaa kyynärvarren kuviteltavalle hoitopöydälle kämmenpohja alaspäin. Hahmo pyörii myötäpäivään, jotta hoitoalue on kameraan päin, ja kamera nousee hieman korkeampaan katsekulmaan. Vaihe on nähtävissä kuvassa 10.
- Vaihe 3, jossa kamera muuttaa zoomausta lähemmäksi hoitoaluetta, ja hahmo pyörii lisää myötäpäivään, jotta hoitoalue on vielä enemmän kohti kameraa. Hahmon pää kääntyy katsomaan hoitoaluetta. Vaihe on nähtävissä kuvassa 11.

- Vaihe 4, jossa hahmon kyynärvarsi pyörii niin, että kämmenpohja on ylöspäin.
- Vaihe 5, jossa kyynärvarsi pyörii takaisin alkuperäiseen asentoon, hahmon pää kääntyy takaisin katsomaan eteen, ja hahmon rotaatio sekä kameran zoom palautuu vaiheen kaksi asentoon.
- Vaihe 6, jossa hahmo nousee seisomaan, ja hahmon rotaatio sekä kameran asento palautuu vaiheen yksi asentoon.

Käyttöliittymän tilojen määrä määritettiin luonnollisesti samaksi. Tosin hahmon tilojen kohdalla lisättiin yksi ylimääräinen tila, jossa määritettiin sellaiset asennonmuutokset, jotka pätivät kaikkiin vaiheisiin. Tilan muutokset sai jatkettua muihin tiloihin käyttämällä tilojen ominaisuutta extend.

Siirtymäanimaatioissa nivelten liikkeiden järjestystä muokattiin muuttamalla niiden easing-kaaria. Suurimmassa osassa animaatioista käytettiin kaarta, joka on nopea alussa, hidastuu keskellä ja nopeutuu taas lopussa. Sellaisiin animaatioihin, joiden haluttiin tapahtuvan ensin, asetettiin alkuun nopea ja sen jälkeen hidastuva kaari. Lopussa tapahtuville valittiin vastakkainen kaari. Kaarien valinnassa yritettiin myös ottaa huomioon vaiheissa taaksepäin liikkuminen.

Asennonmuutokset ja animaatiot tehtiin ensin yhdelle hahmolle, jonka jälkeen ne kopioitiin toiselle. Molemmat asetettiin tapahtumaan samanaikaisesti, jotta hahmoa vaihtaessa asento on täysin sama. Ensimmäisen hahmon näkyvyys eli visible-ominaisuus asetettiin todeksi eli true-arvoon ja toisen epätodeksi eli false-arvoon. 3D-hahmon ikkunan ylle lisättiin sarkainpalkki (tab bar) ja siihen kaksi sarkainpainiketta (tab button) Male ja Female. Painikkeista toinen on päällä ja toinen poissa riippuen valinnasta. Male-painikkeen päällä oloon yhdistettiin ensimmäisen hahmon näkyminen ja toisen näkymättömyys, kuten se oli aiemmin määritetty. Female-painikkeeseen yhdistettiin vastakkaiset käskyt, eli sen painaminen saa toisen hahmon näkyviin. Painikkeet ovat nähtävissä kuvassa 10 3D-hahmojen ikkunoiden yläpuolella.

4.3.4 Hoitoalueen esitys ja hahmonvaihto

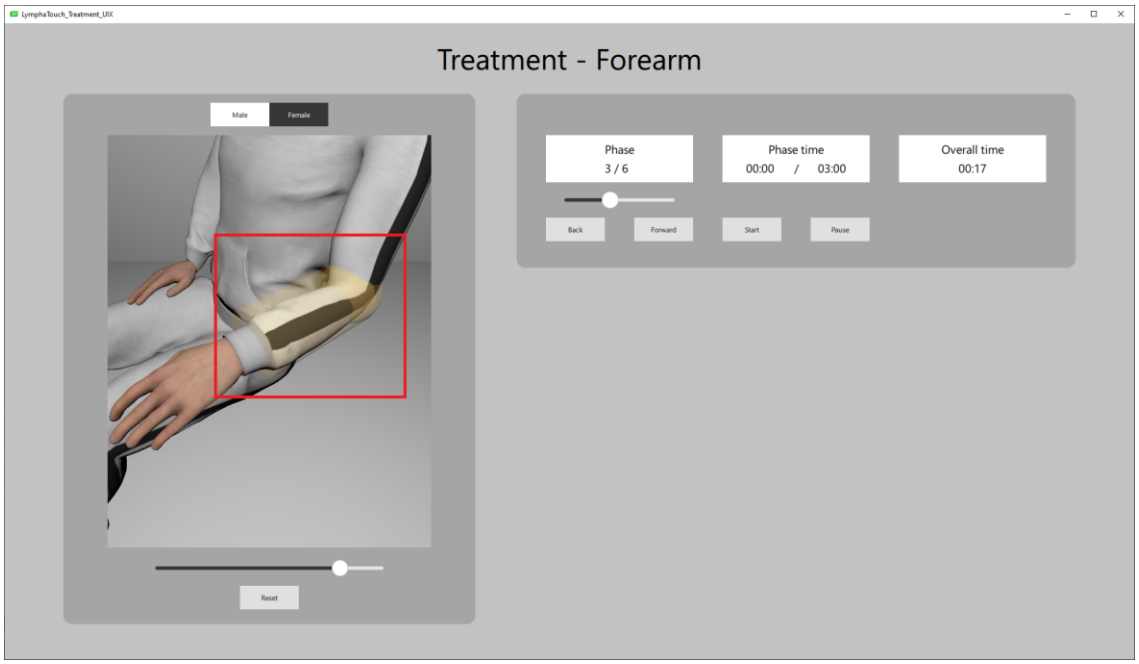
Valittua hoitoaluetta yritettiin ensin esittää Glow-komponentilla, jolla on mahdollista saada aikaa hohtoefekti joissain asseteissa. 3D-assetit eivät kuitenkaan ole niitä. Toisena mahdollisena ideana kokeiltiin käyttää pistemäistä valonlähdettä, jota on jo käytetty 3D-näkymän valaisemiseen. Hahmon kyynärvarren niveleen lisättiin valonlähde, sen leviämisalue pienennettiin ja kirkkautta kasvatettiin. Tämä ei aiheuttanut sellaista selkeää hehkuvaa aluetta, kuin olisi haluttu.

Kolmantena ideana kokeiltiin valmiita 3D-komponentteja. Tällä kertaa kyynärvarren niveleen eli kyynärpäähän lisätiin palloelementti. Elementti asetettiin hieman kyynärpäätä isommaksi, ja sen läpinäkyvyys eli opacity-ominaisuus muutettiin 0,05:een eli 5 prosenttiin. Lisäksi samaan niveleen lisättiin sylinterielementti, ja sen y-akselin sijaintia ja pituutta muokattiin niin, että se alkaa ranteen alapuolelta ja päättyy kyynärpäähän. Läpimitta ja läpinäkyvyys asetettiin samaksi kuin palloelementin. Molemmille elementeille lisättiin aikajana- eli timeline-animaatio, jolla saatiin aikaiseksi pulssimainen efekti. Palloelementin animaation koodin voi lukea esimerkkikoodista 10.

```
Timeline {
     id: timelineTreatmentAreaS
     animations: [
          TimelineAnimation {
               id: timelineAnimationTreatmentGlowS
               running: true
               duration: 2000
               loops: Animation.Infinite
               target: sphereMaterial
               property: "opacity"
               pingPong: true
               to: 0.25
               from: 0.05
               easing.type: Easing.InOutQuad
          }
     ]
     startFrame: 0
     enabled: true
     endFrame: Animation.Infinite
}
```

Esimerkkikoodi 10. Aikajana-animaatio, jossa pallokomponentin materiaalin läpinäkyvyyttä muutetaan 5–25 %:n välillä kahden sekunnin sykleissä loputtomiin.

Hoito-ohjelman käynnistymisestä lähtien hoidettavalle alueelle lisättyjen elementtien läpinäkyvyys muuttuu 5 prosentista 25 prosenttiin ja takaisin. Tämä tapahtuu pulssimaisesti 2 sekunnin sykleissä. Hoitoalueen esitystavan voi nähdä kuvassa 11.



Kuva 11. Hoito-ohjelman vaihe 3, jossa hoidettavan alueen esitystapa näkyy punaisen ruudun sisällä.

Toteutettu hoitoalue liikkuu nivelen liikkeiden mukana, joten sitä ei tarvinnut ottaa huomioon hahmon asentojen muutoksissa ja animaatioissa. Sykkimisen animaatio toimii myös erillään asentojen animaatioista ja toimii normaalisti vaiheiden välillä.

5 Käytettävyysarviointi

Luvun 4 mukaisesti toteutetulle ohjelmistokäyttöliittymälle tehtiin vapaamuotoinen formatiivinen arviointi eli käytettävyysarviointi. Arviointi tehtiin asiantuntija-arviona, missä asiantuntijana toimi itse työn kehittäjä. Arvioinnissa käytettiin kahta metodia standarditarkastusta ja heuristista arviointia. Arviointien läpikäynti ja tulokset selostetaan seuraavissa luvuissa 5.1 ja 5.2. Luvussa 5.3 vedetään yhteen tutkimuskysymyksen kannalta merkittävimmät tulokset.

