

Numeerisen datan visualisointikirjasto

Projektidokumentti – Ohjelmoinnin peruskurssi Y2

Akseli Oinaanoja 656551 (BIO 2.VSK)

akseli.oinaanoja@aalto.fi

22.4.2019

Sisällysluettelo

2.Yleiskuvaus.....1

 Kirjoita luvun otsikko (taso 2)2

 Kirjoita luvun otsikko (taso 3).....3

3. Käyttöohje.....4

 Kirjoita luvun otsikko (taso 2)5

 Kirjoita luvun otsikko (taso 3).....6

2. Yleiskuvaus

Projektin tavoitteena oli rakentaa ohjelmisto, joka pystyy visualisoimaan numeerista dataa luomalla erilaisia graafeja. Ohjelmalla voi visualisoida viivadiagrammeja plot-tyyppisesti, eli ohjelma yhdistää käyttäjän antamat datapisteet viivoin. Datan käyttäjä syöttää ohjelmalle antamalla csv-tiedoston nimen, jossa data on pilkuin eroteltuna ja visualisointitavan valinta "plot;" ensimmäisenä sanana. Ohjelma käsittelee raakadatan ja tallentaa sen erilaisten luokkien tietoihin ja rakentaa tämän perusteella akselien skaalauksen, sekä jakovälien paikat ja niistä piirretyt ruudukkoviivat, jotka helpottavat datan lukemista kuvaajalta. Ohjelmassa on käytetty punaista väriä yhdistämään datapisteet viivalla ja mustaa väriä datapisteiden värinä. Värien selitteet näkyvät kuvaajan oikeassa yläreunassa. Selitteet sekä ruudukon voi käyttäjä halutessaan kytkeä päälle tai pois päältä.

Ohjelma rakentaa datan kuvaajan PyQt5:n QMainWindowiin, johon on lisätty yläreunaan käyttäjälle valintapalkkeja, josta käyttäjä saa asetettua kuvaajalle esimerkiksi akselien nimet. Ruudukon ja selitteiden kytkemisen lisäksi käyttäjä voi myös halutessaan tallentaa kuvaajan kuvatiedostoksi, poistua kuvaikkunasta tai piilottaa valintapalkin näin halutessaan.

Alkuperäisestä suunnitelmasta poiketen, projekti toteutettiin yksinkertaisemmalla luokkarakenteella ja muutamaa lisäominaisuutta korvattiin toisilla. Suurin muutos oli päätös rakentaa kuvaajat suoraan QMainWindowiin luettujen datapisteiden perusteella, sen sijaan, että luotaisiin QGraphicsItem ja lisättäisiin se QGraphicsSceneen. Halusin muuttaa tätä, koska se on mielestäni tehokkaampi ja suoraviivaisempi tapa käsitellä dataa. Työ on lopulta mielestäni toteutettu keskivaikkealla vaikeustasolla, koska työ vastaa lähes täysin keskivaikkeen työn vaatimuksiin ja sisältää muutaman lisäominaisuuden.

3. Käyttöohje

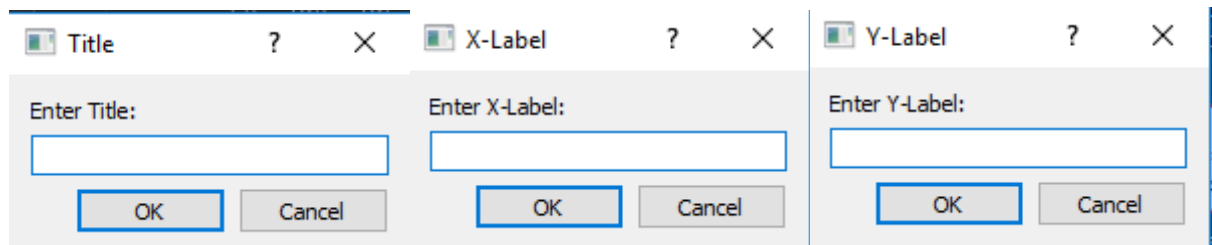
Ohjelma voidaan käynnistää komentoriviltä ajamalla main.py-tiedosto. Ohjelma luo PyQt applikaation ja kysyy käyttäjältä tiedostonimen, josta tiedot haetaan. Tiedoston oletetaan olevan ajettavan ohjelmiston kanssa samassa hakemistossa. Data käsitellään guireader.py-tiedostossa ja siellä luodaan datapisteistä Graph-olio, joka palautetaan GUI.py-tiedostoon, joka rakentaa olion pohjalta visualisoinnin. Käyttäjältä kysytään vielä ennen kuvaikkunan muodostumista tietoja kuvaikkunan otsikoksi sekä akselien selitteille.

