



Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Joni Oksanen

Käyttöliittymäpohja lääkintälaitte- ohjelmistoille pelimoottorin avulla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintäteknikka

Insinöörityö

2.5.2019

Tekijä Otsikko Sivumäärä Aika	Joni Oksanen Käyttöliittymäpohja lääkintälaitteohjelmistoille pelimoottorin avulla 41 sivua 2.5.2019
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Pelisovellukset
Ohjaajat	Lehtori Antti Laiho Pääohjelmistokehittäjä Toni Paavola
<p>Insinööriyössä toteutettiin käyttöliittymäpohja terveysteknologia-alan ohjelmistokehitysyri-tyksen lääkintälaitteohjelmistoille, jotka luovat DICOM-kuvista 3D-malleja matemaattisten mallien avulla. Luotuja malleja voidaan hyödyntää esimerkiksi diagnostiikassa ja hoidon suunnittelussa. Työn tavoitteena oli luoda käyttöliittymäpohja, jolla voitaisiin luoda erilaisia käyttöliittymiä yrityksen ohjelmistotuotteille ja samalla tarjota yhtenäinen pohja tuotteiden yhteisille ohjelmistokomponenteille niiden päivitettävyyden ja keskinäisen yhteensopivuuden parantamiseksi.</p> <p>Insinööriyössä perehdyttiin lääkinnällisten laitteiden ja lääkintälaitteohjelmistojen määritelmiiin, alan tietoteknisiin standardeihin sekä laitteita koskevia säännöksiin. Säännökset koskevat myös käyttöliittymäpohjia, joita käytetään osana lääkintälaitteohjelmistoja. Tämän lisäksi insinööriyössä tutustuttiin lääkintälaitteohjelmistojen käyttöliittymäsuunnitteluun. Käyttöliittymäpohja toteuttiin yrityksen käyttämällä pelimoottorilla, mikä on lääketieteen alalla harvinaista.</p> <p>Käyttöliittymäpohja kattoi käyttöliittymien toteuttamiseen tarvittavat käyttöliittymäelementit ja fontit sekä käyttöliittymiin liittyvät yhteiset ohjelmistokomponentit. Käyttöliittymäelementit toteutettiin vektorigrafiikkaohjelmalla, ja toteutetut elementit lisättiin resursseina pelimoottorissa luotuun projektiin. Käyttöliittymäelementeistä rakennettiin esimerkkikäyttöliittymä toteutettujen ohjelmistokomponenttien toiminnallisuuksien esittelemistä varten.</p> <p>Käyttöliittymäpohja saatiin toteutettua sille asetettujen vaatimusten mukaisesti, ja sen kehittämistä jatketaan tulevaisuudessa. Käyttöliittymäpohjan päivittämistä ei saatu tehtyä yhtä helpoksi, kuin insinööriyön alussa toivottiin, mutta alkuperäiseen tilanteeseen saatiin silti merkittävä parannus.</p>	
Avainsanat	käyttöliittymäpohja, pelimoottori, lääkintälaitteohjelmisto

Author Title Number of Pages Date	Joni Oksanen Implementing a user interface template for medical software with a game engine 41 pages 2 May 2019
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Professional Major	Game Applications
Instructors	Antti Laiho, Senior Lecturer Toni Paavola, Principal Developer, Software
<p>This thesis was conducted for a software development company that works in the medical field. The company's software products reconstruct 3D models created from DICOM images using mathematical models for diagnosis and treatment planning purposes. The goal of the thesis was to design and implement a user interface template to be used in the company's software products. In addition to providing the resources for building diverse user interfaces, the template would provide a foundation for the common software components that are shared across different software to improve upgradability and compatibility.</p> <p>The thesis provides definitions for medical devices and medical device software, introduces the industry IT standards and reviews the regulations governing the development of medical devices. User interface templates also fall under these regulations if they are used in medical software. In addition, user interface design of medical software was explored in the thesis. The thesis was implemented using a game engine which is rare in the medical field.</p> <p>The user interface template consisted of the user interface elements and fonts required to produce diverse user interfaces as well as the common software components that are shared across all different software products. Unity game engine served as the basis for the project. Affinity Designer vector graphics software was used to create the user interface elements which were subsequently added as assets to the Unity project. An example user interface was created with the produced user interface elements to showcase the implemented features.</p> <p>The implemented user interface template met the requirements imposed on it before the start of the project and is going to be further developed in the future. Upgradability of the user interface template suffered a setback during the development process but nonetheless a major improvement compared to the initial situation was achieved.</p>	
Keywords	user interface template, game engine, medical software

Sisällys

1	Johdanto	1
2	Lääkinnälliset laitteet	2
2.1	Lääkintälaitteohjelmisto	2
2.2	Alan tietotekniikan standardit	3
2.3	Lääkinnällisiä laitteita koskevat säännökset	4
3	Käyttöliittymäpohjat	6
3.1	Atomisen suunnittelu	8
3.2	Käyttöliittymäpohjien hyödyt ja haasteet	9
4	Käyttöliittymäpohjan toteuttaminen Unity-pelimootorilla	9
4.1	Vaatimusmäärittely ja tavoitteet	10
4.2	Käytetyt työkalut	12
4.3	Käyttöliittymäelementtien toteutus	15
4.4	Unity-toteutus	20
4.5	Käyttöönotto Unitypackage-pakettien avulla	31
5	Työn tulokset ja käyttöliittymäpohjan tulevaisuus	32
6	Yhteenveto	35
	Lähteet	36

1 Johdanto

Insinööriyön tarkoituksena on toteuttaa käyttöliittymäpohja 3D-mallintamiseen keskittyville lääkintälaitteohjelmistoille. Työn tavoitteena on luoda käyttöliittymäpohjaan ulkoasultaan yhtenäisiä käyttöliittymäelementtejä, jotka ovat linjassa insinööriyön tilanteen yrityksen visuaalisen ilmeen kanssa. Tämän lisäksi käyttöliittymäpohjaan toteutetaan käyttöliittymien toiminnallisuuksia tukevia ohjelmistokomponentteja.

Insinööriyön on tilannut lääkintälaitteohjelmistoja tuottava Disior Oy. Käyttöliittymien kehittämisen yhtenäistämisen ja helpottamisen lisäksi käyttöliittymäpohjan on tarkoitus toimia yhtenäisenä pohjana ohjelmistokomponenteille, joita käytetään kaikissa Disiorin ohjelmissa. Ohjelmistokomponenttien toteutuksien keskittämällä haetaan parannuksia ohjelmistokomponenttien päivitettävyyteen ja yhteensopivuuteen.

Insinööriyön ensimmäisessä osiossa käydään läpi lääkinnällisten laitteiden ja lääkintälaitteohjelmistojen määritelmät sekä alan tärkeimmät tietotekniikan standardit. Määritelmien jälkeen paneudutaan lääkinnällisiä laitteita koskeviin määräyksiin ja asetuksiin Suomessa, Euroopan unionissa ja Yhdysvalloissa.

Seuraavassa osiossa käydään läpi varsinainen käyttöliittymäpohjan toteutus. Työlle asetut tavoitteet ja vaatimukset kerrataan lyhyesti, minkä jälkeen esitellään työssä käytetyt työkalut. Käyttöliittymäpohjan pohjana toimi Unity-pelimoottorilla luotu projekti, johon Affinity Designer -vektorigrafiikkaohjelmalla luodut käyttöliittymäelementit lisättiin. Käyttöliittymäpohjaan toteutetut ohjelmistokomponentit käydään läpi yksitellen, ja lopuksi esitellään käyttöliittymäpohjan avulla toteutettu esimerkkikäyttöliittymä.

Viimeisessä osiossa käydään ensin läpi työn tuloksia ja arvioidaan insinööriyölle asetettujen tavoitteiden ja vaatimuksien täyttymistä. Tämän jälkeen käsitellään, miten käyttöliittymäpohja integroitiin osaksi yrityksen tuoteympäristöä, ja lopuksi pohditaan käyttöliittymäpohjan jatkokehitysmahdollisuuksia.

2 Lääkinnälliset laitteet

Lääkinnälliseksi laitteeksi luokitellaan mikä tahansa instrumentti, laite, kone, implantti, tietokoneohjelma tai muu vastaava esine tai toiminto, joka on sen valmistajan mukaan suunniteltu käytettäväksi ihmisillä jonkin lääkinällisen toiminnon toteuttamiseksi. Näihin lääkinällisiin toimintoihin lukeutuvat

- sairauksien diagnosointi, ehkäisy, monitorointi ja hoitaminen
- vammojen diagnosointi, monitorointi, hoitaminen ja lievittäminen
- anatomian tai fysiologisen prosessin tutkiminen, korvaaminen ja muuttaminen
- elämän ylläpitäminen ja tukeminen
- lääkinällisten laitteiden desinfiointi
- ehkäisy.

Lääkinällisiksi laitteiksi ei lasketa niitä esineitä ja toimintoja, jotka saavuttavat niiden ensisijaisen käyttötarkoituksen farmaseuttisia, immunologisia tai ruuansulatuksen toimintoja hyödyntämällä. (1.) Tyypillisiin lääkinällisiin laitteisiin kuuluvat mm. röntgenlaitteet, erilaiset implantit, happimaskit, katetrit ja kondomit (2; 3).

2.1 Lääkintälaitteohjelmisto

Lääkintälaitteohjelmistoiksi luokitellaan kaikki ne ohjelmistot, jotka suorittavat joitain lääkinällisille laitteille määriteltäviä toimintoja. Lääkintälaitteohjelmisto voi olla itsenäinen lääkinällinen laitteensa tai osa jotain muuta lääkinällistä laitetta. Tämän lisäksi lääkinällisiksi lasketaan ne ohjelmat, joita käytetään lääkinällisten laitteiden valmistamiseen tai huoltamiseen. (4.)

Erityyppisiä lääkinällisohjelmistoja on lukuisia. Tutkimuskäytössä on ohjelmia mm. DNA:n (deoksiribonukleiinihappo) ja RNA:n (ribonukleiinihappo) sekvensoimiseen ja anemian yleisyyden seuraamiseen valtiokohtaisesti (5). Lääkinällisohjelmistojen piiriin kuuluvat myös esimerkiksi sähköisiä lääkeresepitejä välittävät palvelut (6). Perinteisempiä lääkinällisohjelmistoja edustavat sairaaloiden potilastietojärjestelmät, joissa ylläpidetään asiakas- tai potilasasiakirjoja. Omakanta-palvelu on esimerkki potilastietojärjestelmästä, jota voi käyttää sairaalahenkilökunnan lisäksi potilas itse niin halutessaan. Kuvassa 1 on kuvakaappaus Omakanta-potilastietojärjestelmän verkkosivuston etusivusta.

The screenshot shows the Omakanta website homepage. At the top, there is a navigation bar with three tabs: 'Kansalaiset' (selected), 'Ammattilaiset', and 'Järjestelmäkehittäjät'. To the right of the tabs are language options: 'På svenska', 'In English', and 'Other languages'. Below the navigation bar is the 'Kanta' logo and a search bar with the text 'Hae'. To the right of the search bar is a green button labeled 'Kirjautu Omakantaan' with a user icon. Below the navigation bar is a horizontal menu with items: 'Ohjeet ja asiointi', 'Tietojen käyttö ja turvallisuus', 'Tuki', 'Ajankohtaista', and 'Tietoa Kanta-palveluista'. The main content area is titled 'Omakanta' and includes a sidebar on the left with the heading 'OHJEET JA ASIOINTI' and several menu items: 'Tiedot Kannassa', 'Omakanta' (selected), 'Ohjeita uusimispyynnön lähettämiseen', 'Omakannan uudet ominaisuudet', 'Omakannan verkkokoulu', 'Asiointi toisen puolesta', and 'Ulkomailla'. The main text area explains that users can view their health information and prescriptions, request new prescriptions, and store their medical history. It lists what users can see in their Omakanta account: prescriptions, treatments, laboratory and X-ray results, and annual maintenance information. It also lists what users can do in their Omakanta account: request new prescriptions, store their medical history, and request their information to be shared with other European countries. A small image of a woman using a smartphone is shown on the right side of the page. Below the image is a text box with the heading 'Kaikki terveystietosi ja reseptisi yhdestä paikasta' and a link to 'Tutustu verkkokouluun'.

Kuva 1. Omakanta-verkkosivuston etusivu (7).

Diagnostiikkaa tai analytiikkaa tarjoaville lääkintälaitteohjelmistoille erityisen tärkeitä ovat PACS-järjestelmät (Picture And Communication System), joita käytetään lääkinällisten kuvantamisten siirtämiseen kuvantamislaitteiden ja lääkäreitten työasemien välillä sekä kuvien tallentamiseen. PACS-järjestelmien avulla lääkärit pystyvät jakamaan kuvia keskenään ja tarkastelemaan niitä miltä tahansa järjestelmään kytkeytyneeltä päätelaitteelta. (8.) Lääkintälaitteohjelmistoille, joiden toiminnot ovat riippuvaisia lääkinällisistä kuvantamista, on erityisen tärkeää tarjota mahdollisuus kuvien lataamiseen joko käyttäjän työasemalta tai suoraan PACS-järjestelmästä.

2.2 Alan tietotekniikan standardit

DICOM

DICOM-standardi (Digital Imaging and Communications in Medicine) määrittelee, miten lääketieteellisiä kuvantamisia ja muuta digitaalista dataa tulee käsitellä ja hallinnoida (9). DICOM-standardi kattaa niin lääketieteellisten kuvantamisten tallennusformaattien määritelmät kuin kuvantamisten tiedonsiirrossa käytettävät protokollat. DICOM-standardi syntyi vuonna 1993 ACR:n (American College of Radiology) ja NEMA:n (National Electrical Manufacturers Association) aloitteesta. DICOM-standardin päätavoitteena on taata lääketieteellisten kuvantamisten keskinäinen yhteensopivuus riippumat-

ta siitä, minkä valmistajan laitteella kuvantaminen suoritetaan tai millä järjestelmällä kuvia käsitellään. (10.) Nykypäivänä lähes kaikki radio- ja kardiologiset kuvantamis- ja sädehoitolaitteet käyttävät DICOM-standardia (11).

DICOM-standardin mukaisissa tiedostoissa tieto on koottuna erillisiin tietojoukkoihin. DICOM-tiedosto, joka sisältää esimerkiksi rintakehästä otetun röntgenkuvan, sisältää myös mm. potilaan nimen ja tunnistenumeron. Tällä pyritään varmistamaan, että kuvantaminen pystytään aina kohdentamaan oikeaan henkilöön. (12.) Potilaan nimen, tunnistenumeron ja muiden attribuuttien lisäksi DICOM-tiedosto sisältää aina erikoisattribuutin, joka sisältää kuvantamisen pikselidatan. DICOM-standardissa ei ole rajoitettu DICOM-tiedostoon tallennettavien kuvien määrää, joten useita kuvia voidaan tallentaa samaan DICOM-tiedostoon. Myös 3D- ja 4D-kuvadatan tallentaminen DICOM-tiedostoon on mahdollista.

Health Level 7 International -standardit

Health Level 7 International on Health Level 7 -organisaation ylläpitämä kokoelma standardeja elektronisten sairaskertomusten yhteensopivuuden takaamiseksi (13). Health Level 7 International -standardikokoelma tarjoaa sen käyttäjille viitekehyksen elektronisten sairaskertomuksien vaihtoon, integrointiin, jakamiseen ja hakemiseen. Useimmat Health Level 7 International -standardit eivät ole avoimia standardeja, vaan niiden käytöstä joutuu maksamaan vuosittaisen maksun. (14.)