5.1 Standarditarkastus

Tämän työn ohjelmistokäyttöliittymälle tehtiin liitteen 1 mukainen standarditarkastus. Käyttöliittymälle ja sen elementeille kerättiin standardin IEC 62366 teknisestä raportista ja standardin ISO 9241 osasta 161 löytyvät vaatimukset. Vaatimukset jaettiin taulukoihin niin, että yhdessä taulukossa on yleiset vaatimukset koko käyttöliittymälle ja lopuissa taulukoissa vaatimukset yksittäisille elementeille.

Vaatimusten täyttymiselle annettiin arvio. 29 vaatimuksesta 20 täyttyi täysin, kuusi osittain, kaksi ei täyttynyt ollenkaan, ja yhden kohdalla ei pystytty arvioimaan täyttymistä. Kahdesta vaatimuksesta, jotka eivät täyttyneet, ja kuudesta, jotka täyttyivät vain osittain, viisi koettiin ongelmaksi. Ongelmat koottiin taulukkoon, jossa myös määritettiin voiko ongelma johtaa käyttövirheisiin. Viidestä ongelmasta kahden määriteltiin voivan johtaa käyttövirheisiin käyttöliittymän nykyisessä tilassa. Nämä kaksi ongelmaa liittyivät painikkeiden liian pieneen fonttikokoon ja zoomauksen liukusäädöstä puuttuvaan nimikkeeseen.

5.2 Heuristinen arviointi

Standarditarkastuksen jälkeen ohjelmistokäyttöliittymälle tehtiin liitteen 2 heuristinen arviointi. Arvioinnissa käytettiin Nielsenin heuristiikoita, joita on yhteensä kymmenen. Käyttöliittymää arvioitiin soveltaen heuristiikkoja yksi kerrallaan. Arvioinnin dokumentaatio tehtiin sanallisesti ja siinä käsiteltiin sekä ongelmia että

positiivisia havaintoja. Arvioinnissa pyrittiin myös ottamaan huomioon päällekkäiset havainnot heuristiikkojen välillä.

Arvioinnin havainnoista kerättiin ongelmat yhteen taulukkoon, jossa myös määritettiin, voiko ongelma johtaa käyttövirheisiin. Ongelmia löydettiin yhteensä yhdeksän, joista kahdeksan määritettiin voivan aiheuttaa käyttövirheitä käyttöliittymän tämänhetkisessä tilassa. Ongelmia löytyi kuudesta eri heuristiikasta. Kahta heuristiikkaa ei voitu arvioida ollenkaan käyttöliittymän nykyisessä tilassa. Ongelmat liittyivät pääasiassa painikkeisiin, vaiheajastimeen ja 3D-hahmon ikkunaan.

5.3 Arvioinnin yhteenveto

Koko käytettävyysarvioinnissa löydettiin 13 uniikkia ongelmaa. Yksi ongelmista löytyi molempien metodien ongelmalistalta. Molemmista metodeista löytyi myös ongelmia, jotka liittyivät samaan käyttöliittymän elementtiin, mutta ongelmat olivat hieman eri näkökulmasta. Positiivisia huomioita tehtiin lähinnä vain heuristisessa arvioinnissa, koska standarditarkastus oli vain vaatimusten täyttymisen arviointia.

Huomioitavaa tämän tutkimustyön tutkimuskysymyksen kannalta on se, että standarditarkastuksessa ei löydetty suoraan 3D-hahmoon liittyviä vaatimuksia. Standarditarkastus ei siis loppujen lopuksi tarkastellut työn kannalta merkityksellisintä elementtiä, mutta arvioi hyvin kaikkia muita elementtejä. Tarkastus löysi tosin yhden saman ongelman heuristisen arvioinnin kanssa, joka liittyi 3D-hahmoon liittyvään elementtiin.

Seuraavana vaiheena formatiivisessa arvioinnissa olisi arvioida käyttövirheiden vaikutusta turvallisuuteen, tehdä riskianalyysi ja luoda suunnitelma riskien eliminointiin, minimointiin tai niistä informoimiseen [33, s. 41–43]. Tässä työssä arviointi kuitenkin rajattiin vain pelkkään käytettävyysarviointiin, koska sitä käytetään yhtenä työkaluna tutkimuskysymykseen vastaamisessa.

Taulukkoon 3 on kerätty kaikki 3D-hahmoon tai siihen liittyviin elementteihin liittyvät ongelmat ja positiiviset havainnot. Niitä käytetään hahmon hyötyjen ja haittojen arviointiin, ja myöhemmin niiden pohtimiseen luvussa 7.3.

Taulukko 3. Insinöörityössä kehitetyn käyttöliittymän 3D-hahmon ja siihen liittyvien elementtien ongelmat ja positiiviset havainnot käytettävyysarvioinnista.

Ongelmat	**Positiiviset havainnot**
Zoomauksen liukusäätimellä ei ole nimikettä tai otsikkoa.	3D-hahmon asennolla esitetään ohjeita, jotka vaatisivat normaalisti tekstiä tai 2D-kuvia.
Hahmonvaihdon painikkeille ei ole otsikkoa.	3D-hahmon asentoa voi tutkia pyörittämällä tai zoomaamalla, mitä ei voi tehdä 2D-kuvilla.
3D-hahmon nipistyszoomauksen ja pyörityksen toimintoja ei esitetä tai ohjeisteta millään tavalla.	3D-hahmolla voisi mahdollisesti esittää muitakin käyttöliittymän osia, jolla voi kasvattaa minimalismia.
Käyttöliittymä ei kysy varmistusta hahmon asennon palauttamiseen.	3D-hahmon voi vaihtaa lennossa.
Käyttöliittymässä ei ole ohjeistusta elementtien käyttöön.	

3D-hahmossa ja siihen liittyvissä elementeissä havaittiin enemmän ongelmia (5 kpl) kuin positiivisia havaintoja (4 kpl), mutta ongelmat liittyivät enemmän toteutuksen vajavaisuuksiin kuin 3D-mallin haittoihin. Ongelmat myös liittyvät vahvasti toisiinsa. Positiiviset havainnot liittyvät kaikki siihen, mitä 3D-hahmolla voi esittää tai ohjeistaa.

6 Tulokset

Tässä insinöörityössä tehdyn kehitystyön lopputuloksena on ohjelmistokäyttöliittymä lääkinnällisen hoitolaitteen hoito-ohjelmalle. Työ on keskittynyt käyttöliittymän toiminnallisuuteen. Kehitettävät ominaisuudet sovittiin etukäteen työn tilaajan kanssa, ja asetettiin työn tavoitteiksi. Käyttöliittymän toiminnallisuudet ovat se osa-alue, jota työn tilaaja haluaa työstä hyödyntää. Taulukossa 4 on listattuna tavoitteisiin liittyvät vaatimukset ja niiden täyttyminen tässä työssä.

Taulukko 4. Kehitystyön tavoitteiden vaatimuksien täyttyminen.

id	**Vaatimus**	**Täytetty**
KeT1	Jokaiselle hoidon vaiheelle tulee määrittää sille ominainen tila ja näyttää ikkunassa, missä vaiheessa ollaan.	Kyllä
KeT2	Vaiheiden välillä tulee pystyä siirtymään painikkeen painalluksella.	Kyllä
KeT3	Jokaiselle hoidon vaiheelle tulee luoda ajastin, ja määritettävä tavoiteltava aika, johon asti lasketaan aikaa. Lisäksi koko hoito tulee ajastaa.	Kyllä
KeT4	Valittu 3D-hahmon asset tulee saada tuotua näkyviin sille luotuun ikkunaan.	Kyllä
KeT5	3D-hahmoa tulee pystyä zoomaamaan sen ikkunassa, ja se tulee olla mahdollista kahden sormen nipistyksellä kosketusnäytössä (pinch-to-zoom).	Kyllä
KeT6	3D-hahmoa tulee pystyä pyörittämään sekä sivu- että pystysuunnassa käyttäen kosketusnäyttöä.	Kyllä
KeT7	3D-hahmon zoomaus ja pyöritys tulee pystyä palauttamaan vaiheen alun asetuksiin.	Kyllä
KeT8	3D-hahmon asennon tulee pystyä muuttumaan vaiheesta toiseen.	Kyllä
KeT9	3D-hahmon eri vaiheissa olevat asentojen erot tulisia animoitua paikoilleen vaiheiden siirron välillä.	Kyllä
KeT10	Hoito-ohjelman hoitoalue tulee esittää jollain tavalla 3D-hahmon kehossa.	Kyllä

Jokaiselle hoidon vaiheelle on tila ja ikkuna, jossa esitetään sen hetkinen vaihe. Jokaista vaihetta vastaavalla tilalla on oma vaiheluku, joten tilan vaihtaminen vaihtaa myös vaihenumeroinnin. Vaiheiden välillä siirrytään painikkeiden avulla. Painikkeet siirtävät liukusäädintä eteen tai taakse, joka käynnistää sitä arvoa vastaavan tilan ja siten vaiheen. Nämä täyttävät tavoitteet KeT1–2.

