Pinja Similä

Annotointityökalun käytettävyyden kehittäminen

Tieto- ja viestintätekniikka
Insinööri (AMK)
Kevät 2023

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä(t): Pinja Similä

Työn nimi: Annotointityökalun käytettävyyden kehittäminen

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: annotointi, ohjelmistokehitys, käytettävyys, Qt, OpenCV

Opinnäytetyö toteutettiin toimeksiantona Raute Oyj:n Kajaanin yksikölle. Raute Oyj on maailmanlaajuisesti toimiva suomalainen yritys, joka toimittaa puutuoteteollisuudelle teknologiaratkaisuita ja palveluita.

Työn tarkoituksena oli jatkokehittää Qt:lla kehitetyn kuvadatan annotointiin tarkoitetun ohjelmiston käytettävyyttä ja käyttäjäystävällisyyttä, jotta sen hyödyntäminen tutkimusprosessissa olisi tehokkaampaa ja mukavampaa työntekijöille. Tavoitteina oli kehittää ohjelmiston suorituskykyä ja sen työkaluja sekä lisätä kokonaan uusia ominaisuuksia.

Työssä kartoitettiin ohjelmiston käyttäjien tarpeita kehityksen aikana, mikä ohjasi kehityksen suuntaa iteratiivisesti, eli vaatimuksia ei päätetty etukäteen, toisin kuin perinteisessä sovelluskehityksessä on tapana. Käyttäjien palautetta kerättiin kehityksen aikana ja ominaisuuksia kehitettiin siihen pohjautuen.

Ohjelmiston suorituskykyä kehitettiin vaihtamalla alkuperäisessä ohjelmistossa käytettyjä teknologioita tehokkaampiin ja tähän käyttötapaan sopivimpiin vaihtoehtoihin sekä myös refaktoroimalla koodipohjaa. Näillä muutoksilla ohjelman käyttämisestä saatiin yleisesti sulavampaa ja helpompaa, kun sen vastausaika lyhentyi. Piirtämisen toiminto jäi kuitenkin hieman pätkiväksi, vaikka senkin toiminta parantui merkittävästi.

Työkalujen toimintaa parannettiin ja kehitettiin runsaasti uusia erilaisia, jotka mahdollistavat erilaiset operaatiot annotointimerkintöjen tekemisessä ja niiden muokkaamisessa. Uusissa ominaisuuksissa käytettiin hyväksi OpenCV-konenäkökirjastoa. Käyttöliittymään tehtiin muutoksia ja ominaisuuksia, jotka selkeyttivät sen rakennetta ja helpottivat ohjelmiston käyttöä.

Työn tuloksena ohjelmiston käytettävyyttä saatiin kehitettyä merkittävästi sekä suorituskyvyn, että ominaisuuksien osalta. Ohjelmisto toimii sulavammin kuin aikaisemmin ja ominaisuuksia on monipuolisempi valikoima. Tehdyt suorituskykyparannukset tekivät merkintöjen piirtämisestä paljon sulavampaa, mutta siihen jäi vielä hieman kehittämisen varaa. Yleisesti ohjelmisto tuli käyttäjäystävällisempi ja annotointityöstä sillä helpompaa.

KAMK • University of Applied Sciences

Abstract

Author(s): Similä Pinja

Title of the Publication: Improving the Usability of the Annotation Tool

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: annotation, software development, usability, Qt, OpenCV

The thesis was carried out as an assignment for the Kajaani unit of Raute Oyj. Raute Oyj is a Finnish company that operates globally and provides technology solutions and services to the wood products industry.

The purpose of the work was to further develop the usability and user-friendliness of software developed with Qt for annotating image data, in order to make its use in the research process more efficient and comfortable for employees. The goals were to improve the performance of the software and its tools, as well as to add completely new features.

During the development, the needs of the software users were mapped, which guided the direction of the development iteratively, meaning that requirements were not decided in advance, unlike in traditional application development. User feedback was collected during development and features were developed based on it.

The performance of the software was improved by replacing less efficient technologies used in the original software with more efficient and suitable alternatives for this use case, as well as by refactoring the code base. These changes made the use of the program generally smoother and easier, as its response time shortened. However, the drawing function remained somewhat choppy, although its performance also improved significantly.

The functionality of the tools was improved, and many new ones were developed, allowing for various operations in making and editing annotation markings. In the new features, the OpenCV computer vision library was utilized. Changes and features were made to the user interface, which clarified its structure and made the software easier to use.

As a result of the work, the usability of the software was significantly improved in terms of both performance and features. The software runs smoother than before and has a more diverse selection of features. The performance improvements made the drawing of annotations much smoother, but there is still room for improvement in that area. Overall, the software is more user-friendly and annotation work is easier.

Alkusanat

Kiitokset Janne Piriselle työn ohjauksesta ja hyvästä palautteesta kehityksen aikana sekä loistavista ideoista ja vaatimuksista ominaisuuksien suhteen. Kiitokset KAMKin Markku Karppiselle opinnäytetyön ohjauksesta ja tsemppaamisesta.

Kiitokset myös Esko Toloselle ja Aleksi Nissiselle tuesta työn aikana ja kehityksessä sekä hyvistä vinkeistä. Viimeisenä, muttei vähäisimpänä, kiitokset Teemu Moilaselle opinnäytetyön alkuideasta.

Sisällys

Symboliluettelo

Widgetti: Käyttöliittymäelementti, joka omaa toiminnallisuuksia. Näillä elementeillä voidaan luoda ohjelmiston visuaalinen käyttöliittymä.

OpenCV: (Open Source Computer Vision Library) Avoimen lähdekoodin konenäkökirjasto, joka kokoaa sisäänsä erilaisia ohjelmistofunktioita.

1 Johdanto

Opinnäytetyön toimeksiantajana toimi Raute Oyj:n Kajaanin yksikkö. Raute Oyj on maailmanlaajuisesti toimiva suomalainen yritys, joka toimittaa puutuoteteollisuudelle erilaisia teknologiaratkaisuja ja palveluita. Yrityksen tuotevalikoima kattaa viilulle, vanerille ja LVL:lle koko tuotantoprosessin ja on niiden parissa maailmanlaajuinen markkinajohtaja. [1.]

Yksi Raute Oyj:n tuotteiden osa-alue ovat erilaiset analysaattorit, joiden avulla voidaan mitata monia erilaisia puumateriaalin ominaisuuksia. Yksi tapa analysoida viilua on kameralla saadun tiedon perusteella. Linjastolla kulkevasta viilusta otetaan kuvia, jotka järjestelmä analysoi. Tätä kuvatietoa myös kerätään talteen, myöhempää käyttöä varten.

Tähän kuvatietoon halutaan pystyä tekemään merkintöjä piirtämällä eli annotoimaan. Raute Kajaanilla on käytössä tätä varten kehitetty työkalu, annotaatiotyökalu. Ohjelmiston tämänhetkiset ominaisuudet ovat hyvin yksinkertaiset ja eivät taivu yrityksessä tarvittaviin vaatimuksiin. Kuvien merkitseminen on prosessina manuaalinen ja hidas. Tämän takia ohjelmistoa halutaan kehittää, jotta prosessi olisi mahdollisimman helppo työntekijälle, eikä työkalu hankaloittaisi sitä.

Tämän opinnäytetyön tavoitteena on jatkokehittää olemassa olevaa ohjelmistoa, johon Raute Kajaani on jo tehnyt kehitystyötä paremmalle ja käyttäjäystävällisemmälle tasolle, jotta sitä voitaisiin käyttää tehokkaasti tutkimus- ja kehitystyössä. Ohjelmistoon tarvitaan suorituskyvyn parannusta ja uusia ominaisuuksia helpottamaan prosessia. Työn näkökulmana on erityisesti käyttäjäystävällisyys ja -lähtöisyys, joihin on tavoitteena etsiä kehityksen avulla ratkaisuja.

2 Toiminnallinen opinnäytetyö

Toiminnallisen opinnäytetyön prosessissa luodaan tai kehitetään jokin toiminnallinen ratkaisu johonkin ongelmaan tämän vaatimalla ja valitulla tavalla. Prosessi eroaa tutkimuksellisesta prosessista siten, että se tuottaa lopputuloksena toiminnallista sisältöä uuden tiedon sijaan. [2, s. 9.]

Tutkimuksellisessa opinnäytetyössä valitaan tutkimusongelma. Toiminnallisessa työssä taas itse projekti on ikään kuin tutkimusongelma, johon haetaan ratkaisua [2, s. 85]. Yleensä toiminnallinen projekti tehdään toimeksiantajalle, eli se tulee oikeaan tarpeeseen ja tuottaa arvoa tälle taholle. Tämä kehitetty ratkaisu eli tuotos voi olla insinöörialalla esimerkiksi ohjelmisto, laite tai jatkokehitysidea.

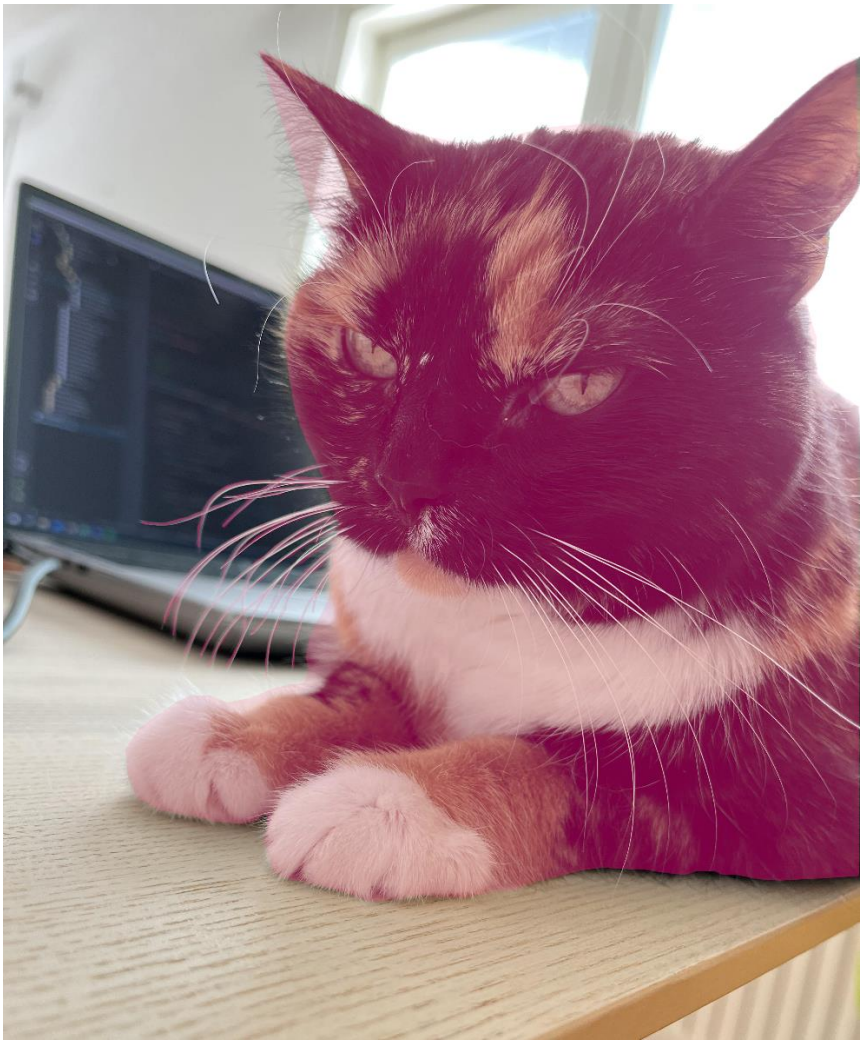
Toiminnalliseen työhön voi kuulua itse tuotoksen kehittämisen taustaksi tutkimuksellinen osuus sekä muunlaista selvitystyötä. Tutkimuksellisen haastattelun sijaan toiminnallisessa opinnäytetyössä voidaan konsultoida ihmisiä tiedonhankintaa varten. Tällaista konsultointia voivat olla muun muassa kokemusten, tiedon ja näkemysten kerääminen työn antavan yrityksen työntekijöiltä. Tietoa voidaan hyödyntää ja jalostaa projektissa ja sen vaatimusten ja kehityssuunnan määrittelyssä. [2, s. 64.] Tässä opinnäytetyössä on konsultoitu yrityksen työntekijöitä, jotta on saatu selvitettyä ohjelmiston käyttöön liittyvät ongelmat, jatkokehittämisen vaatimukset ja uudet ominaisuudet, joille olisi tarvetta. Työntekijöitä on osallistettu kehitykseen, mikä on kehityksen aikana määritellyt suuntaa ja tehtyjä valintoja iteratiivisesti.

Työssä on ikään kuin kaksi osuutta, jotka yhdistyvät: opinnäytetyöraportti ja tuotos. Opinnäytetyön raportissa esitellään tutkimusviestinnällisesti olennaiset asiat tuotoksesta, aiheen valinnasta kehitykseen ja lopulta lopputuloksiin. Työssä kuvataan projektin tavoitteet, mitä tuotokseen on tehty, jotta ratkaistaisiin nämä ongelmat ja miksi ongelmat on ratkaistu sillä tavalla kuin on. [2, s. 82]. Tuotos tehdään projektin vaatimalla tyylillä, eli tässä opinnäytetyössä ohjelmointikoodina ja sen kommentaationa. Tuotos on täten erillinen teos opinnäytetyöraportista, mutta on osa samaa kokonaisuutta. [2, s. 65.]

3 Annotointi

Annotoinnilla tarkoitetaan ihmisjohtoista prosessia, jossa aineistoon tai aineistosta tehdään lisätietoa tuovia tai selittäviä merkintöjä [3, s. 1]. Annotoinnilla halutaan luoda lisätietoa jonkin aineiston sisällöstä muokkaamatta alkuperäistä aineistoa. Tässä prosessissa luotua tietoa voidaan kutsua annotaatioksi. [4, s. 61.]

Annotoinnissa halutaan luoda tarkoituksellisia merkintöjä, jotka lisäävät tietoa tai selityksiä alkuperäiseen aineistoon, jota ei siitä jo löydy. Annotaatiot eivät myöskään ole täysin itsenäistä tietoa, vaan liittyvät aina aineistoon, joka on ollut pohja-aineistona, kuten kuvan 1 esimerkissä, jossa on annotoitu kissa pinkillä värillä. Prosessi lisää tietoa alkuperäiseen aineistoon ja tuo lisäarvoa: kuvasta voidaan nyt erottaa kissa muusta ympäristöstä. [3.]