Health Level 7 International -standardikokoelman tärkeimpiin standardeihin kuuluvat HL7 Version 2, CDA (Clinical Document Architecture), EHR-PHR-järjestelmän toimintamallit ja FHIR (Fast Health Interoperability Resources). HL7 Version 2 -standardi on maailman yleisin potilastietojen ja kliinisen informaation välittämiseen käytetty standardi. (13.) CDA-standardi on XML-pohjainen merkitsemisstandardi, jonka tehtävänä on määrittää kliinisten dokumenttien rakenne ja semantiikka sekä se, kuinka niitä tulee ohjelmoida (14).

2.3 Lääkinnällisiä laitteita koskevat säännökset

Ennen lääkinällisen laitteen markkinoille viemistä tulee sen valmistajan pystyä todentamaan tuotteen turvallisuus, käyttötarkoitukseen soveltuvuus, suorituskyky ja luotetta-

vuus. Laite ei saa myöskään vaarantaa potilaan tai tuotteen käyttäjän terveyttä. Suomessa Valvira valvoo, noudattavatko lääkinnälliset laitteet niille asetettua lainsäädäntöä ja säännöksiä (15). Kansallisten määräysten lisäksi Suomessa myytävien laitteiden tulee noudattaa Euroopan unionin asettamia säännöksiä.

Säännökset Euroopan unionissa

Euroopan Unionin alueella myytäviä lääkinnällisiä laitteita säätelee vuonna 2017 voimaan tullut lääkintälaitteasetus MDR (Medical Device Regulation). Ennen lääkintälaitteasetuksen käyttöönottoa laitteiden säätelyssä käytettiin lääkintälaitedirektiivi MDD:tä (Medical Device Directive). Säännökset eivät poikkea merkittävästi toisistaan, mutta MDR tuo mukaan uusia vaatimuksia, joiden täyttäminen vaatii laitteiden valmistajilta entistä enemmän työtä. (16.)

Lääkintälaitteasetuksen vaatimusten täyttäminen koostuu useista vaiheista. Heti laitteen kehityskaaren alkuvaiheessa tulee päättää lääkinnällisen laitteen käyttötarkoitus ja luokitus (16). Luokituksia on neljä sen mukaan, kuinka suuri riski tuotteen käytöstä aiheutuu potilaalle, tuotteen käyttäjälle tai sivullisille henkilöille. Mitä korkeampi luokitus laitteelle asetetaan, sitä laajemmat ja tiukemmat määräykset sen tulee täyttää. (18.) Lääkintälaitteasetuksen näkökulmasta lääkintälaitteohjelmistoja koskevat samat vaatimukset kuin muitakin vastaavan luokituksen omaavia lääkinnällisiä laitteita (19, s. 10). Luokituksesta riippumatta kaikki laitteet vaativat teknisen dokumentaation ja niiden on läpäistävä kliininen evaluaatio, jossa laitteen toimintaa testataan kliinisessä ympäristössä (20). Jos laite läpäisee kliinisen evaluaation, voidaan laitteen vaatimustenmukaisuus todentaa EU:n hyväksymän ilmoitetun laitoksen toimesta. Suomessa lääkinnällisten laitteiden vaatimustenmukaisuuden todentavia ilmoitettuja laitoksia on kaksi, SGS Fimko Oy ja Eurofins Expert Services Oy (21).

Vaatimusmäärittelyt eivät koske pelkästään valmistettavaa laitetta, vaan myös laitetta valmistavan yrityksen tulee täyttää sille asetetut vaatimukset. Lääkinnällisiä laitteita kehittävilta yrityksiltä vaaditaan mittavien riskien- ja laadunhallintajärjestelmien perustamista, dokumentointia ja toimeenpanemista (22; 23). Laite voidaan CE-merkitä (Conformité Européenne), kun sekä laite että sitä valmistava yritys täyttävät niille asetetut vaatimukset (19, s. 27).

Säännökset Yhdysvalloissa

Yhdysvallat on yksi merkittävimmistä lääkinnällisten laitteiden markkina-alueista Euroopan unionin lisäksi. Yhdysvalloissa lääkinnällisten laitteiden myyntiä säätelee Yhdysvaltain elintarvike- ja lääkevirasto FDA (Food and Drug Administration). Euroopan unionissa myytävien laitteiden tavoin Yhdysvalloissa myytävälle laitteille määritellään luokitus niiden aiheuttamien riskien suuruuden perusteella. (24.) Periaatteen tasolla FDA:n tuotteelle myöntämä hyväksyntä vastaa Euroopan unionissa myönnettävää CE-merkintää (25).

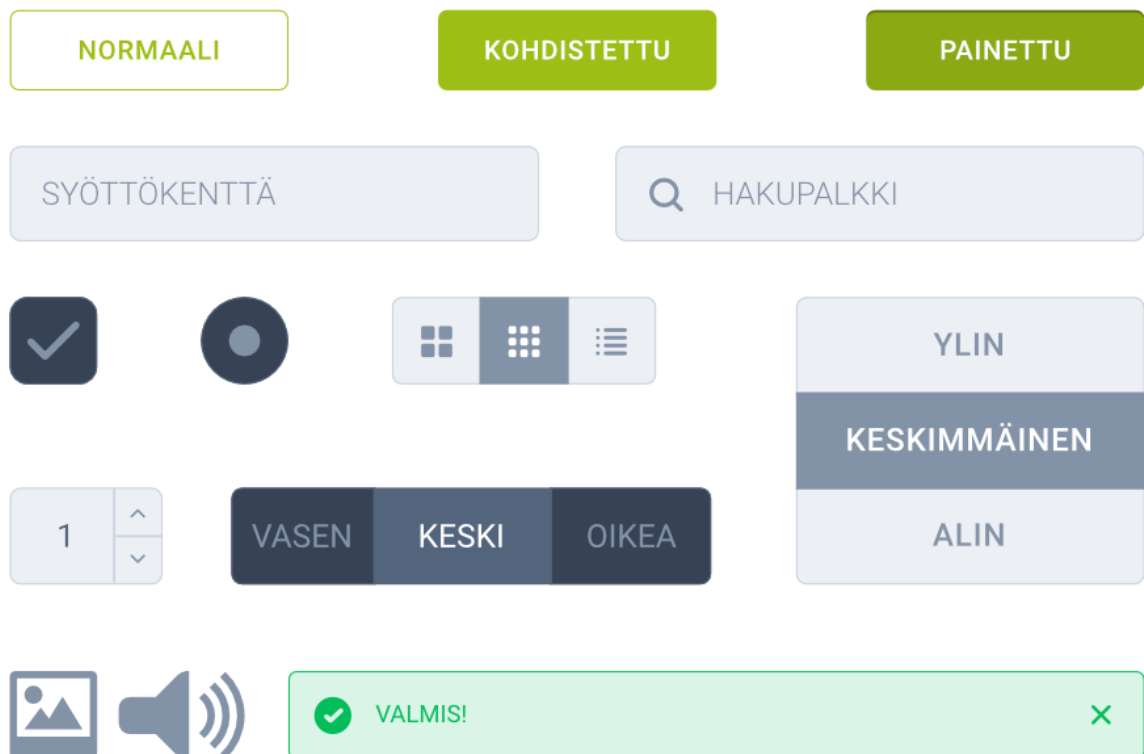
Kaikkien lääkinnällisiä laitteita valmistavien ja myyvien yritysten ja organisaatioiden tulee rekisteröityä FDA:n yhtenäistä rekisteröitymis- ja listausjärjestelmää (FDA's Unified Registration and Listing System) hyväksikäyttäen. Rekisteröitymisen joutuu vahvistamaan järjestelmään vuosittain. Rekisteröityminen järjestelmään ei ole ilmaista, vaan FDA laskuttaa merkittävän summan rekisteröitymisestä koituvista kuluista. Rekisteröitymisen lisäksi organisaation tulee nimetä Yhdysvalloissa yrityksen nimissä toimiva agentti, jonka tehtävän on toimia välikätenä FDA:n ja yrityksen välisessä yhteydenpidossa, vastata laitteeseen liittyviin kysymyksiin yrityksen puolesta ja avustaa FDA:ta järjestämään tarkistuksia yrityksen tuotantolaitoksiin. (26.)

Class II- tai III-luokkaan kuuluvien laitteiden tulee toimittaa FDA:lle markkinoille saatamista edeltävä ilmoitus 510(k). Ilmoituksessa voidaan todentaa, että markkinoille tuotava laite vastaa käyttötarkoitukseltaan ja/tai ominaisuuksiltaan markkinoilla jo olevaa tuotetta siinä määrin, että sen voidaan olettaa olevan yhtä turvallinen ja toimintavarma. Jos FDA päättää, ettei tuote vastaa riittävästi olemassa olevaa hyväksynnän saannutta laitetta, joutuu yritys hakemaan FDA:lta markkinoille saattamista edeltävää hyväksyntää (Premarket Approval) tai todentamaan laitteen turvallisuuden kliinisillä tutkimuksilla. Lääkintälaitteasetuksen tapaan FDA vaatii lääkintälaitteita valmistavalta yritykseltä laadunhallintajärjestelmää. (25.)

3 Käyttöliittymäpohjat

Käyttöliittymäpohjalla (engl. user interface template tai -kit) tarkoitetaan työkalua, jota voidaan käyttää hyväksi digitaalisessa suunnittelussa ja käyttöliittymien toteutuksessa. Kuvassa 2 on havainnollistettu erilaisia käyttöliittymäelementtejä, joita käyttöliittymä-

pohjat tyypillisesti sisältävät. Yksittäisten elementtien lisäksi käyttöliittymäpohja voi sisältää myös suurempia käyttöliittymäkokonaisuuksia, kuten navigointipalkkeja, taulukoita ja alavetovalikkoja. Jotkin käyttöliittymäpohjat sisältävät jopa pieniä käyttöliittymäkokonaisuuksia ja ovatkin jossain määrin rinnastettavissa valmiisiin käyttöliittymäteemoihin. Käyttöliittymäpohjaan sisällytettyjä elementtejä yhdistää yleensä yhtenäinen väriteema, joka on usein käyttäjän muokattavissa. (27; 28; 29.)



Kuva 2. Grade UI Kit -käyttöliittymäpohjan käyttöliittymäelementtejä.

Käyttöliittymäpohjien elementit toteutetaan yleensä rasterigrafiikkaohjelmalla, kuten Adobe Photoshopilla tai Sketchillä. Elementit voidaan luoda myös vektorigrafiikkaohjelmalla ja tallentaa rasterigrafiikkana lopullista käyttöä varten. Grafiikkaohjelmia käyttämällä käyttöliittymäprototyypin luominen on vaivatonta käyttöliittymäpohjan elementtejä hyödyntäen (30). Prototyypin luomisen helppouden vuoksi suunnitteluvaiheessa käyttöliittymästä tehtävien rautalankamallien tekeminen voidaan jättää kokonaan väliin (31).

3.1 Atominen suunnittelu

Atomisen suunnittelun keskiössä on perusajatuksena, että käyttöliittymät voidaan hajottaa yksinkertaisiksi rakennuspalikoiksi samalla periaatteella kuin kaikki universumin materia voidaan hajottaa alkuaineiksi ja edelleen atomeiksi (32). Käyttöliittymät rakentuvat samaan tapaan yksittäisistä elementeistä ja niiden eri yhdistelmistä kuin elementit ja molekyylit rakentuvat yksittäisistä atomeista ja niiden yhdisteistä.

Atominen suunnittelu voidaan jakaa karkeasti viiteen tasoon. Nämä viisi tasoa ovat

- atomi
- molekyyli
- organismi
- malli
- sivu.

Alimmalla tasolla pienimpänä rakennuspalikkana on atomi, josta esimerkkejä ovat painikkeet, tekstikentät ja otsikot. Atomeja yhdistelemällä muodostuu molekyylejä, jotka ovat astetta monimutkaisempia ja yksityiskohtaisempia elementtejä (esim. osoitekenttä). Vastaavasti molekyylejä yhdistelemällä pystytään luomaan organismeja, jotka ovat suurempia kokonaisuuksia, kuten kirjautumislomakkeita ja taulukoita. Erilaisista organismeista rakennetaan malleja, jotka kuvaavat, mistä palasista käyttöliittymä koostuu ja missä ne ovat toisiinsa nähden. Sivun malli on erityinen malli, joka vastaa valmiin käyttöliittymän ulkoasua. (29; 32.)

Tavoitteena atomisessa suunnittelussa on luoda modulaarinen ja helposti päivitettävä käyttöliittymä. Kun käyttöliittymä on rakennettu pienistä palasista, yhden palasen poistaminen tai muokkaaminen ei vaadi muuhun käyttöliittymään kajoamista. Myös tiedostorakenne ja ohjelmistokoodi pysyvät modulaarisina atomista suunnittelua hyödyntämällä, jolloin ohjelmistokoodin ylläpidettävyys paranee. Näitä peruseriaatteita käytetään usein hyväksi käyttöliittymäpohjien suunnittelussa ja käytössä. Käyttöliittymäpohja suunnitellaan niin, että sen elementtejä voidaan yhdistellä ja luoda laajempia ja laajempia kokonaisuuksia. Lopulta eri tasojen elementtejä yhdistelemällä pystytään luomaan kokonaisia käyttöliittymiä. (32; 33.)

3.2 Käyttöliittymäpohjien hyödyt ja haasteet

Etuina käyttöliittymäpohjien käyttämisessä ovat käyttöliittymien toteuttamisen nopeus, ulkoasun yhtenäisyys ja edullisuus. Käyttöliittymäpohjista on apua erityisesti pienemmillä yrityksillä. Käyttöliittymäkehityksessä säästetään merkittävästi aikaa, koska kaikkia käyttöliittymäelementtejä ei tarvitse suunnitella ja toteuttaa itse. Käyttöliittymän ulkoasu pysyy myös helpommin yhtenäisenä, kun elementit on jo alun alkujaan suunniteltu yhdessä (28).

Halvimmillaan käyttöliittymäpohjia saa jopa ilmaiseksi, mutta tyypillisesti valmiit käyttöliittymäpohjat maksavat muutamia kymmeniä euroja. Jos haluaa täysin personoidun käyttöliittymäpohjan, joutuu suunnittelijalle maksamaan työstä jopa tuhansia euroja. (30.) Tämän takia onkin järkevää miettiä, onko mahdollista esimerkiksi laajentaa ja muokata valmista käyttöliittymäpohjaa omien tarpeiden mukaiseksi personoidun käyttöliittymäpohjan teettämisen sijaan.

Käyttöliittymäpohjien käytössä on mukana myös uhkia. Koska käyttöliittymäpohjien kattavuus vaihtelee paljon eri pohjien välillä, on mahdollista, että jokin käyttäjän tarvitsema käyttöliittymäelementti puuttuu pohjasta kokonaan. Tällöin puuttuva elementti joudutaan lisäämään pohjaan jälkikäteen ja käyttöliittymäpohja on työmäärän vähentämisen sijaan lisännyt sitä. (27.) Käyttöliittymäpohjia käytettäessä on lisäksi riskinä, että tuotteen ulkoasusta tulee liian samanlainen jonkin muun tuotteen kanssa, jossa on käytetty samaa tai samankaltaista pohjaa. Valmiita käyttöliittymäpohjia onkin suositeltavaa käyttää käyttöliittymäsuunnittelussa ja toteutuksessa työkaluna, jonka päälle käyttöliittymän lopullinen ulkoasu rakennetaan. (28.)