Sekä koko hoidolle että vaiheille on tehty ajastimet. Niissä hyödynnetään Qt-kehyksen valmista ajastinkomponenttia, jolla ajettiin syklisesti aikaa edistävää funktiota. Molemmat ajastimet on sijoitettu vaiheiden ikkunaan. Koko hoidon ajastin lähtee käyntiin hoito-ohjelman auetessa, ja vaiheajastimen saa käyntiin painikkeella. Toisella painikkeella voi myös laittaa vaiheajastimen tauolle, jos sille on tarvetta. Painikkeet lisättiin ylimääräisenä ominaisuutena vaatimukseen KeT3 nähden, jotta hoitovaiheen voi aloittaa silloin, kun on valmis. Vielä viimeisen tavoitteen KeT3-vaatimuksen täyttämiseksi jokaiselle vaiheelle on oma maksimiaika, jonka käyttöliittymä hakee vaiheikkunaan. Vaiheajastin pysähtyy maksimiajan tullessa täyteen. Toisena ylimääräisenä ominaisuutena maksimiajan tullessa täyteen vaiheajan teksti muuttuu lihavoiduksi ja väriltään punaiseksi. Tällä ilmoitetaan vaiheen päättyminen.

Käyttöliittymään tuotiin Mixamosta haettu asset 3D-hahmoksi, joka asetettiin omaan ikkunaan vaiheikkunan viereen, mikä täyttää tavoitteen KeT4. Hahmoa voi zoomata kahdella eri tavalla: liukusäätimellä ja nipistyseleellä kosketusnäytöllä. Näistä jälkimmäinen on osa tavoitteen KeT5 vaatimusta, ja täyttää sen. Nipistys on liitetty liukusäätimeen, joten molemmat liikkuvat aina oikeassa suhteessa toisiinsa. Nipistyksen kokoa seurataan Qt-kehyksen valmiilla PinchArea-komponentilla. Zoomaus muuttaa 3D-ympäristön kameran etäisyyttä hahmosta.

Hahmoa voi pyörittää sekä sivu- että pystysuunnassa, kuten tavoite KeT6 vaatii. Sivusuuntaisessa pyörityksessä hahmo pyörii oman akselinsa ympäri, ja pyörityksen määrä määräytyy kosketuksen x- eli sivusuuntaisen liikkeen mukaan 3D-hahmon ikkunan päällä. Pystysuuntaisessa pyörityksessä liikutetaan 3D-ympäristön kameraa hahmon edessä puoliympyrän muodossa, ja pyörityksen määrä määräytyy kosketuksen y- eli pystysuuntaisen liikkeen mukaan 3D-hahmon

ikkunan päällä. Kameran asento on otettu huomioon myös zoomauksessa. Kosketuksen x- ja y-koordinaattien muutosta seurataan ajastimen ajamalla funktiolla.

Tavoitteen KeT7 mukaisesti kaikki zoomauksessa ja pyörityksessä tehdyt muutokset hahmon ja kameran asennolle voi palauttaa vaiheen alussa olleisiin arvoihin. Palautus tehdään 3D-hahmon ikkunan alla olevan painikkeen avulla. Jokaisella vaiheella on omat arvonsa, ja käyttöliittymä ottaa nämä arvot aina muistiin vaiheen alussa. Painike palauttaa myös liukusäätimien asennon, jotta ne vastaavat palautettuja arvoja.

Hahmon asento muokkaantuu vaiheitten välillä, ja vaiheen vaihtuessa hahmo animoituu uuteen asentoon tavoitteiden KeT8–9 mukaisesti. Asennot on suunniteltu kuvitteellisen kyynärvarren hoito-ohjelman mukaan. Asennon muutokset on tehty muokkaamalla 3D-hahmon nivelien akselien rotaatioita, ja animaatioissa käytettiin Qt-kehyksen valmiita animointikomponentteja. Vaiheitten välisissä animaatioissa muuttuu välillä myös kameran asento.

Hoito-ohjelman hoitoaluetta esitetään Qt-kehyksen valmiilla 3D-komponenteilla, jotka on asetettu hoitoalueen nivelen sisään. Niiden koko on hieman hoitoaluetta suurempi, ja niiden läpinäkyvyys on asetettu niin, että hoitoalue näkyy alta. Esitysalue asetettiin pulssimaisesti sykkimään käyttämällä aikajana-animaatiota. Animaatio muuttaa 3D-komponenttien läpinäkyvyyttä 5–25 %:n välillä edestakaisin 2 sekunnin sykleissä. Tämä täyttää viimeisen tavoitteen KeT10.

Vielä yhtenä ylimääräisenä ominaisuutena tavoitteiden lisäksi käyttöliittymässä voi vaihtaa hahmoa 3D-hahmon ikkunan yläpuolella olevilla painikkeilla. Painikkeet muuttavat hahmojen näkyvyyttä päälle ja pois niin, että vain toinen niistä näkyy. Molemmilla hahmoilla on täysin identtiset asennot ja animaatiot, joten ne ovat aina samassa asennossa.

7 Pohdinta

Tämän insinöörityön pohdinta on jaettu neljään osa-alueeseen. Niistä ensimmäiset kaksi käsittelee työssä käytettyjä menetelmiä, kolmannessa käsitellään työn tutkimuskysymystä ja siihen liittyviä tuloksia, ja viimeisessä pohditaan, miten tätä aihetta kannattaisi tutkia tulevaisuudessa.

7.1 Qt-kehitys

Käyttöliittymien kehitys Qt Design Studio -ohjelmistolla ei ollut ennalta tuttua ennen tätä insinöörityötä. Qt-kehys ja QML-ohjelmointikieli olivat myös uutta. Tämä vaikutti paljon siihen, millä tavalla kehitystä lähestyttiin. Sen sijaan, että ominaisuuksia olisi suoraan kehitetty tavoitteiden mukaiseksi, aloitettiin paljon pienemmästä. Harjoittelun kannalta ominaisuuksia yritettiin saada ensin toimimaan tavalla millä hyvänsä. Tällä opittiin ohjelmiston ja kehyksen toimintaa, jotta ominaisuudet saatiin lopulta kehitettyä tavoitteiden mukaisiksi.

Hankaluuksia aiheutti kehityksen alussa se, että Qt:n dokumentaatio ei ollut kovin aloittelijaystävällinen. Eri komponenttien ja tyyppien toimintaa joutui yleensä kokeilemaan eri tavoin sen sijaan, että olisi löytänyt tarvittavan tiedon dokumentaatiosta. Yksi Qt Design Studion tärkeistä näkymistä 3D-Editor löytyi vasta kehityksen myöhemmässä vaiheessa työn tilaajan ohjaajan avustuksella.

Kehitetyssä käyttöliittymässä on edelleen joitain huonoja ohjelmointiratkaisuja johtuen Qt-kehyksen kummallisuuksista, joille ei ole löytynyt vastauksia heidän dokumentaatiostansa. Ongelmat liittyvät komponenttien ominaisuuksien kutsumiseen toisesta tiedostosta. Esimerkiksi 3D-hahmon niveliin ei pääse käsiksi hahmon tiedoston ulkopuolella. Työn tilaajan ohjaaja kyseli joistain ongelmista Qt:lta tuen kautta, mutta kyselyihin joko ei vastattu tai annettiin geneerisiä vastauksia, mistä ei loppujen lopuksi ollut apua.

Edellä mainituista vaikeuksista huolimatta, kun kehityksessä päästiin vauhtiin, sujui se suhteellisen hyvin ja nopeasti. Kehityksessä lopulta meni noin puolet

vähemmän aikaa, kuin alun perin suunniteltiin. Qt Design Studion eri näkymät ja työkalut nopeuttivat työskentelyä, ja kehyksen valmiit tyypit ja komponentit toimivat suurimmaksi osaksi niin, kuin niiden odottaisikin toimivan. Dokumentaation tulkinta myös helpottui työn edetessä. Yksi tarvittava tyyppi ilmoitusviestin kehitykseen tosin puuttui kehyksen uusimmasta versiosta 6, koska version 5 ominaisuuksia tuodaan uudempaan versioon pikkuhiljaa päivitysten muodossa. Vaiheajan ilmoitusviestiä ei lopulta saatu kehitettyä, koska päivitys viivästyi.

Käyttöliittymä ei tule sellaisenaan käyttöön siihen tarkoitettuun laitteeseen. Kehitystyöstä hyödynnetään eri toiminnallisuuksien toteutustapoja, ja niitä sovelletaan varsinaiseen tulevaan käyttöliittymään. Työllä on näytetty työn tilaajalle, miten tavoitellut toiminnallisuudet voi tehdä, ja samalla tuotu esiin mahdollisia ongelmia alustan kanssa.

7.2 Käytettävyysarviointi

Käytettävyysarviointi ei ollut työn alkuperäisessä suunnitelmassa, vaan suunnitelma siitä kehkeytyi taustatyötä tehdessä. Toteutetun käyttöliittymän tuloksia haluttiin jollain tavalla verrata kerättyyn tutkimustaustaan. Käytettävyysarviointi koettiin parhaaksi tavaksi saada vertailtavia tuloksia.

Asiantuntija-arvio oli sopiva kehityksen tilaan nähden, mutta parempia tuloksia olisi saatu suuremmalla määrällä asiantuntijoita. Aikaa ei tosin ollut tarpeeksi asiantuntijoiden keräämiseen ja arviointien järjestämiseen. Se olisi ollut mahdollista, jos käytettävyysarviointia olisi suunniteltu varhaisemmasta vaiheesta lähtien.

Valituista metodeista heuristinen arviointi toimi paremmin tutkimuskysymykseen nähden. Standarditarkistukseen valitut standardit eivät sisältäneet sellaisia vaatimuksia, joilla olisi voinut suoraan arvioida 3D-mallien käyttöä. Muita sopivia standardeja tähän saattaa olla olemassa, mutta niistä ei oltu tietoisia.

