Piirustusohjelma
projektin dokumentti

Kaisa Voutilainen

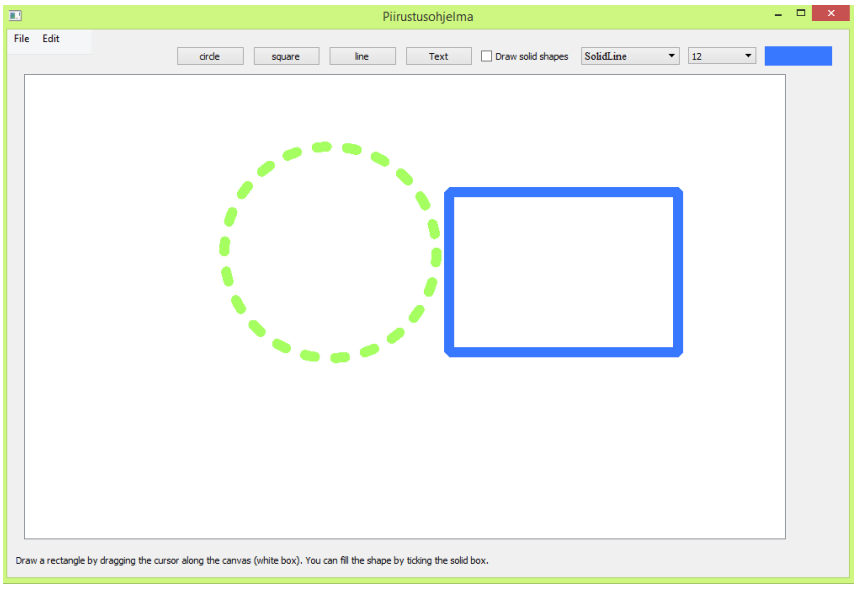
481629

AUT, 2. vuosikurssi

5.5.2016

2. Yleiskuvaus

Projektin aiheena on piirustusohjelma, joten projektissa on siis luotu hiirellä toimiva graafinen piirustusohjelma. Kuten suunnitelmassa on mainittu, piirustusohjelmalla voidaan piirtää viivoja, ympyröitä, soikioita, neliöitä ja suorakaiteita. Lisäksi jokaiselle muodolle voidaan etukäteen määritellä oma värinsä. Muutoksena suunnitelmaan, ohjelmassa voidaan myös päättää, piirretäänkö muodosta vain ääriviiva vai täytetäänkö se, ja värin valitseminen toimii klikkaamalla tämänhetkisen värin ilmaisevaa suorakaidetta, joka avaa käyttäjälle värinvalitsemis dialogin. Piirustukseen voidaan myös lisätä tekstiä, jonka värin ja fonttikoon saa itse päättää. Kaikkia lisättyjä objekteja voidaan myös jälkeenpäin siirtää edes takaisin ruudulla.



Kuva 1 Screenshot toteutuneesta piirustusohjelmasta

Suunnitelman mukaan piirustusohjelmassa on myös toimiva undo-ominaisuus, joka poistaa viimeisimmän kuvaan tehdyn muutoksen. Lisäksi ohjelmassa on mahdollista tallentaa piirretty kuva ja ladata se tiedostosta myöhemmin, sekä luoda uusi piirustus. Lisäyksenä tähän, ohjelmassa on myös mahdollista tuoda nykyiseen piirustukseen .png tai .jpg tiedosto sekä viedä oma piirustus .png muotoon.

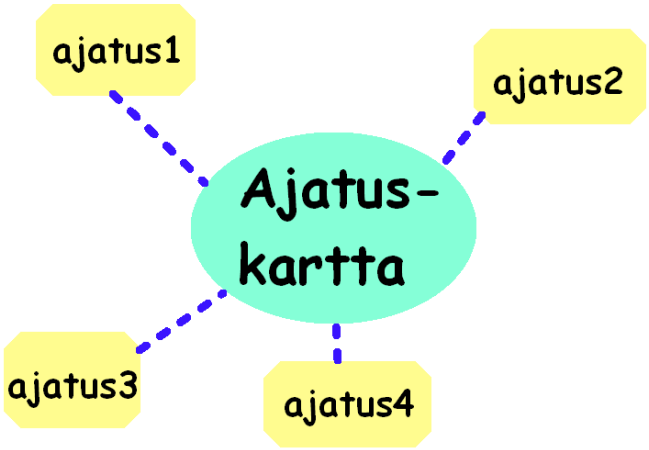
Projekti on toteutettu keskivaikeana työnä, kuten suunniteltu. Ohjelman pitäisi täyttää keskivaikean työn vaatimukset, ja siihen on lisätty myös ominaisuuksia, joita siinä ei vaadittu. Työ ei kuitenkaan täytä kaikkia vaikean työn vaatimuksia.