4 Käyttöliittymäpohjan toteuttaminen Unity-pelimoottorilla

Insinööriyön toimeksiantaja Disior Oy tuottaa lääkäreille ja sairaaloille 3D-ohjelmistoja diagnostiikkaan ja hoidon suunnitteluun. Ohjelmistoissa lähtötietoina käytetään potilaasta otettua magneetti-, tietokonetomografia- tai kartiokeilakuvaa, josta muodostetaan 3D-malli. 3D-mallista ohjelmisto luo rekonstruktoidun matemaattisen mallin ja graafisen esityksen. Mallinnuksesta voidaan hakea esimerkiksi kovakudosten akseleita ja mittoja, joita tarvitaan diagnostiikassa, hoidon suunnittelussa ja leikkaustulosten arvioinnissa (34).

Käyttöliittymäpohjan tarve tuli ilmeiseksi kun silmäkuopan murtumia analysoivan sovelluksen rinnalla alettiin itsenäisesti kehittää toista sovellustuotetta. Toiseen tuoteprojektiin päivitettyt ominaisuudet jouduttiin kopioimaan projektista toiseen manuaalisesti ja erinäisten sovituskonfliktien takia kehittäjien aikaa tuhlaantui päivitysprosessiin huomattavissa määrin.

Tämän ongelman ratkaisemiseksi päädyttiin kehittämään käyttöliittymäpohja, joka yhdistäisi sovellustuotteiden yhteiset toiminnallisuudet yhden pohjan alle, ja samalla yhdenäistäisi eri ohjelmien käyttöliittymien ulkoasut. Käyttöliittymäpohjan suunnittelussa pyrittiin ottamaan huomioon Disiorin nykyisten ja tulevien sovelluksien tarpeita ja vaatimuksia.

Hieman perinteisistä käyttöliittymäpohjista poiketen käyttöliittymäpohjaan sisällytettiin käyttöliittymäelementtien lisäksi myös ohjelmistokomponentteja. Sisällytettyjen komponenttien tarkoituksena oli tarjota käyttöliittymäpohjassa kehittäjille valmiiksi ne toiminnallisuudet, joita kaikissa Disiorin sovellustuotteiden käyttöliittymissä tarvitaan. Tällöin toiminnallisuuksia ei tarvitse erikseen toteuttaa jokaiseen tuotteeseen uudelleen ja niiden päivittäminen onnistuu keskitetysti yhdestä paikasta.

Projektin alkuperäisenä tavoitteena oli liittää käyttöliittymäpohja osaksi tuoteprojekteja versionhallintajärjestelmää hyödyntämällä, mutta tästä toteutuksesta jouduttiin työn aikana luopumaan vastaan tulleiden ongelmien vuoksi. Versionhallinnan sijaan käyttöliittymäpohjan integrointi suoritettiin Unitypackage-paketteja hyödyntämällä.

4.1 Vaatimusmäärittely ja tavoitteet

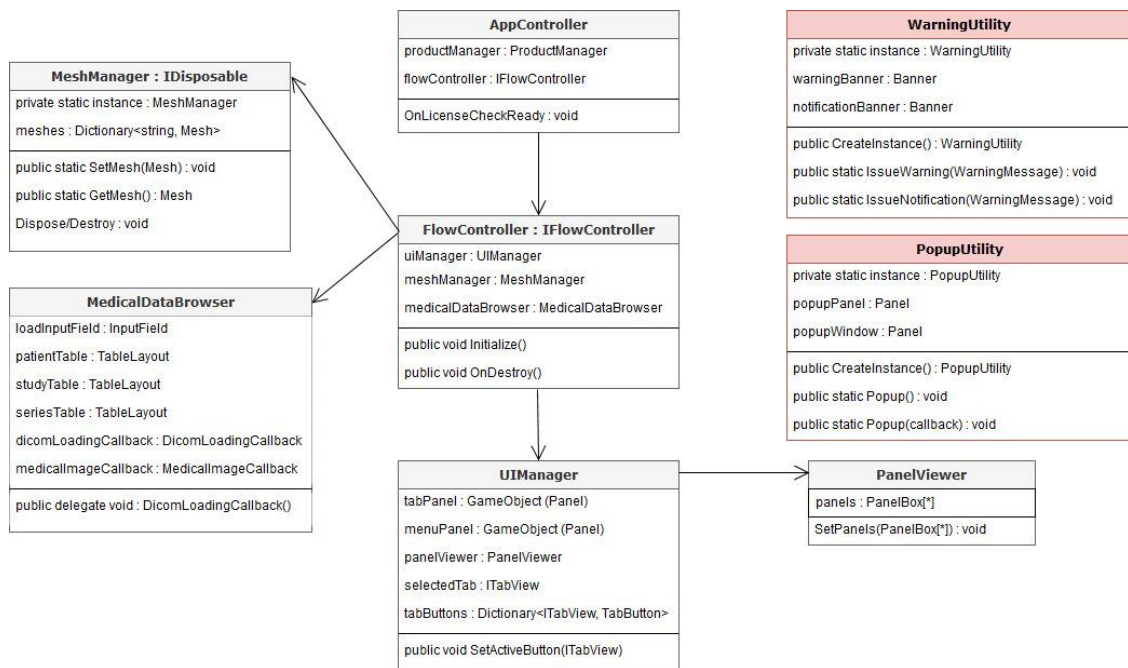
Työn alkuvaiheessa sovittiin yhdessä yrityksen sovelluskehittäjien kanssa käyttöliittymäpohjan vaatimusmäärittelystä. Käyttöliittymäpohjaan toteutettaisiin ne graafiset elementit ja niiden toiminnallisuudet, jotka vaaditaan lopullisten tuotteiden käyttöliittymien rakentamista varten. Käyttöliittymäelementtien lisäksi pohjaan toteutettaisiin ohjelmistokomponentteja tukemaan käyttöliittymäpohjan toimintaa.

Käyttöliittymäpohjalle asetettiin seuraavat vaatimukset:

- Käyttöliittymäpohjan tulee pystyä muodostamaan 3D-malleja DICOM-kuvista.

- Käyttöliittymän tulee mukautua eri näyttöresoluutioille ja kuvasuhteille.
- Valikkopaneelien tulee olla piilotettavissa ja kytkettävissä pois päältä niin ettei käyttäjä pysty vaikuttamaan niihin.
- Pohjan tulee sisältää tiedostoselain DICOM-tiedostojen selaamista ja lataamista varten.
- Käyttöliittymäelementtien tulee olla rakennettu niin, että niitä on helppo muokata ja yhdistellä.
- Käyttöliittymän avulla pitää pystyä ilmoittamaan käyttäjälle erilaisista vika-tilanteista ja pyytämään käyttäjältä syötteitä.

Tavoitteena työssä oli luoda nämä vaatimukset täyttävä käyttöliittymäpohja, jonka päälle niin olemassa olevat kuin tulevat sovellustuotteet voitaisiin rakentaa. Ennen käytännön työn aloittamista käyttöliittymäpohjan ohjelmistokomponenteista laadittiin ohjelmistoarkkitehtuurikaavio luokkatasolla. Kuvan 3 arkkitehtuurikaaviossa ovat kuvattuna ohjelmistokomponenttien riippuvaisuudet toisistaan ja se, miten käyttöliittymän toiminnallisuudet on suunniteltu.



Kuva 3. Käyttöliittymäpohjan ohjelmistokomponenttien arkkitehtuurikaavio luokkatasolla. Punaiset luokat ovat piilotetun yksittäisoliomallin toteuttavat luokat.

Arkkitehtuurikaavio auttoi osaltaan havainnollistamaan, mitä kaikkia toiminnallisuuksia käyttöliittymäpohjaan tulee toteuttaa ja missä järjestyksessä mikäkin toiminto on järkevintä ottaa työn alle. Työnkulku alkoi AppController-luokan toteuttamisella, ja tämän jälkeen työ eteni aina kaavion hierarkiassa seuraavana olevan luokan toteutuksella.

4.2 Käytetyt työkalut

Unity

Unity on järjestelmäriippumaton pelimoottori ja integroitu ohjelmointiympäristö interaktiivisen median luomista varten. Pelimoottorin ja integroidun ohjelmointiympäristön lisäksi Unity tarjoaa Visual Studio Community -koodieditorin skriptien luomista, testaamista ja kääntämistä varten, mutta myös muiden koodieditorien käyttöä tuetaan. (35.) Unity tukee sovelluksien julkaisemista 27 eri alustalle. Tuettuihin alustoihin ja käyttöjärjestelmiin lukeutuvat mm. Windows, iOS, Android, Playstation 4, Xbox One ja Nintendo Switch (36).

Ensimmäinen Unity-versio julkaistiin Tanskassa 6.6.2005 David Helgason, Joachim Anten ja Nicholas Francisin toimesta Applen Mac OS X -käyttöjärjestelmälle (37, s. 8). Tähän päivään mennessä Unitystä on julkaistu seitsemän eri versiota ja satoja pienempiä päivityksiä (38). Työssä käytettiin Unity-versiota 2018.3.10f1.

Unity tarjoaa WYSIWYG-periaatteeseen nojaavan graafisen editorin (kuva 4), jonka tehtävänä on suoraviivaistaa ja helpottaa Unityllä tehtävää kehitystyötä puhtaasti ohjelmointipohjaiseen kehitystyöhön verrattuna. Editorissa voi mm. luoda käyttöliittymiä, lisätä projektiin resursseja kuten kuvia, valmiita 3D-malleja ja lisäosia, käsitellä 3D-malleja reaaliajassa ja hallinnoida animaatioita (39). Luoduille objekteille voidaan lisätä erilaisia ohjelmointikomponentteja niiden toiminnallisuuksien kasvattamiseksi. Komponenttien muuttujien muokkaaminen onnistuu suoraan editorissa ohjelmakoodiin kajoamatta, jos muuttujat on koodissa julkistettu julkisina tai SerializeField-attribuutilla (40).



Kuva 4. Unity Pro 2018.3.10f1 -version käyttöliittymä.

Ohjelmointikielenä Unityssä käytetään C#-kieltä, mutta itse pelimoottori on ohjelmoitu C++-kielellä. Niin C# kuin C++ ovat olio-ohjelmointikieliä. Oletuksena kaikki käyttäjän luomat skriptit perivät Unityn oman MonoBehaviour-luokan. MonoBehaviour-luokka mahdollistaa mm. komponenttien liittämisen skripteihin ja objektien poistamisen manuaalisesti. Unity on asteittain siirtymässä MonoDevelop-koodieditorista Visual Studio Community -koodieditoriin, mikä mahdollistaa tulevaisuudessa .NET 4.6 -sovelluskehiksen ja C# 6.0 -standardin käyttöönoton. Molemmat teknologiat tuovat mukanaan uusia ominaisuuksia kehitysympäristöön. (35.)

Pelinkehityksen lisäksi Unityä käytetään mm. CAD-ohjelmien korvikkeena arkkitehtuurissa ja tutkimuskäytössä tuloksien visualisoinnissa. Unityn käyttäjilleen teetättämässä kyselyssä jopa kolmannes kyselyyn vastanneista käyttäjistä käytti Unityä johonkin muuhun tarkoitukseen kuin pelien kehittämiseen (37, s. 17–18).

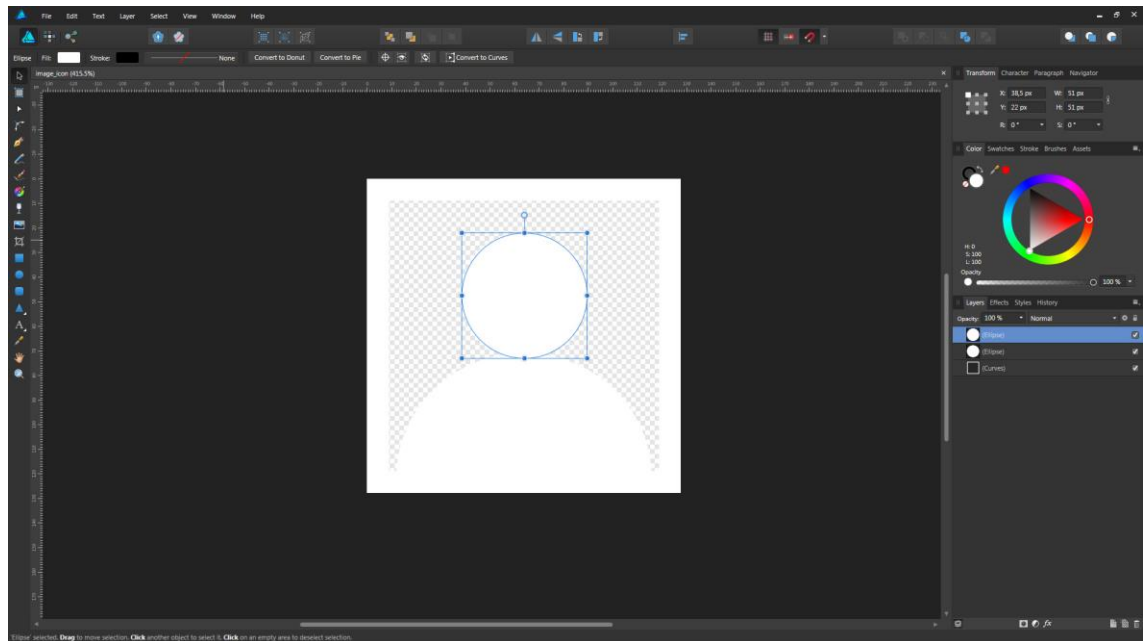
Disiorin sovellustuotteiden käyttöliittymä on toteutettu Unityllä, ja siksi se toimi myös insinööriyön pääasiallisena työkaluna. Alun perin Disiorilla päädyttiin käyttämään Unityä sen tarjoamien 3D-mallintamisominaisuuksien ja Disiorin työntekijöiden aikaisempien kokemusten perusteella. Vaikka Unityn käyttäminen on lääkintälaitteohjelmistojen kehityksessä poikkeuksellista, ei Disior ole ainut yritys, joka on hyödyntänyt Unityä terveysteknologian alalla (41).

Affinity Designer

Affinity Designer on Serif Europan vuonna 2014 julkaisema ammattilaistason vektorigrafiikkaohjelma. Alun perin Affinity Designer oli saatavilla vain Applen macOS-käyttöjärjestelmälle, mutta vuonna 2016 se julkaistiin myös Microsoft Windows -käyttöjärjestelmälle. Affinity Designerilla voi vektorigrafiikan luomisen lisäksi editoida rasterigrafiikkaa. Affinity Designerin ominaisuuksiin lukeutuvat tiedostojen tallentaminen lukuisilla eri formaateilla, persoonat, joilla voi tarkastella vektorigrafiikkaelementtejä rasteriformaatissa, kyky zoomata kuvaa jopa miljoonaprosenttisesti ja yli 8 000 toimintoa kattava työhistoria. (42.)

Käyttöliittymäelementit toteutettiin vektorigrafiikkaohjelmalla toteutettavien elementtien yksinkertaisuuden ja vektorigrafiikkaohjelmien tarjoamien helppokäyttötoimintojen takia. Kaikki toteutetut käyttöliittymäelementit olivat yksinkertaisia matemaattisia muotoja noudattavia objekteja, joiden toteuttamiseen vektorigrafiikkaohjelmat ovat omiaan. Vektorigrafiikkaohjelmilla on helppo tallentaa erikokoisia rasterikuvia lopullisia käyttökohteita varten ilman, että kuvan skaalaaminen heikentäisi kuvanlaatua. (43.)