7.3 3D-mallien hyödyt ja haitat käyttöliittymissä

3D-mallien hyötyjä ja haittoja ohjelmistokäyttöliittymissä on pyritty valaisemaan olemassa olevan tutkimustaustan sekä kehitetyn käyttöliittymän käytettävyysarvioinnin kautta. Tutkimustaustassa toistuvana teemana ilmeni 3D-visualisaatioiden tuoma suurempi informaation määrä verrattuna 2D-visualisaatioihin. Tämä ei ole sellaisenaan tulkittavissa hyödyksi tai haitaksi, vaan se riippuu täysin sovelluksesta ja sen käyttäjistä.

Yksi suuremman informaation määrän hyödyistä, jotka havaittiin taustatutkimuksessa, on tehokkaampi tilan käyttö. Jos käyttöliittymän tila on rajallinen, 3D-ratkaisuilla voidaan mahduttaa suurempi määrä ominaisuuksia tai informaatiota. Tätä johtopäätöstä tukee tämän työn käytettävyysarvioinnin tulokset, jossa havaittiin, että 3D-hahmolla voidaan antaa käyttäjälle sellaisia ohjeita esimerkiksi potilaan asennosta, mitkä normaalisti vaatisivat tekstiä tai 2D-kuvia. 2D-kuviin nähden 3D-hahmo toimii paremmin, koska hahmoa voi pyörittää ja zoomata, mikä helpottaa ymmärtämään asennon yksityiskohtaisesti.

Toinen hyöty, joka on havaittu tutkimuksissa, on se, miten 3D-visualisaatioilla voidaan hyödyntää ihmisten avaruudellista muistia ja havainnointikykyä. Tämä on huomattu karttoihin liittyvissä sovelluksissa sekä 2D- ja 3D-opetusmateriaalia vertailevassa tutkimuksessa. Toteutetun käyttöliittymän kohdalla ei löydetty tällaista tulosta, mutta voidaan ainakin hypoteettisesti arvata, että hahmon asennon hahmottaminen olisi helpompaa 3D-mallista kuin 2D-kuvasta tai tekstiohjeista.

Tutkimusten mukaan suuremman informaation määrän varjopuolena voi olla käyttöliittymän monimutkaistuminen ja kognitiivinen ylikuormitus. Monimutkaistumista voi osittain puoltaa toteutetussa käyttöliittymässä havaitut ongelmat. 3D-mallin toiminnallisuudet voi vaatia monimutkaisempia interaktioita käyttöliittymän kanssa kuin vastaavat 2D-toteutukset. Toimintojen kuten nipistyszoomauksen ja pyörityksen esittäminen ja ohjeistaminen tulee huomioida erityisellä tavalla.

Kognitiivinen ylikuormitus voi mahdollisesti lamauttaa käyttäjän tai hidastaa käyttöliittymän käyttöä ja sisäistämistä. Voi olla sovelluksia, jossa informaation määrää on syytä rajata, jotta tätä ei tapahdu. Tällöin voi olla syytä yksinkertaistaa 3D-visualisaatiota tai jättää ne kokonaan käyttämättä. Tässä insinöörityössä toteutetussa käyttöliittymässä tosin ei kognitiivisen ylikuormituksen vaaraa ole, kun käytetään vain yhtä 3D-mallia yhdessä tarkoituksessa.

Onko 3D-malleista käytettävyyden kannalta hyötyä ohjelmistokäyttöliittymien toimintaan vai onko niiden käyttö vain visuaalinen lisä? Hyötyjä hyvin selkeästi on sekä tutkimustaustan että kehitystyön käytettävyysarvioinnin perusteella, mutta niin on myös mahdollisia haittoja. 3D:n hyödyntämistä tulee siis harkita toteutuksen mukaan, ja kehityksessä kannattaa tehdä vertailevaa käyttäjätutkimusta sen määrittämiseen. Tämän työn toteutuksessa voidaan todeta, että 3D-mallista on enemmän hyötyä kuin haittaa. Havaitut ongelmat liittyivät lähinnä toteutuksen vajavaisuuteen, kun taas hyödyt liittyivät vahvasti 3D-visualisaation hyötyyn.

7.4 Tulevaisuuden tutkimus

Aiheena tämä insinöörityö on ensimmäistä laatuaan Metropolian hyvinvointi- ja terveysteknologian pääaineessa. Vastaavanlaista käyttöliittymän kehitystä ja 3D-mallin hyödyntämistä sekä arviointia ei ole aiemmin tehty. Tämä näkyi siinä, miten suunnitelma hieman muuttui työn aikana. Yksi tutkimuskysymys jätettiin pois varhaisessa vaiheessa, ja käytettävyysarviointi tuotiin suunnitelmaan mukaan hieman sen jälkeen.

Jos tällaisesta aiheesta tehdään insinööritöitä jatkossa, kannattaisi käytettävyysarviointi ottaa suunnitelmaan alusta lähtien. Tämän avulla on mahdollista suunnitella arvioinnista laajemman ja järjestää käyttäjätutkimusta tai useamman asiantuntijan asiantuntija- tai paneeliarvioita.

Kokonaisuudessaan 3D-mallien hyödyllisyyden tutkimuksessa olisi hyvä nähdä lisää vertailevaa tutkimusta, missä sama käyttöliittymä toteutettaisiin sekä 2D-

että 3D-ratkaisuna. Niissä kannattaisi fokusoida yhden elementin tai osa-alueen arvioimiseen. Esimerkiksi tämän insinöörityön 3D-mallin olisi voinut toteuttaa myös 2D-kuvana tai tekstimuotoisina ohjeina, ja tehdä niistä vertailevaa käyttäjätutkimusta. Monissa vertailevissa tutkimuksissa 2D-toteutukset on muutettu kokonaisvaltaisesti 3D-toteutuksiksi, mutta yksittäisten elementtien vertailua löytyi suhteellisen vähän. Tästä variaatiosta voisi saada arvokasta tutkimustietoa.

Lähteet

1 LymphaTouch Oy. Verkkoaineisto. Kauppalehti. Saatavana osoitteessa: <https://www.kauppalehti.fi/yritykset/yritys/lymphatouch+oy/19789777>. Luettu 5.3.2022.

2 Alipaineella tehoa terapiaan. Tuote-esite. LymphaTouch Oy. Saatavana osoitteessa: <https://irp.cdn-website.com/af9f9e3f/files/uploaded/Lympha-Touch_esite.pdf>. Luettu 5.3.2022.

3 Hoitoalueet. Verkkoaineisto. LymphaTouch Oy. Saatavana osoitteessa: <https://www.lymphatouch.com/fi/hoitoalueet>. Luettu 5.3.2022.

4 Clinical proof book. E-kirja. LymphaTouch Oy. Saatavana osoitteessa: <https://irp-cdn.multiscreensite.com/af9f9e3f/files/uploaded/Lympha-Touch%20Clinical%20Proof%20Book.pdf>. Luettu 5.3.2022.

5 Saraci, Petref. Qt Design Studio – The New Age of UI Development. Blogikirjoitus. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/blog/2018/06/26/qt-design-studio-the-new-age-of-ui-development>. Julkaistu 26.6.2018. Luettu 26.2.2022

6 About Us. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/company>. Luettu 26.2.2022.

7 About Trolltech. 2008. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/archives/4.3/trolltech.html>. Luettu 26.2.2022.

8 Haartmann, Thomas. Qt Design Studio 3.0 Released. Blogikirjoitus. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/blog/qt-design-studio-3.0-released>. Julkaistu 3.2.2022. Luettu 26.2.2022.

9 Knoll, Lars. Qt 6.0 Released. Blogikirjoitus. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/blog/qt-6.0-released>. Julkaistu 8.12.2020. Luettu 26.2.2022.

10 Turunen, Tuukka. Qt 6.2.3 Released. Blogikirjoitus. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/blog/qt-6.2.3-released>. Julkaistu 31.1.2022. Luettu 15.4.2022.

11 Licensing. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/licensing/>. Luettu 26.2.2022.

12 Qt and C++. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/product/qt6/qml-book/ch16-qtcpp-qtcpp>. Päivitetty 23.8.2021. Luettu 26.2.2022.

13 JavaScript. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/product/qt6/qml-book/ch15-javascript-javascript>. Päivitetty 5.10.2021. Luettu 27.2.2022.

14 Selecting Modes. Ohjekirja. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtdesignstudio/creator-modes.html>. Luettu 26.2.2022.

15 User Interface. Ohjekirja. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtdesignstudio/creator-quick-tour.html>. Luettu 26.2.2022.

16 Design Views. Ohjekirja. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtdesignstudio/creator-using-qt-quick-designer.html>. Luettu 26.2.2022.

17 Form Editor. Ohjekirja. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtdesignstudio/qtquick-form-editor.html>. Luettu 27.2.2022.

18 3D Editor. Ohjekirja. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtdesignstudio/studio-3d-editor.html>. Luettu 27.2.2022.

19 Navigator. Ohjekirja. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtdesignstudio/qtquick-navigator.html>. Luettu 27.2.2022.

20 States. Ohjekirja. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtdesignstudio/qtquick-states-view.html>. Luettu 27.2.2022.

21 Haartmann, Thomas. Qt Design Studio 1.5 released. Blogikirjoitus. Qt Group Oyj. Saatavana osoitteessa: <https://www.qt.io/blog/qt-design-studio-1.5-released>. Julkaistu 26.5.2020. Luettu 26.2.2022.