3. Käyttöohje

Ohjelma käynnistetään kaksoisklikkaamalla projektin src-kansiosta tiedostoa main.py. Kun ohjelma on käynnistynyt, eli piirustusohjelman ruutu on ilmestynyt näytölle, voi sillä alkaa heti piirtämää. Piirtäminen tapahtuu yksinkertaisesti valitsemalla, mitä haluaa piirtää ja klikkaamalla sitä vastaavaa nappulaa. Valittavia vaihtoehtoja ovat: ympyrä, neliö, viiva ja teksti. Ympyrä-vaihtoehdosta voi piirtää ympyröitä ja soikioita, neliö-vaihtoehdosta voi piirtää neliöitä tai suorakaiteita, viiva-vaihtoehdosta voi piirtää viivoja ja teksti-vaihtoehdosta voi kirjoittaa tekstiä.

Piirtäminen toimii yksinkertaisesti samoin kuin muissakin piirustusohjelmissa. Kirjoittaminen puolestaan toimii siten, että käyttäjä klikkaa haluamaansa kohtaa piirtoalueella ja alkaa kirjoittaa. Kirjoittamista varten ei ilmesty erikseen indikaattoria, mihin tekstiä pitäisi kirjoittaa. Jos oletusasetukset eivät ole käyttäjän mieleen, voi hän muuttaa kynän väriä, leveyttä ja piirrettävän viivan tyyliä päänäkymän ylärivissä olevista valikoista.

Jos piirtäessä haluaa poistaa lisättyjä muotoja, tapahtuu se valitsemalla ”Edit”-valikosta undo tai käyttämällä toimintoa vastaavaa näppäinyhdistelmää. Piirretty kuva voidaan tallentaa ”File”-valikosta käsin. Valikosta löytyy myös toiminnot tallennetun kuvan lataamiseen, kuvatiedoston lisäämiseen, kuvan viemiseen .png-muodossa sekä uuden piirustuksen luomiseen. ”Edit”-valikosta käsin voidaan myös muuttaa piirustusalueen kokoa ja valita ”select”-toiminto, jonka avulla käyttäjä voi valita ja siirtää piirtämiään muotoja ja tekstiä.



Kuva 2 Piirustusohjelmalla piirretty suunnitelmassa esitetty ajatuskartta, joka on exportattu käyttäen ohjelman toimintoja

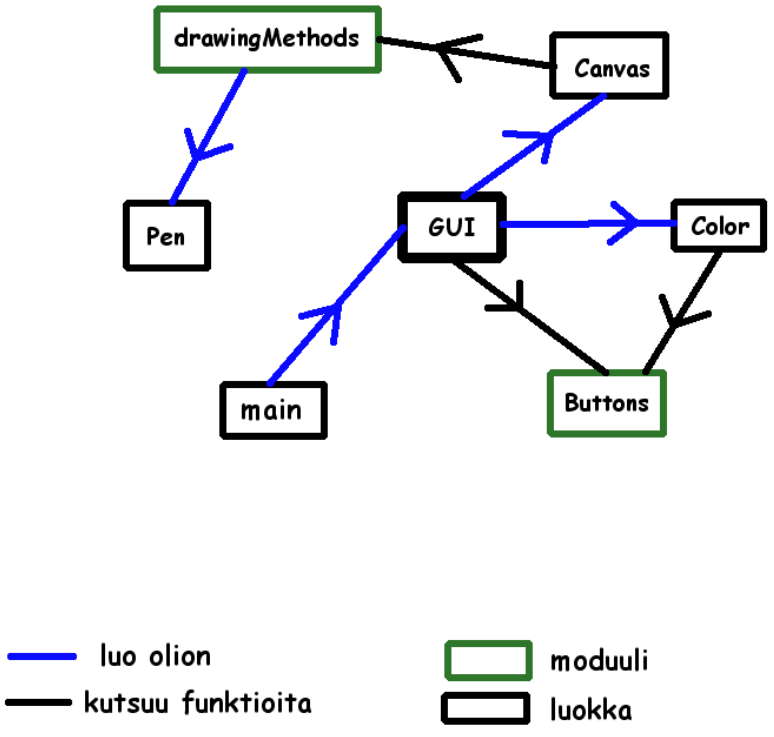
4. Ohjelman rakenne

Koska kyseessä on graafinen ohjelma, tärkein ominaisuus sen rakenteessa on käyttöliittymä, joka ottaa vastaan käyttäjän inputteja ja kutsuu niiden seurauksena tarvittavia funktioita ja metodeita. Ohjelmassa käytetyt luokat löytyvät alla esitetystä taulukosta:

Luokan nimi	Periytyy	Käyttötarkoitus	Keskeiset metodit
GUI	QWidget	ohjelman graafinen käyttöliittymä	**initUI(self)**, jossa alustetaan koko graafinen käyttöliittymä
			buttonWrapper(self), joka ajetaan aina kun nappuloita painetaan, kutsuu Buttons-moduulissa olevaa klikkaus-funktiota
Color	QFrame	klikattava valittua väriä ilmaiseva laatikko GUI:ssä	**mousePressEvent(self, event)**, joka on uudelleen implementoitu metodi tunnistamaan, että hiirellä on klikattu, kutsuu Buttons-moduulissa olevaa klikkaus-funktiota
Canvas	QGraphicsView	piirustusalue GUI:ssä	**createPic(self, …)**, joka ajetaan ladattua kuvaa piirrettäessä. Ottaa parametreikseen ominaisuuksia, joita tarvitaan graafisten objektien uudelleenpiirtämiseen
			mouseMoveEvent(self, event), uudelleen implementoitu metodi hiiren liikkeiden seuraamiseen, päivittää piirrettävien muotojen leveyttä ja korkeutta tai siirrettävien muotojen sijaintia
			mousePressEvent(self, event), uudelleen implementoitu metodi, jossa havaitaan hiiren klikkaus. Tällä määritellään piirrettävän muodon aloituspiste, siirrettävän muodon alkusijainti tai kirjoitettavan tekstin alkupiste
			mouseReleaseEvent(self, event), uudelleen implementoitu metodi, jolla havaitaan hiiren klikkauksen loppumista. Tällä lisätään tallennusta varten oleellista informaatiota listaan
			keyPressedEvent(self, event), uudelleen implementoitu metodi painettujen näppäinten havaitsemiseen. Tällä havaitaan ja päivitetään kirjoitettavaa tekstiä jos toiminto on valittuna
Pencil	QPen	piirtämiseen käytettävä itse määritelty kynä	

Ohjelma sisältää myös kaksi moduulia, jotka sisältävät funktioit, joita kutsutaan tarvittaessa. Buttons-moduulissa on tärkeänä funktio Clicked, jota kutsutaan aina kun jotain nappulaa tai toimintoa klikataan. Tämä funktio selvittää, mitä on painettu ja kutsuu sen seurauksena muita Buttons-moduulissa olevia funktioita. Funktioita löytyy tallennukselle, lataamiselle, peruutustoiminnolle, kuvien viemiselle ja tuomiselle sekä piirtoalueen muuttamiselle.

Piirtäminen puolestaan hoidetaan moduulin drawingMethods kautta. Moduuli sisältää funktiot ellipsin, nelikulmion, viivan ja tekstin piirtämiselle. Nämä funktiot taas luovat piirtämistä varten Pencil-olion, jonka ominaisuudet muuttuvat käyttäjän valintojen mukaan.



Kuva 3 Ohjelmalla piirretty kaavio ohjelman rakenteesta ja suhteista

5. Algoritmit

Piirustusohjelmassa algoritmeja tarvitaan lähinnä piirtämiseen ja tallennukseen sekä lataamiseen. Piirtämisessä toteutus on hyvin epäoptimaalinen: kun hiiren klikin havaitaan painuneen pohjaan ja alkavan liikkua, piirretään uusi kuva muuttuneilla ominaisuuksilla (leveys/korkeus) ja poistetaan edellinen lisätty objekti ruudulta. Nelikulmioita piirrettäessä koordinaatteja tulee tarkastella tarkemmin, koska jostain syystä QGraphicsRectItem ei tue negatiivisia korkeuksia tai leveyksiä.

Vaihtoehtoinen toteutustapa piirtämiselle olisi ollut yksinkertaisesti uuden objektin lisääminen hiiren klikillä ja sen ominaisuuksien päivittäminen update()-komennolla hiiren liikkuessa. Päädyin tämänhetkiseen ratkaisuun ennen kuin löysin dokumentaatiosta update()-komennon, joten ominaisuus oli jo toteutettu. Lisäksi aikaa oli löydön hetkellä niin vähän jäljellä, että en halunnut lähteä muuttamaan ohjelman tärkeintä ominaisuutta täysin erilaiseen toteutukseen.