Työn alkuvaiheessa vektorigrafiikan luomiseen harkittiin Gravit Designer -vektorigrafiikkaohjelmaa, mutta lyhyen testijakson aikana ohjelman ilmaisversion rajoitukset tulivat kuitenkin selvästi vastaan. Affinity Designer valikoitui lopulta työn vektorigrafiikkaohjelmaksi sen tarjoamien kattavien ominaisuuksien, Gravit Designeria selkeämmän käyttöliittymän (kuva 5) ja hintansa vuoksi, joka on erityisesti Adobe Illustratoriin verrattuna erittäin kilpailukykyinen (42).

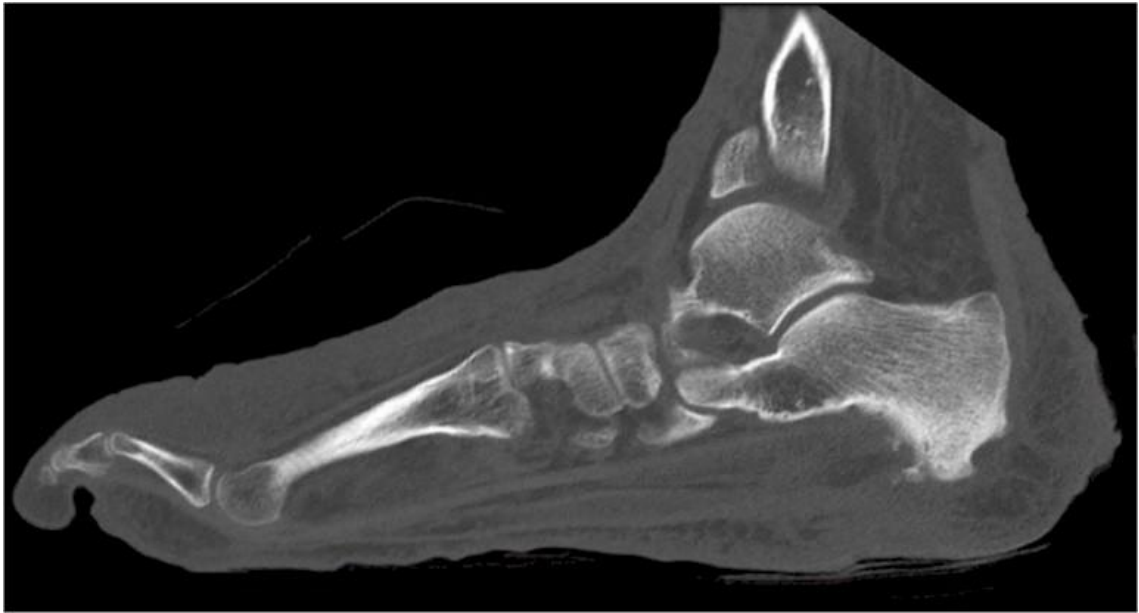


Kuva 5. Affinity Designer -vektorigrafiikkaohjelman käyttöliittymä.

4.3 Käyttöliittymäelementtien toteutus

Käyttöliittymäelementtien toteutus alkoi elementtien värimaailman valitsemisella. Disiorin ohjelmien alkuperäisessä värimaailmassa oli vaikutteita niin tummista kuin vaaleista käyttöliittymäteemoista ja kontrastiero eri käyttöliittymäelementtien välillä oli suurta. Vaaleita käyttöliittymäteemoja katsellessa ihmisen pupillit pienenevät, jotta silmään ei pääsisi liikaa valoa. Vastaavasti tummilla teemoilla pupillit laajenevat, jotta verkkokalvoille kerääntyisi enemmän valoa selkeän kuvan luomiseksi. (44.) Alkuperäinen värimaailma täten pakotti käyttäjän mukautumaan jatkuvasti kontrastiltaan vastakkaisiin väreihin, mikä rasittaa käyttäjän silmiä tarpeettomasti. Tästä syystä käyttöliittymäelementtien värimaailma päädyttiin yhtenäistämään ja käyttämään vain yhden teeman mukaisia värejä.

Tumman värimaailman valitsemista vaalean sijaan tukivat useat syyt. DICOM-kuvat ovat taustaväriiltään mustia, ja kuvannetut ruumiinosat näkyvät kuvissa tyyppisesti valkoisen ja harmaan sävyinä (kuva 6), mutta myös erilaisia väripaletteja käyttäviä DICOM-kuvia on olemassa (45).



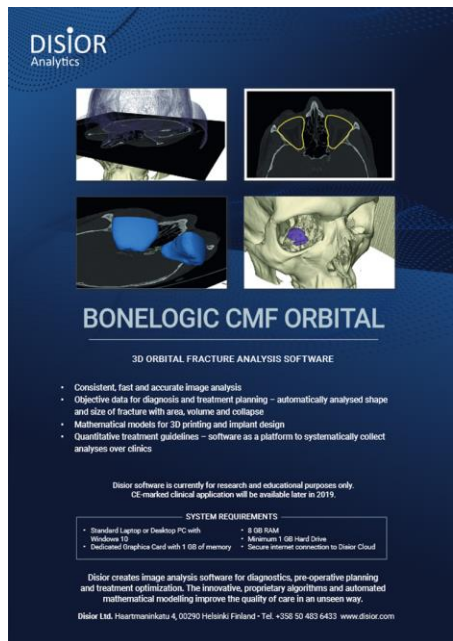
Kuva 6. Tyypillinen nilkasta otettu tietokonetomografiakuva (45).

Disiorin ohjelmissa on 3D-mallien tarkastelun lisäksi mahdollista tarkastella myös DICOM-kuvia. DICOM-kuvien tarkastelu tummaa taustaa vasten on täten miellyttävämpää pienemmän kontrastivaihtelun takia. DICOM-kuvista luotava 3D-malli kuvaa tyypillisesti ihmisen luita, jolloin on luonnollista, että 3D-malli on väriltään valkoinen. Valkoinen malli erottuu tummasta taustasta paremmin ja yksityiskohtaisempana kuin vaaleasta, mikä entisestään tukee tumman värimaailman käyttämistä.

Kun tekstiä luetaan suuria määriä kerrallaan, on vaaleaa tekstiä vaikeampi lukea tummalta pohjalta kuin lukea tummaa tekstiä vaalealta pohjalta. Tekstinlukuun keskittyvissä ohjelmissa ei tästä syystä suositella käytettäväksi tummia teemoja. (47.) Disiorin ohjelmat painottuvat kuitenkin pitkälti 3D-mallien manipulointiin ja tarkasteluun eikä käyttöliittymissä ole juurikaan tekstiä, jolloin tumman käyttöliittymäteeman käyttäminen ei muodostu ongelmaksi tekstin luettavuuden kannalta.

Tumman ja vaalean värimaailman yhdistelmästä yksinomaan tummaan värimaailmaan siirtymisen lisäksi myös käyttöliittymäelementtien värisävyt päädyttiin uusimaan yksilöllisemmän lopputuloksen saavuttamiseksi. Vaikutteita värisävyihin haettiin Disiorin markkinointimateriaaleista. Kuvassa 7 on esitetty Disiorin Bonelogic CMF Orbital -tuotteen esite sekä työssä valitut värisävyt. Värisävyiksi valikoitui erilaisia sinisen sävyjä, jotka ovat linjassa markkinoinnissa käytettävien värien kanssa. Sininen väri mielletään rauhalliseksi, turvalliseksi, avoimeksi ja luotettavaksi (48). Erityisesti kliinisessä

ympäristössä nämä kaikki ovat toivottavia ominaisuuksia. Sininen on tämän lisäksi värisokeille hyvin sopiva väri (49). Värimaailma haluttiin pitää mahdollisimman neutraalina ja puhtaana, minkä takia värimaailmaan ei valittu sinisen lisäksi muita värisävyjä. Värimaailman tiukka rajaaminen on erityisen tärkeää tummissa käyttöliittymäteemoissa (50).



#0F1934

#16203B

#1C3363

#063B73

#074489

#0A61C3

#325CB2

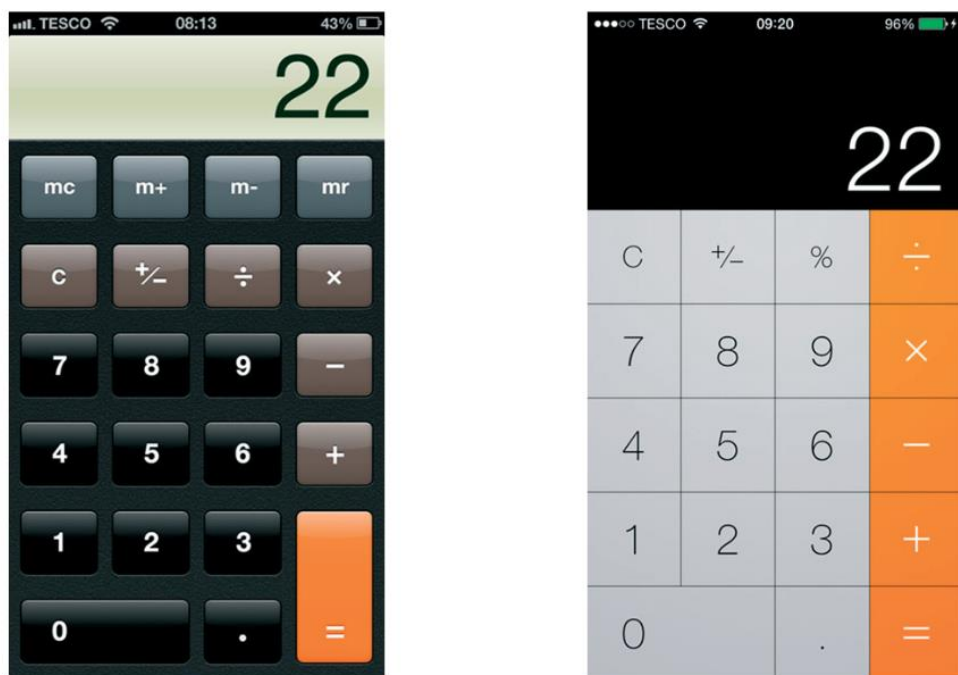
#13141A

Kuva 7. Värimaailman inspiraationa käytetty Disiorin markkinointimateriaali ja käyttöliittymäelementeissä käytetyt värit heksadesimaaleineen.

Taustaväreinä käytettiin tummempia ja matalamman värikylläisyyden omaavia värejä. Tummemman värin käyttäminen taustalla helpottaa muiden elementtien korostamista kirkkaammilla väreillä (51). Värikylläisyydeltään valittu taustaväri on keskiluokkaa. Värikylläisyys ilmoittaa, kuinka voimakas väri on, ja mitä matalampi värin värikylläisyys on, sitä harmaampi se on. Värikylläisyydeltään matalampi väri on miellyttävämpi silmälle kuin korkean värikylläisyyden omaava väri (51). Värin voimakkuutta voi laskea myös sen kirkkautta laskemalla, jolloin väri muuttuu harmaamman sijasta valkoisemmaksi. Affinity Designer tarjoaa mahdollisuuden säätää värin sävyä, värikylläisyyttä ja kirkkautta liukusäätimillä, jos värjäysasetuksista valitsee HSL (Hue Saturation Luminance) Slider -vaihtoehdon.

Kirkkaampia ja voimakkaampia sinisen sävyjä käytettiin korostevärinä niissä käyttöliittymäelementeissä, joihin käyttäjän huomio halutaan kohdentaa (51). Tietyissä mielessä asioiden korostaminen voimakkailla väreillä perustuukin nimenomaan siihen, että voimakkaat värit ärsyttävät silmiä enemmän ja täten saavat käyttäjän reagoimaan ärsykeeseen.

Käyttöliittymäelementtien muotoilussa noudatettiin minimalistista flat design -tyyliä. Minimalistisuudella pyrittiin ajankohtaiseen, selkeään ja kliniseen ulkoasuun. Flat design -tyyliin päädyttiin sen erinomaisen mukautuvuuden ja käytettävyyden takia (52, s. 41–45). Käyttöliittymäkehittäjän näkökulmasta käyttöliittymäelementtien toteuttaminen flat design -tyyliin on nopeaa ja elementtien välisen tyylin pitäminen yhtenäisenä helppoa. Lääkintälaiteohjelmistoissa flat design -tyylin käyttö on harvinaista, minkä takia sitä hyödyntävät ohjelmat erottuvat kilpailijoistaan helpommin. Kuvassa 8 on havainnollistettu flat design -tyylin ja perinteisempää tyyliä edustavan skeuomorfismin eroja.



Kuva 8. Vasemmalla skeuoformismia edustava iOS 6 -käyttöjärjestelmän laskinsovellus. Oikealla flat design -tyyliä edustava iOS 7 -käyttöjärjestelmän laskinsovellus (52, s. 37).

Fonttina käyttöliittymäelementeissä käytettiin Roboto-fonttiperhettä. Roboto on Googlen jakelema pääteviivaton fonttiperhe. Fonttiperheeseen kuuluu kuusi eripaksuuksista fonttia ja jokaisesta fontista on tarjolla lisäksi kursivoitu versio. Roboto-fonttiperheestä on julkaistu kaksi johdannaista fonttiperhettä, kapeampi Roboto Condensed ja pääte-

viivallinen Roboto Slab. Roboto oli käytössä Disiorin ohjelmissa jo työn aloitusvaiheessa, eikä sen vaihtamista toiseen fonttiin nähty tarpeelliseksi. Fontin selkeys ja päätevii-vattomuus ovat lääkintälaiteohjelmistoissa käytettävissä fonteissa toivottavia piirteitä (53).

Hyvin usein käyttöliittymän ulkoasu ja estetiikka ajavat käytettävyyden edelle käyttöliittymäelementtien suunnittelussa, mutta lääkintälaiteohjelmistoissa tilanne on päinvas-tainen (49). Käyttäjäkeskeisyys ja ohjelman väärinkäytön riskin minimointi ovat lääkin-tälaiteohjelmistokehityksen keskiössä käyttöliittymäsuunnittelun lisäksi myös kaikilla muilla osa-alueilla. Käyttöliittymä suunnitellaan ohjelman loppukäyttäjät aina mielessä pitäen, jotta ohjelman käyttö on heille mahdollisimman intuitiivista. (54.) Käyttöliittymäs-tä aiheutuvat mahdolliset väärinkäyttötilanteet tunnistetaan, analysoidaan ja käsitel-lään. Tällä menettelyllä pyritään välttämään väärinkäyttötilanteet kokonaan tai mini-moimaan niistä aiheutuvia seuraamuksia (55). Lopuksi kaikki riskienhallinnan vaiheet kirjataan tuotteen laatudokumentointiin.

Yksittäisissä käyttöliittymäelementeissä käyttäjäkeskeinen ajattelutapa on otettu huo-mioon mm. kuvakkeiden suunnittelussa ja siinä, miten kuvakkeita käytetään yhdessä muiden käyttöliittymän elementtien kanssa sekä käyttöliittymäelementtien yleisessä ulkoasussa. Kuvakkeet suunniteltiin yksinkertaisiksi ja yleisilmeeltään universaalisti tunnistettavia kuvakkeita vastaaviksi (56). Tällä pyrittiin minimoimaan riskiä, jossa käyt-täjä tunnistaa johonkin käyttöliittymään toimintoon liittyvän kuvakkeen väärin. Niissä tapauksissa, joissa yksiselitteistä kuvaketta ei ollut mahdollista toteuttaa, liitettiin ku-vakkeen läheisyyteen tekstiseloste (53).