22 Visinescu, Lucian; Sidorova, Anna; Jones, Mary & Prybutok, Victor. 2015. The influence of website dimensionality on customer experiences, perceptions and behavioral intentions: An exploration of 2D vs. 3D web design. Information & Management 52(1), s. 1–17. Saatavana osoitteessa: <https://doi.org/10.1016/j.im.2014.10.005>.

23 Oulasvirta, Antti; Estlander, Sara & Nurminen, Antti. 2009. Embodied interaction with a 3D versus 2D mobile map. Pers Ubiquit Comput 13, s. 303–320. Saatavana osoitteessa: <https://doi.org/10.1007/s00779-008-0209-0>.

24 Liao, Hua; Dong, Weihua; Peng, Chen & Liu, Huiping. 2016. Exploring differences of visual attention in pedestrian navigation when using 2D maps and 3D geo-browsers. Cartography and Geographic Information Science 44(6), s. 474–490. Saatavana osoitteessa: <https://doi.org/10.1080/15230406.2016.1174886>.

25 Rottermanner, Gernot ym. 2020. Design and Evaluation of a Tool to Support Air Traffic Control with 2D and 3D Visualizations. 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), s. 885–892. Saatavana osoitteessa: <https://doi.org/10.1109/VR46266.2020.00011>.

26 Settapat, Sittapong; Achalakul, Tiranee & Ohkura, Michiko. 2011. The Usability Evaluation of Web-Based 3D Medical Image Visualization. Teoksessa: Marcus, Aaron (toim.). Design, User Experience, and Usability, Theory, Methods, Tools and Practice, DUXU 2011, s. 507–516. Lecture Notes in Computer Science 6770. Berliini, Heidelberg: Springer. Saatavana osoitteessa: <https://doi.org/10.1007/978-3-642-21708-1_57>.

27 Kim, Kyungdoh; Proctor, Robert & Salvendy, Gavriel. 2011. Comparison of 3D and 2D menus for cell phones. Computers in Human Behavior 27(5), s. 2056–2066. Saatavana osoitteessa: <https://doi.org/10.1016/j.chb.2011.04.014>.

28 Ark, Wendy; Dryer, Christopher; Selker, Ted & Zhai, Shumin. 1998. Representation Matters: The Effect of 3D Objects and a Spatial Metaphor in a Graphical User Interface. Teoksessa: Johnson, Hillary; Nigay, Lawrence & Roast, Christopher (toim.). People and Computers XIII, s. 209–219. Lontoo: Springer. Saatavana osoitteessa: <https://doi.org/10.1007/978-1-4471-3605-7_13>.

29 Wang, Hao-Chuan; Chang, Chun-Yen & Li, Tsai-Yen. 2007. The comparative efficacy of 2D- versus 3D-based media design for influencing spatial visualization skills. Computers in Human Behavior 23(4), s. 1943–1957. Saatavana osoitteessa: <https://doi.org/10.1016/j.chb.2006.02.004>.

30 Punchoojit, Lumpapun & Hongwarittorrn, Nuttanont. 2017. Usability Studies on Mobile User Interface Design Patterns: A Systematic Literature Review. Advances in Human-Computer Interaction 2017. Saatavana osoitteessa: <https://doi.org/10.1155/2017/6787504>.

31 SFS-EN ISO 9241-11. Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts. 2018. Helsinki: Suomen Standardisoimisliitto.

32 SFS-EN 62366-1. Medical devices – Part 1: Application of usability engineering to medical devices. 2015. Helsinki: Suomen Standardisoimisliitto.

33 IEC TR 62366-2. Medical devices – Part 2: Guidance on the application of usability engineering to medical devices. 2016. Geneve: International Electrotechnical Commission.

34 Nielsen, Jakob. 1994. Usability Inspection Methods. Teoksessa: Conference Companion on Human Factors in Computing Systems (CHI '94). Association for Computing Machinery, New York, NY, USA, s. 413–414. Artikkeli. Saatavana osoitteessa: <https://doi.org/10.1145/259963.260531>.

35 Nielsen, Jacob & Landauer, Thomas. 1993. A mathematical model of the finding of usability problems. Teoksessa: Proceedings of the INTERACT '93 & CHI '93 Conference on Human Factors in Computing Systems (CHI '93). Association for Computing Machinery, New York, NY, USA, s. 206–213. Saatavana osoitteessa: <https://doi.org/10.1145/169059.169166>.

36 SFS-EN ISO 9241-161. Ergonomics of human-system interaction – Part 161: Guidance on visual user-interface elements. 2016. Helsinki: Suomen Standardisoimisliitto.

37 Nielsen, Jakob. 10 Usability Heuristics for User Interface Design. 1994. Verkkoaineisto. Nielsen Norman Group. Saatavana osoitteessa: <https://www.nngroup.com/articles/ten-usability-heuristics/>. Päivitetty 15.11.2020. Luettu 8.3.2022.

38 State QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick-state.html>. Luettu 28.2.2022.

39 Timer QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtqml-timer.html>. Luettu 28.2.2022.

40 UI Files. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qtcreator/creator-quick-ui-forms.html>. Luettu 28.2.2022.

41 Qt Quick Dialogs. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qtquickdialogs-index.html>. Luettu 28.2.2022.

42 MessageDialog QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc-snapshots.qt.io/qt6-dev/qml-qtquick-dialogs-messagedialog.html>. Luettu 28.2.2022.

43 Characters. Verkkoaineisto. Mixamo. Saatavana osoitteessa: <https://www.mixamo.com/#/?page=1&type=Character>. Ladattu 2.3.2022.

44 View3D QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick3d-view3d.html>. Luettu 28.2.2022.

45 PinchArea QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick-pincharea.html>. Luettu 1.3.2022.

46 MouseArea QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick-mousearea.html>. Luettu 28.2.2022.

47 Maknun, C.L., Rosjanuardi, R. & Jupri, A. 2019. From ratios of right triangle to unit circle: an introduction to trigonometric functions. Journal of Physics: Conference Series 1157(2), 022124. Saatavana osoitteessa: <https://doi.org/10.1088/1742-6596/1157/2/022124>.

48 Node QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick3d-node.html#details>. Luettu 28.2.2022.

49 Transition QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick-transition.html>. Luettu 28.2.2022.

50 Animation QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick-animation.html>. Luettu 28.2.2022.

51 PropertyAnimation QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick-propertyanimation.html>. Luettu 28.2.2022

52 NumberAnimation QML Type. Verkkoaineisto. Qt Group Oyj. Saatavana osoitteessa: <https://doc.qt.io/qt-6/qml-qtquick-numberanimation.html>. Luettu 28.2.2022

Standarditarkastus

Qt Design Studio -ohjelmistolla luodulle käyttöliittymälle tehtiin standarditarkastus osana formatiivista arviointia. Käyttöliittymä luotiin LymphaTouchin hoitolaitteen kosketusnäytöllä ajettavaksi, ja se kattaa yksittäisen hoito-ohjelman. Hoito-ohjelman näytössä on 3D-hahmo omassa ikkunassaan ja siihen liittyvät elementit sekä ikkuna hoidon vaiheiden seuraamiseen, niissä eteenpäin liikkumiseen ja hoidon ajoittamiseen. Tässä dokumentissa käydään läpi ensin itse tarkastus luvussa 1 ja sen jälkeen sen tulokset liittyen formatiiviseen arviointiin luvussa 2.

1 Tarkastus

Tarkastus tehtiin hyödyntäen kahta standardia: IEC 62366:n teknistä raporttia [1] ja ISO 9241:n osaa 161 [2]. Ensimmäinen näistä antoi ohjelmistokäyttöliittymälle muutaman yleisen vaatimuksen (taulukko 1) ja toinen vaatimuksia käyttöliittymän yksittäisille elementeille (taulukot 2–11). Tarkastuksessa käsitellään vain niitä elementtejä, joita on käytetty toteutetussa käyttöliittymässä.

Taulukossa 1 on standardin IEC 62366 teknisen raportin vaatimuksia ohjelmistokäyttöliittymälle. Taulukossa kerrotaan lisäksi niiden täyttyminen käsiteltävässä käyttöliittymässä.

Taulukko 1. Standardin IEC 62366 teknisen raportin [1, s. 47] vaatimukset ohjelmistokäyttöliittymälle.

id	Vaatimus	Täytetty
IEC1	Jokaisella ruudulla tulee olla tarkoitusta kuvaava otsikko, jotta käyttäjä voi ymmärtää oman sijaintinsa käyttöliittymän rakenteessa ja edistymisen sen hetkisessä tehtävässä.	Osittain
IEC2	Mikäli käyttöliittymässä oleva toiminallisuus vaatii vähintään kolmen sekunnin odotuksen, tulee ruudulla olla jonkinlainen merkki odotuksen edistymisestä.	Kyllä
IEC3	Käyttöliittymän jokaisessa ruudussa tulee olla vähintään yksi dynaaminen elementti, jotta käyttäjä voi huomata ohjelmiston kaatumisen.	Kyllä
IEC4	Käyttöliittymässä olevan tekstin tulee olla kooltaan vähintään 14 pikseliä, jotta luettavuus on turvattu myös huonompinäköisille.	Ei

IEC1:n koetaan vain osittain täyttyneeksi, koska toteutetun hoito-ohjelman lisäksi käyttöliittymän muita osia ei ole vielä tehty, joten varsinaista rakennetta ei ole. Hoito-ohjelmalla on kuvaava otsikko, mutta sijaintia rakenteessa ei voi arvioida. IEC2 täyttyy, koska käyttöliittymässä ei toistaiseksi ole toimintoa, joka vaatisi yli kolmen sekunnin odotusta. Tosin hahmon vaihto ensimmäisen kerran on lähellä tätä. IEC3:n dynaamisena elementtinä toimii pulssimaisesti animoituva hoitoalue. Lisäksi itse 3D-hahmo toimii dynaamisena elementtinä, koska sitä pystyy pyörittämään ja zoomaamaan. IEC4:n vaatimus ei täyty, koska painikkeiden tekstin koko on vain 9, mikä on niiden automaattinen koko, kun elementit lisää.