Kuva 1. Kissa annotoituna

Annotointi prosessina on sitä, että halutaan tehdä merkintöjä aineistoon joko ihmisvoimalla tai vaihtoehtoisin menetelmin, vaikkapa tekoälyllä tai konenäöllä. Prosessin kulku voi vaihdella laajasti eri käyttökohteiden mukaan. [3.]

Samankaltaisesta sisällöstä käytetään myös termiä metatieto. Metatieto on yksinkertaisesti määriteltynä tietoa aineistosta [5]. Tietokoneiden kontekstissa puhutaan usein myös metadatasta. Termeinä nämä ovat rajaavampia kuin annotaatio, jonka merkitykseen sisältyy se, että annotaatio on ikään kuin sisältöä itsessään eli sisältää lisätietoa [4, s. 81]. Se täyttää myös metadatan määritelmän [3].

Annotaatioita tehdään erilaisiin tarkoituksiin. Teknologiateollisuudessa tällaisia tarkoituksia voivat olla tekoälyn tai konenäön kehittäminen, internetsivun saavutettavuuden parantaminen tai tiedonetsintä ja hakukoneet.

Tekoälyn kehittämiseen ihmisen tekemiä annotaatioita käytetään esimerkkiaineistona tekoälylle käsiteltäväksi koneoppimista varten, kuten esimerkiksi silmänpohjakuviin merkittyjä merkkejä silmäsairaudesta [6, s. 1]. Internetsivun saavutettavuutta voidaan parantaa vaikkapa kuviin liittyvillä tekstimuotoisilla annotaatioilla, joita näkövammaisten näytönlukuohjelmat ymmärtävät ja siten sivu on heille tämän ansiosta saavutettavampi [7]. Tiedonetsintää ja hakukoneita varten annotaatiot auttavat sisällön indeksoinnissa sekä tehostavat tiedonetsintää, aivan kuten kirjastojen hakuluettelot ennen kuin oli kehitetty digitaalisia tai automatisoituja hakujärjestelmiä.

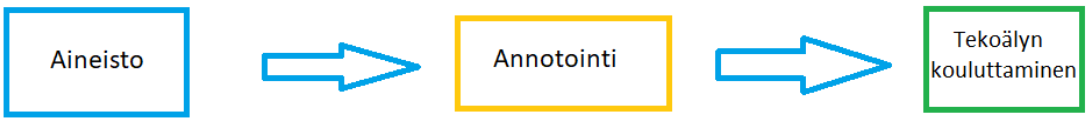
Annotointia voidaan toteuttaa manuaalisena, puolimanuaalisena tai automaattisena. Käytetyt tavat ja työkalut vaihtelevat prosessin tavan mukaan laajasti.

Manuaalinen annotaatio on ihmisen tekemää työtä, jossa merkitään dataan, eli tämän työn tapauksessa kuvatiedostoihin, halutut eri erotellut ominaisuudet. Prosessi voi olla hyvinkin työläs, sillä annotoitavat aineistokokonaisuudet voivat olla isoja ja annotointi voi viedä paljon aikaa. Tämän takia on yritetty kehittää muita vaihtoehtoisia tapoja täysin manuaalisen työn rinnalle tai sen korvaamiseksi.

Puolimanuaalinen annotaatio taas voi olla esimerkiksi prosessi, jossa tietokonealgoritmille syötetään esimerkkiannotaatio. Tämän perusteella algoritmi käsittelee aineiston ja merkitsee sinne löytämänsä eri ominaisuudet. Tämä parhaassa tapauksessa vähentää ihmisen tekemän työn määrää. Ihminen tarkistaa ja korjaa koneen tekemät pohjamerkinnät tarkoiksi. [8, s. 90.]

Automaattinen kuva-annotaatio on seuraava askel kehityksessä puolimanuaalisen jälkeen. Automaattisessa annotaatiossa tietokonealgoritmi on kehitetty ja hienosäädetty tarkoitukseensa niin tarkaksi, että algoritmi osaa toimia tarvittavalla tarkkuudella itsenäisesti. [8, s. 90.]

Tässä projektissa kyse on manuaalisesta annotoinnista, eli kaikki merkinnät tehdään ihmisen toimesta. Tällä tavoin luotua materiaalia voidaan käyttää esimerkiksi tekoälyn kouluttamiseen, kuten kuva 2 havainnollistaa.



Kuva 2. Annotoinnin osuus prosessissa

4 Ohjelmistojen käytettävyys

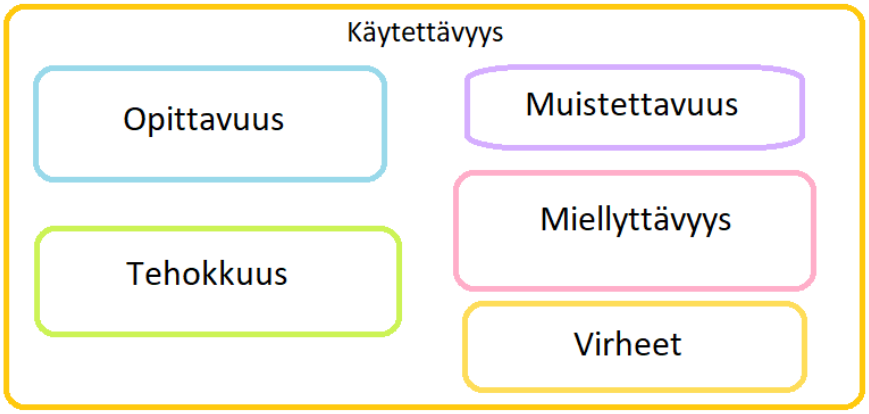
Ohjelmistokehityksen yksi olennaisimmista asioista on käytettävyys, johon tässä työssä on ollut tarkoituksena hakea ja toteuttaa ratkaisuja jatkokehityksen keinoin. Käytettävyys tai käyttäjäystävällisyys on ominaisuus, jolla tarkoitetaan ohjelmiston soveltuvuutta ja käyttökelpoisuutta johonkin sille suunniteltuun tehtävään [9].

Standardi ISO 9241-11:2018 määrittelee käytettävyyden käsitteen vapaasti suomennettuna seuraavasti: ”Kuinka vaikuttavasti, tehokkaasti ja tyydyttävästi järjestelmää, tuotetta tai palvelua voidaan käyttää määritettyjen käyttäjien toimesta tavoitteiden saavuttamiseen määritellyssä käyttökontekstissa.” [10.]

Jakob Nielsen taas määrittelee käytettävyyden näin: ”Käytettävyys on laatuominaisuus, joka arvioi, kuinka helppo käyttöliittymien käyttö on.” [11.]

4.1 Osa-alueet

Käytettävyys koostuu erilaisista laadullisista osa-alueista. Kuvassa 3 on esiteltynä Jakob Nielsenin viisi tärkeimmäksi määriteltyä: opittavuus, tehokkuus, muistettavuus, miellyttävyys ja virheet. [11.]



Kuva 3. Käytettävyyden osa-alueita [Kuvan tiedot: 11.]

Opittavuus on sitä, miten helppo ohjelmistoa on oppia käyttämään. Opittavuus koostuu erilaisista asioista, kuinka helppoa ohjelmistoa on käyttää ensimmäisen kerran, kuinka jyrkkä oppimiskäyrä on ja kuinka tehokkaasti on mahdollista oppia käyttämään ohjelmistoa. Ohjelmiston tekninen tehokkuus ja pienet vasteajat ovat tässäkin tärkeä vaikutin. Jos ohjelmisto toimii hitaasti, se vaikeuttaa opittavuutta. Toimintoja on vaikeampi oppia, jos kerkeää unohtamaan, mitä oli tekemässä, sillä ihmisen työmuisti on hyvin rajallinen. Mitä hitaammaksi käyttäminen koetaan, sitä vähemmän halutaan etsiä ja kokeilla uusia toimintoja. [12, 11, 13.]

Tehokkuus on määritelty siten, kuinka nopeasti käyttäjä pystyy suorittamaan tehtävän opittuaan käyttämään ohjelmistoa, eli kuinka hyvin se on suunniteltu tehtäväänsä. Ammattikäyttöön suunnitelluilla ohjelmilla voi olla huonompi opittavuus, mutta korkeampi tehokkuus. Tällaisesta esimerkki ovat sovellukset, joita käytetään monimutkaisilla komennoilla, jotka kuitenkin nopeuttavat työtä. Tehokkuuteen vaikuttaa myös tekninen suorituskyky, joka vaikuttaa operaatioiden suoritusnopeuteen, eli voi hidastaa työn tehokkuutta. Jos ominaisuudet aiheuttavat pitkää odottamista, käyttäjä voi vältellä niiden käyttämistä. [14 s. 30, s. 41–42, 11, 13.]

Muistettavuuden osa-alue mittaa sitä, kuinka nopeasti käyttäjä pystyy palaamaan ohjelmiston pariin, kun ei ole käyttänyt sitä hetkeen. Käyttämiseen pitäisi pystyä palaamaan ilman, että joutuu opettelemaan asioita uudelleen. Muistettavuuteen vaikuttavat käyttöliittymä ja sen selkeys. Siinä voidaan pitää yleisimmin käytettyjä työkaluja esillä, ja niillä helposti muistettavat nimet tai ikonit. Käyttäjät eivät nimittäin välttämättä edes muista ulkoa käyttöliittymän sisältöä, mutta osaavat käyttää sitä, kun pääsevät sen äärelle. [11, 14 s. 31–32.]

Virheillä tarkoitetaan sitä, kuinka paljon ja kuinka vakavia virheitä käytettäessä voi tehdä tai tapahtua ja kuinka helposti niistä voidaan toipua. Virheillä ei tarkoiteta vain käyttäjän tekemiä virheellisiä valintoja, vaan myös tilanteita, jotka hidastavat käyttöä. Virheiden määrään vaikuttavat esimerkiksi, kuinka opittava ohjelmisto on, miten selkeä käyttöliittymä on tai ohjaako ohjelmisto työskentelyä. [11, 14 s. 32–33.]

Miellyttävyys on yksinkertaisesti sitä, miten miellyttävää ohjelmiston käyttäminen yleisvaltaisesti on. Kuinka nopeasti ja tehokkaasti se toimii, vaikuttaa miellyttävyyteen välittömästi. Muita vaikuttavia asioita voivat olla esimerkiksi ulkonäkö ja kuinka korkea opittavuus on. [11, 15, 14 s. 33–34.]

4.2 Käytettävyyden edistäminen

Käytettävyyteen voidaan vaikuttaa erilaisin keinoin. Tärkein näistä on tiedon kerääminen käyttäjäkokemuksesta. Käyttäjällä on konkreettista tietoa siitä, mikä ohjelmistossa ei toimi, mikä käyttöä hankaloittaa ja minkälaisille työkaluille olisi tarvetta. Ohjelmistokehittäjällä ja ohjelmiston käyttäjillä voi myös olla erilainen kuva näistä asioista, odotettu käytettävyys ei korreloi koetun käytettävyyden kanssa [16, s. 15–16].

Iteratiivinen kehitys, jonka avulla tämä työ tehtiin, mahdollistaa käytettävyyden kehittämisen parhaalla tavalla. Laadittaessa suunnitelma ja vaatimukset etukäteen, ei ominaisuuksiin voi vaikuttaa yhtä tarkasti. On mahdotonta suunnitella heti kättelyssä käyttöliittymää, jossa ei ole ongelmia. [17.]

Palautetta voidaan kerätä erilaisista asioista, kuten aikaisemmin esitellyistä käytettävyyden eri osa-alueista, tai myös tarkemmin määritellyistä aiheista, kuten käyttöliittymästä tai työkaluista. Palautteen pohjalta voidaan esimerkiksi ohjelmiston visuaalista rakennetta selkeyttää. Tämä helpottaa toimintojen ymmärtämistä, niiden löytämistä ja kokonaisuuden hahmottamista. Käyttämisen oppiminen helpottuu, eikä kaikkea tarvitse muistaa ulkoa, jos käyttöliittymä on mahdollisimman intuitiivinen.

4.3 Kehittämisen hyödyt

Kehittäminen voi tuoda ohjelmiston käyttäjälle monia erilaisia hyötyjä sekä ihmisten että koneiden näkökulmasta. Palautteen ottaminen käyttöön ja kehittäminen sen perusteella auttaa kohdentamaan resurssit juuri sinne, missä niitä tarvitaan.

Yksi tärkeimmistä hyödyistä on kohonnut käyttäjätyytyväisyys. Kun ohjelmistoa on helppo käyttää, käyttäjät voivat olla todennäköisemmin mieltyneitä sen käyttämiseen. Käyttäjät myös jatkavat todennäköisemmin helppokäyttöisten ohjelmistojen käyttöä, mikä voi lisätä tuottavuutta ja tehokkuutta.

Toinen hyöty, jota halutaan saavuttaa käytettävyyden kehittämisestä, on käyttäjän tehokkuuden kasvu. Helppokäyttöisyyden kehittäminen voi lisätä tehokkuutta. Kun ohjelmistoa on helppo käyttää ja ymmärtää, käyttäjät voivat suorittaa tehtäviä nopeammin ja tehokkaammin. Tämä voi lisätä tehokkuutta, koska käyttäjät käyttävät vähemmän aikaa ohjelmiston ymmärtämisen kanssa

kamppailemiseen ja enemmän aikaa tavoitteidensa saavuttamiseen. Lisäksi vähentämällä tehtävien suorittamiseen tarvittavaa aikaa ja vaivaa, käyttäjät voivat olla tuottavampia työssään ja heillä on enemmän aikaa keskittyä muihin tärkeisiin tehtäviin.

Myös ohjelmiston suorituskykyä kehittämällä voidaan vaikuttaa käytettävyyteen. Täten myös tekninen tehokkuus voi nousta. Ihmiset eivät pidä odottamisesta, varsinkaan tietokoneiden parissa ja jos työskentelyn parissa turhautuu ohjelmistoon, ei sitä välttämättä haluta käyttää enää ollenkaan. Tässäkin tapauksessa kehittäminen vähentää muuta ajankäyttöä ja mahdollistaa keskittymisen varsinaisiin työtehtäviin. Ohjelmiston suorituskyvyn kehittäminen nostaa siten osaltaan myös käyttäjien tehokkuutta. [18, 19.]