Käyttöliittymäelementtien käytännön toteutus Affinity Designerilla oli verrattain suoravii-vaista elementtien minimalistisen ja yksinkertaisen muotoilun vuoksi. Ruudukoiden ja nappauksen (engl. snapping) avulla symmetristen muotojen luominen oli helppoa. To-tuusarvo-operaattoreita hyödyntämällä monimutkaisempia elementtejä (kuten kuvak-keita) pystyttiin toteuttamaan vaivattomasti piirtämällä yksinkertaisia muotoja päällekkäin ja sitten poistamalla elementtien päällekkäisyydet kuvasta (57).

Valmiit elementit ja kuvakkeet tallennettiin myöhemmin tehtäviä muokkauksia varten .afdesign-tiedostoformaattissa, joka on Affinity Designerin oma työtiedostoformaatti. Käyttöliittymäpohjaan lisättävät kuvat tallennettiin läpinäkyvyyttä tukevassa PNG-kuvaformaattissa. Koska Unityssä kuvien väriä voidaan vaihtaa Image-komponentin

color-muuttujan arvoa muuttamalla, Affinity Designerissa kaikki yksiväriset kuvat tallennettiin valkoisina. Image-komponentissa valittu väri piirtyy valkoiseen kuvaan muuttumattomana, mutta muun värisissä kuvissa kuvan pohjaväri paistaa valitun värin läpi ja täten sekoittuu osaksi lopullista näytöllä näkyvää väriä.

4.4 Unity-toteutus

Suurin osa käytännön työstä tehtiin Unity-pelimoottorilla. Käyttöliittymäpohjaa varten Unityllä luotiin uusi projekti ja Affinity Designerilla toteutetut käyttöliittymäelementit ja valitut fontit lisättiin projektiin resursseina. Käyttöliittymäelementtien ja fonttien lisäksi Unity-projektiin lisättiin resursseina valmiiksi olemassa olevia grafiikoita, varjostimia, materiaaleja ja Ookii.Dialogs.WinForms.1.0.0 -kirjasto. Ookii.Dialogs-kirjastoa käytettiin tiedostoselainikkunoiden avaamiseen ohjelman aikana. Ookii.Dialogs hyödyntää Windowsin omia tiedostoselainikkunoita, mikä tekee tiedostoselaimen käytöstä käyttäjäystävällistä, kun sen ulkoasu ja toiminnot ovat useimmille käyttäjille entuudestaan tuttuja (58).

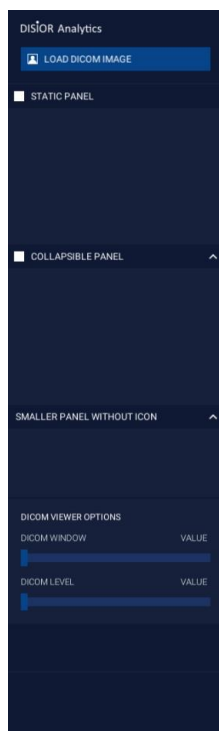
App- ja FlowController-luokat

Ohjelman yleistä toimintaa hallinnoivat App- ja FlowController-luokat. ApplicationController-luokan pääasiallinen tarkoitus on suorittaa toimintoja, jotka ovat kaikille tuoteprojekteille yhteisiä. Näihin toimintoihin lukeutuu mm. käyttäjän lisenssin tarkistaminen. ApplicationController-luokan oliio tarkistaa ohjelman käynnistyksen yhteydessä, onko asiakkaan lisenssi vielä voimassa, ja tämän jälkeen kutsuu IFlowController-rajapinnan toteuttavaa oliiota, joka alustaa ohjelman käyttöliittymän (olettaen, että käyttäjän lisenssi oli edelleen voimassa). Se, mitä kaikkea näytöllä halutaan esittää ohjelman alussa, vaihtelee tuotteesta toiseen. Tämän takia IFlowController-rajapinnan toteuttava luokka ja alustus-funktio määritellään jokaisessa tuoteprojektissa erikseen.

IFlowController-rajapinnan toteuttavan luokan tehtävänä on hallinnoida ohjelman varsinaista työkulkua. Yhtenä Flow-luokan tärkeimmistä toiminnoista on huolehtia olioiden välisten yhteyksien alustamisesta ohjelman käynnistyessä. Flow-luokan vastuulla on lisäksi DICOM-tiedostoselaimen lukeman DICOM-kuvan vastaanottaminen ja MarchingCubes-algoritmilta lähettäminen. MarchingCubes-ajon valmistuttua Flow-luokka ottaa algoritmin tuottamat solmupiste- ja kolmiotaulukot vastaan ja lähettää ne Mesh-Manager-oliolle, joka luo taulukoista Unityn Mesh-luokan oliion.

Valikkopaneeli ja PanelBox-luokka

Valikkopaneelissa olevien yksittäisten paneelien määrä ja toteutus vaihtelevat ohjelman toimintojen ja vaatimusten mukaan. Tästä syystä käyttöliittymäpohjan valikkopalkista päätettiin tehdä geneerinen säiliö yksittäisille paneeleille, joissa valikkopaneelin varsinaiset toiminnallisuudet sijaitsevat. Valikkopaneelin toteutuksessa käytettiin hyväksi Unityn asemointiryhmäkomponentteja (engl. Layout Group). Lisäämällä vertikaalinen asemointiryhmä valikkopaneelille saatiin sen sisälle lisättävät objektit ryhmittymään oikeaan järjestykseen automaattisesti. Asemointiryhmiä käytettäessä lapsielementtien sijaintia ei myöskään tarvitse erikseen määrittellä. Esimerkkikäyttöliittymässä lapsielementtien ryhmittyminen asetettiin asemointiryhmän asetuksista vasempaan ylänurkkaan. Tällä asetuksella kaikki lapsielementit ryhmittyvät peräkkäin välejä jättämättä valikkopaneelin yläreunaan (kuva 9) ja jättävät mahdollisen tyhjän tilan valikkopaneelin alareunaan.



Kuva 9. Esimerkkikäyttöliittymän valikkopaneeli, jossa erikokoiset yksittäiset paneelit ovat ryhmittäytyneet valikkopaneelin yläreunaan.

Ryhmittelyasetuksen lisäksi asemointiryhmillä on neljä asetusta lapsielementtien skaalaukseen liittyen. Child Controls Size -asetukset leveydelle ja korkeudelle määrittävät saako lapsielementti manuaalisesti määrittellä oman kokonsa vai ei. Child Force Expand -asetukset vaikuttavat siihen skaalaako asemointiryhmä lapsielementit täyttä-

mään paneelissa olevan tyhjän tilan. (59.) Valikkopaneelin asemointiryhmä asetettiin estämään lapsielementtien manuaalisen koon määrittäminen, jotta lapsielementit skaalautuisivat automaattisesti sisältönsä mukaan. Asemointiryhmä pakotettiin tämän lisäksi venyttämään lapsielementtejä leveysuunnassa. Tällä haluttiin varmistaa, että valikkopaneelin sisällä olevat yksittäiset paneelit olisivat aina oikeanleveyisiä ja ettei kehittäjien tarvitsisi huolehtia leveyden asettamisesta jokaisen paneelin kohdalla erikseen.

Jokaiseen valikkopaneelin alaisuudessa olevaan yksittäiseen paneeliin lisättiin komponenttina PanelBox-luokka. Sen tehtävänä on pitää yksittäisten paneelien ulkoasu yhtenäisenä ja tarjota kehittäjille paneelien toimintaan liittyviä perustoiminnallisuksia esim. paneelien esittämiseen ja piilottamiseen. Koska kaikki yksittäiset paneelit sisältävät PanelBox-komponentin ja sen mukanaan tuomat perustoiminnallisuudet, voidaan paneeleita käsitellä geneerisinä objekteina valikkopaneelissa niiden sisällöstä riippumatta.

Tavallinen PanelBox-luokan toteuttava paneeli sisältää kolme pääelementtiä: ylätunnisteen, sisältökentän ja jakajan. Pääelementtien asemointi hoidettiin valikkopaneelin tapaan vertikaalista asemointiryhmää hyödyntämällä. Ylätunniste sisältää paneelin otsikkotekstin ja toimii samalla painikkeena, jolla paneelin sisältökenttä voidaan piilottaa. Paneelin otsikkotekstin viereen voidaan myös lisätä paneelin sisältöä kuvaava kuvake. PanelBox-luokan asetuksista on mahdollista kytkeä sisältökentän piilottaminen kokonaan pois päältä, jos paneelin sisällön piilottaminen käyttäjän toimesta halutaan estää. Otsikkotekstin vieressä oleva kuvake voidaan myös piilottaa suoraan komponentin asetuksista editorin Inspector-paneelistä. Luokalle lisättiin Enable- ja Disable-funktiot, joiden avulla koko paneelin tai pelkästään sen sisällön voi kytkeä pois päältä. Tätä ominaisuutta voidaan käyttää hyväksi esimerkiksi ohjelman työkulun ohjaamisessa.

Välilehtipaneeli ja ITabView-rajapinta

Välilehtipaneeli toteutettiin hyvin samanlaisella periaatteella kuin valikkopaneeli. Välilehtipaneeli toimii säiliönä varsinaisille välilehdille ja horisontaalisen asemointiryhmän avulla ryhmittelee välilehdet paneeliin sisällä automaattisesti. Välilehtien automaattinen asemoituminen on havainnollistettu kuvassa 10. Välilehtipaneelin RectTransform-komponentin ankkurointiesiasetus (engl. Anchor Preset) asetettiin yläreunan horisontaaliseen venytykseen, jonka avulla välilehtipaneeli täyttää aina näytön yläreunan näytön kuvasuhteesta ja resoluutioista riippumatta.



Kuva 10. Esimerkkikäyttöliittymän välilehtipaneeli, jossa kaksi välilehteä ovat automaattisesti ryhmittäytyneet valikkopaneelin vasempaan reunaan.

ITabView-rajapinnan tarkoituksena on tarjota yhteinen rajapinta välilehtien näyttämiseen ja piilottamiseen. ITabView-rajapinnalla on kaksi metodia, GetPanels ja SetVisible. SetVisible-metodille annetaan parametrina totuusarvomuuttuja, joka määrittää, näytetäänkö vai piilotetaanko välilehti. GetPanels-metodi on UIManager-luokan hyödyntämä funktio, joka palauttaa kaikki välilehdelle kuuluvat valikkopaneelit, jotta ne voidaan asettaa aktiiviseksi välilehden vaihtamisen yhteydessä.

UI- ja MeshManager-luokat

UIManager-luokka huolehtii käyttöliittymän perusoperaatioista, kuten välilehtien vaihtamisesta, valikkopaneelien asettamisesta ja käyttöliittymän alustamisesta, kun ohjelma käynnistetään. Luokka sisältää valikko- ja välilehtipaneelien peliobjektit sekä totuusarvomuuttujan, jonka arvo määrittää, piilotetaanko kaikki hierarkiassa valikkopaneelin alla olevat PanelBox-luokan toteuttavat oliot alustuksen aikana. Nämä muuttujat paljastetaan Unityn editorille SerializeField-attribuutilla, jolloin kehittäjät pystyvät asettamaan muuttujien arvoja suoraan editorin Inspector-paneelistä.

Alustuksen yhteydessä UIManager etsii kaikki välilehtipaneelin alaisuudessa olevat TabButton-luokan painikkeet ja lisää painikkeisiin kuuntelijan, joka kutsuu välilehteä vaihtavaa funktioita, kun painiketta painetaan. Tämän lisäksi UIManager luo sanakirjatyypin listan (engl. Dictionary) kaikista TabButton-luokan painikkeista ja ITabView-rajapinnan toteuttavista olioista, jotka kuuluvat kullekin painikkeelle. Sanakirjalistaa käytetään hyväksi silloin, kun Flow-luokan olio haluaa aktivoida välilehden, jota ei ole vielä kytketty päälle. Flow-luokka lähettää UIManagerille ITabView-rajapinnan toteuttavan olion, ja UIManager hakee sanakirjalistasta sitä vastaavan TabButton-luokan painikkeen ja asettaa sen aktiiviseksi hierarkiassa välilehden vaihtamisen lisäksi.

MeshManager-luokka huolehtii nimensä mukaisesti verkkojen hallinnoimisesta ja luomisesta. MeshManager-luokka pitää kirjaa kaikista luoduista verkoista sanakirjatyypisessä listassa. Verkon luomisen yhteydessä MeshManagerille annetaan merkkijonomuuttujassa verkon nimi, jota käytetään verkkojen yksilöimiseen sanakirjalistassa. MeshManagerille toteutettiin GetMesh-funktio, joka ottaa vastaan parametrina halutun verkon nimen ja vertaa sitä sanakirjalistassa oleviin merkkijonoihin. Onnistuneen haun seurauksena funktio palauttaa referenssin löydettyyn verkkoon.

Verkon hakemisen lisäksi MeshManageriin toteutettiin samalla periaatteella toimiva DestroyMesh-funktio, joka poistaa halutun verkon. DestroyMesh-funktio ei käytä Mesh-luokan Clear-toiminnallisuutta verkon poistamiseen, sillä tämä toiminnallisuus ei poista itse Mesh-luokan objektia vaan ainoastaan tyhjentää objektin sisältämät muuttujat ja asettaa niiden koot nolliksi. Clear-toiminnallisuuden sijasta verkot poistetaan Object.Destroy-funktiolla, jolla pystyy poistamaan minkä tahansa olion, joka perii Object-luokan. Mesh ja kaikki muut Unityn omat komponentit perivät Object-luokan. (60; 61.) Destroy-funktioita käyttämällä varmistetaan siitä, ettei ohjelman muistiin jää lojumaan viittauksia tyhjiin Mesh-luokan olioihin.

NotificationUtility- ja PopupUtility-työkaluluokat

Käyttöliittymäpohjaan toteutettiin erityyppisiä ilmoituksia näytölle tulostava NotificationUtility-luokka ja ponnahdusikkunoita luova PopupUtility-luokka. NotificationUtility-luokan ensisijaisena tarkoituksena on ilmoittaa käyttäjälle mahdollisista vikatilanteista ja muista ilmoitusluontoisista asioista, kuten lisenssin loppumispäivästä ohjelman käynnistämisen yhteydessä. PopupUtility-luokka hallinnoi ponnahdusikkunoiden luontia ja takaisinkutsufunktioiden välittämistä ponnahdusikkunoita kutsuneille luokille.

Molemmat työkaluluokat käyttävät piilotettua yksittäisoliotointamallia jonotusjärjestelmän toteuttamiseksi. Rajoittamalla työkaluluokkien määrän yhteen varmistetaan siitä, että yksi instanssi luokasta ottaa vastaan ja käsittelee kaikki sille kohdistetut funktiokutsut. (62.) Jonotusjärjestelmät toimivat molemmissa työkaluluokissa samalla periaatteella.