Tallennus toteutetaan käymällä läpi piirrettyjen objektien lista ja ottamalla ylös, mikä muoto on kyseessä. Sen lisäksi käydään läpi kaksi apulistaa, joista otetaan jokaiselle objektille ylös oleellisia ominaisuuksia, kuten koordinaatit, leveys, korkeus, väri ja kynän asetukset. Nämä lisätään listaan siten, että jokaiselle objektille on oma alkio, joka sisältää yhtenäisen tekstijonon, jossa ominaisuudet erotellaan pilkulla. Lopuksi lista tallennetaan tekstitiedostoon. Esimerkki tallennetun tiedoston sisällöstä löytyy otsikon 7 alta.

Latauksessa puolestaan luetaan ensin koko tiedosto ja jaetaan sen alkiot split()-komennolla. Sen jälkeen jokaisen alkion sisältö jaetaan pilkun mukaan alkioihin uuteen listaan ja tästä voidaankin lukea ensimmäinen alkio, joka määrittelee, mikä muoto on kyseessä. Seuraavaksi kutsutaan Canvas-luokan metodia, joka sitten annettujen alkioiden perusteella piirtää oikeanlaiset muodot.

Tallennukseen ja lataamiseen oli varmasti löytynyt parempikin tapa, mutta en ehtinyt perehtyä kovin syvällisesti esimerkiksi QGraphicsItem-luokkaan, mistä olisin saanut muotojen ominaisuuksia helpommin.

6. Tietorakenteet

Käytin ohjelmaa tehdessä listoja ja taulukoita tiedon varastoimiseen. Piirrettyjen objektien säilöminen toimi helposti jo implementoidun metodin QGraohicsScene.AddItem() avulla, sekä objektien saaminen listasta toimi metodilla QGraphicsScene.items(), joka palauttaa listan kaikista objekteista. Tässä siis käytin listoja.

Loin myös omia listoja piirrettyjen kuvioiden ominaisuuksien tallennukseen. Näihin listoihin lisäsin jokaiselle kuviolle oman listan ominaisuuksista, lopputuloksena oli taulukko. Mielestäni tämä oli paras ja helpoin tapa toteuttaa tarvitsemani tietojen tallennus.

7. Tiedostot

Ohjelma tallentaa kuvia tekstitiedostoon ja lataa ne myös sieltä. Vaikka tallennustiedosto onkin tekstitiedosto, ohjelma antaa tiedostopäätteeksi .kbv, jotta voidaan helposti tunnistaa, mitkä tiedostot ovat ohjelmaltani, koska tiedostojen lataaminen vaatii juuri tietynlaisen rakenteen tekstitiedostossa. Rakenne tiedostossa on seuraavan lainen:

```
900.0,550.0;['textTekstiä,136,390,135,391,#b561ff,True,DashDotLine
,12,', 'line,331,304,713,358,#b561ff,True,DashDotLine,12,',
'rect,451,72,726,225,#000000,True,SolidLine,PenWidth,',
'ellipse,129,63,307,229,#000000,False,SolidLine,PenWidth,']
```

Tiedostosta siis löytyy lista alkioista, jotka sisältävät kuvion tyypin ja ominaisuudet. Aivan ensimmäisenä tiedostossa lukee piirustuksen piirustusalueen koko (tässä 900x550). Kuviot on listattu niin, että viimeisimmäksi lisätty kuvio on listan ensimmäinen alkio. Jos kuvion tyyppi on ’text’, on heti tyypin perään kirjoitettu kuvion sisältämä teksti. Tässä tapauksessa siis käyttäjä on kirjoittanut kuvaan viimeisenä tekstiä, jonka sisältö on ollut ”Tekstiä”.

Ensimmäisenä piirustukseen on lisätty ellipsi, jonka vasen yläkulma on pisteessä (129, 63), leveys on 307, korkeus on 229, väri on #000000, kuvio ei ole täytetty eli on piirretty vain ääriviiva, kynän viivana on yhtenäinen viiva (mahdolliset viivatyypit löytyvät QPen dokumentaatiosta), eikä kynän leveyttä ole muutettu.