Taulukoissa 2–11 on standardin ISO 9241 osan 161 vaatimuksia ohjelmistokäyttöliittymän elementeille. Taulukko 2 kertoo vaatimukset ja niiden täyttymisen liukusäätimien osalta. Liukusäädin on analoginen elementti, joka mahdollistaa arvojen valitsemisen jatkuvasta arvojoukosta liikuttamalla valitsinosaa osoittimella [2, s. 14–15].

Taulukko 2. Standardin ISO 9241 osan 161 [2, s. 14–15] vaatimukset liukusäädinelementeille (analogue form element/slider).

id	Vaatimus	Täytetty
ISO1	Käyttöliittymän tulee tarvita rajoitettuja arvoja.	Kyllä
ISO2	Käyttöliittymän tulee tarvita jatkuvia arvoja ja niiden välitöntä palautetta.	Kyllä
ISO3	Käyttöliittymässä tulee olla tarpeeksi tilaa elementille ja sen mahdollisille arvoille.	Kyllä
ISO4	Liukusäätimen arvoja tulee käyttää vain syötteinä.	Osittain

Jokainen käyttöliittymän kolmesta liukusäätimestä tarvitsee rajatun arvoalueen. Zoomaus halutaan rajoittaa niin, että se ei mene hahmon sisään tai liian kauas. Pystysuuntainen pyöritys halutaan pitää -90…90 asteen välillä, jotta se pysähtyy suoraan hahmon alle ja ylle. Vaiheiden liukusäädössä halutaan olevan vaiheiden lukumääräinen arvoalue. Kaikki kolme liukusäädintä tarvitsevat myös jatkuvia arvoja sekä välitöntä palautetta niiden muutoksille ja niille on suunniteltu tarpeeksi tilaa näytöllä. ISO4 täyttyy tosin vain osittain johtuen vaiheiden liukusäädöstä. Liukusäätö ei käytä arvoja syötteinä vaan tekevät käyttöliittymän sisällä vertailua.

Taulukossa 3 käydään läpi nimikkeelle asetettuja vaatimuksia. Nimikkeellä tarkoitetaan elementille annettua otsikkoa, joka on tekstiä, mitä ei voi muokata [2, s. 36–37].

Taulukko 3. Standardin ISO 9241 osan 161 [2, s. 36–37] vaatimukset nimikkeelle (label).

id	Vaatimus	Täytetty
ISO5	Kaikissa käyttöliittymän elementeissä paitsi osoittimessa tulee olla nimike.	Osittain
ISO6	Nimikkeen tulee olla lyhyt ja ytimekäs.	Kyllä
ISO7	Nimikkeen muotoilu tulee toteuttaa niin, että se tukee luettavuutta.	Osittain
ISO8	Nimike tulee visualisoida niin, että sen tunnistaa nimikkeeksi ja erottaa muusta tekstistä.	Ei

Lähes kaikilla käyttöliittymän elementeillä on nimikkeet paitsi liukusäätimillä. Vaiheiden liukusäätimen tarkoituksen voi kuitenkin ymmärtää yllä olevan elementin nimikkeen kontekstista. Pyörityksen liukusäädin on piilotettu, koska pyörityst on tarkoitus tehdä vain kosketuksella. Zoomauksen liukusäätimen kontekstia ei kuitenkaan voi suoraan ymmärtää, ja se kaipaisi nimikettä. Kaikki käytetyt nimikkeet ovat lyhyitä ja ytimekkäitä ja niiden muotoilu tukee luettavuutta suurimmaksi osaksi. Ainoana mahdollisena ongelmana voi olla painikkeiden tekstin fontin pienuus. ISO8 ei täyty, koska nimikkeitä ei ole visuaalisesti eroteltu mitenkään muusta tekstistä.

Taulukossa 4 on vaatimukset tulosteruuduille. Tulosteruudut määritellään elementeiksi, jotka esittävät muuttuvaa informaatiota [2, s. 43–44].

Taulukko 4. Standardin ISO 9241 osan 161 [2, s. 43–44] vaatimukset tulosteruudulle (output pane).

id	Vaatimus	Täytetty
ISO9	Esitettävän tiedon ei tule olla muokattavaa.	Kyllä
ISO10	Elementti tulee toteuttaa niin, että se selkeästi esittää, että sen tiedot ei ole muokattavissa.	Osittain

Tulosteruutuja käytettiin vain ajastimien esitykseen. Niiden tiedot eivät ole muokattavia, joten ISO9 täyttyy. Elementtejä ei ole tosin esitetty erityisesti sellaisella tavalla, että teksti ei varmasti olisi muokattavaa, mutta ajastimien konteksti pitää siitä huolen. Tämän takia nähdään, että ISO10 on osittain täytetty.

Taulukko 5 käsittelee komentopainikkeiden vaatimuksia. Komentopainikkeiden aktivoinnilla annetaan käskyjä tai tehdään toimintoja käyttöliittymässä [2, s. 48].

Taulukko 5. Standardin ISO 9241 osan 161 [2, s. 48] vaatimukset komentopainikkeille (push button/command button).

id	**Vaatimus**	**Täytetty**
ISO11	Systeemillä tulee olla käynnistettäviä toimintoja.	Kyllä
ISO12	Jokaisella painikkeella tulee olla nimike.	Kyllä
ISO13	Painikkeen nimikkeen tulee olla lyhyt ja ytimekäs.	Kyllä
ISO14	Jos systeemin toiminnan käynnistäminen vaatii ylimääräistä informaatiota, painikkeen suunnittelun tulee osoittaa se.	Kyllä
ISO15	Painikkeen tulee aktivoitua yksittäisellä aktivointitapahtumalla.	Kyllä

Kaikki toteutetun käyttöliittymän komentopainikkeista on liitetty johonkin systeemin toiminnallisuuteen tai useampaan. Jokaisella painikkeella on myös lyhyt ja ytimekäs nimike. Minkään systeemin toiminnon käynnistys ei vaadi ylimääräistä informaatiota, joten ISO14 täyttyy. Toimintopainikkeet toimivat yhdellä aktivointitapahtumalla eli painalluksella automaattisesti Qt kehyksessä, joten myös ISO15 täyttyy.

Taulukossa 6 on vaatimukset kirjoitussuojatuille tekstikentille. Ne määritellään kentiksi, jotka sisältävät tietoa, mitä ei voi muokata [2, s. 50–51].

Taulukko 6. Standardin ISO 9241 osan 161 [2, s. 50–51] vaatimukset kirjoitussuojatuille tekstikentille (read only field/protected field).

id	Vaatimus	Täytetty
ISO16	Käyttöliittymän tulee tarvita kirjoitussuojattua tekstiä.	Kyllä
ISO17	Kirjoitussuojatun tekstikentän tulee pystyä keskittymään niin, että sen voi lukea huononäköisten aputeknologialla.	Ei tietoa
ISO18	Kirjoitussuojattu tekstikenttä tulee suunnitella niin, että sen erottaa kirjoitussuojatuksi.	Osittain

Kirjoitussuojattuja tekstikenttiä tarvittiin vaihenumeroiden esitykseen ja osaan nimikkeistä. Vaatimuksen ISO17 täyttymistä ei pystytty varmistamaan. Samoin kuin nimikkeiden kohdalla kirjoitussuojattuja tekstikenttiä ei suunniteltu erottumaan muusta tekstistä, mutta jälleen konteksti auttaa tunnistamaan ne tekstisuojatuiksi. Tämän takia ISO18 nähdään osittain täyttyneeksi.

Taulukossa 7 käydään läpi otsikoille asetettuja vaatimuksia. Otsikko on tekstiä, joka kertoo käyttöliittymän tai sen osan nimestä ja/tai sisällöstä [2, s. 64–65].

Taulukko 7. Standardin ISO 9241 osan 161 [2, s. 64–65] vaatimukset otsikoille (title).

id	Vaatimus	Täytetty
ISO19	Otsikkoa tulee käyttää, kun käyttöliittymä käyttää ikkunoita, tai kun käyttöliittymän sisällön esitys käyttää koko ruudun tilan.	Kyllä
ISO20	Otsikon tulee olla ”ennen” sen sisältöä tai tietoa.	Kyllä
ISO21	Otsikko tulee esittää tavalla, joka osoittaa eron käyttöliittymän muihin osiin.	Kyllä

Toteutettu käyttöliittymän osa eli hoito-ohjelman näyttö on suunniteltu käyttämään koko kosketusnäytön ruudun, johon sitä on kehitetty. Siihen lisätty otsikko

on ennen sisältöä eli sen yläpuolella. Otsikko erotetaan muusta tekstistä sen koolla sekä sen sijainnilla.

Taulukko 8 käsittelee vaihtopainikkeiden vaatimuksia. Vaihtopainikkeilla voidaan vaihdella kahden eri tilan välillä [2, s. 65–66].