Myös käyttöliittymän ulkonäön kehittäminen voi nostaa käyttäjätyytyväisyyttä. Esteettinen käyttöliittymän ulkonäkö voi jopa harhauttaa käyttäjää uskomaan, että ohjelmisto toimii paremmin kuin se oikeasti toimii. Käyttäjältä voi myös jäädä huomaamatta ongelmia ohjelmiston toiminnassa, koska esteettinen ulkonäkö vie huomion. Tätä ilmiötä kutsutaan nimellä ”aesthetic-usability effect”. [20.]

5 Piirtonäyttö

Piirtonäytöt ovat näyttöjä, jotka on tarkoitettu erilaisiin graafisiin töihin. Niiden perusajatus on se, että niihin on sisällytetty erilaisia kosketusominaisuuksia. Nämä ominaisuudet voivat toimia sormella ja/tai näytön omalla kynällä kosketettaessa. Näyttöjä ei tule sekoittaa tablettitietokoneisiin, vaan ne ovat kokonaan eri asia, vaikka niiden englanninkielinen nimi on ”drawing tablet”. Näytöt ovat vain näyttöjä ja tarvitsevat tietokoneen toimiakseen, vaikka niissä voikin olla älykkäitä ominaisuuksia.

Piirtonäyttöä käytetään yleisimmin siinä mukana tulevalla kynällä. Kun kynää käytetään parin millin päässä näytön pinnasta, se liikuttaa tietokoneen kursoria. Kynällä näyttöä painettaessa se klikkaa kyseistä kohtaa, kuten tavallinen tietokoneen hiiri vasemmalla painikkeella. Oikean painikkeen klikki voidaan tehdä käyttäen kynässä olevaa nappia.

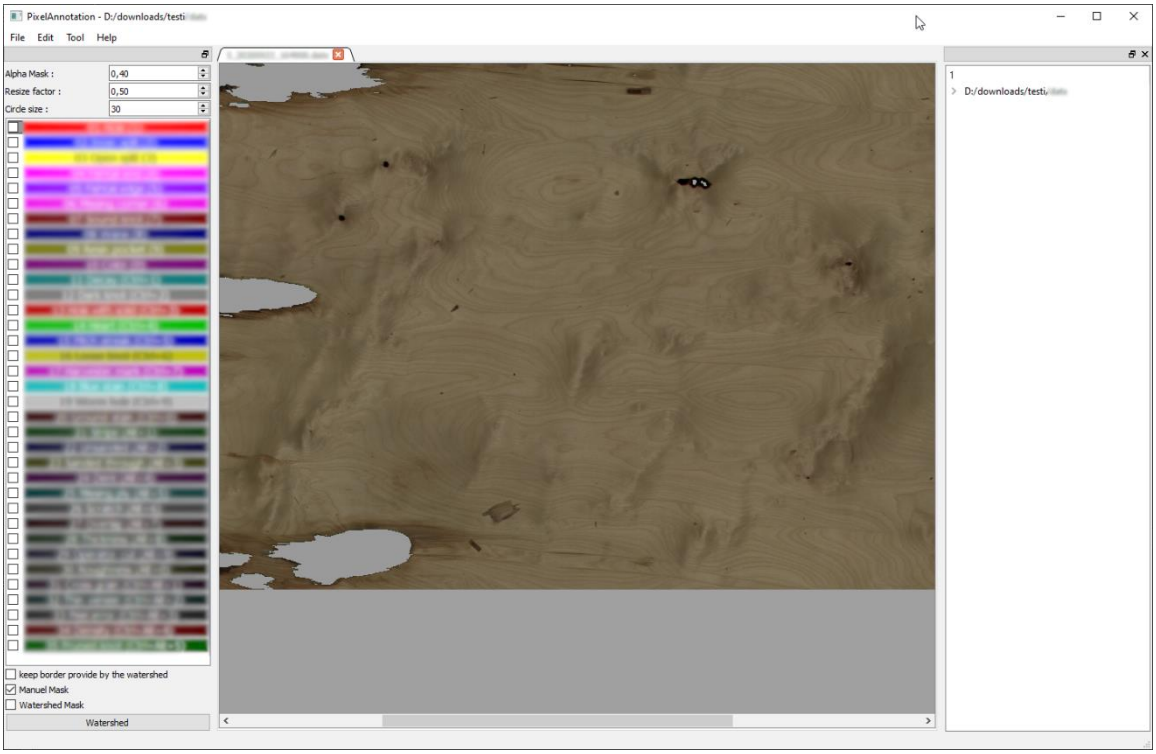
Yksi suosituimmista piirtonäyttövalmistajista on Wacom. Kuvassa 4 on esitettynä esimerkkilaite heiltä. Myös Raute Oyj:llä annotoinnin tekijällä on käytössä Wacomin valmistama piirtonäyttö. Kehityksessä haluttiin ottaa huomioon juuri siinä olevat ominaisuudet, kuten näppäimet ja makrojen määrittäminen. Makroihin voidaan määrittää pikatoimintoja ja -näppäinyhdistelmiä, joita voidaan sitten näytön nappia tai kynää käyttäen aktivoida.

Kuva 4. Wacom-piirtonäyttö [21]

6 Kehitettävä sovellus

Olemassa oleva ohjelmisto pohjautuu avoimen lähdekoodin ohjelmistoon, jonka on alkuperin kehittänyt Amaury Bréhéret Qt-ohjelmistonkehitystyökalulla [22]. Ohjelmistoon oltiin kuitenkin jo Raute Kajaanin toimesta tehty jatkokehitystyötä, eli täysin alkuperäinen ohjelmisto ei ollut tämän työn pohjana.

Kuvassa 5 esiintyy kuvankaappaus sovelluksesta tämän prosessin alussa. Käyttöliittymä koostui viidestä osiosta: värilistasta käyttöliittymän vasemmassa reunassa, asetusvalintaruuduista ylävasemmalla sekä alavasemmalla, yläpalkista, tiedostolistauksesta oikealla ja piirtoalueesta keskellä.



Kuva 5. Kehitettävän sovelluksen käyttöliittymänäkymä

Värilistassa on listattuna erilaiset merkitsemisvärit ja niiden tyyppien nimet. Listasta valitaan klikkaamalla, millä värillä halutaan merkitä. Värien vasemmalla puolella olevilla valintaruuduilla pystyy vaikuttamaan siihen, mitkä värit näkyvät annotoitavan kuvan päällä.

Asetusvalintaruuduissa on ylhäällä kolme erilaista asetusta, joilla kaikilla on numeraaliset arvot. Asetuksia ovat väritason läpinäkyvyys eli alpha-arvo, zoomauksen kerroin, joka ei oikein toiminut oikein, ja piirtokynän tai -ympyrän koko.

Tiedostolistaukseen listataan auki oleva yläkansio (joka on kuvassa suljettuna), tiedostot sen alla ja niiden sisältö. Ne sisältävät värikuvan ja tallennetut värikerrokset. Jokainen kerros tallennetaan omaan kuvatiedostoonsa mustavalkoisena, valkoinen edustaa merkintää ja musta väri merkitsemätöntä aluetta.

Yläpalkissa on neljä kohtaa, file eli tiedosto, edit eli muokkaa, tool eli työkalu ja help eli apua. Tiedosto-kohdan alta löytyvät esimerkiksi tiedostojen avaamiseen ja tallentamiseen liittyvät toiminnot. Muokkaa-kohdasta taas löytyy muokkaamiseen liittyviä toimintoja, kuten ”undo” ja ”redo”, eli peruuta ja tee uudelleen. Työkaluvalikon alta ei löytynyt oikeastaan mitään ja apukohdan alta löytyi lisätietoikkuna sovelluksesta.

Piirtoalue näyttää tiedostolistauksesta valitun kuvatiedoston ja sen päällä valittuna olevat mahdolliset värikerrokset. Merkitseminen tapahtuu klikkaamalla pohjassa ja piirtämällä kursorilla alueen päällä ja toiminnallisuus oli kovin hidas ja pätkivä, eli saadakseen katkeamattoman merkinnän, piti käyttäjän piirtää hitaasti. Erilaisia merkitsemiseen liittyviä toiminnallisuuksia piirtämällä merkitsemisen, pyyhkimisen ja värin vaihtamisen lisäksi ei oikeastaan ollut.

7 Vaaditut ominaisuudet

Tämän työn näkökulmana, opinnäytetyön laajuuden huomioon ottaen, oli kehittää erityisesti ohjelmiston olemassa olevia toiminnallisuuksia sekä lisätä uusia. Projektin tarkoituksena oli kehittää ja parantaa olemassa olevaa ohjelmistoa, jonka käyttötarkoituksena on kuvadatan annotointi koulutusdatan luomiseksi tekoälyn kehitystä varten.

Käyttäjällä on työssään käytössä piirtonäyttö, jolla ohjelmistoa käytetään. Piirtonäyttö on kuin kosketusnäyttö, jota käytetään sormen sijaan sen omalla kynällä, joka toimii ikään kuin hiiren tapaan ja tämä tuli ottaa huomioon erilaisissa työkaluissa ja käyttöliittymäelementeissä.

Projektin perusvaatimuksena oli ohjelmiston yleisen käytettävyyden parantaminen ja suorituskyvyn tehostaminen, sillä suorituskykyongelmat olivat keskeisin käyttämistä hankaloittava tekijä. Esimerkiksi itse annotointia kynällä ja zoomaamista tuli nopeuttaa, jotta molempien toiminta olisi sulavampaa ja pysyisi mahdollisimman hyvin käyttäjän syötteen perässä. Ominaisuuksien ideointi ja kehittäminen tapahtui ketterän sovelluskehityksen keinoin, eli niitä ei määritelty ennen kehittämisen aloittamista, vaan sen aikana.

Kehitettävän sovelluksen tapauksessa toiminnallisuuksista monet olivat melkein käyttökelvottomia tarkoitukseensa, eli kaipasivat kehitystä tai muutosta. Siitä löytyvät työkalut olivat hyvinkin yksinkertaiset ja erilaisiin tehtäviin tarvittiin uudenlaisia työkaluja helpottamaan niitä ja nopeuttamaan siten annotointiprosessia.

Käyttöliittymässä oli tavoitteina kehittää sen rakennetta, soveltuvuutta ja selkeyttä. Kehityksen aikana käyttäjäpalautetta kerättiin säännöllisesti henkilöiltä, jotka ohjelmistoa käyttivät. Palaute ohjasi kehityksen suuntaa sen aikana, sillä sovellus oli koko ajan käytössä ja kehitystapa oli ketterä, joten esimerkiksi työkalujen kaikkia vaatimuksia ei suunniteltu ennen kehityksen aloittamista.

7.1 Kuvanmuokkaustyökalut

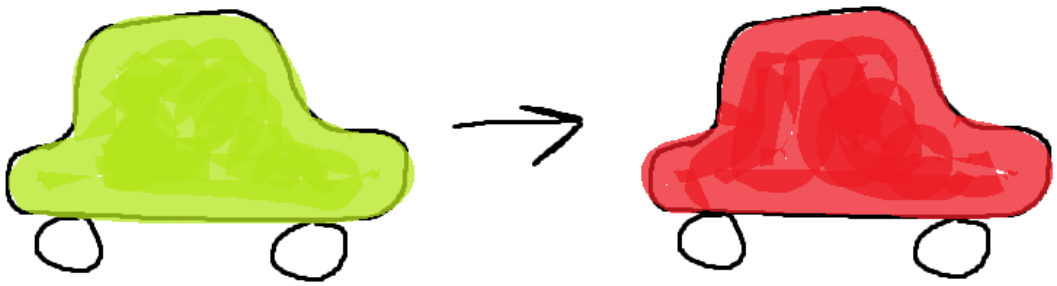
Annotointia varten tarvittiin ohjelmistoon vanhojen parantamisen lisäksi täysin uusia kuvanmuokkaustyökaluja ja ominaisuuksia kuvan siirtelyyn ja kohdistamiseen, helpottamaan ja nopeuttamaan annotointiprosessia, jotta se sujuisi mahdollisimman tehokkaasti.

Tarvittiin täyttötyökalu, josta olisi kaksi erilaista versiota. Ensimmäisellä käyttäjä pystyisi täyttämään suljetun alueen kerralla klikkaamalla täytettävää kohtaa, ja toisella taas kaikki merkitystä kuvasta löytyvät suljetut alueet. Täyttötyökalut eivät siis toiminnassaan huomioisi annotoitavaa värikuvaa, vaan annotoinnin.

Kuvan siivoamista helpottamaan ideoitiin pyyhkimistyökalu, jolla käyttäjä voisi painiketta painamalla pyyhkiä kerralla automaattisesti pois kaikki sellaiset annotointimerkinnät, jotka on tehty kohtiin, missä ei itse viilua ole, eli käytännössä taustavärin kohdalta.

Isojen alueiden merkitsemiseen haluttiin neliötäyttötyökalu, jolla käyttäjä pystyisi vain kursorilla painamalla ja vetämällä halutun neliön nurkasta nurkkaan merkitsemään isoja alueita kerralla. Työkalu täyttäisi luodun neliön käytössä olevalla värillä.

Työskentelyn nopeuttamiseen ja virheiden korjaamiseen tarvittiin tyypinvaihtotyökalu, joka toimisi käyttäjän syötteellä siten, että käyttäjä pystyisi vaihtamaan haluamansa jo tehdyn merkinnän tyypin/värin, kuten kuvassa 6 on havainnollistettu.



Kuva 6. Annotointimerkinnän tyypin vaihtaminen

7.2 Muut työkalut

Muita lopullisia tarvittavia ns. ”Quality of life”, eli prosessia helpottavia työkaluja olivat erityisesti piirtonäytölle sopivaksi suunniteltuja ominaisuuksia. Niitä olivat erityisesti kuvan siirtelyyn ja kohdistamiseen liittyvät työkalut.

Näitä olivat kursorilla vetämällä näkyvän kuvan kohdan siirtäminen, kursorilla vetämällä kuvan zoomaaminen, käyttöliittymään plus- ja miinusnapit zoomaamista varten sekä myös napit erilaisille kynän koon ja zoomauksen tason pika-asetuksille. Kaikki nämä käyttöliittymänapit piti suunnitella sopivan kookkaiksi, jotta niiden käyttäminen piirtonäytön kynällä olisi käyttäjäystävällistä.

Kaikkien työkalujen kohdalla vaatimuksena oli myös, että käyttäjän syötteen jälkeen itse muutokset tapahtuisivat mahdollisimman nopeasti, millisekunneissa. Tämä vaatimus huolehti siitä, että uudet ominaisuudet parantaisivat ohjelmiston käytettävyyttä, mikä työn perimmäisenä tavoitteena oli.