Koska ohjelmassa on toteutettuna myös kuvien tuominen ja mahdollinen vieminen, pystyy se avaamaan kuvatiedoston (.png ja .jpeg) ja myös tallentamaan .png muotoon.

8. Testaus

Ohjelman luonteesta ja rakenteesta johtuen sitä ei juuri pystytä yksikkötestaamaan, koska koko ohjelma on riippuvainen graafisesta käyttöliittymästä. Sille voidaan kuitenkin keksiä testitapauksia, ja tietysti jokaista ominaisuutta on myös huolella testattu ohjelmoinnin ohella heti kun ne on implementoitu. Lisäksi lopuksi ohjelmaa tuli testattua niin monella eri tapaa kuin mieleen tuli ideoita, jotta kaikki rikkimenevät kohdat löytyisivät ja ne saisi korjattua.

Suunnitelmassa mainittuja testitapauksia ovat esimerkiksi peruutustoiminnon käyttäminen, kun ei ole piirretty mitään; tallentaminen ja lataaminen sekä piirtämisen yrittäminen piirtoalueen ulkopuolella. Ohjelman koodauksesta ja rakenteesta johtuen peruutustoiminto ei yritä poistaa olemattomia kuvioita, koska se on ohjelmoitu tarkistamaan ensin, onko mitään poistettavaa. Piirtoalueen ulkopuolellekaan ei ole mahdollista piirtää, koska piirtoalustana käytetään

QGraphicsViewiä, joka tunnistaa vain sen alueella suoritetut hiirenklikkaukset. Näitä ominaisuuksia on kuitenkin testattu käyttöliittymässä ja ne toimivat kuten niiden kuuluukin.

Tallennus ja lataus puolestaan eivät välttämättä heti toimi oikein, joten niitä täytyi testata paljon. Paras tapa huomata, toimivatko tallennus ja lataus oikein, on piirtää ensin kuva, joka sisältää kaikkea mahdollista: kaikkia kuvioita, eri värejä, eri viivatyyppejä, eri kynän leveyksiä ja tallentaa kuva. Koska kuva ei katoa tallennettaessa, voidaan päälle ladata heti juuri tallennettu kuva. Jos kuvas muuttuu, tallennuksessa tai latauksessa on jotain vikaa. Ensimmäisillä toteutuksilla näin testatessa kuva muuttui useinkin ja muutosten perusteella oli suhteellisen helppo löytää, missä vika oli. Lopullisessa ohjelmassa kuitenkin ladattu kuva näyttä täysin samalta kuin tallennettukin, ja peruutustoimintokin poistaa kuviot viimeisimmästä ensimmäiseen.

9. Ohjelman tunnetut puutteet ja viat

PyQt:n version valinnasta ja liian vähäisestä perehtymisestä johtuen en saanut kaikkea toteutettua niin hienosti kuin olin suunnitellut tai halunnut, joten ohjelma tietysti sisältää heikkouksia. Kaikista suurin vika on piirustusten vieminen (export). Kyseinen toiminto onnistuu tallentamaan piirustuksen oikein kuvaksi (kuva.png kovakoodatusti), mutta aiheuttaa jostain syystä aivan satunnaisesti C++ Visual run-time Errorin, syynä ”pure virtual function call”, ja kaataa ohjelman. Ominaisuus on kuitenkin yleensä ainakin ensimmäisellä yrityskerralla toimiva ja hieno lisäys, joten halusin jättää sen lopulliseen ohjelmaan. PyQt: versiossa 4 ominaisuuden olisi voinut toteuttaa eri tavalla, joka toimii täydellisesti joka kerralla.

Muita heikkouksia ohjelmassa ovat esimerkiksi ei niin nätti ulkoasu, peruutustoiminnon epätäydellinen toiminta latauksen jälkeen, se että kuvioista voi valita vain päällimmäisen vaikka ne olisivat vain ääriviivoja, rajattu piirtoalue, piirtämisen ja tekstin lisäämisen epäoptimaalinen toteutus, se että tietyt merkkijonot kuvaan lisättynä hajottavat tiedoston ladatessa sekä se, että kuviot jättävät jälkeensä outoja jälkiä.

Koska ohjelma on graafinen, olisi toivottavaa, että sen ulkoasu olisi myös hieno ja käyttäjäystävällinen. Valitettavasti käyttöliittymä on kuitenkin mahdollisesti aluksi hankala käyttäjälle, eikä kovin esteettinen. Ongelmaa helpottaakseni olen lisännyt ohjelman alareunaan info-tekstin auttamaan käyttäjää ja intuitiivisia näppäinyhdistelmiä, kuten Ctrl+S ja Ctrl+Z. Ongelman voisi korjata perehtymällä paremmin PyQt:n layouteihin ja hyödyntämällä niitä oikein.