Ohjelma on rakennettu siten, että kuvaajan osat (selitteet, koordinaatit, ruudukko) skaalautuvat ikkunan koon mukaan. Käyttäjä voi halutessaan muokata paljolti kuvaajaa yläpalkin valinnoista. Valintaikkunasta File aukeaa mahdollisuus sulkea tiedosto, valinnasta Action voi halutessaan kytkeä ruudukon, selitteet tai antaa akseleille nimet (Grid, Legend, Labels), valinnasta Save käyttäjä voi tallentaa kuvatiedostona kuvaajan (.png-tiedostona). Valintaikkunasta Hide voi piilottaa yläpalkin.

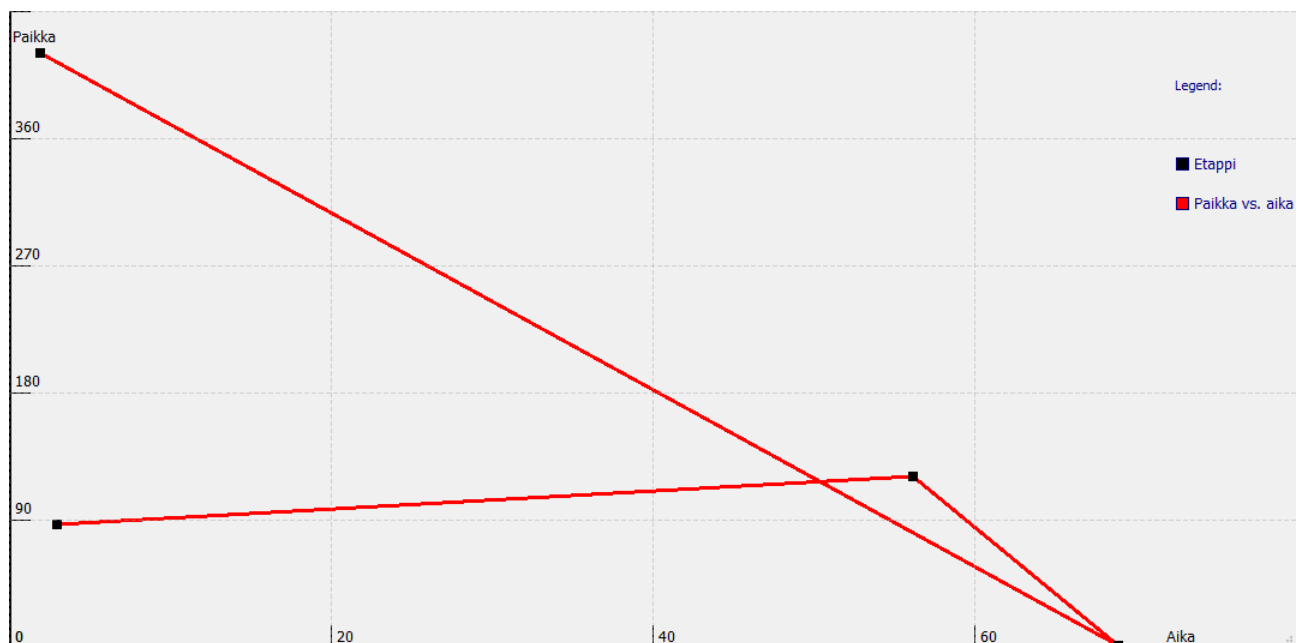
Jokaiseen edellä mainittuun toimintoon on myös lisätty pikanäppäin, esimerkiksi Save toiminnon voi myös suorittaa painamalla Ctrl+S. Pikanäppäimet lukevat valikossa painikkeen vieressä.

Ohjelmaa ajettaessa avautuu seuraavanlainen ikkuna, johon tekstikenttään voi syöttää haluamansa otsikon kuvaajalle.

Vastaavanlainen ikkuna avautuu vielä kahdesti, pyytäen käyttäjää syöttämään x- ja y-akselien selitteet. Nämä voi myös ohittaa painamalla OK, jolloin kuvaajan otsikoksi tulee default-asetuksena python ja akseleille ei selitteitä.