Työkaluluokka ottaa ensin vastaan kutsun esimerkiksi ponnahdusikkunan luomisesta. Tämän jälkeen kutsuttu ponnahdusikkuna lisätään jonoon ja ponnahdusikkunoiden esittämistä hallinnoiva totuusarvomuuttuja asetetaan todeksi. Tämän totuusarvomuut-

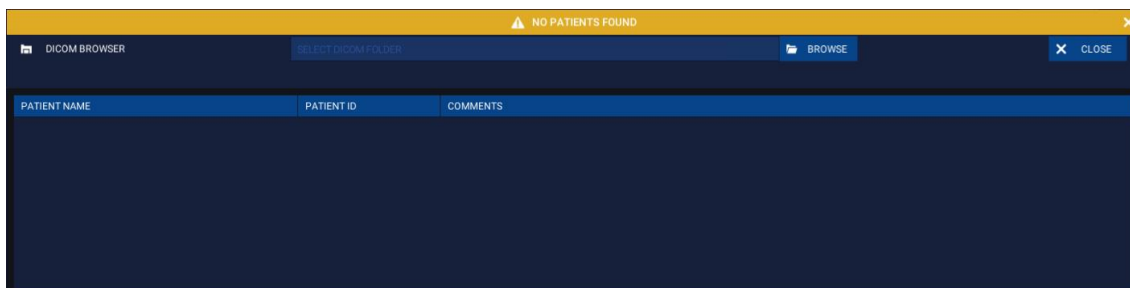
tujan arvo tarkastetaan jokaisen kuvaruudun aikana työkaluluokan Update-funktiossa. Jos totuusarvomuuttujan arvo on tosi, avataan jonossa seuraavaksi vuorossa oleva ponnahdusikkuna. Funktio, jolla ponnahdusikkuna avataan, on MonoBehaviour-luokalta peritty ominaisuus, jolloin sitä pystytään kutsumaan vain ohjelman pääsäikeestä (63). Koska Update-funktio on myös MonoBehaviour-luokalta peritty funktio, käsitellään se aina ohjelman pääsäikeessä. Tällä toteutustavalla mahdollistetaan ponnahdusikkunoiden luominen muistakin säikeistä kuin pääsäikeestä.

Ponnahdusikkunaluokkia toteutettiin projektiin kaksi, ja luokkia lisätään tulevaisuudessa aina tarpeen vaatiessa lisää. Kaikki ponnahdusikkunaluokat perivät IPopup-rajapinnan. Rajapinnan käyttämisellä varmistuttiin siitä, että PopupUtility-luokan jonotusjärjestelmä osasi käsitellä kaikkia ponnahdusikkunoita niiden käyttötarkoituksista ja ominaisuuksista riippumatta. Aina ponnahdusikkunan auetessa, sen taakse avataan koko näytön peittävä paneeli, joka on väriltään musta ja osittain läpinäkyvä. Paneelin tarkoituksena on tummentaa muita näytöllä näkyviä asioita ja täten korostaa näytölle avautunutta ponnahdusikkunaa.

IPopup-rajapinta sisältää Open- ja Close-funktiot sekä Action-tyypin tapahtuman OnExit. Open-funktion tehtävänä on avata ponnahdusikkuna ja välittää sille ponnahdusikkunakutsussa määritetyt aloitusparametrit. Open-funktion parametrilistaan kuuluvat taulukollinen totuusarvo- ja merkkijonomuuttujia. Se, kuinka monta totuusarvo- tai merkkijonomuuttujaa kukin ponnahdusikkuna ottaa vastaan, vaihtelee ponnahdusikkunasta toiseen. Kirjoittamattomana sääntönä ponnahdusikkunoille on myös mahdollista välittää tyhjä parametrilista, jolloin ponnahdusikkuna luodaan ennalta määritetyillä oletusarvoilla. Kehittäjien työn helpottamiseksi jokaisen ponnahdusikkunan kutsufunktion XML-kommenttiin listattiin kaikki yksittäiset parametrit, jotka ponnahdusikkuna ottaa vastaan. XML-kommenttien avulla kommentteja voidaan lukea koodieditorissa minkä tahansa luokan lähdetiedostossa viemällä hiiren kohdistin käytetyn funktiokutsun päälle. (58.)

Close-funktio nimensä mukaisesti sulkee ponnahdusikkunan, ja se kytketään yleensä ponnahdusikkunassa oleviin peruutus- ja sulkemispainikkeisiin. OnExit-tapahtuma kytketään tyyppillisesti Close-funktion lisäksi funktioon, jota kutsutaan, kun käyttäjä painaa ponnahdusikkunassa olevaa varmistuspainiketta. PopupUtility-työkaluluokassa OnExit-tapahtuma on kytketty funktioon, joka piilottaa ponnahdusikkunapaneelin, jos jonotuslistalla ei ole jäljellä muita ponnahdusikkunoita.

NotificationUtility-luokan toteutus pohjautui olemassa olevaan toteutukseen, jonka toiminnallisuuksia laajennettiin. Erilaisia ilmoitustyypppejä luokalla on kolme: vikatilanne, varoitus ja tiedotus. Kutsufunktiossa määritellään ilmoituksen tyyppi ja viesti, minkä jälkeen ilmoitusluokka lisää sen jonoon. Ilmoitus esitetään joko välittömästi, jos jonossa tai näytöllä ei ole aktiivisena muita ilmoituksia, tai sitten kun käyttäjä on sulkenut kaikki aikaisemmat ilmoitukset. Kuvassa 11 on esitetty näytön ylälaitaan, esimerkikikäyttöliittymän välilehtipalkin päälle, ilmestynyt varoitusilmoitus.



Kuva 11. Ohjelman luoma varoitus, kun käyttäjä on valinnut kansion, joka ei sisällä yhtäkään DICOM-tiedostoa.

Ilmoituksien taustaväriä käytettiin ilmoitustyypeille tyypillisiä värejä, jotka poikkeavat käyttöliittymän muusta väriytyksestä merkittävästi (56). Ilmoitukset erottuvat väriytyksensä avulla muusta käyttöliittymästä selkeästi ja täten kiinnittävät käyttäjän huomion helpommin.

DICOM-tiedostojen selaaminen

DICOM-tiedostojen selaamista varten käyttöliittymäpohjaan toteutettiin DICOM-selainnäkyvä, josta käyttäjä pystyy valitsemaan haluamansa DICOM-tiedoston. Ensimmäiseksi DICOM-selaimessa valitaan haluttu kansio, josta DICOM-tiedostoja etsitään Ookii.Dialogs-kirjaston muodostaman kansioselainikkunan avulla. Selain etsii ja kirjaa kaikki DICOM-tiedostot valitusta kansioista ja kaikista sen alikansioista.

DICOM-tiedostojen tietomalli rakentuu neljästä tasosta. Ylimpänä hierarkiassa on potilas, joka sisältää potilaalle ominaisia attribuutteja, kuten potilaan nimen. Seuraavana hierarkiassa ovat potilaalle suoritettut tutkimukset. Yhdelle potilaalle voi olla suoritettuna yksi tai useampia tutkimuksia. Tutkimukset niin ikään rakentuvat yhdestä tai useammasta sarjasta, jotka kattavat yksittäisen kuvantamisen tai muun toimenpiteen. Sarja voi kuulua vain yhteen tutkimukseen kerrallaan. Tietomallin pienimmät osat ovat

komposiittiohjelma-instanssit, jotka esimerkiksi tietokonetomografiakuvantamisissa ovat kuvantamisen yksittäisiä kuvia. (64.) Koska ohjelma vaatii kokonaisen sarjan 3D-mallin muodostamista varten, on komposiittiohjelma-instanssitason elementtien esittäminen DICOM-selaimessa turhaa.

DICOM-tiedostojen lukeminen ei ole natiivisesti tuettuna Unityssä. Tämän puutteen paikkaamiseksi käyttöliittymäpohjaan asennettiin Fellow Oak DICOM -jaettu kirjasto. Fellow Oak DICOM -kirjastosta käytetään yleisimmin nimitystä fo-dicom. Fo-dicom on avoimeen lähdekoodiin pohjautuva kirjasto DICOM-tiedostojen lukemista ja kirjoittamista varten. Tuettujen kehitysympäristöjen listalla ovat .NET, Universal Windows Platform, iOS, Android, Mono ja Unity. (65.)

Fo-dicom-kirjastoa hyödyntämällä DICOM-tiedostoista listattiin 3D-mallien muodostamiseen sopivien sarjojen tunnistamista helpottavia attribuutteja. Löydetyt DICOM-tiedostot jaoteltiin potilas-, tutkimus- ja sarjataulukoihin. Taulukoiden asemointi toisiinsa nähden on esitetty kuvassa 12. Kun potilas valittiin, tutkimustaulukkoon listattiin kaikki potilaalle suoritettut tutkimukset ja edelleen, kun yksi potilaan tutkimuksista valittiin, sarjataulukoon listattiin kaikki valittuun tutkimukseen kuuluvat sarjat. Taulukot toteutettiin TableLayout-lisäosan avulla, joka hyödyntää Unityn omia horisontaalisia ja vertikaalisia asemointiryhmiä taulukoiden luomisessa. Jos jokin taulukoista täyttyi alkioista, luotiin taulukon vasempaan reunaan vierityspalkki, jonka avulla taulukon yli meneviä alkioita voitiin tarkastella.

The screenshot shows a DICOM browser interface with three empty tables. The top table has columns for Patient Name, Patient ID, and Comments. The middle table has columns for Study ID, Accession Number, Study Date, Study Time, and Study Description. The bottom table has columns for Series Number, Series Date, Series Time, Image Type, Modality, and Series Description. A 'LOAD' button is visible at the bottom right of the interface.

PATIENT NAME	PATIENT ID	COMMENTS

STUDY ID	ACCESSION NUMBER	STUDY DATE	STUDY TIME	STUDY DESCRIPTION

SERIES NUMBER	SERIES DATE	SERIES TIME	IMAGE TYPE	MODALITY	SERIES DESCRIPTION

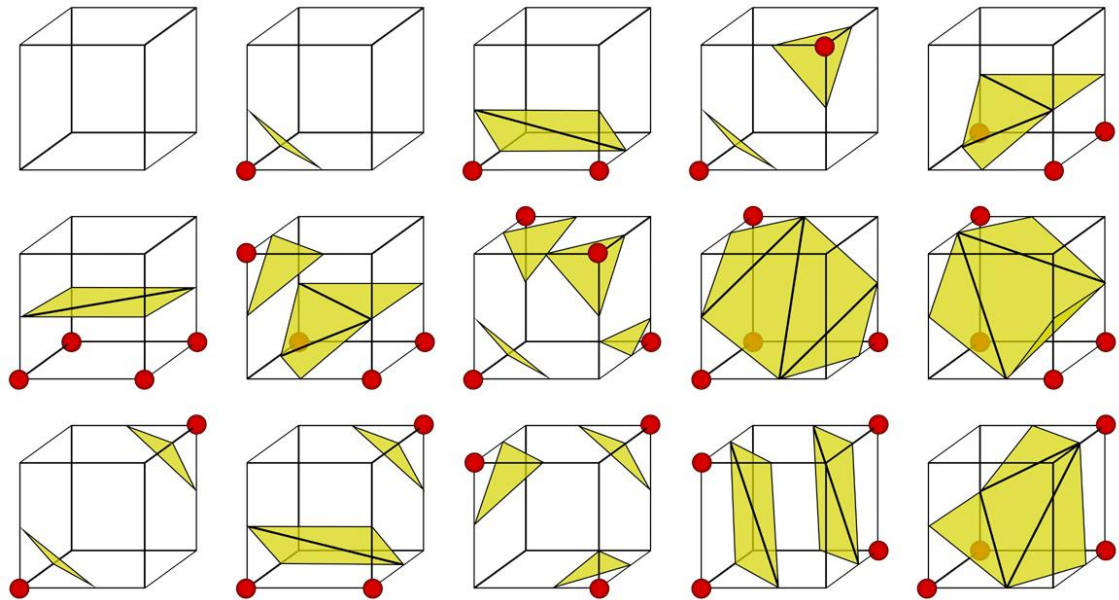
Kuva 12. DICOM-selaimen potilas-, tutkimus- ja sarjataulukot. Anonymiteettisuojaajan säilyttämiseksi taulukoita ei ole täytetty DICOM-tiedostoista luettavilla tiedoilla.

3D-mallin muodostaminen DICOM-tiedostoista

3D-mallin muodostamiseen 2D-kuvista käytettiin MarchingCubes-algoritmia. Sen on kehittänyt General Electric vuonna 1985. Alun perin General Electric patentoi algoritmin, mutta patentti on sittemmin rauennut, ja algoritmi on nykyään kaikkien käytettävissä. MarchingCubes käyttää raja-arvokvantamista 3D-mallin muodostamiseen. Algoritmille annetaan raja-arvo, jota suuremmat arvot lasketaan malliin kuuluviksi ja muut arvot hylätään. (66.) Raja-arvosta käytetään myös englanninkielistä nimitystä isovalue. Tätä nimitystä ei tule kuitenkaan sekoittaa valokuvaamisessa käytettävään ISO-arvoon, joka kuvaa kameran kennon valoherkkyyttä (66). Yksinkertaistettuna DICOM-kuvissa tämä tarkoittaa sitä, että kuvassa valkoisella olevat alueet ovat mallin sisällä ja mustat ovat sen ulkopuolella.

Yksittäiset kolmiot MarchingCubes-algoritmi muodostaa ottamalla 8 arvoa peräkkäisistä kuvista ja muodostamalla niistä imaginäärisen kuution. Tämän jälkeen kuution jokaisen solmupisteen (engl. vertex) arvoa verrataan algoritmille annettuun raja-arvoon ja määritellään, onko se pinnan sisä- vai ulkopuolella. Koska kuutiossa on aina kahdeksan solmupistettä ja niiden arvoilla on vain kaksi eri tilaa, erilaisia tapoja, joilla pinta voi muodostua, on 256. Solmupisteiden kiertosuunta ei vaikuta muodostuvan pinnan topo-

logiaan, jolloin erilaisten pintojen määrä laskee 128:aan, mikä nopeuttaa algoritmin toimintaa. Tämän lisäksi kuution symmetrisyyden ansiosta eri tapauksia voidaan muodostaa samoista pinnoista pyörittämällä kuutiota eri asentoihin. Näiden optimointien jälkeen ainutlaatuisia tapauksia jää jäljelle vain 14. (66.) Algoritmin ainutlaatuiset tapaukset ovat esitettyinä kuvassa 13.



Kuva 13. MarchingCubes-algoritmin ainutlaatuiset tapaukset visualisoituina (70).

Vaikka yksittäisiä DICOM-kuvia katsellaan useimmiten kaksiulotteisessa ympäristössä, ne sisältävät pikselien sijaan kolmiulotteisia vokseleita (68 s. 17). Yksivärisissä DICOM-kuvissa vokselit on tallennettu joko etumerkittöminä tai etumerkillisinä 12–16-bittisinä kokonaislukuina kolmiulotteiseen matriisiin. Jos vokselit ovat etumerkillisiä, ovat niiden arvot tyypillisesti väliltä -1024 – 3071 . Nämä arvot vastaavat lääketieteessä käytetyn HU-asteikon (Hounsfield Unit) arvoja. Asteikossa esimerkiksi ilman arvo on -1000 ja veden 0 . Luiden HU-arvot vaihtelevat 200 ja 2000 :n välillä riippuen siitä kuinka tiheä luu on kyseessä. (69.)