Vaikka vaadittu peruutustoiminto toimiikin täysin oikein uudelle piirustukselle, halusin silti lisätä toiminnon myös ladatuille piirustuksille. Aluksi ominaisuus toimi kuten pitikin, mutta lisäsin myöhemmin ohjelmaan kuvioiden siirtämisominaisuuden, jonka käyttöä ei tallenneta. Tämä siis tarkoittaa sitä, että jos piirustuksessa on liikuteltu kuvioita ennen tallennusta, ei ladatussa kuvassa tiedetä, että niillä on ollut eri sijainti aiemmin, joten peruutustoiminnon käyttö ladattuun kuvaan vain poistaa kuvioita viimeisimmästä ensimmäiseen, mutta ei siirrä niitä ensin niiden alkuperäisille paikoilleen. Ladatun kuvan päälle piirretyt kuviot ja niiden siirtely kuitenkin otetaan huomioon ja toimii peruutustoiminnossakin. Tämän ongelman voisin korjata lisäämällä olemassa olevan workLogin tallennukseen.

Kuvioiden liikuttelua ei keskivaikean työn vaatimuksissa ollut, mutta lisäsin sen, koska se on tärkeä ominaisuus piirustusohjelmassa. Koska käytän kuvion valinnassa QGraphicsScene:n valmista metodia itemAt(), käyttäjä voi valita vain päällimmäisen kuvion ja liikuttaa sitä tapauksessa, jossa on esimerkiksi pieni ympyrä piirrettynä ensin ja isompi neliö piirrettynä sen päälle ja neliö on piirretty vain ääriviivoilla. Tapauksessa näyttäisi siltä, että ympyrää voisi siirtää, mutta koska kuviot on määritelty täyttämään koko kuvion alueen, ei ympyrää pysty siirtämään ennen kuin neliö on ensin siirretty pois sen päältä. Ongelman voisi korjata mahdollisesti koodaamalla oman metodin, joka pystyy valitsemaan kuvioita läpinäkyvien kuvioiden läpi.

Ohjelmassa piirtoalue on rajattu siten, että siihen ei koskaan ilmesty scrollbareja. Olisi kuitenkin parempi, jos käyttäjä pystyisi määrittelemään itse aivan niin suuren piirtoalueen kuin hän haluaa. Ongelma kuitenkin oli se, että kun piirtoaluetta scrollasi, koordinaatit, joita käytetään piirtämiseen, muuttuivat vääriksi. Päätin siis tehdä toteutuksen, joka toimii, mutta rajaa valitettavasti käyttäjän mahdollista piirtoaluetta. Ongelman voisi varmasti korjata perehtymällä paremmin siihen, miten QGraphicsView ja QGraphicsScene toimivat yhdessä ja muuntelemalla vaikka koordinaatteja tarpeen mukaan.

Koodin kannalta mainittavaa on piirto- ja tekstinlisäysominaisuuksien epäoptimaalinen toteutus. Kuten algoritmien kohdalla on mainittu, kuvioiden piirtäminen toteutetaan edellisen kuvion päivittämisen sijaan piirtämällä kokonaan uusi kuvio ja poistamalla edellinen. Ohjelmaa käytettäessä ominaisuus näyttää toimivan ihan hyvin, mutta toteutus ei ole siisti. Kuvioita voisi oikeasti päivittää itemeillä olevalla metodilla update(), ja tämä toteutus veisi vähemmän rivejäkin. Jos jatkaisin projektia ja minulla olisi aikaa perehtyä myös QGraphicsViewin päivittämiseen, toteuttaisin kuvioiden piirtämisen päivittämällä.

Tekstinlisäysominaisuus on ylimääräinen ominaisuus, mutta mielestäni oleellinen. Se on kuitenkin toteutettu monimutkaisesti, eikä se ole käyttäjällekään intuitiivinen. Kun tekstin piirto on valittuna, luodaan koodissa heti uusi tekstiolio, johon päivitetään tekstiä käyttäjän inputin mukaan. Klikattuun kohtaan ei kuitenkaan piirretä minkäänlaista indikaattoria, joten käyttäjä ei välttämättä tajua alkaa kirjoittaa ja saattaa klikata uudesta kohdasta, jolloin lisätään aivan uusi teksti-olio. Ongelmaa helpottamaan ohjelma kertoo info-tekstissä, miten tulisi toimia. Jos jatkaisin projektia, korjaisin puutteen lisäämällä jonkinlaisen indikaattorin ja optimoimalla tekstin lisäystoiminnallisuutta.