7.3 Käyttöliittymämuutokset

Piirtonäyttöystävällisyyden ja helppokäyttöisyyden sekä käyttöliittymän selkeyden takia ohjelmistoon tarvittiin myös kaikki nämä työkalut ja painikkeet kokoava työkalulaatikko, jonka käyttäjä pystyisi ikään kuin irrottamaan käyttöliittymästä ja siirtämään haluamaansa kohtaan näytöllä.

Käyttöliittymään haluttiin jokin toiminto, jolla voisi seurata tai merkitä, mitkä tiedostot ovat annotoitu ja mitkä ovat vielä työlistalla. Siitä suunniteltiin olemassa olevan tiedostolistauksen perusteella toinen lista sen alapuolelle, johon valmiit annotoidut tiedostot siirtyvät, kun käyttäjä merkitsee ne valmiiksi.

8 Käytetyt työkalut

8.1 Qt

Qt on Qt Groupin kehittämä ohjelmistonkehitystyökalu alustariippumattomaan ohjelmistokehitykseen. Qt Creator on Qt:n tarjoama ohjelmointiympäristö, johon on integroitu erilaiset Qt:n ohjelmistokehitystoiminnot, kuten debugger-työkalut ja käyttöliittymätyökalu Design. [23, 24.]

Qt tarjoaa laajan kirjaston ohjelmistokehitystä varten ja helpottaa erityisesti käyttöliittymän toteuttamista ja mahdollistaa erilaisten muutosten tekemisen siihen nopeasti. Kirjastoihin kuuluu erilaisia ”widgettejä”, joiden avulla voidaan rakentaa haluttu käyttöliittymä, ja niitä voidaan tarvittaessa muokata projektin tarpeisiin luomalla niistä johdettuja luokkia. Qt:n Design-työkalu tarjoaa laajat ja muokattavat ominaisuudet graafisen käyttöliittymän suunnitteluun ja toteuttamiseen widgettien avulla. Johdettua luokkaa käyttämällä voidaan toteuttaa uusia tai muokata widgetin olemassa olevia ominaisuuksia ja toimintaperiaatteita.

Qt:ssa on myös ominaisuus nimeltä ”signals and slots”. Ominaisuutta käytetään objektien väliseen kommunikaatioon. Tämä ominaisuus on se, joka erottaa Qt:n isoimmin muista kilpailevista alustoista. Signaalit ovat asioita, joita voidaan lähettää, kun jossain objektissa tai widgetissä tapahtuu jotakin; ja slotit ovat funktioita, jotka reagoivat niihin määritellyllä tavalla. Signaaleja voidaan yhdistää niin moneen eri slottiin kuin halutaan. Myöskään sloteilla ei ole mitään rajoitusta, kuinka monta signaalia yhteen voidaan yhdistää. Kuten muitakin widgettien ominaisuuksia, myös näitä voidaan muokata tai luoda kokonaan uusia. [25.]

Signaalien ja slottien lisäksi Qt tarjoaa ominaisuuden nimeltä ”events”, eli tapahtumat. Tapahtumat ovat objekteja, jotka muodostavat tapahtumajärjestelmän. Tätä järjestelmää käytetään käyttöliittymässä tapahtuvien tapahtumien hallintaan, ja niiden perusteella funktioiden ja toimintojen suorittamiseen. Nämä tapahtumat edustavat asioita, joista ohjelmiston tarvitsee tietää ja edustavat asioita, jotka tapahtuvat ohjelmiston sisällä tai ulkopuolisesta toiminnasta. Esimerkiksi hiiren klikkaus jonkin widgetin päällä on tällainen tapahtuma. Tapahtumista saadaan irti tarvittaessa erilaisia attribuutteja, kuten hiiren klikkauksesta sen sijainti koordinaatistossa. [26.]

Qt valittiin tämän työn toteutukseen, koska pohjalla oleva projekti oli rakennettu sillä. Erilaiseen alustaan siirtyminen olisi ollut tarpeetonta ja hankalaa, koska ohjelmisto oli Qt:n kirjastoista riippuvainen.

8.2 OpenCV

OpenCV (Open Source Computer Vision Library) on avoimen lähdekoodin konenäkökirjasto, joka kokoaa sisäänsä erilaisia ohjelmistofunktioita [28]. Se on suunnattu erityisesti erilaisille konenäköratkaisuille. Kirjasto on kirjoitettu C++-ohjelmointikielellä ja on alustariippumaton. OpenCV:n kehittämisen tavoitteena on ollut tarjota erilaisia tehokkaita reaaliajan kuvankäsittelyfunktioita ja korkean tason algoritmeja, kuten muotojen löytämistä. [27, s. 3.]

Kirjasto mahdollistaa muun muassa reaaliajassa videomateriaalista ihmiskasvojen löytämisen ja tunnistamisen, kuvasta tunnistettujen liikkuvien objektien seuraamisen, panoraamakuvien muodostamisen erillisistä kuvista automaattisesti. [28, 27 s. 13.]

Tämän työn ohjelmistossa kuitenkin kyse on manuaalisesta työstä, eikä automaattisesta objektien tunnistuksesta, mutta ohjelmiston toiminta perustuu siihen, että annotaation kuvadata pidetään erillisellä pikselikerroksella. OpenCV-kirjasto valittiin tämän työn toteuttamiseen työkaluksi, koska se mahdollistaa nopeat muutokset kuvadatalle, mikä on tärkeää, kun kyseessä on reaaliajassa toimiva ohjelmisto. Tällä tavoin saavutetaan mahdollisimman saumaton kokemus käyttäjälle, kun käyttöliittymässä näkyvä kuva saadaan päivitettyä hyvin pienellä viiveellä.

9 Sovelluksen käytettävyyden kehittäminen

Sovelluksen jatkokehitys toteutettiin Qt:n avulla C++-kielellä, sillä alkuperäinen pohjana oleva ohjelmistoprojekti oli toteutettu Qt:lla. Työn tarkoituksena oli parantaa ja kehittää olemassa olevan ohjelmiston käytettävyyttä. Sovelluksen kehityksessä keskityttiin ottamaan huomioon käyttäjäystävällisyys ja käyttäjän työväline, piirtonäyttö.

9.1 Suorituskyvyn parantaminen

Alkuperäisessä ohjelmistossa itse värikuvan päälle merkitseminen, käyttäjän liikkeiden välittäminen piirtofunktiolle ja kuvan esittäminen oli toteutettu QLabel-widgetistä periytettyä ImageCanvas-widgettiä käyttäen. Tämä aiheutti suorituskykyongelmia ohjelmiston toiminnassa.

QLabel on widgetti, joka on tarkoitettu tekstin tai kuvan näyttämiseen eli yksinkertaiseen grafiikkaan [29]. Tämä aiheutti suorituskykyongelmia ohjelmistoa käyttäessä, sillä QLabelia ei ole tarkoitettu muuttuvan grafiikan esittämiseen, vaikka se onkin mahdollista toteuttaa. Nämä ongelmat vaikeuttivat käyttäjän työtä.

Suorituskykyä parannettiin korvaamalla QLabel-pohjainen ImageCanvas-luokka omiin mukautettuihin kahteen luokkaan, ImageView- ja ImageViewItem-luokkaan, joiden avulla värikuvan näyttäminen ja sen päälle piirtäminen mahdollistettiin.

ImageView-luokka perustuu QGraphicsView-widgettiin. QGraphicsView on widgetti, joka on tarkoitettu alustaksi QGraphicsScene-widgetin sisällön näyttämiseen. QGraphicsScene taas on widgetti, joka mahdollistaa näkymän, jolla voidaan näyttää suuri määrä QGraphicsItem-objekteja, kuten esimerkiksi kuvia tai tekstiä.

ImageView-luokka erottuu alkuperäisestä QGraphicsViewistä siten, että sille on lisätty erilaisia funktioita ja muokattu tai korvattu olemassa olevia. Sille on toteutettu alkuperäisen ohjelmiston ominaisuuksista esimerkiksi värikuvan lataaminen ja näyttäminen, värikerrokset sisältävien ImageViewItem-objektien näyttäminen ja näiden zoomaaminen.

ImageViewItem-luokka perustuu QGraphicsItem-luokkaan, ja sitä käytetään maskikuvien hallintaan ja muokkaamiseen. Se eroaa alkuperäisestä pohjalla olevasta luokasta siten, että siihen on

lisätty paljon uutta toiminnallisuutta ja muokattu olemassa olevia perittyjä funktioita. Kaikki alkuperäisen ohjemiston annotoimiseen liittyvät toiminnallisuudet siirrettiin ja toteutettiin uudelleen ImageViewItem-luokkaan.

Alkuperäisessä ohjelmistossa muutoksien piirtäminen värikerrokselle oli toteutettu QPainter-luokan avulla, ja sen toiminta säilytettiin tämän työn versiossakin. QPainter on luokka, joka mahdollistaa grafiikan piirtämisen widgetteihin ja graafiseen käyttöliittymään. Siitä löytyy valmiita funktioita erilaisten muotojen piirtämiseen, kuten suorakulmioiden, ympyröiden ja viivojen.

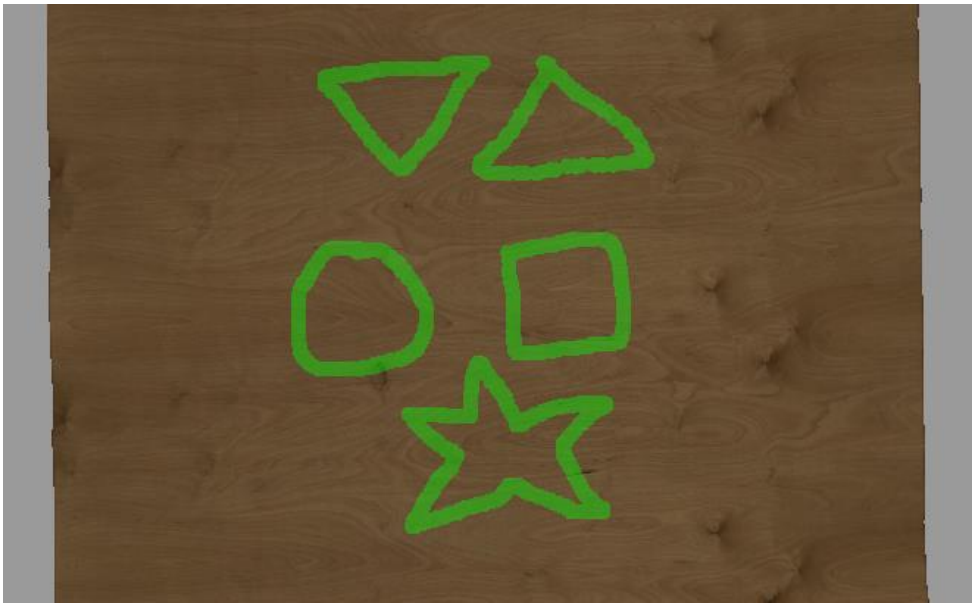
9.2 Kuvankäsittelytyökalut

Ohjelmistoon kehitettiin lopulta useita erilaisia kuvankäsittelytyökaluja, joilla pystytään tehokkaasti tekemään merkintöjä erilaisin menetelmin ja muokkaamaan olemassa olevia. Kaikki nämä työkalut ovat uusia lisättyjä ominaisuuksia. Muutokset tehtiin aikaisemmin luotuihin ImageView- ja ImageViewItem-luokkiin.

9.2.1 Täyttötyökalut

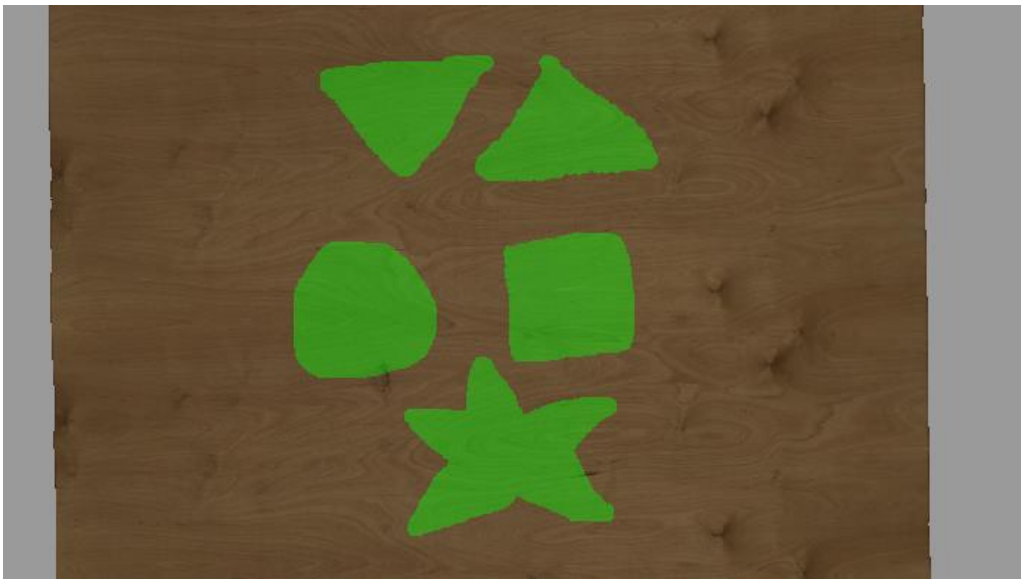
Täyttötyökalujen kehityksessä hyödynnettiin OpenCV-konenäkökirjastoa, jolla pystyttiin tekemään nopeita muutoksia värikerrokseen käyttäen lähdemateriaalina annotoitavaa kuvaa. Näin varmistettiin työkalujen käyttömukavuus. Työkaluja kehitettiin lopulta kolme erilaista: kaikki suljetut muodot kerralla täyttävä, valitun kohdan täyttävä ja valitun suorakulmion täyttävä.

Kaikki suljetut muodot täyttävällä työkalulla pystyy nimensä mukaan täyttämään kerralla kaikki suljetut muodot, joita valitulle värikerrokselle on merkitty. Kuvassa 7 on kuvattu esimerkkimuotoja ennen täyttämistä.



Kuva 7. Muodot ennen täyttämistä

Kuvassa 8 voidaan nähdä samat muodot täyttöominaisuuden käytön jälkeen. Ominaisuus suorittuu välittömästi sen aktivoinnin jälkeen, viiveettömästi ja tarkasti.