Disiorin ohjelmissa analysoidaan luita, minkä takia MarchingCubes-algoritmi ajetaan oletuksena arvolla 300 . Kaikki DICOM-kuvat eivät kuitenkaan ole HU-skaalan mukaisia, minkä takia ennen algoritmin ajamista käyttäjältä kysytään ponnahdusikkunan avulla, millä arvolla käyttäjä haluaa mallin muodostettavan. Kyselyn avulla käyttäjä voi tarvittaessa muodostaa 3D-mallin niistäkin DICOM-kuvista, jotka eivät noudata HU-skaalaa.

Mallikenttä ja käyttöliittymävalmiselementti

Unity-projektin sisälle luotiin mallikenttä, jossa toteutettuja ohjelmistokomponentteja voitiin testata. Mallikenttä toimi samalla myös käyttöliittymäpohjan toiminnallisuuden esittelyssä. Mallikenttään rakennettiin esimerkkikäyttöliittymä Disiorin olemassa olevien ohjelmien pohjalta. Esimerkkikäyttöliittymään sisällytettiin ominaisuuksiin kuuluivat valikko- ja välilehtijärjestelmä, DICOM-tiedostoselain ja 3D-mallin muodostaminen MarchingCubes-algoritmillä. Kuvassa 14 ovat esitettynä esimerkkikäyttöliittymään toteutetut erilaiset valikkopaneelit, välilehtipaneeli ja MarchingCubesilla nilkan tietokone-tomografiakuvasta muodostettu 3D-malli.



Kuva 14. Nilkan tietokonetomografiakuvasta muodostettu 3D-malli esimerkkikäyttöliittymässä.

Esimerkkikäyttöliittymän ydinominaisuuksista luotiin valmiselementti (engl. prefab). Tämän valmiselementin uuteen kenttään lisäämällä saa käyttöönsä kaikki tarvittavat perustoiminnot uusiin tuoteprojekteihin yhden elementin viemisellä projektin hierarkiaan. Valmiselementti sisällytettiin käyttöliittymäpohjasta luotuun Unitypackage-pakettiin, jotta sen sisältämät sisäiset linkitykset eivät mitätöityisi, kun valmiselementti siirretään tuoteprojektiin.

Valmiselementtiin sisällytetyt toiminnallisuudet voidaan ottaa osaksi yrityksen vakioitua toimintaohjetta sen sijaan, että jokaisen tuotteen kohdalla käyttöliittymän elementit ja toiminnallisuudet dokumentoitaisiin erikseen. Tällöin tuotteiden kanssa, jotka käyttävät

valmiselementtiä, voidaan käyttöliittymiä koskevissa dokumenteissa viitata vakioidun toimintaohjeen osaan, jossa valmiselementin toiminnallisuudet ovat dokumentoituina.

4.5 Käyttöönotto Unitypackage-pakettien avulla

Alun perin käyttöliittymäpohja oli tarkoitus implementoida osaksi tuoteprojekteja Git-versionhallintajärjestelmän alimoduuleita käyttäen, mutta työn edetessä tämä osoittautui käytännössä mahdottomaksi. Ongelmaksi muodostuivat Unityn jokaisesta tiedostosta luomat metatiedostot ja tarkemmin ottaen metatiedostoihin tallennettavat GUID-tunnisteet (Globally Unique Identifier). Unity käyttää metatiedostojen GUID-tunnisteita komponenttien toisiinsa liittämiseen esimerkiksi valmiselementeissä ja kenttien hierarkiassa. GUID-tunnisteet ovat jokaiselle projektille ainutlaatuisia eli samalla tiedostolla on kahdessa eri projektissa toisistaan eroavat tunnisteet (64). Jos käyttöliittymäpohja lisittäisiin alimoduulina useampaan projektiin samanaikaisesti, yrittäisi jokainen projekti vuorollaan vaihtaa alimoduulissa sisällytettyjen komponenttien metatiedostojen tunnisteita, mikä rikkoisi komponenttien väliset sisäiset linkitykset. Tämä myös rikkoisi Unityn omaa ohjesääntöjä, joissa kaikki resurssit suositellaan lisääväksi editorin kautta. (71.)

Git-versionhallintajärjestelmän alimoduulien sijasta käyttöliittymäpohjan käyttöönotto päätettiin toteuttaa Unitypackage-pakettien avulla. Ne toimivat periaatteeltaan hyvin samantyyppisesti kuin perinteiset zip-paketit (72). Paketin pystyy luomaan suoraan Unityn editorin kautta. Tiedostot, jotka halutaan sisällyttää pakettiin, valitaan paketin luomisen yhteydessä. Unitypackage-pakettiin on mahdollista sisällyttää valmiselementtejä ilman, että niiden sisäiset linkitykset hajoavat siirron aikana tai sen jälkeen. Pakettiin sisällytettiin kaikki käyttöliittymäpohjaan toteutetut toiminnot ja lisätyt resurssit esimerkiksi käyttöliittymää lukuun ottamatta.

Toisessa projektissa paketin saa lisättyä osaksi projektia vaivattomasti ”Import Package” -toimintoa käyttämällä. Paketin valitsemisen jälkeen ohjelma avaa ikkunan, jossa näytetään mahdolliset ristiriidat olemassa olevien resurssien kanssa ja käyttäjä voi valita haluaako hän jättää jonkin resurssin tuomisen väliin. (72.)

Käyttöliittymäpohjan päivitettävyyys ei Unitypackage-paketteja hyödyntämällä ole yhtä korkealla tasolla, kuin mitä se olisi ollut alimoduuleita yhdessä versionhallinnan kanssa käyttämällä. Toiminnallisuuksien päivittäminen onnistuu keskitetysti käyttöliittymäpoh-

jassa, mutta muutokset eivät päivyty automaattisesti projekteihin, joissa käyttöliittymäpohja on käytössä. Unitypackage-paketteja käytettäessä paketin joutuu muutoksien tekemisen jälkeen luomaan uudestaan ja lisäämään erikseen jokaiseen tuoteprojektiin.

Tästä huolimatta käyttöliittymäpohja toi merkittävän parannuksen alkuperäiseen tilanteeseen, jossa jokainen toiminnallisuus jouduttiin yksitellen päivittämään kuhunkin projektiin erikseen. Muutosten yhteensopivuudesta ei tällä tavalla ollut takeita kun projektien toiminnallisuuksia kehitettiin eteenpäin täysin erillään toisistaan. Käyttöliittymäpohjan käyttöönoton jälkeen muutokset yhteisiin toiminnallisuuksiin tehdään käyttöliittymäpohjassa olevaan toteutukseen eikä tuoteprojektiin, jolloin taataan yhteensopivuus kaikkien muiden käyttöliittymäpohjaa käyttävien projektien kanssa. Positiivisena puolella Unitypackage-pakettien käytössä on lisäksi vikatilanteiden nopeampi havaitseminen kun uusi käyttöliittymäpohjaversio lisätään tuoteprojektiin käsin sen sijaan, että versionhallinta hoitaisi päivittämisen automaattisesti.

5 Työn tulokset ja käyttöliittymäpohjan tulevaisuus

Insinööriyössä saatiin toteutettua sille asetettujen tavoitteiden mukainen käyttöliittymäpohja, jonka avulla Disiorin tuotteiden kehitysprosessia saadaan tehostettua ja nopeutettua. Erilaisia käyttöliittymäelementtejä toteutettiin parisenkymmentä. Käyttöliittymäelementeistä saatiin hyvin eri kuvasuhteille ja resoluutioille mukautuvia, eikä työn aikana tullut vastaan tilannetta, jossa elementit olisivat mukautuessaan vääristyneet. Käyttöliittymäpohjaan toteutetut kuvakkeet ovat esitettyinä kuvassa 15.



Kuva 15. Käyttöliittymäpohjaan toteutetut käyttöliittymäelementit.

Käyttöliittymäpohjaan luotu esimerkkikäyttöliittymä toimi luotettavasti, eikä testauksen aikana ilmennyt ohjelmointivirheitä. Esimerkkikäyttöliittymän valikkopaneeliin pystyttiin ongelmitta lisäämään erikokoisia ja eri toiminnallisuuksia sisältäneitä paneeleita. Esimerkkikäyttöliittymään sisällytetyllä DICOM-tiedostoselaimella saatiin selattua DICOM-tiedostoja ja lähetettyä DICOM-kuvista luettu data MarchingCubes-algoritmilte. Tämän lisäksi DICOM-tiedostoselain varoitti käyttäjää mm. vikatilanteista, joissa käyttäjä yritti ladata vain yhden kuvan sisältäneen röntgenkuvan.

Käyttöliittymäpohjan päivitettävyydestä ei valitettavasti saatu niin helppoa ja yksinkertaista, kuin työn alussa toivottiin. Tapa, jolla Unity luo ja käsittelee kaikista resursseista luotuja metatietotiedostoja, pakotti luopumaan alkuperäisestä suunnitelmasta, jossa käyttöliittymäpohja olisi integroitu osaksi tuoteprojekteja Git-versionhallinnan alimoduuleita hyödyntämällä. Alimoduulien sijaan käyttöliittymän integrointi ja päivittäminen päätettiin toteuttaa Unitypackage-pakettien avulla. Unitypackage-paketin joutuu luomaan uudestaan aina, kun paketin sisältöön tehdään muutoksia, ja päivitetty paketti täytyy lisätä kuhunkin tuoteprojektiin manuaalisesti. Manuaalisten päivityksien hitauden kääntöpuolena on vikatilanteiden luotettavampi havaitseminen päivitysprosessin aikana.

Integrointi yrityksen tuoteympäristöön

Käyttöliittymäpohjaa lähdettiin integroimaan osaksi yrityksen tuoteympäristöä välittömästi sen valmistuttua. Työssä toteutettujen toimintojen lisäksi käyttöliittymäpohjaan liitettiin sen valmistumisen jälkeen muutamia muita komponentteja ja lisäosia. Nämä komponentit ja lisäosat käsittävät sellaisia toimintoja, jotka eivät liity käyttöliittymän toimintaan, minkä takia niitä ei sisällytetty insinööriyöhön. Kun kaikki tarvittavat komponentit saatiin liitettyä osaksi käyttöliittymäpohjaa, lähdettiin käyttöliittymäpohjaa integroimaan osaksi Disiorin nilkan ja jalan alueen murtumien analysointiohjelmää.

Integrointityö alkoi käyttöliittymän uudelleenluomisella käyttöliittymäpohjan valmisen päälle. Tämän jälkeen ohjelman toiminnot, jotka olivat myös käyttöliittymäpohjassa, päivitettiin ja vaihdettiin käyttöliittymäpohjan versioihin. Raportin kirjoittamishetkellä integrointityö on vielä toistaiseksi kesken.

Jatkokehitysmahdollisuudet

Trello-projektinhallintajärjestelmään luotiin oma taulunsa käyttöliittymäpohjaa varten. Kehittäjien on Trello-laudan avulla helppo pitää kirjaa jo toteutetuista toiminnoista ja kirjata ohjelmointivirheitä, toivomuksia ja uusia toimintoja, joita käyttöliittymäpohjaan lähdetään toteuttamaan. Toteutetut toiminnot ovat omassa listassaan, josta pystytään jäljittämään, kuka kehittäjä on toteuttanut minkäkin toiminnon ja milloin.

Toivomuslistalla on mm. ponnahdusikkunatyökaluluokan palauttamien takaisinkutsufunktioiden muuttaminen generisiksi funktioiksi ja DICOM-tiedostoselaimen esittämien vakioitujen DICOM-metadatakenttien vaihtaminen käyttäjän määriteltäviksi, jotta kukin käyttäjä voi tarkastella DICOM-tiedostoista niitä metadatakenttiä, joita haluaa. Suunnitelmassa on myös mukautetun painikeluokan toteuttaminen. Sen avulla yksittäiset painikkeet voitaisiin kytkeä päälle ja pois päältä. Unityn Button-valmisluokassa on tämän toiminnallisuuden toteuttava Interactable-tila, joka ei kuitenkaan tilanvaihdoksessa johda toivotunlaiseen visuaaliseen ulkoasuun. Tästä syystä mukautettu painikeluokka, jolle voidaan määrittää halutunlainen visuaalinen ulkoasu, on tarpeellinen.

6 Yhteenveto

Insinööriyössä toteutettiin käyttöliittymäpohja Disior Oy:n lääkintälaitteohjelmistoille Unity-pelimoottoria ja Affinity Designer -vektorigrafiikkaohjelmaa hyödyntäen. Tavoitteena oli luoda pohja, joka on mahdollista integroida osaksi Disiorin nykyisiä ja tulevia ohjelmistotuotteita. Työn tavoitteiksi asetettiin helposti muokattavien käyttöliittymäelementtien toteuttaminen, käyttöliittymän toiminnallisuuksien yhtenäistäminen ja käyttöliittymäpohjaprojektin toteuttaminen, jotta toteutettujen ominaisuuksien päivittäminen voitaisiin hoitaa keskitetysti yhdessä paikassa.

Käyttöliittymäpohjan toteutus onnistui kaikkien sille asetettujen vaatimusten mukaisesti, ja tuloksena saatiin käyttövalmis käyttöliittymäpohja. Käyttöliittymäpohjan toiminnallisuuksia testattiin jatkuvasti kehitystyön aikana pohjan toimivuuden varmistamiseksi. Testaamisen lisäksi toiminnallisuudet kommentoitiin huolellisesti, jotta käyttöliittymäpohjan käyttöönotto ja jatkokehittäminen olisi mahdollisimman helppoa myös yrityksen muille sovelluskehittäjille. Käyttöliittymäpohjan päivittämisestä ei saatu aivan yhtä tehokasta ja helppoa, kuin työn alussa toivottiin, mutta alkuperäiseen tilanteeseen saatiin silti merkittävä parannus.

Insinööriyön valmistumisen jälkeen käyttöliittymäpohja otettiin osaksi yrityksen kehitysympäristöä ja käyttöliittymäpohjaa alettiin integroida osaksi Disiorin sovellustuotteita. Käyttöliittymäpohjan tarpeet muuttuvat jatkuvasti, kun uusia sovellustuotteita aletaan kehittää, ja tarkoituksena onkin, että käyttöliittymäpohjan olemassa olevia toiminnallisuuksia päivitetään ja parannellaan ja uusia toiminnallisuuksia lisätään aina mahdollisuuksien tullen.