Taulukko 8. Standardin ISO 9241 osan 161 [2, s. 65–66] vaatimukset vaihtopainikkeille (toggle button).

id	**Vaatimus**	**Täytetty**
ISO22	Yhden tai useamman systeemin ominaisuuden tulee olla joko tosi tai epätosi.	Kyllä
ISO23	Painike tulee esittää tavalla, jolla sen kaksi tilaa voidaan selkeästi erottaa toisistaan.	Kyllä
ISO24	Käyttäjän tulee pystyä selvittämään, mikä on painikkeen tila.	Osittain
ISO25	Painikkeen esitys tulee olla käyttöliittymän suunnittelussa johdonmukainen.	Kyllä

Vaihtopainikkeita käytetään toteutetussa käyttöliittymässä hahmon vaihtamiseen. Hahmojen näkyvyys vaihtuu todesta epätodeksi päittäin painikkeiden tilan mukaan. Tosi-tila esitetään vaalealla painikkeella ja epätosi tummalla, joten ISO23 täyttyy. Käyttöliittymä ei kuitenkaan erikseen kerro tätä. Hahmojen perusteella voi tosin päätellä painikkeiden tilat. Tämän takia nähtiin vaatimuksen ISO24 olevan ainakin osittain täytetty. Painikkeiden esitys on johdonmukainen, kun niitä on vain kertaalleen toteutuksessa.

2 Tulokset

Formatiivisen arvioinnin tuloksena on arvioinnissa havaitut ongelmat, jotka voivat aiheuttaa käyttövirheitä, ja sitä kautta vaikuttaa turvallisuuteen. Taulukossa 9 on standarditarkastuksessa havaitut ongelmat. Ongelmia oli niiden vaatimusten kohdalla, jotka eivät täyttyneet tai täyttyivät osittain. Kaikkia osittain täyttyneitä ei kuitenkaan koettu ongelmaksi. Niitä ongelmia, jotka johtuvat käyttöliittymän kehityksen vaiheesta, ei otettu huomioon.

Taulukko 9. Standarditarkastuksessa havaitut käyttöliittymän ongelmat.

Ongelma	**Vaatimus ID:t**	**Liittyvät elementit**	**Aiheutuvat käyttövirheet**
Painikkeissa on liian pienikokoinen teksti.	IEC4, ISO7	Nimikkeet (komento- ja vaihtopainikkeet)	Huononäköinen käyttäjä painaa vääriä painikkeita, ja ei saa ohjelmistoa toimimaan tarkoitetulla tavalla.
Vaiheiden liukusäädin käyttää arvojaan muuten kuin syötteenä.	ISO4	Liukusäädin (vaiheikkuna)	Ei odotettavia käyttövirheitä nykyisessä käyttöliittymässä
Zoomauksen liukusäätimestä puuttuu nimike.	ISO5	Liukusäädin (zoomaus)	Käyttäjä ei hyödynnä liukusäädintä, koska luulee sen tekevän jotain muuta.
Eri tekstielementtien muotoilua ei ole visuaalisesti eroteltu.	ISO8, ISO18	Nimikkeet, kirjoittussuojatut tekstikentät	Ei odotettavia käyttövirheitä nykyisessä käyttöliittymässä
Valintapainikkeiden tilaa ei pysty selvittämään kokeilematta.	ISO24	Valintapainikkeet	Ei odotettavia käyttövirheitä nykyisessä käyttöliittymässä

Standarditarkastuksessa löytyi viisi käytettävyysongelmaa, joista kahden määritettiin voivan aiheuttaa käyttövirheitä käyttöliittymän tämänhetkisessä tilassa. Ongelmia löytyi neljästä eri elementtityypistä.

Lähteet

1 IEC TR 62366-2. Medical devices – Part 2: Guidance on the application of usability engineering to medical devices. 2016. Geneve: International Electrotechnical Commission.

2 SFS-EN ISO 9241-161. Ergonomics of human-system interaction – Part 161: Guidance on visual user-interface elements. 2016. Helsinki: Suomen Standardisoimisliitto.

Heuristinen arviointi

Qt Design Studio -ohjelmistolla luodulle käyttöliittymälle heuristinen arviointi osana formatiivista arviointia. Käyttöliittymä luotiin LymphaTouchin hoitolaitteen kosketusnäytöllä ajettavaksi, ja se kattaa yksittäisen hoito-ohjelman. Hoito-ohjelman näytössä on 3D-hahmo omassa ikkunassaan ja siihen liittyvät elementit sekä ikkuna hoidon vaiheiden seuraamiseen, niissä eteenpäin liikkumiseen ja hoidon ajoittamiseen. Arvioinnissa käytettiin Nielsenin heuristiikkoja, jotka on listattu ja selitetty taulukossa 1.

Taulukko 10. Nielsenin [1] kymmenen heuristiikkaa

Nro	**Heuristiikka**	**Kuvaus**
1	Järjestelmän tilan näkyvyys	Käyttäjää tulee informoida sijainnista käyttöliittymässä ja järjestelmän tilasta.
2	Järjestelmän ja todellisuuden vastaavuus	Käyttöliittymässä tulee käyttää käyttäjille tuttua ja intuitiivista kieltä ja suunnittelua.
3	Käyttäjän kontrolli ja vapaus	Käyttäjillä tulee olla vapaus kumota tekoja, ja palata tilasta tai ruudusta takaisin.
4	Johdonmukaisuus	Käyttöliittymän kielen ja toimintojen tulee olla johdonmukaisia.
5	Virheiden ehkäisy	Käytössä tulevat virheet tulee minimoida suunnittelussa.
6	Muistin sijaan tunnistus	Ulkomuistin tarve tulee minimoida käyttöliittymässä.
7	Joustavuus ja tehokkuus	Käyttäjille tulee tarjota eri vaihtoehtoja tehdä toimintoja ja mahdollisia oikopolkuja.
8	Minimalismi	Kaikki tarpeeton informaatio tulee poistaa käyttöliittymän ruuduista.
9	Virheiden viestintä	Virheistä tulee viestiä hyvin, ja tarjota niille ratkaisua.
10	Apu ja dokumentaatio	Ohjeiden tarve tulee minimoida, ja ohjeiden tulee olla nopeita löytää sekä helppo ymmärtää.

Heuristiikkoja käsitellään yksi kerrallaan seuraavissa luvuissa 1–10. Jokaista heuristiikkaa on arvioitu erikseen. Arvioinnin tulokset liittyen formatiiviseen arviointiin käydään läpi luvussa 11.

1 Järjestelmän tilan näkyvyys

Käyttöliittymä antaa otsikollaan tiedon siitä, että kyseessä on hoito, ja se tulisi tehdä kyynärpäälle, mutta sijaintia muuhun rakenteeseen nähden ei ilmene. Tämä on tosin ymmärrettävää ottaen huomioon sen, että muuta rakennetta ei vielä ole olemassa.

Ikkunassa näkyy, että missä vaiheessa ollaan ja kuinka kauan hoito on kestänyt Vaiheen voi ajoittaa ehdotettuun maksimiaikaan nähden. Omassa ikkunassaan olevan 3D-hahmo vaihtaa asentoa vaiheen mukaan. Systeemi siis kertoo, missä vaiheessa ollaan, miten se tulisi ajoittaa, ja hahmon asento antaa ainakin osittain ohjetta potilaan asennolle. Vaiheen ajan päättyminen ilmoitetaan aikatekstin lihavoitumisella ja punaiseksi värjäytymisellä. Tämä voi olla ihan hieman huomaamaton ilmoitus, varsinkin kun otetaan huomioon, että hoidon aikana ei koko ajan katsota ruutua kohti.

2 Järjestelmän ja todellisuuden vastaavuus

Käyttöliittymässä on vähän tekstiä, mutta niiden otsikoiden ja nimikkeiden puolesta, mitä näytössä on, käytetään hyvin yleisiä termejä. Painikkeet tekevät pääasiassa niitä toiminallisuuksia, mitä termiltä odottaisi. Hahmoa vaihtavat painikkeet voisivat kaivata otsikkoa niiden yläpuolelle lisäselvennykseksi, mutta niidenkin kontekstista voi realistisesti odottaa, että suurin osa käyttäjistä ymmärtää niiden toiminnan.

Koska liukusäätimiä voi käyttää moneen tarkoitukseen, niiden näkeminen ei suoraan kerro niiden toiminnallisuudesta. Vaiheen liukusäätimen tarkoituksen ymmärtää todennäköisesti kontekstista, mutta zoomauksen liukusäätöä ei. Sen toiminnallisuuden saa selville vain kokeilemalla, mikä ei ole optimaalista suunnittelua. Liukusäädin kaipaisi siis nimikettä tai otsikkoa.

3D-hahmon kohdalla vaiheiden erinäiset asennot auttavat antamaan ohjeistusta hoidon suorittamiseen. Potilaalle tarkoitetun asennon ymmärtää ilman sanoja, ja asentoja voi tutkia tarkemmin muuttamalla hahmon zoomausta, tai pyörittämällä sitä eri suuntiin. Tämän voisi nähdä hyödyksi verrattuna asentojen esitykseen 2D-kuvilla. Hahmon sukupuolen voi myös valita vastaamaan potilaan sukupuolta, mutta se ei asennon antamiin ”ohjeistuksiin” vaikuta millään tavalla.