Kuva 8. Muodot täyttämisen jälkeen

Ominaisuudessa hyödynnettiin OpenCV-kirjaston muodonetsintäfunktioita. Kirjastosta löytyvällä ”findContours”-funktiolla etsitään mustavalkoisesta maskikuvasta suljettujen muotojen ääriviivat. Muodot tallentuvat vektoriin, josta jokainen käydään läpi ja niistä piirretään täytetyt versiot samaisesta kirjastosta löytyvällä ”drawContours”-funktiolla. Operaatiosta syntynyt matriisi

muunnetaan QPixmap-pikselikartaksi. Kuvan 4 muotojen täyttämistä varten tehty pikselikartta on esitettynä kuvassa 9.

Kuva 9. Pikselikartta muotojen täyttämistä varten

Kuvassa olevaan merkintään muutoksien tekemistä varten kehitettiin piirtofunktio, joka pystyy käyttämään pikselikarttaa ikään kuin siveltimenä ja piirtämään sen sisällön muotoisen merkinnän. Funktio ottaa parametreikseen värikerroksen tyypin, jolle halutaan piirtää ja pikselikartan, jonka sisältö halutaan piirtää.

Piirtofunktiota käyttäen aikaisemmassa operaatiossa syntynyttä pikselikarttaa käytetään muutoksien piirtämiseen värikerrokselle, ja tuloksena muodot täyttyvät.

Toinen täyttötyökalu toimii siten, että käyttäjä pystyy valitsemaan yksittäisen muodon klikkaamalla suljetun muodon keskeltä. Toiminto sen jälkeen täyttää tämän muodon välittömästi. Kuvissa 10 ja 11 on esitetty kuvat muodosta ennen ja jälkeen operaation.

Kuva 10. Suljettu muoto

Kuva 11. Työkalulla täytetty suljettu muoto

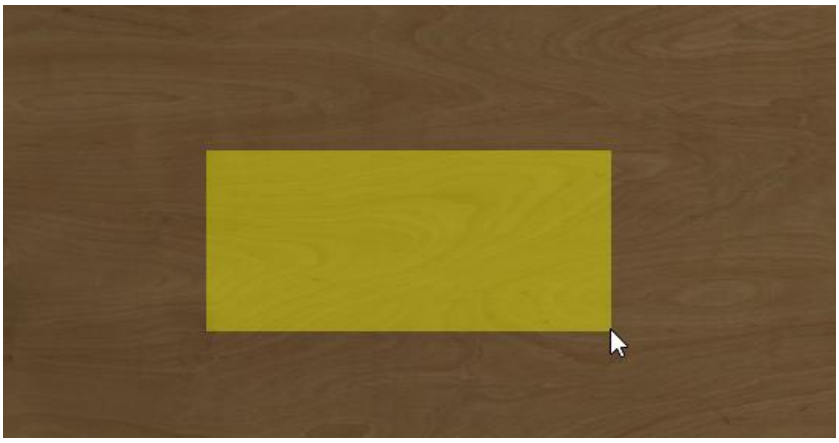
Toiminto toimii teknisesti siten, että työkalun ollessa käytössä ja käyttäjän klikatessa jotakin kohtaa, tuon kohdan koordinaatit tallennetaan muuttujaan. Olemassa olevasta maskikuvasta otetaan kopio ja OpenCV-kirjaston funktion avulla siinä oleva valittu muoto täytetään valkoisella ja korvataan alkuperäinen maskikuva tällä uudella.

Toiminto ei tarkista, onko sen ympärillä oikeasti mitään piirrettyä muotoa, eli sillä pystyy myös täyttämään koko kuvan, jos klikkaa jonkin muodon ulkopuolelta tai tyhjää kuvaa. Tämä on kuitenkin tarkoituksenmukaista työkalun ollessa sananmukaisesti täyttötyökalu.

Kolmannella täyttötoiminnolla käyttäjä pystyy täyttämään suorakulmion muotoisen alueen kerralla, ilman aikaisempaa olemassa olevaa merkintää. Työkalu toimii vetämällä kursorilla suorakulmion nurkasta nurkkaan, kuvassa 12 esitetyllä tavalla.

Kuva 12. Täytettävän alueen valitseminen

Kun käyttäjä päästää hiiren napista irti, toiminto täyttää valitun alueen käytössä olevalla värillä. Kuvassa 13 voidaan nähdä valittu alue täytön jälkeen täytettynä keltaisella värillä.



Kuva 13. Valittu alue täytön jälkeen

Teknisesti suorakulmiotäyttötyökalu on toteutettu siten, että kun alueen valitseminen aloitetaan, otetaan talteen tuon pisteen koordinaatit. Tässä on hyödynnetty Qt:n tarjoamaa signaaliominaisuutta, eli kun käyttäjä klikkaa, niin siitä välittyy signaali, jonka perusteella voidaan tehdä operaatioita. Kun käyttäjä päästää irti, siitäkin välittyy signaali, josta otetaan koordinaattipisteet talteen. Näiden koordinaattien välille suorakulmio piirretään värikerrokselle QPainter-luokan ”drawRect”-funktion avulla.

9.2.2 Hover/toggle-piirtäminen

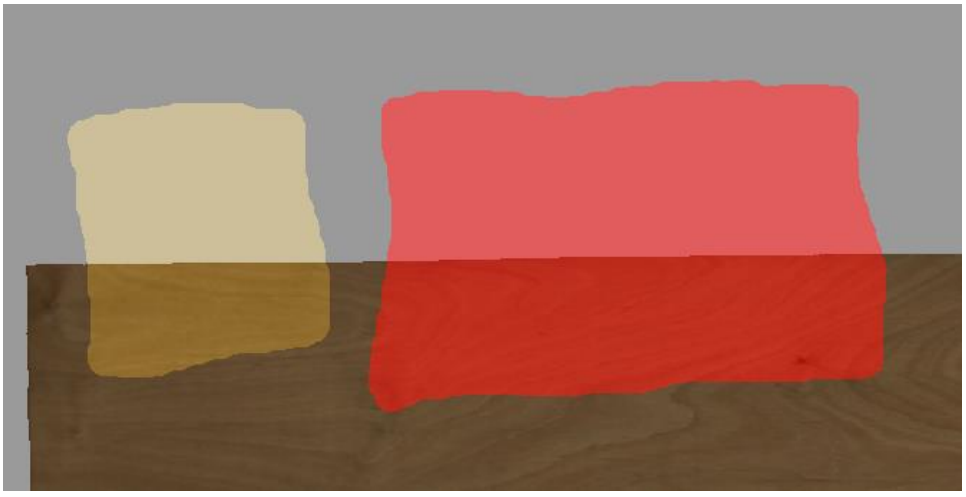
Kehityksen aikana selvisi toive työkalulle, jolla pystyisi merkitsemään yhtäjaksoisesti painamatta hiiren nappia koko aikaa, eli kerran klikkaamalla päälle- ja poislaitettavasta piirtotilasta. Tällainen toteutettiin käyttäen Qt:n tarjoamaa tapahtumajärjestelmää hyväksi. Sieltä löytyy ”hover”-tapahtuma eli leijumistapahtuma. Leijumista tässä kontekstissa on kursorin siirtäminen jonkin widgetin päällä klikkaamatta sitä.

Kun käyttäjä klikkaa piirtoalueen päällä kerran työkalun ollessa käytössä, piirtäminen aloitetaan. Tavallisen piirtämisen tapauksessa sitä jatkettaisiin vain, kun käyttäjä piirtää alueen päällä hiiren nappi pohjassa, mutta tässä työkalussa sitä jatketaan niin kauan, kun käyttäjä leijuttaa alueen päällä, kunnes hän klikkaa aluetta uudestaan lopettaakseen.

Jos käyttäjä siirtyy alueen päältä pois lopettamatta piirtämistä ensin, sitä jatketaan, kun hän palaa alueen päälle. Piirtämisen voi lopettaa myös vaihtamalla valittuna oleva työkalu johonkin toiseen, jolloin piirtämistä ei jatketa käyttäjän palatessa alueen päälle.

9.2.3 Siivoamistyökalu

Siivoamistyökalu toimii siten, että kun toiminto aktivoidaan napista, se pyyhkii merkinnät kohdista, missä ei ole värikuvaa eli annotoitavaa materiaalia. Sitä käytetään helpottamaan reunojen siivoamista, jolloin itse merkitsemisessä ei tarvitse olla niin tarkka. Työkalu nopeuttaa annotoijan työtä. Kuvassa 14 on esitetty kaksi muotoa piirrettynä annotoitavan objektin reunan yli.



Kuva 14. Kaksi muotoa piirrettynä viilukuvan yli

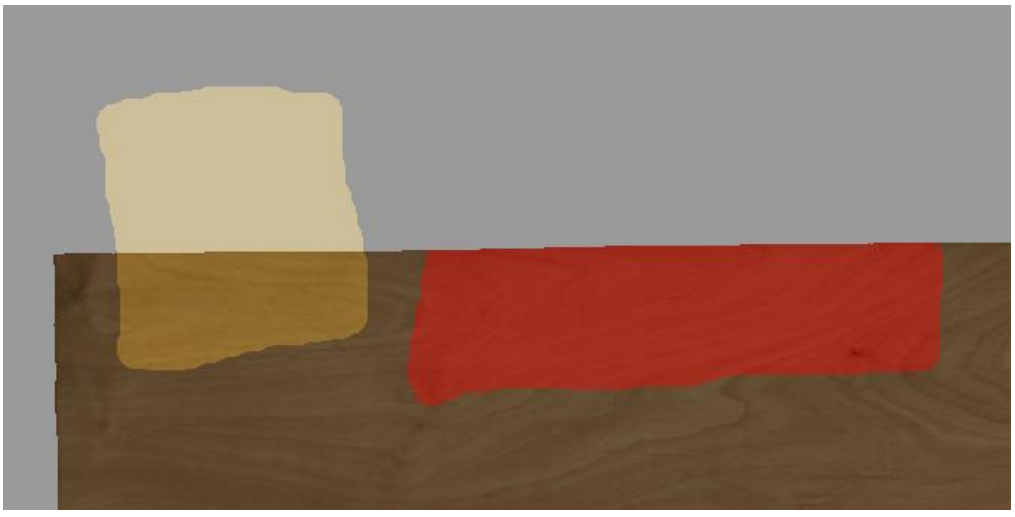
Työkalua varten ohjelmisto luo tiedoston avaamisen yhteydessä pikselikartan, joka on alpha-arvoltaan käänteinen alkuperäiseen värikuvaan verrattuna, eli värilliset pikselit ovat läpinäkyviä ja läpinäkyvät läpinäkymättömiä. Näin itse siivoamistyökalu toimii nopeammin, kun pikselikarttaa ei tarvitse luoda vasta työkalua käytettäessä.

Pikselikartta luodaan siten, että alkuperäisesti Qt:n QImage-muuttujaan ladattu värikuva muunnetaan OpenCV:n mat- eli matriisimuuttujaksi. Tämän jälkeen matriisin data muunnetaan värillisestä mustaharmaaksi ja vielä BGRA-muotoon, alfa- eli läpinäkyvyyskanavan omaavaksi.

Kuvan väriarvoja yksinkertaistetaan porrastamalla niitä OpenCv:n threshold- eli kynnysarvofunktiolla, jotta käsittely onnistuu konenäköfunktioilta paremmin. Sitten ohjelmisto etsii matriisista kaikki valkoiset pikselit, ja muuntaa niiden kaikki väriarvot (kaikki 4 kanavaa) nollaan. Prosessissa syntynyt matriisi muunnetaan ja talletetaan Qt:n QPixmap-muotoiseen muuttujaan käyttöä varten.

Itse pyyhkimistä varten kehitettiin funktio, joka käyttää parametreinään valittuna olevaa värikerrosta, johon sitä käytetään ja QPixmap-pikselikarttaa, jota halutaan käyttää pyyhkimiseen. Funktiossa käytetään QPainter-luokkaa muutoksien tekemiseen.

Ohjelmisto käyttää tätä pikselikarttaa pyyhkimisfunktion parametrina, joka sen perusteella piirtää muutoksen värikerrokseen, eli muuttaa olemassa olevissa merkinnöistä pikselit, jotka jäävät läpinäkyvän alle värittömiksi, eli pyyhkii ne. (Kuva 15)

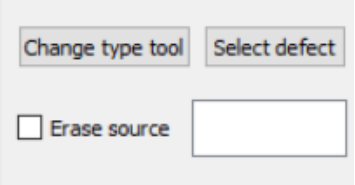


Kuva 15. Toinen muodoista pyyhitty

9.2.4 Merkinnän tyypinvaihto

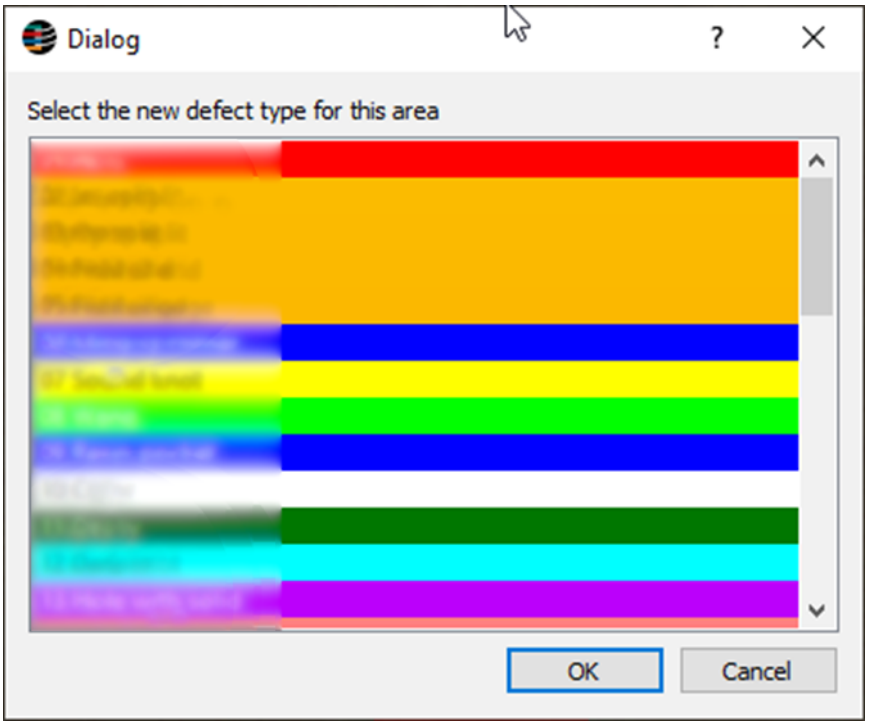
Työkalu jo tehtyjen merkintöjen tyypin, eli värin, vaihtamiseen käytti myös OpenCV:n ominaisuuksia. Työkalulle määritettiin kaksi asetusta: määränpää-värikerros ja siirretäänkö vai kopioidaanko valittu muoto määränpäähän, eli pyyhitäänkö muoto alkuperäisestä lähteestä.