Lähteet

- 1 Medical devices. Verkkoaineisto. Maailman terveysjärjestö WHO. <https://www.who.int/medical_devices/full_definition/en/>. Luettu 17.4.2019.
- 2 Porche, Demetrius J. 1998. Condom effectiveness. Journal of the Association of Nurses in AIDS Care Volume 9 Issue 3, s. 91–94.
- 3 Medical devices. Verkkoaineisto. European Commission. <https://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/medical-devices_en>. Luettu 17.4.2019.
- 4 Software as a Medical Device (SaMD). 2019. Verkkoaineisto. Food and Drug Administration. <<https://www.fda.gov/medicaldevices/digitalhealth/softwareasamedicaldevice/default.htm>>. Päivitetty 2.4.2019. Luettu 17.4.2019.
- 5 Bernik, Ksenia. 2017. Medical Software Types and Examples. Verkkoaineisto. Intersog Inc. <<https://ehealth.intersog.com/blog/medical-software-types-and-examples>>. 19.7.2017. Luettu 18.4.2019.
- 6 In-Depth Guide to Key Types of Healthcare Software. 2019. Verkkoaineisto. RubyGarage. <<https://rubygarage.org/blog/types-of-healthcare-software>>. 3.1.2019. Luettu 18.4.2019.
- 7 Omakanta - Kanta.fi. Verkkoaineisto. Kansaneläkelaitos. <<https://www.kanta.fi/omakanta>>. Luettu 19.4.2019.
- 8 Rouse, Margaret. 2016. What is PACS (picture archiving and communication system). Verkkoaineisto. SearchHealthIT.com. <<https://searchhealthit.techtarget.com/definition/picture-archiving-and-communication-system-PACS>>. Päivitetty 7/2018. Luettu 18.4.2019.
- 9 Rouse, Margaret. 2011. What is DICOM (Digital Imaging and Communications in Medicine). Verkkoaineisto. SearchHealthIT.com. <<https://searchhealthit.techtarget.com/definition/DICOM-Digital-Imaging-and-Communications-in-Medicine>>. Päivitetty 6/2018. Luettu 19.4.2019.
- 10 What is DICOM? Verkkoaineisto. NeoLogica. <<https://www.neologica.it/eng/Tutorial/TutorialDICOM>>. Luettu 19.4.2019.
- 11 Overview. Verkkoaineisto. National Electrical Manufacturers Association. <<https://www.dicomstandard.org/about/>>. Luettu 19.4.2019.
- 12 Key concepts. Verkkoaineisto. National Electrical Manufacturers Association. <<https://www.dicomstandard.org/concepts/>>. Luettu 19.4.2019.

- 13 Rouse, Margaret. 2015. HL7 (Health Level Seven International). Verkkoaineisto. SearchHealthIT.com. <<https://searchhealthit.techtarget.com/definition/Health-Level-7-International-HL7>>. Päivitetty 7/2015. Luettu 19.4.2019.
- 14 Health Level 7. Verkkoaineisto. Diatom Enterprises. <<https://www.diatomenterprises.com/technologies/health-level-7/>>. Luettu 19.4.2019.
- 15 Health technology. 2015. Verkkoaineisto. Valvira. <<https://www.valvira.fi/web/en/healthcare/health-technology>>. Päivitetty 3.5.2015. Luettu 19.4.2019.
- 16 What's changed compared to the MDD. Verkkoaineisto. EU MDR. <<http://eumdr.com/whats-changed/>>. Luettu 20.4.2019.
- 17 EU MDR - Regulation (EU) 2017/745. Verkkoaineisto. EU MDR. <<http://eumdr.com/>>. Luettu 20.4.2019.
- 18 Medical Device Classification in Europe. Verkkoaineisto. Emergo by UL. <<https://www.emergobyul.com/services/europe/european-medical-device-classification>>. Luettu 20.4.2019.
- 19 Åman, Anu. 2017. Itsenäisen lääkintälaitteohjelmiston CE-merkintä. Insinööriyö. Oulun ammattikorkeakoulu. Theseus-tietokanta.
- 20 Step 4: Complete the clinical evaluation. Verkkoaineisto. EU MDR. <<http://eumdr.com/step-4/>>. Luettu 20.4.2019.
- 21 Vaatimustenmukaisuuden arviointi. 2019. Verkkoaineisto. Valvira. <https://www.valvira.fi/terveydenhuolto/terveysteknologia/tuotteen_markkinoille_s_aattami-nen/terveydenhuollon_laitteet_ja_tarvikkeet/vaatimustenmukaisuuden_arviointi>. Päivitetty 25.7.2019. Luettu 20.4.2019.
- 22 Risk Management process. Verkkoaineisto. EU MDR. <<http://eumdr.com/risk-management-process/>>. Luettu 20.4.2019.
- 23 Quality Management System. Verkkoaineisto. EU MDR. <<http://eumdr.com/quality-management-system/>>. Luettu 20.4.2018.
- 24 Borad, Anand. 2018. An Overview of FDA Regulations for Medical Devices. Verkkoaineisto. <<https://www.einfochips.com/blog/an-overview-of-fda-regulations-for-medical-devices/>>. 22.2.2018. Luettu 20.4.2019.

- 25 Conley, Dan. 2015. Two Paths for Medical Device Approval: FDA vs. CE. HealthManagement.org, Volume 15 - Issue 2, 2015. Verkkoaineisto. <<https://healthmanagement.org/c/healthmanagement/issuearticle/two-paths-for-medical-device-approval-fda-vs-ce>>. Luettu 20.4.2019.
- 26 U.S. Agents. 2017. Verkkoaineisto. Food and Drug Administration. <<https://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/HowtoMarketYourDevice/RegistrationandListing/ucm053196.htm>>. Päivitetty 23.12.2017. Luettu 20.4.2019.
- 27 UI kit. Verkkoaineisto. InVision. <<https://www.invisionapp.com/design-defined/ui-kit/>>. Luettu 6.4.2019.
- 28 Cousins, Carrie. 2013. Pros and Cons of Working With Design Kits. Verkkoaineisto. Design Shack. <<https://designshack.net/articles/graphics/pros-and-cons-of-working-with-design-kits/>>. 23.9.2013. Luettu 5.4.2019.
- 29 Dolan, Red. 2016. What is a UI template and why use one? Verkkoaineisto. A Medium Corporation. <<https://medium.com/claritydesignsystem/what-is-a-ui-template-and-why-use-one-e5d455ad64c5>>. 5.12.2016. Luettu 5.4.2019.
- 30 Durand, Claude. 2016. The Benefits of Using a UI Kit. Verkkoaineisto. Shopify. <<https://www.shopify.com/partners/blog/104547526-the-benefits-of-using-a-ui-kit>>. 26.4.2016. Luettu 6.4.2019.
- 31 Dmitriy, Bunin. 2018. 5 Reasons UI Kits are Awesomely Helpful! Verkkoaineisto. A Medium Corporation. <<https://medium.com/sketch-app-sources/5-reasons-ui-kits-are-awesomely-helpful-24ea2a1d6d71>>. 22.10.2018. Luettu 7.4.2019.
- 32 Frost, Brad. 2016. Atomic Design. E-Kirja. Shopify.
- 33 Bray, Richard. 2018. 10 reasons why you should be using Atomic Design. Verkkoaineisto. <<https://www.creativebloq.com/web-design/10-reasons-you-should-be-using-atomic-design-61620771>>. 26.3.2018. Luettu 7.4.2019.
- 34 Snäll, Johanna; Narjus-Sterba, M.; Toivari, M.; Wilkman, T. & Thorén, H. 2019. Does postoperative orbital volume predict postoperative globe malposition after blow-out fracture reconstruction? A 6-month clinical follow-up study. Oral and Maxillofacial Surgery March 2019, Volume 34, Issue 1, s. 27–34.
- 35 Tuliper, Adam. 2014. Unity : Developing Your First Game with Unity and C#. Verkkoaineisto. <<https://msdn.microsoft.com/en-us/magazine/dn759441.aspx>>. 9/2014. Luettu 10.4.2019.
- 36 The world's leading real-time creation platform. Verkkoaineisto. Unity Technologies. <<https://unity3d.com/unity>>. Luettu 9.4.2019.

- 37 Haas, John. 2014. A History of the Unity Game Engine. Interactive Qualifying Project (IQP). Worcester Polytechnic Institute.
- 38 Unity download archive. Verkkoaineisto. Unity Technologies. <<https://unity3d.com/get-unity/download/archive>>. Luettu 10.4.2019.
- 39 Game Project Elements of Unity 3D. Verkkoaineisto. Studytonight. <<https://www.studytonight.com/3d-game-engineering-with-unity/elements-of-unity3d>>. Luettu 11.4.2019.
- 40 Adding a C# Script to Unity Game Project. Verkkoaineisto. Studytonight. <<https://www.studytonight.com/game-development-in-2D/basics-of-unity-script>>. Luettu 11.4.2019.
- 41 Build quality learning apps, games and simulations. Verkkoaineisto. Unity Technologies. <<https://unity.com/solutions/edtech>>. Luettu 10.4.2019.
- 42 Roth, Emma. 2019. Why Affinity Designer Is The Best Adobe Illustrator Alternative. Verkkoaineisto. MakeUseOf. <<https://www.makeuseof.com/tag/affinity-designer-best-adobe-illustrator-alternative/>>. 14.2.2019. Luettu 21.4.2019.
- 43 Cousins, Carrie. 2018. Vector vs. Raster: What Do I Use? Verkkoaineisto. Design Shack. <<https://designshack.net/articles/layouts/vector-vs-raster-what-do-i-use/>>. 18.7.2018. Luettu 21.4.2019.
- 44 How Eyes Work. Verkkoaineisto. American Optometric Association. <<https://www.aoa.org/patients-and-public/resources-for-teachers/how-your-eyes-work>>. Luettu 13.4.2019.
- 45 C.7.6.3 Image Pixel Module. Verkkoaineisto. National Electrical Manufacturers Association. <http://dicom.nema.org/medical/dicom/2016e/output/chtml/part03/sect_C.7.6.3.html>. Luettu 13.4.2019.
- 46 Medical Imaging - Foot and Ankle CT. Verkkoaineisto. South Western Sydney Local Health District. <https://www.swslhd.health.nsw.gov.au/medicalImaging/serv_CT_Extremities_Foot.html>. Luettu 22.4.2019.
- 47 Tubik Studio. 2018. Light or Dark UI? Tips to Choose a Proper Color Scheme for User Interface. Verkkoaineisto. UX Planet. <<https://uxplanet.org/light-or-dark-ui-tips-to-choose-a-proper-color-scheme-for-user-interface-9a12004bb79e>>. 13.4.2018. Luettu 14.4.2019.
- 48 Sandu, Bogdan. 2016. Create a Better App by Finding the Right UI Color Scheme. Verkkoaineisto. Vandelay Design. <<https://www.vandelaydesign.com/better-app-ui-color-schemes/>>. 16.10.2016. Luettu 14.4.2019.

- 49 Nguyen, Kimberly. Medical Device UX Design process: 4 considerations for optimal User Interface usability. Verkkoaineisto. StarFish Product Engineering Inc. <<https://starfishmedical.com/blog/medical-device-ux-design-process-4-considerations-optimal-user-interface-usability/>>. Luettu 14.4.2019.
- 50 Babich, Nick. 2018. The Power of the Dark Side: Dark User Interfaces. Verkkoaineisto. <<https://www.shopify.com/partners/blog/dark-ui>>. 16.3.2018. Luettu 15.4.2019.
- 51 Why You Should Avoid Bright, Saturated Background Colors. 2018. Verkkoaineisto. UX Movement. <<https://uxmovement.com/content/why-you-should-avoid-bright-saturated-background-colors/>>. 22.5.2018. Luettu 14.4.2019.
- 52 Joutjärvi, Jutta. 2015. Skeuomorfismi ja flat design. Maisterin opinnäyte. Aaltoyliopisto. Aaltodoc-tietokanta.
- 53 Ng, Valerie & Tilliss, Jon. 2018. Balancing Aesthetics and Usability in Medical User Interface Design: 9 Key Trends. Verkkoaineisto. Emergo by UL. <<https://www.emergobyul.com/blog/2018/07/balancing-aesthetics-and-usability-medical-user-interface-design-9-key-trends>>. 8.7.2018. Luettu 12.4.2019.
- 54 Yalanska, Marina. Small Elements, Big Impact: Types and Functions of UI Icons. Verkkoaineisto. Tubik Studio. <<https://tubikstudio.com/small-elements-big-impact-types-and-functions-of-ui-icons/>>. Luettu 13.4.2019.
- 55 Mallette, Tracy. 2014. Risk Management in Medical Device Software Development. Verkkoaineisto. KMC Systems. <<http://www.kmcsystems.com/blog/bid/106735/don-t-risk-life-and-limb-medical-device-software-development-risk-management>>. 16.5.2014. Luettu 14.4.2019.
- 56 Graphical Symbols Booklet. 2013. E-kirja. Organisation Internationale de Normalisation (ISO).
- 57 Joining objects. Verkkoaineisto. Serif Europe. <<https://affinity.help/designer/en-US.lproj/index.html?page=pages/ObjectControl/join.html?title=Joining%20objects>>. Luettu 20.4.2019.
- 58 Ookii.Dialogs. Verkkoaineisto. Ookii.org. <<http://www.ookii.org/software/dialogs>>. Luettu 16.4.2019.
- 59 Manual: Vertical Layout Group. Verkkoaineisto. Unity Technologies. <<https://docs.unity3d.com/Manual/script-VerticalLayoutGroup.html>>. Luettu 17.4.2019.
- 60 Scripting API: Mesh. Verkkoaineisto. Unity Technologies. <<https://docs.unity3d.com/ScriptReference/Mesh.html>>. Luettu 17.4.2019.

- 61 Scripting API: Object.Destroy. Verkkoaineisto. Unity Technologies. <<https://docs.unity3d.com/ScriptReference/Object.Destroy.html>>. Luettu 17.4.2019.
- 62 Skeet, Jon. 2011. Implementing the Singleton Pattern in C#. Verkkoaineisto. C# in Depth. <<https://csharpindepth.com/articles/Singleton>>. Päivitetty 12.2.2011. Luettu 15.4.2019.
- 63 Paczkowski, Lukasz. 2016. Serialization, MonoBehaviour constructors and Unity 5.4. Verkkoaineisto. Unity Technologies. <<https://blogs.unity3d.com/2016/06/06/serialization-monobehaviour-constructors-and-unity-5-4/>>. 6.6.2016. Luettu 15.4.2019.
- 64 C.3 Standard Query/Retrieve Information Models. Verkkoaineisto. National Electrical Manufacturers Association. <http://dicom.nema.org/dicom/2013/output/chtml/part04/sect_C.3.html>. Luettu 21.4.2019.
- 65 Fellow Oak DICOM for .NET, .NET Core, Universal Windows, Android, iOS, Mono and Unity. 2018. Verkkoaineisto. GitHub, Inc. <<https://github.com/fo-dicom/fo-dicom>>. Luettu 16.4.2019.
- 66 Anderson, Ben. An Implementation of the Marching Cubes Algorithm. Verkkoaineisto. Carleton College. <<http://www.cs.carleton.edu>>. Luettu 18.4.2019.
- 67 Saari, Mikko. 2012. Valokuvauksen perusteita: ISO-herkkyys. Verkkoaineisto. Mikko Saari. <<https://www.mikkosaari.fi/iso-herkkyys/>>. 13.6.2012. Luettu 18.4.2019.
- 68 Oliver, Marilène. 2008. Flesh to Pixel, Flesh to Voxel, Flesh to XYZ. Verkkoaineisto. Marilène Oliver. <https://marileneoliver.com/_library/_uploaded/Chapter000Introduction.pdf>. 8/2008. Luettu 18.4.2019.
- 69 What are Hounsfield Units (HU)? Verkkoaineisto. Materialise. <<https://www.materialise.com/en/faq/what-are-hounsfield-units-hu>>. Luettu 18.4.2019.
- 70 Hamilton, Howard J. Isovox: A Brick-Octree Approach to Indirect Visualization. Verkkoaineisto. <https://www.researchgate.net/publication/228454597_Isovox_A_Brick-Octree_Approach_to_Indirect_Visualization>. Luettu 22.9.2019.
- 71 Smith, Forest. 2015. Managing Meta Files in Unity. Verkkoaineisto. ForrestTheWoods.com. 8.12.2015. Luettu 19.4.2019.
- 72 Manual: Asset packages. Verkkoaineisto. Unity Technologies. <<https://docs.unity3d.com/Manual/AssetPackages.html>>. Luettu 19.4.2019.