The image shows three separate dialog boxes for configuring a plot. Each box has a title bar with a green icon, a question mark, and a close button (X). The first box is titled 'Title' and contains a text input field labeled 'Enter Title:' with 'OK' and 'Cancel' buttons. The second box is titled 'X-Label' and contains a text input field labeled 'Enter X-Label:' with 'OK' and 'Cancel' buttons. The third box is titled 'Y-Label' and contains a text input field labeled 'Enter Y-Label:' with 'OK' and 'Cancel' buttons.



Esimerkki mielenkiintoisesta tilanteesta, jossa mitataan kappaleen paikkaa ajan suhteen ohjelman avulla. Kuvan kaikki selitteet on luotu valintapalkin valinnoilla ja kuva otettu Save-toiminnolla.

4. Ulkoiset kirjastot

Ohjelma hyödyntää PyQt5-kirjaston QtCore:a, QtGui:a, sekä QtWidgets:ia. Lisäksi ohjelmaan tuodaan Pythonin sisältämä math-kirjasto ainoastaan akselien jakovälien pyöristämiseen kymmenien tarkkuudella (ceil-komento).

5. Ohjelman rakenne

Ohjelma on keskeisesti kolmiosainen. Kolme tiedostoa, jotka vastaavat ohjelman toiminnasta ovat main.py, guireader.py, sekä GUI.py. Main.py luo ohjelman applikaation, eli on ajamisen

kannalta tärkeä, mutta ei juuri muuten osallistu keskeisiin toiminnallisuuksiin. Main.py luo PyQt applikaation sekä GUI-luokan, joka saa parametrinaan tiedostonimen, josta lukea tiedot.

5.1 GUI

GUI-luokka perii PyQt5:n QMainWindow-luokan, johon kuvaaja rakentuu. Valitsin QMainWindow-luokan kuvaajan pohjaksi, koska se on mielestäni yksinkertaisin ja tehokkain tapa piirtää kuvaaja (vaihtoehtona olisi ollut myös esim. QGraphicsScene). GUI luo alustusmetodissaan GUIreader-olion, joka poimii annetusta tiedostosta datan ja palauttaa oleelliset tiedot sisältävän Graph-olion GUI:lle. (GUIreader palauttaisi GraphCollection-olion, joka voi sisältää useamman Graph-olion, jos ohjelma visualisoisi dataa muutenkin kuin viivadiagrammein.)

GUI-luokassa on useita attribuutteja, joista moni määritellään vasta myöhemmissä metodeissa. GUI-luokan paintEvent-metodi, joka yliajaa QMainWindowin oman paintEventin, on keskeisin metodi, joka päivittyy useissa eri tilanteissa (esimerkiksi ikkunaa venytettäessä) ja joka kutsuu kaikki keskeisimmät piirtometodit.

GUI luokassa on esimerkiksi seuraavanlaisia metodeja:

Initialize_ui: Alustaa kuvaikkunan luomisen

Tässä määritellään muutama attribuutti, esim.

Self.grid: kuvastaa ja säätelee akselien jakovälien määrää ja täten ruudukon resoluutiota. Oletuksena arvo 5.

Self.filename: GUI- luokkaa kutsuttaessa annettu parametri, tiedostonimi, josta data luetaan.

paintEvent: Tässä metodissa tapahtuvat asiat toistuvat useita kertoja (päivittyvät), joten tämä sisältää kaikista keskeisimmät piirtometodit, kuten

draw_lines: piirtää punaiset viivat datapisteiden välille sekä datapisteet mustina pisteinä.

draw_axis: piirtää pieniä viivoja merkitsemään jakovälejä x- ja y-akseleilla

draw_grid: piirtää apuruudukon taustalle, jos self.gridO = True

draw_legend: piirtää selitteet graafeille, pisteille ja viivoille, jos self.legend = True

(myös muille graafeille, jos kyseessä olisi esim. pylväs- tai ympyrädiagrammeja. Nämä olisivat tallennettuna self.legends listaan, joka täyttyisi GraphCollection-olion sisältämästä Graph-olioiden datasta.)

`find_axis_scales`: laskee datasta suurimman ja pienimmän arvon ja skaalaa näiden tietojen mukaan akselien jakovälit (pyöristäen kymmenen tarkkuudelle).

Tässä määritellään `self.x_bin` (jakovälit) ja `self.x_range` (lukuväli, jolla data-arvot sijaitsevat) attribuutit ja vastaavat y-arvoille.

`build_menu`: rakentaa kuvaajaikkunan yläreunaan valintapalkin, jossa erilaisia painikkeita ja toimintoja käyttäjälle. Pohjana toimii attribuutti `self.menubar`, joka on `QMenuBar`-luokan instanssi, sekä `QAction`-luokan painikkeet, jotka ilmenevät `QIcon`-luokan avulla. Yhdistää painikkeet pikanäppäimiin sekä funktioihin, jotka vastaavat painikkeiden toimivuudesta.