Kuvassa 16 on esiteltynä tyypinvaihdon käyttöliittymä, jossa löytyy työkalun aktivointinappi, kohdevärin valintanappi, asetus lähteen poistamiselle kopioinnin jälkeen ja laatikko näyttämään, mikä kohdeväri on valittuna.



Kuva 16. Tyypinvaihdon käyttöliittymä työkalulaatikossa

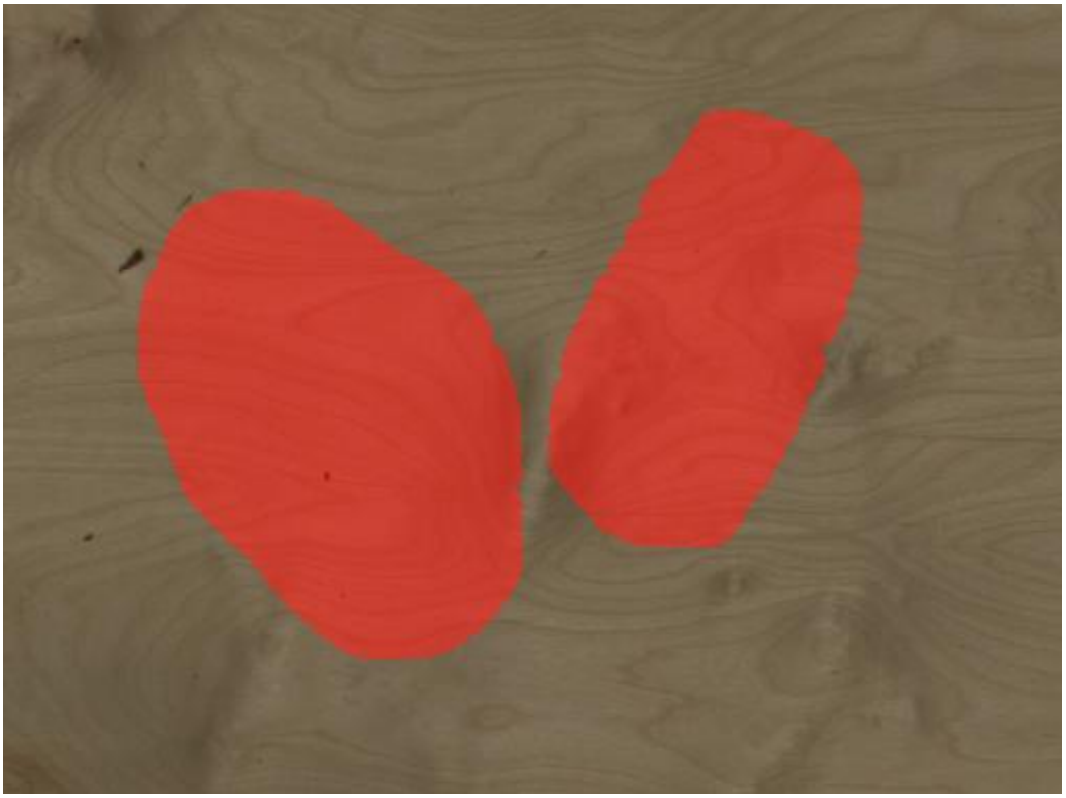
Kohdevärin valintanapista aukeaa dialogi-ikkuna, johon on listattu kaikki värivaihtoehdot ja josta haluttu väri valitaan. Nappia uudelleen painamalla dialogi aukeaa taas, ja valitun värin voi vaihtaa. Kuvassa 17 on esitetty kyseinen dialogi-ikkuna.



Kuva 17. Tyypinvaihdon dialogi-ikkuna

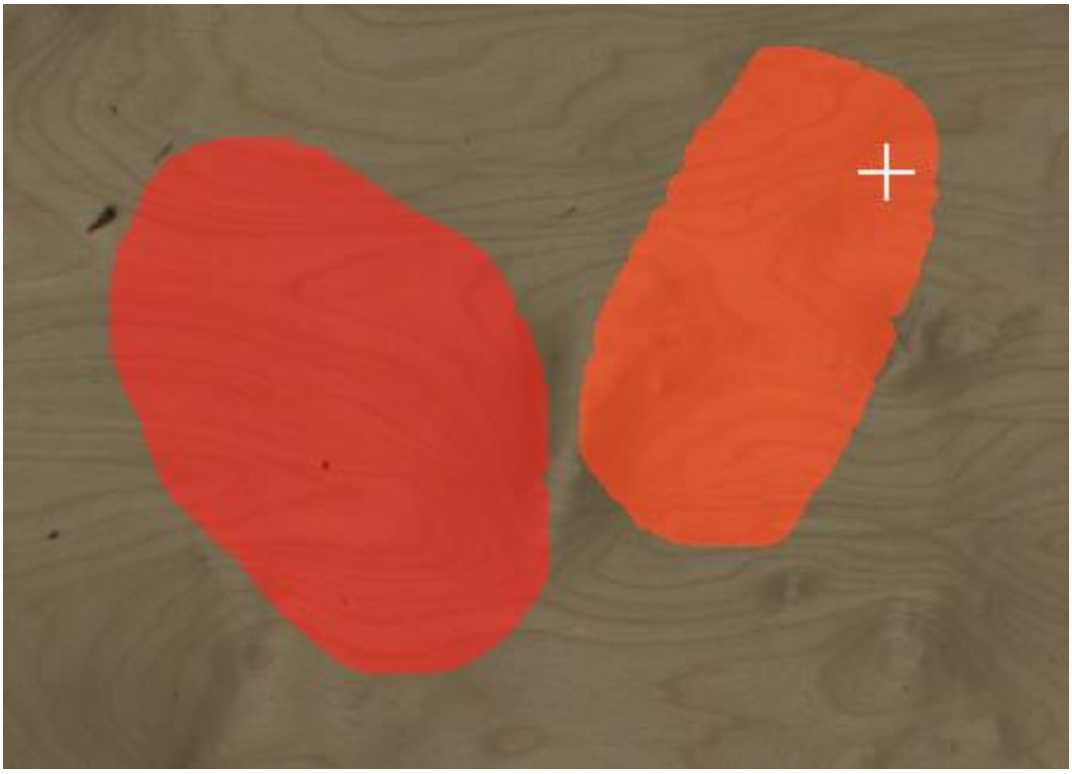
Kun työkalu on valittuna käytössä, käyttäjä voi klikata jo tehtyjä merkintöjä ja ohjelmisto muuttaa ne valituksi väriksi. Valittu väri ja työkalu pysyvät käytössä, kunnes käyttöön otetaan joku muu työkalu tai kunnes käyttäjä vaihtaa väriä, eli monta erillistä muotoa voidaan valita peräkkäin.

Kuvassa 18 on esitettynä kaksi muotoa, joiden väri on punainen.



Kuva 18. Kaksi samanväristä muotoa

Kuvassa 19 on esitetty, miltä samat muodot kuin kuvassa 18, näyttävät sen jälkeen, kun toinen niistä (oikeanpuoleinen) on kopioitu myös toisen erilaisen tyypin kerrokselle, jonka värinä on keltainen. Kaksi erityyppistä merkintää näkyy kuvassa päällekkäin oikeanpuolimmaisessa muodossa tehden sen väristä oranssin.



Kuva 19. Tyypin vaihtaminen kopioimalla

Teknisesti työkalu toimii siten, että kun käyttäjä klikkaa muotoa, prosessi taustalla alkaa kopioimalla käytössä olevan värikerroksen data OpenCv:n matriisimuuttujaan.

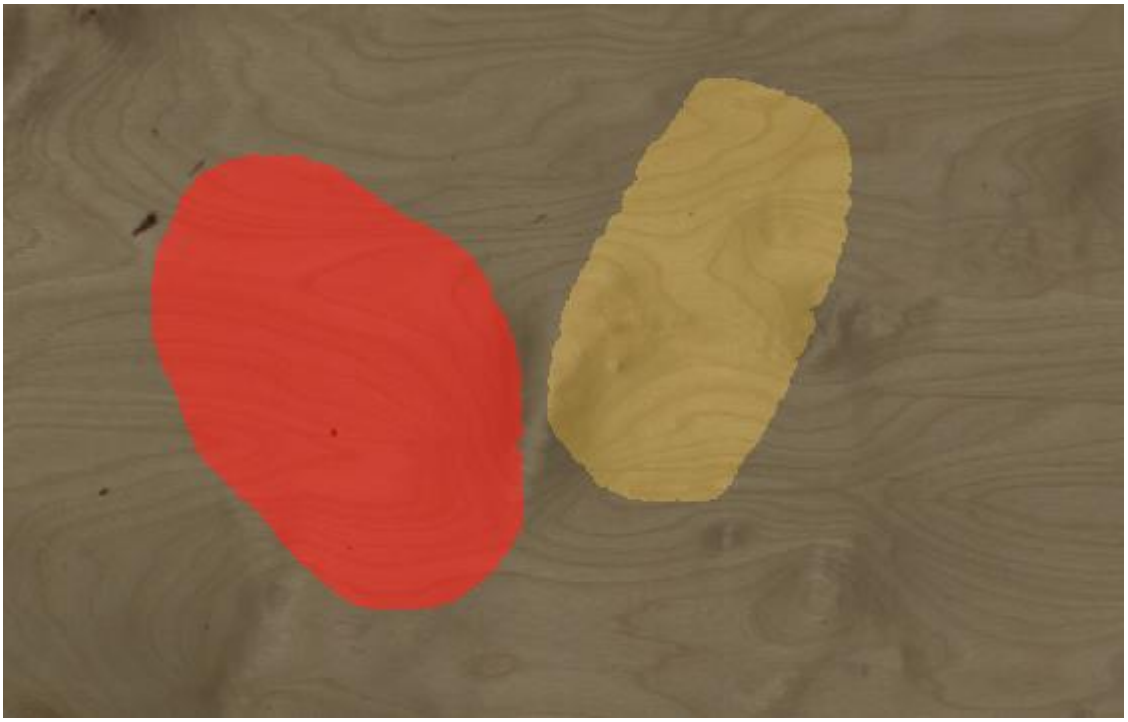
Käyttäjän valitsemasta kohdasta ulospäin, eli valittu merkintä, täytetään samanvärinen alue, tietyllä väriarvolla käyttäen OpenCV:n floodfill- eli täyttöfunktiota. Kuvasta tehdään maski ja lopputuloksessa kaikki muu on valkoista ja valittu merkintä on musta, kuten kuvan 20 esimerkki havainnollistaa.

Kuva 20. Maskikuva, jossa siirrettävä muoto

Tätä maskia käytetään sitten maalaamaan kohdevärikerrokselle uusi merkintä, pyyhkimättä kuitenkaan sillä mahdollisesti olevia olemassa olevia merkintöjä.

Jos kuvassa 16 esitelty ”Erase source”-valintaruutu on valittuna, tämän prosessin aikana myös pyyhitään lähdekerroksesta valittu muoto. Pyyhkimiseen käytetään samaa maskikuvaa, hieman muokattuna. Valkoisten pikselien kaikkien neljän kanavan arvot muutetaan nollaan ja maski annetaan parametriksi samalle pyyhkimisfunktiolle, jota käytettiin siivoamistyökalussa.

Kuvassa 21 nähdään samat muodot kuin aikaisemmin, mutta oikealla oleva muoto on kopioitu keltaiselle kerrokselle ja pyyhitty alkuperäiseltä värikerrokselta.



Kuva 21. Tyypin vaihtaminen siirtämällä

9.3 Muut työkalut

Kehityksen aikana myös muita työkaluja saatiin kehitettyä suorituskykyongelmien ratkomisen ja kuvanmuokkaustyökalujen kehityksen jälkeen. Kehityksessä siirrettiin näkökulma erilaisiin yleisesti ohjelmiston käyttöä helpottaviin ominaisuuksiin.

9.3.1 Kuvassa navigointi

Piirtopöydällä työskentelyn helpottamiseksi ja annotoitavassa kuvassa navigoimiseen, kuten zoomaamiseen ja kuvan liikutteluun, kehitettiin uusia ominaisuuksia. Muutokset tehtiin aikaisemmin luotuihin ImageView- ja ImageViewItem-luokkiin.

Kuvan zoomaamisen porrastetusti piirtonäyttöä käytettäessä käyttöliittymään lisättiin isot napit, joihin on näytön kynällä helppo osua, nappien toiminnoille lisättiin myös pikanäppäimet. Ominaisuus vaati käyttöliittymämuutoksien lisäksi muutoksia koodin toiminnallisuuksiin.

Teknisesti tämä ominaisuus toimii siten, että kun nappia painetaan, se lähettää signaalin, johon slotti vastaa, ja suorittaa toiminnon. Toiminnossa käytetään ImageViewin perittyä funktiota, ”scale” eli skaalaa. Funktiolle annetaan parametriksi arvot, joilla skaalataan x- ja y-akseleilla, eli kerrotaan nykyinen skaala tällä arvolla. Sisään zoomatessa luvut ovat arvoltaan enemmän kuin yksi ja ulos alle yksi.

Toinen erityisesti piirtonäyttöä varten kehitetty zoomausominaisuus on ”drag zoom” eli vetozoom. Se toimii siten, että sen ollessa käytössä, käyttäjän vetäessä kuvan päällä kynällä tai hiirellä nappi pohjassa oikealle tai vasemmalle, kuvaa zoomataan lähemmäs tai kauemmas.