Vielä yksi mahdollinen vika on se, että jos käyttäjä haluaa kirjoittaa tekstiä, joka sisältää merkkejä kuten ”\’,” tai ”;”, joita käytetään ladattavan tiedoston jakamiseen osiin, ohjelma ei ladattaessa tunnista tiedostoa oikeanlaiseksi tiedostoksi vaikka se olisikin tallennettu aivan oikein, koska se jakaa tuollaisten merkkien takia tiedostossa olevan tekstin vääränlaisiin osiin eikä kykene sen takia luomaan kuvaa uudestaan. Ongelman voisi korjata käsittelemällä tekstiä ennen tallennusta.

Haitta, joka on tullut testauksessa vain pari kertaa vastaan, mutta on mahdollinen, on se, että erilaisia viivan tyylejä käytettäessä piirretyn viivan siirtely edes takas jättää jälkeensä värivanan. En ole varma, mistä tämä johtuu, koska vana ei ole mikään erillinen olio tai kuvio. Vana ei tule mukaan tallennuksessa eikä kuvan viemisessä, mutta piirrettäessä se jää piirtoalueelle näkyviin.

10. 3 parasta ja 3 heikointa kohtaa

2 parasta

Yksi hyvistä puolista ohjelmassani on se, että siihen pystytään tuomaan kuvatiedosto, jonka päälle pystyy piirtämään. Tämä on hieno ominaisuus, koska se mahdollistaa jonkin tasoista ”kuvan muokkausta”. Esimerkki ohjelmalla ”muokatusta” ja exportatusta kuvasta löytyy projektin kansiosta ”examples”.

Toinen hyvä puoli ohjelmassa on kuvioiden siirtäminen. Vaikka kuviota ei voikaan siirtää pois toisen alta, kuten puutteissa on kuvailtu, on tärkeää, että kuvioita pystytään siirtämään jos sen piirtäminen oikeaan kohtaan ei onnistunutkaan.

2 heikointa

Ehdottomasti heikoin kohta ohjelmassa on piirtämisen toteutus. Kuten aiemmin on jo selitetty, toteutus on epäoptimaalinen ja sen voisi toteuttaa paljon kätevämmin.

Toinen ikävä heikkous ohjelmassa on se, että kun ruudun laittaa suureksi, infotekstin osa splitteristä pysyy tarpeettoman suurena eikä anna tilaa piirustusalueelle. Tämän takia suurennetulla ruudullakaan ei saada kovin korkeita kuvia aikaan.

11. Poikkeamat suunnitelmasta

Poikkesin ohjelman toteutuksessa paljon suunnitelmasta. Suunnitelman tekovaiheessa en ollut vielä perehtynyt PyQt:een, joten olin ajatellut toteutuksen koodin kannalta hyvinkin erilaiseksi, koska en tiennyt vielä silloin, mitä kaikkea käytettävä kirjasto sisältää. Myös suunniteltu ajankäyttö ja sen jakaminen koko projektinteko ajalle poikkesi suunnitelmasta, koska sain ajanpuutteen vuoksi aloitettua projektin tekemisen kunnolla vasta viikkoa ennen deadlinea. Lisäksi käytin projektin tekemiseen noin puolet suunnittelemastani ajankäytöstä. Pääpiirteittäisestä työjärjestyksestä en kuitenkaan juuri poikennut.

12. Toteutunut työjärjestys ja aikataulu

Projektin toteutus seurasi suurin piiretin suunnitelman järjestystä.

31.3. (2h)

1. Loin alustavan graafisen käyttöliittymän, jossa ei ollut toiminnallisuuksia.

10.4. (2,5h)

1. Parantelin graafista käyttöliittymää ja lisäsin Canvas-luokan sekä Buttons-luokan

21.4. (1h)

1. Yritin saada GUI:ta ja Canvas:ia toimimaan yhdessä

30.4. (12,5h)

1. Muutin Button-luokan pelkäksi moduuliksi ja sain nappuloihin toiminnallisuutta.
2. Lisäsin projektiin main.py, josta ohjelma käynnistetään
3. Lisäsin piirto-ominaisuuden, joka ei toiminut vielä halutulla tavalla
4. Parantelin piirto-ominaisuutta toimimaan kuten sen kuuluukin