`toggle_grid`: Grid valikon Grid-painike on yhdistetty tähän metodiin, joka muuttaa `self.gridO:n` boolean arvon (`True/False`) vastakkaiseksi, kuin mikä se painamishetkellä oli.

`set_legend`: toimii samoin kuin `toggle_grid`, mutta attribuutille `self.legend`

`toggle_menu`: poistaa yläpalkin näkyvistä, mikäli käyttäjä näin haluaa.

`save_image`: Avaa tiedostohakemiston ja tallentaa kuvaajan `QPixmap`-luokan `grab`-metodilla sekä tallentaa kuvan .png-muodossa `QFileDialog`illa osoitettuun hakemistoon käyttäjän antamalla nimellä. Metodi kutsuu ensin `toggle_menu` funktiota, joten yläpalkki ei näy kuvassa.

`label_check`: Kysyy käyttäjältä uudet selitteet x- ja y-akseleille ja päivittää kuvaikkunan.

5.2 GUIreader

GUIreader avaa annetun tiedoston ja lukee sen ensimmäisen sanan, joka on erotettuna puolipisteellä sitä seuraavasta csv-datasta, eli pilkuin erotelluista arvoista. Tällä hetkellä GUIreader tunnistaa kolme sanaa, "plot", "column" ja "pie", mutta suorittaa vain "plot"-alkuisen tiedoston datan lukemisen. Datan lukeminen ei ole merkittävästi erilaista column tai pie- graafin tapauksissa, mutta näitä ei projektiin ehditty implementoida. Ohjelmaan ei kuitenkaan olisi vaikeaa lisätä näitä rakenteita, sillä siinä on valmiina Point-luokka, johon voi asettaa x-attribuutiksi merkkijonon numeerisen arvon sijaan. Merkkijonon voisi yhtä hyvin lukea kuin numeronkin csv-tiedostossa, kunhan pilkku vain erottaa näitä. GraphCollection-luokkaan voisi esimerkiksi lisätä jokaisen eri Point-luokan instanssin ja palauttaa GraphCollection-luokka GUI:lle Graph-luokan sijaan.

Datan lukeminen tapahtuu rivi kerrallaan, kunnes saavutaan tyhjälle riville. Tämän ehdon takia tiedostoa rakennettaessa rivivälin käyttö ei ole aivan vapaata, mutta sen sijaan välilyöntejä tai muuta whitespace tietoa saa tiedostossa olla vapaasti, ohjelma parsii nämä pois. Tiedosto myös olettaa, että tunnistinsana (esim. "plot") löytyy ensimmäiseltä riviltä tiedostoa. Ohjelmaa rakennettaessa vähemmälle huomiolle jäi mahdolliset poikkeavuudet csv-tiedoston rakenteessa. Mielestäni tämä on kuitenkin hyväksyttävää, sillä ohjelmaa käytettäessä voisi olettaa käyttäjän ymmärtävän jotain sen käytöstä.

6. Algoritmit

6.1

Ohjelmassa tarvittiin x- ja y-akselien skaalauksessa sekä erilaisten viivojen tai graafien piirtämisessä algoritmeja. Akselien skaalauksessa `find_axis_scales` -metodi tekee suurimman työn. Tämä käy läpi for-loopilla datasetin ja etsii x- ja y-arvojen minimi- ja maksimiarvot. Näistä saadaan lukuväli, jossa arvot sijaitsevat. Maksimiarvo pyöristetään math-kirjaston `ceil`-funktion avulla lähimpään kymmenlukuun luettavuuden helpottamiseksi, saadaan attribuutit `x_range` ja `y_range`. Akselien jakovälit saadaan jakamalla esimerkiksi `x_range` `self.gridin` arvolla ja kertomalla aina kukin saatu arvo sen osuudella. Tämä on toteutettu siten, että akselien leikkauspiste on aina kuitenkin origossa.