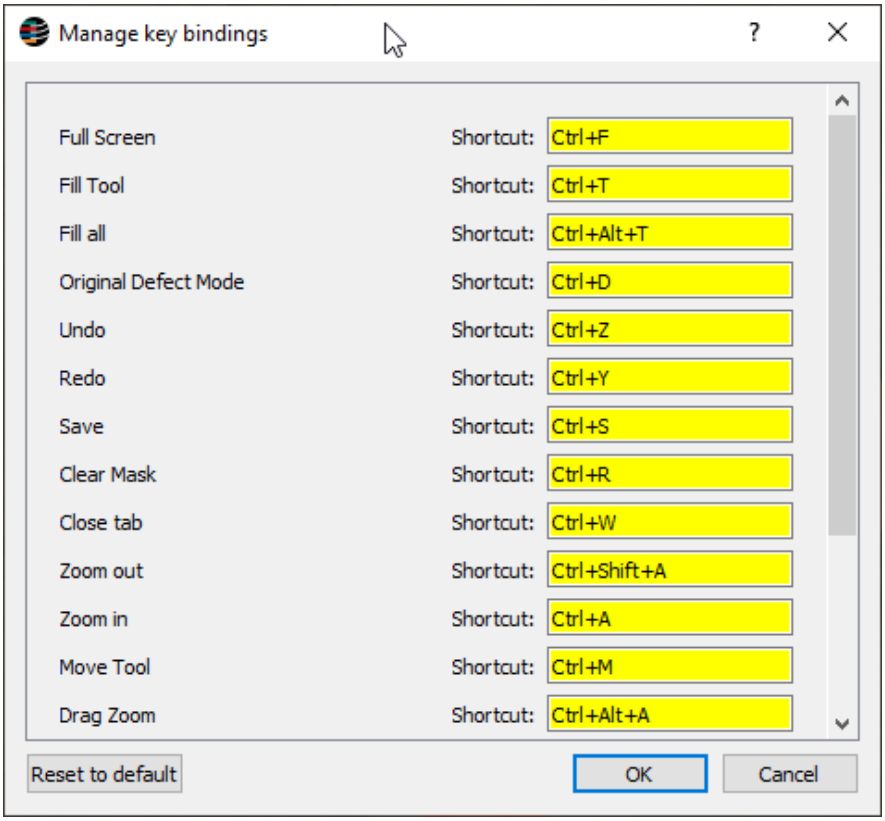
Teknisesti ominaisuus toimii taustalla siten, että klikkauksen jälkeen otetaan tarkasteluun alkuperäisen klikkauspisteen koordinaatti ja nykyisen raahauksenaikaisen pisteen koordinaatti. X-akselin erotuksella päätetään, zoomataanko ulos- vai sisäänpäin. Zoomaaminen tapahtuu ImageView-luokan perityllä ”scale”-skaalausfunktiolla, jota käytettiin myös painikezoomaustoiminnossa.

Kuvan siirtelyyn näkymässä kehitettiin yksinkertainen raahaustyökalu, jolla kuvan siirtely on kätevää. Työkalun ollessa käytössä, kursorilla voi napata kuvasta kiinni ja siirtää sitä haluamaansa kohtaan näkymän sisällä. Prosessi taustalla toimii siten, että raahatessa hiiritapahtumista seurataan, kuinka paljon arvo muuttuu ja sen perusteella päivitetään näkymän reunoilla olevia vierityspalkkeja, joiden muuttuminen päivittää sitä, mikä kohta kuvasta näkyy.

9.3.2 Muokattavat pikanäppäimet

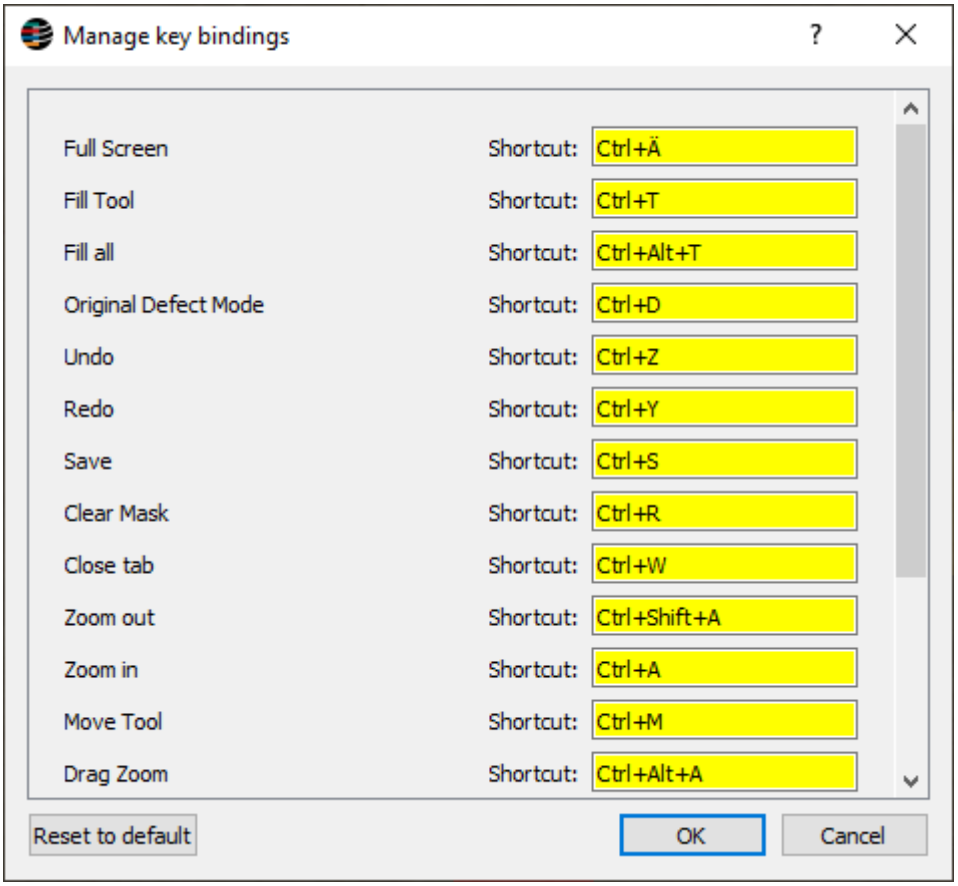
Kaikille uusille ominaisuuksille ja suurimmalle osalle vanhoista on määritelty myös pikanäppäimet, joiden avulla toiminnot voidaan laittaa päälle ja pois nopeasti. Näistä pikanäppäimistä haluttiin kehittää muokattavia, jotta käyttäjä pystyisi kustomoimaan ohjelmiston toimintaa omaan käyttöönsä sopivaksi. Piirtonäytön ominaisuuksien avulla pikanäppäinten käytön voi ohjelmoida siihenkin, joten muokattavuudesta on useita eri hyötyjä.

Muokattaville pikanäppäimille kehitettiin oma dialogi-ikkuna niiden hallitsemiseen ja muokkaamiseen. Kuvassa 22 näkyy kyseinen dialogi-ikkuna, jossa on listassa kaikki mahdolliset toiminnot, joille voidaan määrittää mukautettu pikanäppäin. Ikkunasta pystyy myös palauttamaan pikanäppäimien asetukset takaisin oletusasetuksiin.



Kuva 22. Muokattavien pikanäppäinten dialogi-ikkuna

Pikanäppäimen tai -näppäinyhdistelmän asettaminen toimii siten, että käyttäjä valitsee keltaisesta kohdasta sen asetuksen, jota haluaa muokata. Tämän jälkeen hän painaa näppäimistöllään haluamaansa uutta näppäinyhdistelmää ja toiminto kaappaa sen. Vanha yhdistelmä korvataan uudella käyttöliittymässä ja kun käyttäjä painaa ok, näppäinyhdistelmät tallennetaan. (Kuva 23)



Kuva 23. "Full screen" -pikanäppäimen asetus muutettu

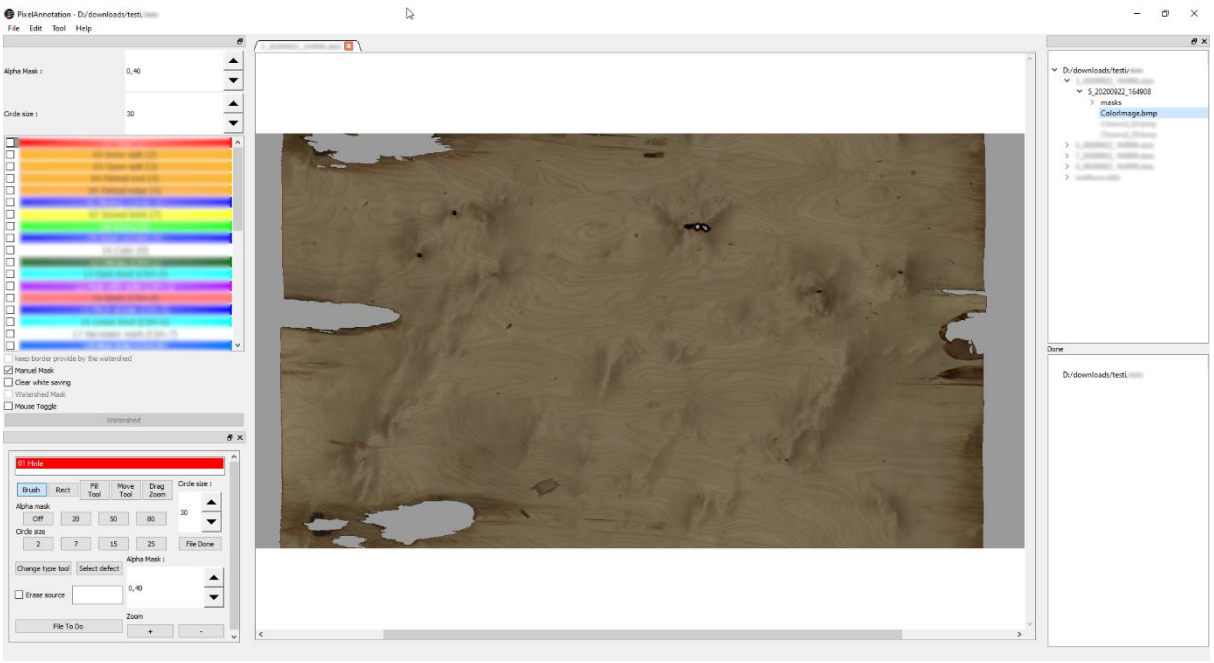
Toimintoa varten kehitettiin KeyCatcher-widgetti, joka ikään kuin nappaa käyttäjän näppäimistöllä antaman syötteen ja esittää sen tekstinä laatikossa. Luokan pohjana käytettiin tavallista yhden rivin QLineEdit-tekstikenttäwidgettia, jota yleensä käytetään vain käyttäjän tekstisyötteen keräämiseen.

Käyttäjän painaessa dialogi-ikkunan ”OK”-nappia, keltaisissa ruuduissa olevien syötteiden perusteella vaihdetaan käytössä olevat pikanäppäimet. Tätä varten ohjelmistoa ei tarvitse avata uudelleen, vaan asetukset voidaan ottaa käyttöön ajonaikaisesti. Uudet pikanäppäimet myös tallennetaan asetustiedostoon, josta ohjelmisto ottaa ne käyttöön käynnistyessään. Jos ohjelmistoa avattaessa asetustiedostoa ei ole olemassa tai se on esimerkiksi poistettu, ottaa se käyttöön pikanäppäimille oletusarvot.

Ikkunasta löytyvää reset-nappia käytettäessä ohjelmistoon kovakoodatut pikanäppäinten oletusarvot palautetaan ikkunaan. Ne otetaan käyttöön vain, jos käyttäjä tämän jälkeen hyväksyy muutokset painamalla ”OK”-nappia.

9.4 Käyttöliittymämuutokset

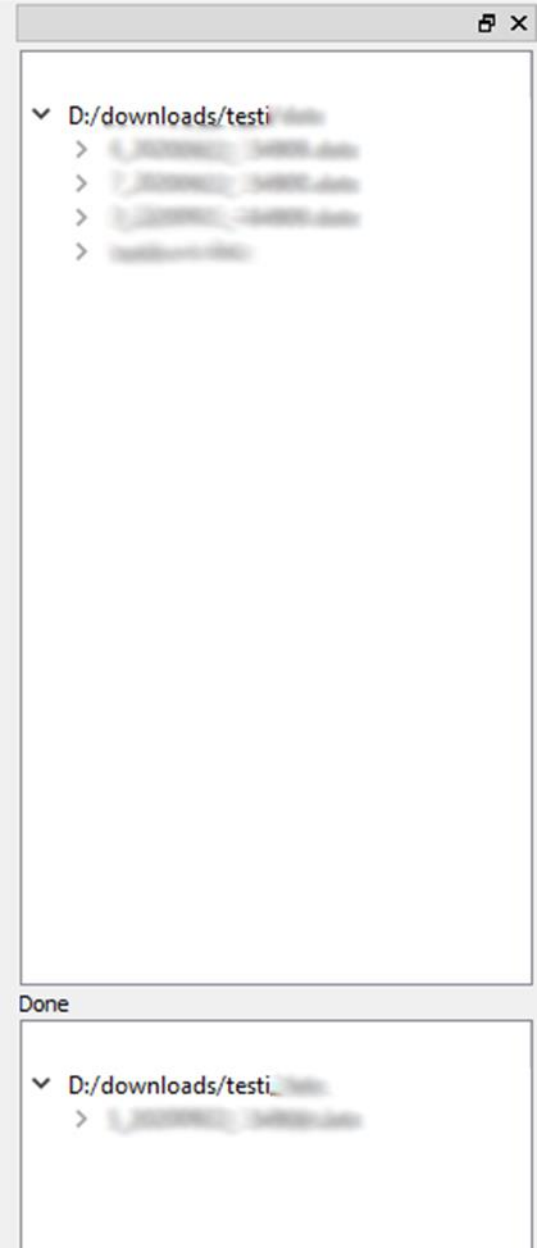
Ohjelmiston käyttöliittymään syntyi kehityksen aikana useita muutoksia, kuten kuvasta 24 voidaan nähdä. Kuvassa on esiteltynä käyttöliittymä kehityksen päätteeksi. Voidaan huomata, että siihen on lisätty erilaisia nappeja, elementtejä ja toimintoja sekä käyttöliittymän rakennetta on muunneltu.



Kuva 24. Ohjelmiston lopullinen käyttöliittymänäkymä

9.4.1 Edistymisen seuranta

Annotointiprosessissa tekijällä voi olla kerrallaan työn alla isojakin tiedostokokoelmia ja tiedostojen välillä siirtyessä voi olla vaikeaa pysyä kartalla, mitkä tiedostot ovat valmiita ja mitkä vaativat vielä työtä. Tätä tarkoitusta varten käyttöliittymään kehitettiin toiminto, jolla tiedostot voidaan jakaa työn alla oleviin ja valmiisiin. Alkuperäisen tiedostolistauksen alle lisättiin toinen samanlainen, johon valmiiksi merkityt tiedostot siirtyvät. (Kuva 25)



Kuva 25. Edistymisen seurannan tiedostolistausnäkymä

Käyttöliittymään lisättiin kaksi nappia toiminnolle: toisella voi merkitä tiedoston valmiiksi ja toisella takaisin tehtäväksi. Tämä nappi sen takia, jos jonkin tiedoston merkitsee vahingossa valmiiksi.