Zoomauksessa ja hahmon pyörityksessä on tosin mahdollisia ongelmia. Kosketusnäytöt ovat nykyään hyvin yleisiä, mutta nipistyszoomaus ja hahmon pyöritys kosketuksella eivät välttämättä ole ilmiselviä kaikille. Niiden mahdollisuudesta tulisi kertoa käyttäjälle jollain tavalla esimerkiksi kontekstivihjeillä.

3 Käyttäjän kontrolli ja vapaus

Toimintojen ja tilojen perumista/palauttamista käyttöliittymä tukee monella tavalla. Itse käyttöliittymän rakenteessa ei pääse takaisin, mutta se johtuu siitä, että muuta rakennetta ei vielä ole. Vaiheissa voi eteenpäin siirtymisen lisäksi palata takaisin. Vaiheajastimen voi pysäyttää, mutta sitä ei voi nollata muuten kuin siirtymällä vaiheissa eteen ja taakse, mikä on liian monimutkainen tapa siihen. Tästä voi syntyä mahdollisia ongelmia.

Hahmolle tehdyt zoomaukset ja pyöritykset voi palauttaa alkuperäiseen tilaan. Alkuperäinen tila vastaa sitä, mikä se on vaiheen alussa. Hahmon voi vaihtaa, missä vaiheessa tahansa, ja sen voi perua vaihtamalla hahmon uudelleen. Hahmon asento säilyy vaihdosta huolimatta.

4 Johdonmukaisuus

Kuten heuristiikassa 2 todettiin, käyttöliittymän nimikkeissä ja otsikoissa käytetyt termit vastaavat niiden yleistä käyttöä, ja niitä käytetään samojen toimintojen tekoon monissa muissa käyttöliittymissä. Jotkin elementit vielä kaipaavat nimikkeitä tai otsikoita, kuten zoomauksen liukusäätö ja mahdollisesti hahmonvaihdon painikkeet, mutta ne eivät liity tähän heuristiikkaan. Johdonmukaisuus

käyttöliittymän sisällä ei vaadi arviointia, koska käyttöliittymässä on vasta yksi toteutettu osa.

5 Virheiden ehkäisy

Nielsenin [1] mukaan heuristiikan 3 tukeminen suunnittelussa auttaa ehkäisemään virheitä, joten kaikki siinä mainitut onnistumiset ja ongelmat pätee myös tähän heuristiikkaan. Niitä ei käydä uudelleen, mutta ne voi lukea kappaleesta 3. Tosin vaiheiden ajastamiseen liittyen ajastimen käynnistämisessä voisi kysyä käyttäjän varmistusta, jotta sen aikaa ei kulu vahinkopainallukseen. Varmistusta voisi kysyä myös vaiheissa siirrossa, ja sitä voisi harkita hahmon asennon palauttavan Reset-painikkeen kohdalla. Näiden kohdalla vahinkopainallukset ovat myös ongelma.

Käyttöliittymän käytössä ei ole havaittu sellaisia virheitä, jotka kaipaisivat virheilmoituksia. Käyttöliittymässä ei siis ole toistaiseksi virheilmoituksia.

6 Muistin sijaan tunnistus

Käyttöliittymässä on suurimmaksi osaksi käytetty hyvin nimikkeitä ja otsikoita, mikä vähentää muistamisen tarvetta, mutta zoomauksen liukusäädöllä ei ole nimikettä, ja hahmonvaihdon painikkeet eivät täysin suoraan kerro, mitä tekevät. Vaiheiden liukusäädöllä ei myöskään ole nimikettä, mutta sen tarkoituksen ymmärtää kontekstista.

3D-hahmon nipistyszoomauksen ja pyörityksen mahdollisuutta ei esitellä, millään tavalla, joten se on muistettava ominaisuus. Ikkunaan tulisi lisätä joko ohjeita tai kontekstivihjeitä, joilla osoitetaan nämä mahdolliset toiminnot. Itse hahmon vaihtuvat asennot eri vaiheissa antavat sellaisia ohjeita potilaan asennolle, mitkä normaalisti vaatisivat tekstimuotoisia ohjeita tai 2D-kuvia.

7 Joustavuus ja tehokkuus

Käyttöliittymällä ei tällä hetkellä ole erillisiä asetuksia, pikanäppäimiä tai oikoreittejä, mutta ei niitä tämänhetkisessä tilassa myöskään tarvitse. Tämä heuristiikka tulee ajankohtaisemmaksi myöhemmässä kehitysvaiheessa.

8 Minimalismi

Käyttöliittymässä ei havaittu mitään ylimääräistä elementtiä tai informaatiota. Pystysuuntaiselle pyöritykselle on liukusäädin, mutta sen näkyvyys on poistettu, koska sitä ei haluta käyttävän pyörittämiseen. Kuten aiempien heuristiikkojen kohdalla on todettu, jotkin elementit tarvitsevat jopa lisäinformaatiota.

3D-hahmon asennot eri vaiheissa antavat ohjeita, joita muuten tarvitsisi kirjoittaa jonnekin käyttöliittymään tai esittää 2D-kuvilla. Hahmolla voisi mahdollisesti esittää muita vastaavia asioita, mikä parantaisi minimalismia edelleen.

9 Virheiden viestintä

Kuten heuristiikassa 5 mainittiin, käyttöliittymässä ei olla tähän mennessä havaittu sellaisia virheitä, jotka kaipaisivat virheilmoituksia.

10 Apu ja dokumentaatio

Käyttöliittymällä ei tällä hetkellä ole ohjeita missään kontekstissa. Kontekstit, joissa pienimuotoisia ohjeita voitaisiin kaivata, ovat 3D-hahmon nipistys-zoomaus ja pyöritys, zoomauksen liukusäätö, hahmonvaihdon painikkeet, hahmon asennon palautuksen Reset-painike ja mahdollisesti vaiheiden liukusäätö. Käyttöliittymässä voisi olla painike, joka näyttää lyhyitä ohjeita eri elementtien toiminnalle.

11 Tulokset

Formatiivisen arvioinnin tuloksena on arvioinnissa havaitut ongelmat, jotka voivat aiheuttaa käyttövirheitä, ja sitä kautta vaikuttaa turvallisuuteen. Taulukossa 1 on heuristisessa arvioinnissa havaitut ongelmat. Niitä ongelmia, jotka johtuvat käyttöliittymän kehityksen vaiheesta, ei otettu huomioon.

Taulukko 1. Heuristisessa arvioinnissa havaitut käyttöliittymän ongelmat

Ongelma	**Heuristiikat**	**Liittyvät elementit**	**Aiheutuvat käyttövirheet**
Vaiheajan päättymisen ilmoitus on liian huomaamaton.	1	Ajastin (vaihe)	Käyttäjä ei huomaa ajan päättymistä, ja jatkaa vaihetta liian pitkään.
Zoomauksen liukusäätimellä ei ole nimikettä tai otsikkoa.	2, 6	Liukusäädin (zoomaus)	Käyttäjä ei hyödynnä liukusäädintä, koska luulee sen tekevän jotain muuta.
Hahmonvaihdon painikkeille ei ole otsikkoa.	2, 6	Valintapainikkeet (Male ja Female)	Ei odotettavia käyttövirheitä nykyisessä käyttöliittymässä
3D-hahmon nipistyszoomauksen ja pyörityksen toimintoja ei esitetä tai ohjeisteta millään tavalla.	2, 6, 10	3D-hahmon ikkuna	Käyttäjä ei hyödynnä toimintoja, vaikka haluaisi tutkia hahmon asentoa eri kulmista tai eri zoomauksella.
Vaiheajastin nollautuu vain siirtymällä edestakaisin vaiheissa.	3, 5	Ajastin (vaihe)	Käyttäjä ei osaa nollata vaiheaikaa, jolloin ajastin ei ajasta koko hoidon vaihetta.
Käyttöliittymä ei kysy varmistusta vaiheajan käynnistämiseen.	5	Komentopainike (Start)	Käyttäjä vahingossa käynnistää ajastimen ennen kuin on valmis.
Käyttöliittymä ei kysy varmistusta vaiheen vaihtamiseen.	5	Komentopainikkeet (Back ja Forward)	Käyttäjä siirtyy vahingossa seuraavaan tai edelliseen vaiheeseen, vaikka ei ole valmis.
Käyttöliittymä ei kysy varmistusta hahmon asennon palauttamiseen.	5	Komentopainike (Reset)	Käyttäjä vahingossa palauttaa hahmon asennon, vaikka oli löytänyt itselleen paremman.
Käyttöliittymässä ei ole ohjeistusta elementtien käyttöön.	10	Kaikki käyttöliittymän toiminnalliset elementit	Käyttäjä ei ymmärrä joidenkin elementtien käyttöä.

Heuristisessa arvioinnissa löydettiin yhdeksän ongelmaa, joista kahdeksan määritettiin voivan aiheuttaa käyttövirheitä käyttöliittymän tämänhetkisessä tilassa. Ongelmia löytyi kuudesta eri heuristiikasta. Eniten ongelmia (4 kpl) oli heuristiikoissa 5 ja 6. Heuristiikalla 2 oli kolme ongelmaa ja heuristiikoilla 1, 3 ja 10 yksi ongelma.

Lähteet

1 Nielsen, Jakob. 10 Usability Heuristics for User Interface Design. 1994. Verkkoaineisto. Nielsen Norman Group. Saatavana osoitteessa: <https://www.nngroup.com/articles/ten-usability-heuristics/>. Päivitetty 15.11.2020. Luettu 8.3.2022.