5. Toteutin undo-toiminnallisuuden sekä uuden piirustuksen luomisen

6. Aloitin tallennuksen toteutusta

1.5. (12,5h)

1. Toteutin tallennuksen loppuun

2. Toteutin lataus-ominaisuuden

3. Lisäsin kuvioiden värin muuttamismahdollisuuden

4. Toteutin kuvatiedostojen tuomisen ja viemisen (export, import)

2.5.(8h)

1. Muutin GUI:ta uusien ominaisuuksien lisäämiseksi, lisäsin menuBarin

2. Lisäsin kuvioihin täyttömahdollisuuden sekä kynän ominaisuuksien muuttamisen

3. Lisäsin tekstin lisäysmahdollisuuden

3.5. (10h)

1. Lisäsin piirretyt tekstit tallennukseen ja lataamiseen

2. Lisäsin mahdollisuuden valita ja siirrellä kuvioita

3. Muutin undo-toimintoa

4. Lisäsin mahdollisuuden muuttaa piirtoalueen kokoa

5. Muutin GUI:ta nätimmäksi

4.5.-5.5.(6h + 10h)

1. Parantelin, siistin ja kommentoin koodia

2. Korjasin pari käytettävyysongelmaa

3. Kirjoitin dokumentin

6.5. (6h)

1. Korjasin vielä pari puutetta latauksesta sekä kuvioiden siirtelystä.

13. Arvio lopputuloksesta

Lopullinen ohjelma on mielestäni käyttäjälle hieman epäintuitiivinen, toimiva piirustusohjelma, josta löytyy käytettävyyden kannalta oleellisimmat toiminnot piirtämiseen. Ohjelmasta löytyy puutteita ja heikkouksia, jotka toki saisi korjattua ajan kanssa, mutta se kuitenkin täyttää vaatimukset, eikä sen pitäisi kaatua muissa tilanteissa kuin joskus exporttaamisen jälkeen.

Toki ohjelmassa on hyviäkin puolia: sillä pystyy piirtämään monipuolisia kuvia monipuolisen värivalikoiman ja kynäasetusten vuoksi. Vaikka ohjelmassa ei olekaan maalipurkkia, voi sillä silti piirtää täytettyjä kuvioita. Myös ajatuskarttojen ja kaavioiden piirtäminen onnistuu, koska tekstiä ja viivoja voi lisätä ja tarvittaessa siirrellä. Kuvien päälle piirtäminenkin onnistuu jos sille on tarvetta. Tällaiset kuvat vain täytyy exportata, koska niiden tallennusta tekstipohjaiseen muotoon ei ole toteutettu.

Jos jatkaisin projektia, muuttaisin sen varmaan ensin muotoon, jossa käytetään PyQt4 kirjastoa, jotta exportin saisi toteutettua hienosti. Muidenkin ominaisuuksien pitäisi siinä toimia. Seuraavaksi korjaisin tunnetut viat ja puutteet. Sen lisäksi uusia piirto-ominaisuuksia voisi lisätä helpommin. Muutosten ja laajennuksen pitäisi olla mahdollista, erityisesti jos tuntee koodin. Tällä hetkellä esimerkiksi tallennus ja lataus ovat kyllä toimivia, mutta hyvin monimutkaisesti toteutettuja. Nämä voitaisiin mahdollisesti hoitaa yksinkertaisemminkin, jos tuntisi käsiteltävien objektien metodeja paremmin. Uusien ominaisuuksien lisääminen tallennukseen ja lataukseen on nytkin mahdollista, mutta lisäyksen jälkeen vanhoja tiedostoja ei enää pystytä lataamaan ohjelmalla.

14. Viitteet

Projektin alkuvaiheessa käytin alkuun päästäkseni tutoriaaleja:

http://zetcode.com/gui/pyqt5/

Projektin edetessä, kun tarvitsin ratkaisuja ongelmiin tai esimerkkejä toteutukseen, apua löytyi seuraavilta sivustoilta:

http://pyqt.sourceforge.net/Docs/PyQt4/index.html

http://stackoverflow.com/

http://www.qtcentre.org/content/

http://doc.qt.io/qt-5/

15. Liitteet

Lähdekoodi löytyy projektikansion src-kansiosta. Lisäksi projektikansiossa on examples-niminen kansio, josta löytyy esimerkkitallennus, kuvia ohjelmasta sekä tiedostoja joilla voi testata ohjelmaa.