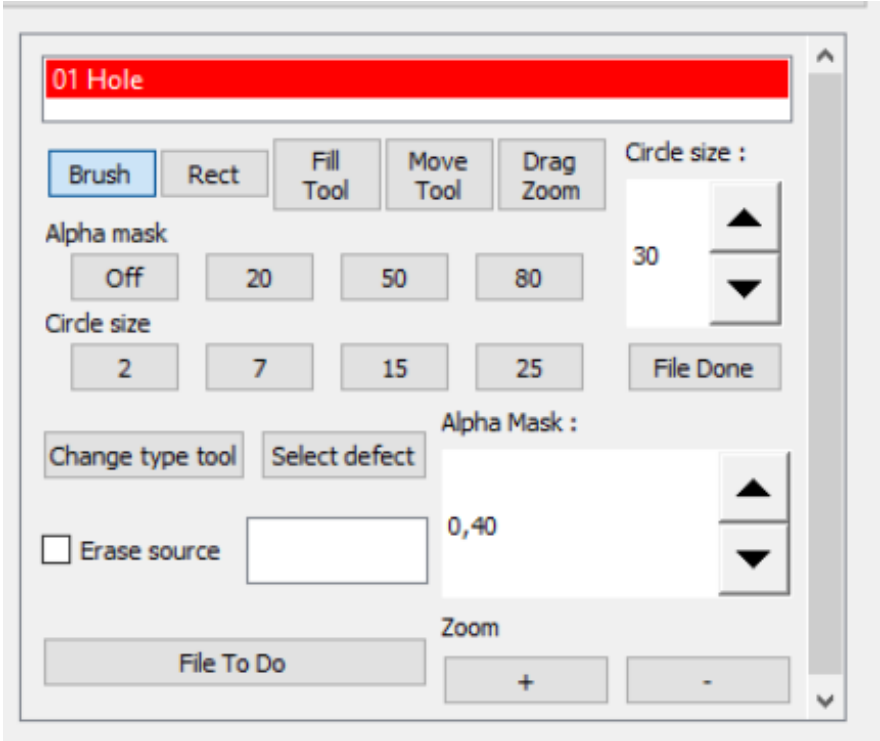
Taustalla toiminto toimii siten, että kun käyttäjä avaa kansion käytön aluksi ja ohjelmisto lataa tiedostolistaukseen kansion sisältämien tiedostojen nimet, ne ladataan myös valmis-tiedostolistaukseen, mutta piilotettuna. Kun tiedostoja merkitään valmiiksi, ne laitetaan näkyviin listaukseen, eli erillistä datan siirtoa tai poistamista ei tarvitse tehdä, joten tiedostonimien hallinta puurakenteessa on helpompaa.

9.4.2 Työkalulaatikko

Kaikkien uusien toimintojen listaamiseksi, sekä piirtonäytön käytön helpottamiseksi ohjelmistoon kehitettiin ne kaikki kokoava työkalulaatikko. Kuvassa 26 on lähempi kuvankaappaus työkalulaatikosta, joka löytyy myös aikaisemmin kuvan 24 vasemmasta alanurkasta.

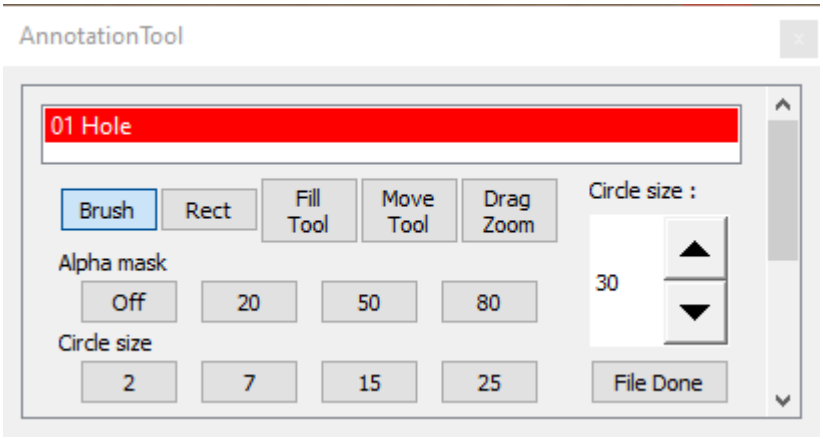
Ylhäältä alas kuvattuna työkalulaatikkoon sisällytettiin muun muassa näkymä siitä, mikä väri on käytössä tällä hetkellä, napit yleisimmille työkaluille, pikanapit ja asetukset maskin ja siveltimen asetuksille, tyypinvaihtotyökalu, napit tiedoston valmiiksi kuittaamiseen ja pikanapit zoomaukselle.

Yleisimpiä työkaluja ovat normaali sivellintyökalu (Brush), suorakulmiotäyttötyökalu (Rect), täyttötyökalu (Fill Tool), raahaustyökalu (Move Tool) ja raahauszoom (Drag Zoom).



Kuva 26. Työkalulaatikon näkymä

Työkalulaatikko kehitettiin käyttäen Qt:n design-työkalua, pohjana käytettiin QScrollArea-widgettiä, joka on alue, joka mahdollistaa widgetin scrollaamisen. Tämän avulla työkalulaatikko on mahdollista irroittaa käyttöliittymästä ja siirtää haluamaansa kohtaan näytöllä. Myös sen kokoa pystyy skaalaamaan pienemmäksi, jos haluaa esimerkiksi piilottaa joitain toimintoja. (Kuva 27)



Kuva 27. Työkalulaatikko pienennettynä

10 Yhteenveto

Työn tavoitteena oli jatkokehittää ohjelmistoa, johon Raute Kajaanissa oli jo tehty jonkin verran kehitystyötä ja joka oli käytössä annotoinnin työkaluna. Tätä työtä tehdään isoissa dataseteissä manuaalisesti, joten prosessin tehostaminen työkalun käytettävyyttä kehittämällä oli haluttua. Tarkoituksena oli kehittää ohjelmiston piirtämisen suorituskykyä ja ominaisuuksia käyttäjäystävällisyyden edistämiseksi ja piirtonäytön käyttö työssä tuli ottaa huomioon kehityksessä.

Ohjelmiston suorituskykyä onnistuttiin tehostamaan alkuperäiseen verrattuna huomattavasti erilaisilla muutoksilla ohjelmistoon. Ohjelmiston tarkoitukseen toimimattomia komponentteja vaihdettiin paremmin sopiviin, jotka on tehty monimutkaisen grafiikan näyttämiseen sekä refaktoroitiin koodipohjaa. Tähän kului työssä eniten aikaa, sillä näiden rakenteellisten ja loogisten muutosten toteuttaminen vaati uusien teknologioiden opettelua ja soveltamista, jotta lopullinen ratkaisu toimisi samalla tavalla kuin vanha, mutta tehokkaammin.

Ominaisuuksia saatiin työn aikana kehitettyä monia erilaisia, vanhoja paremmaksi sekä myös kokonaan uusia. Suorituskykyä parantamalla tavallisen piirtämistyökalunkin sujuvuus parantui huomattavasti, mutta jäi vielä vähän pätkiväksi, eikä tähän tämän työn aikataululla löydetty ratkaisua. Sen sijaan haluttiin keskittyä kehittämään piirtämistä tukemaan erilaisia työkaluja annotointiin ja annotaatioiden muokkaamiseen. Nyt esimerkiksi merkintöjä pystyy täyttämään eri tavoin, lisäämään suorakulmiona kerrallaan ja niiden tyyppiä pystyy vaihtamaan.

Käyttöliittymää ja sen ominaisuuksia muokkaamalla pyrittiin saamaan ohjelmiston käyttäjäystävällisyyttä paremmalle tasolle, missä onnistuttiin suurilta osin. Kuvassa liikkumiseen ja tarkentamiseen kehitettiin erilaisia uusia työkaluja, jotka ottavat piirtonäytön ominaisuudet huomioon. Pikanäppäimistä saatiin täysin muokattavat, joka lisää ohjelmiston muokattavuutta jokaisen käyttäjän omaan käyttötapaan.

Ohjelmiston jatkokehityksen tavoitteisiin päästiin suurelta osin ja sen yleistä käytettävyyttä ja suorituskykyä pystyttiin työn aikana kehittämään. Lopputuloksena ohjelmisto toimii alkuperäistä sulavammin, siinä on useita uusia nopeasti toimivia käyttäjäystävällisyyttä lisääviä ominaisuuksia ja sen käyttäminen työssä on tehokkaampaa.

Lähteet

1. Company [Internet]. raute.com. 2021 [viitattu 6.1.2023]. Saatavilla: https://www.raute.com/company/

2. Vilkka Hanna, Airaksinen Tiina. Toiminnallinen opinnäytetyö. Tammi; 2003.

3. MacMullen, W. John. Annotation as Process, Thing, and Knowledge: Multi-domain studies of structured data annotation. SILS Technical Report TR-2005-02. Chapel Hill: University of North Carolina, School of Information and Library Science, Technical Report Series; 2005.

4. Nagao K. Digital Content Annotation and Transcoding. Norwood: Artech House; 2003.

5. Kansalaisyhteiskunnan pitkäaikaissäilytysopas - Metatieto ja dokumentointi [Internet]. Xamk.fi. 2019 [viitattu 9.10.2022]. Saatavilla: https://digitalia.xamk.fi/digiopas/4

6. Ramesh P, Subramaniam T, Ray P, Devadas A, Ramesh S, Ansar S, et al. Utilizing human intelligence in artificial intelligence for detecting glaucomatous fundus images using human-in-the-loop machine learning. Indian Journal of Ophthalmology. 2022 70(4):1131-8 Saatavilla:https://kamezproxy01.kamit.fi:2405/login.aspx?direct=true&db=a9h&AN=155970066&site=ehost-live

7. Web Content Accessibility Guidelines (WCAG) 2.1 [Internet]. W3.org. 2018 [viitattu 20.10.2022]. Saatavilla: https://www.w3.org/TR/WCAG21/

8. Bianco S, Ciocca G, Napoletano P, Schettini R. An interactive tool for manual, semi-automatic and automatic video annotation. ResearchGate. Elsevier; 2015 Saatavilla: https://www.researchgate.net/publication/272391587_An_interactive_tool_for_manual_semi-automatic_and_automatic_video_annotation

9. käytettävyys | TEPA-hakutulos erikoisalojen sanastoista ja sanakirjoista [Internet]. Termipankki.fi. 2023 [viitattu 22.2.2023]. Saatavilla: https://termipankki.fi/tepa/fi/haku/k%C3%A4ytett%C3%A4vyys

10. Iso.org. ISO 9241-11:2018(en) Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts. 2023. Saatavilla: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en

11. Usability 101: Introduction to Usability [Internet]. Nielsen Norman Group. 2016 [viitattu 24.2.2023]. Saatavilla: https://www.nngroup.com/articles/usability-101-introduction-to-usability/

12. How to Measure Learnability of a User Interface [Internet]. Nielsen Norman Group. 2019 [viitattu 24.2.2023]. Saatavilla: https://www.nngroup.com/articles/measure-learnability/

13. QA (quality assurance) & UX (user experience) [Internet]. Nielsen Norman Group. 2013 [viitattu 24.2.2023]. Saatavilla: https://www.nngroup.com/articles/quality-assurance-ux/

14. Nielsen J. Usability engineering. Morgan Kaufmann Publishers Inc.; 1994.

15. How to Measure Learnability of a User Interface [Internet]. Nielsen Norman Group. 2019 [viitattu 24.2.2023]. Saatavilla: https://www.nngroup.com/articles/measure-learnability/

16. Thielsch MT, Engel R, Hirschfeld G. Expected usability is not a valid indicator of experienced usability. PeerJ Computer Science 1:e19; 2015. https://doi.org/10.7717/peerj-cs.19

17. Iterative Design of User Interfaces [Internet]. Nielsen Norman Group. 2023 [viitattu 24.2.2.2023]. Saatavilla: https://www.nngroup.com/articles/iterative-design/

18. The Benefits of User Experience :: UXmatters [Internet]. Uxmatters.com. 2017 [viitattu 25.2.2023]. Saatavilla: https://www.uxmatters.com/mt/archives/2017/12/the-benefits-of-user-experience.php

19. Benefits of User-Centered Design | Usability.gov [Internet]. Usability.gov. 2023 [viitattu 25.2.2023]. Saatavilla: https://www.usability.gov/what-and-why/benefits-of-ucd.html

20. The Aesthetic-Usability Effect [Internet]. Nielsen Norman Group. 2017 [viitattu 23.2.2023]. Saatavilla: https://www.nngroup.com/articles/aesthetic-usability-effect/

21. Levajsics N. Unsplash [Valokuva] 2017 [viitattu 1.4.2023] Saatavilla: https://unsplash.com/photos/1vwwZ-BmmrE

22. Bréhéret Amaury. PixelAnnotationTool [Internet]. GitHub. 2017 [viitattu 21.10.2022]. Saatavilla: https://github.com/abreheret/PixelAnnotationTool

23. Qt for Investors [Internet]. Investors.qt.io. 2022 [viitattu 17.9.2022]. Saatavilla: https://investors.qt.io/fi/

24. Qt Creator [Internet]. Www.qt.io. 2022 [viitattu 17.9.2022]. Saatavilla: https://www.qt.io/product/development-tools?hsLang=en

25. Signals & Slots | Qt Core 6.4.3 [Internet]. Doc.qt.io. 2023 [viitattu 19.3.2023]. Saatavilla: https://doc.qt.io/qt-6/signalsandslots.html

26. The Event System | Qt Core 6.4.3 [Internet]. Doc.qt.io. 2023 [viitattu 2.4.2023]. Saatavilla: https://doc.qt.io/qt-6/eventsandfilters.html

27. Baksheev K. Realtime Computer Vision with OpenCV. Queue [Internet]. 2012 Apr;10(4). Saatavilla: https://dl.acm.org/doi/pdf/10.1145/2181796.2206309

28. About - OpenCV [Internet]. OpenCV. 2020 [viitattu 17.10.2022]. Saatavilla: https://opencv.org/about/

29. QLabel Class | Qt Widgets 6.4.3 [Internet]. Doc.qt.io. 2023 [viitattu 17.3.2023]. Saatavilla: https://doc.qt.io/qt-6/qlabel.html