Python-koodina yhden jakovälin lauseke saa esimerkiksi seuraavanlaisen muodon:

```
k * int(math.ceil((self.y_range / self.grid) / 10.0) * 10)
```

6.2

Viivadiagrammi

Viivat, jotka yhdistävät annettuja datapisteitä piirretään PyQt:n `drawLine`-komennolla, joka tarvitsee kahden pisteen väliset koordinaatit. Tässä täytyy ottaa huomioon se, että `QMainWindow`:n koordinaatisto on asetettu siten, että sen origo on käyttäjän näkökulmasta vasemmassa yläreunassa ja oikean alareunan koordinaatit voidaan ilmaista esim. `width()` ja `height()` metodeilla. Tämänlaisen skaalauksen etuna on se, että ikkunaa venytettäessä tai kokoa muuttaessa `paintEvent` päivittyy ja myöskin siis kaikki viivat skaalautuvat ikkunan mukana, eikä tarvitse tyytyä vain tiettyyn ikkunan kokoon.

7. Tietorakenteet

Tiedoston lukemisen suorittava luokka `GUIreader`, muodostaa ohjelman tietorakenteet. Tietorakenteita on jo selvennetty hieman aiemmissa kappaleissa, erityisesti luokkarakenteista puhuttaessa. Kokonaisuudessaan ohjelman tietorakenteet ovat yksinkertaisia, ohjelmassa käytetään muuttuvia listoja olioiden attribuutteina. Ainoastaan kuvatiedosto on muuttumaton tietorakenne, jos sellainen luodaan ohjelman suorituksen aikana.

Ohjelman pääpiirteisenä ideana on ylin luokka, Graph-olio. Tämä kuvastaa siis mitä tahansa datan visualisointia kuvaikkunassa ja täten sisältää kaiken numeerisen datan, jota ohjelma käyttää, tallennettuna dynaamiseen listaan. Tämä Graph-olio on tallennettuna GraphCollection-olion lista-attribuuttiin. GraphCollection toimii siis eräänlaisena kokoajana. Tätä luokkaa en lopullisessa versiossa käyttänyt, koska viivadiagrammien piirtämiseen riitti varsin hyvin yksi Graph-olio visualisointia kohden, mutta se on olemassa laajennettavuutta varten.

Kokonaisuudessaan tietorakenne mahdollistaa datan siirrettävyyden tiedostosta toiseen ja sen helpon lukemisen. Dataparit pysyvät kiinni toisissaan järjestyksessä ja jokaisen kuvaajan muodostaminen onnistuu erikseen. Tämä mahdollistaa myös virheiden löytämisen ja poistamisen, sillä yksittäinen piste tai pistejono on helppo jättää piirtämisen ulkopuolelle. Myös yksittäisen pisteen tai jonon poistaminen on yksinkertaista. Tietorakenne on hyvin sovellettavissa ja laajennettavissa, mikäli käyttäjä niin haluaa.

8. Tiedostot

Ohjelma lukee tavallista csv-tiedostoa, kuitenkin sillä poikkeuksella, että ensimmäisellä rivillä sijaitsevan ensimmäisen sanan tulisi olla tunnistesana "plot", "pie" tai "column" ja sanan perään kuuluu puolipiste ";". Tämän jälkeen tiedostolle voi syöttää haluamiaan datapisteitä, vuorotellen x- ja y-koordinaatteja, alkaen ensimmäisestä x-koordinaatista.

Tiedostoon voi syöttää merkkijonodataa, mutta tämä tarkistuu "plot" graafin tapauksessa ja aiheuttaa errorin. Ohjelmalle voi syöttää niin negatiivisia kuin positiivisiakin datapisteitä, mutta ohjelma on suunniteltu piirtämään vain positiivisia arvoja, eli käytössä on vain karteesisen koordinaatiston I neljännes. Lukuarvojen ei kuitenkaan tarvitse olla kokonaislukuja.

Tällä hetkellä ohjelma muuttaa kaikki negatiiviset arvot itseisarvoikseen, jotta kuvaaja näyttäisi järkevältä.

9. Testaus

Ohjelmaa testattiin antamalla mielivaltaista dataa luettavaksi ja tulostamalla nämä sekä niistä luodut lukuvälit. Kuten aiemmin mainittu, ohjelman syötetiedoston rakenteen testaaminen jäi vähäiseksi sopivien käyttäjää koskevien oletusten vallitessa.

Testaus osoittautui varsin helpoksi tehdä työn edetessä, koska koodin tuloksen näki usein suoraan piirrettyinä ikkunassa. Testaus sujui suunnilleen niin kuin odotinkin ja suunnittelin, vaikka syötetiedoston testaus jäikin vähäisemmälle.

10. Ohjelman tunnetut puutteet ja viat

Ohjelman suurin puute on negatiivisten arvojen mallintaminen, toisin sanoen koordinaatiston laajentaminen. Tämä olisi ollut melko helppo toteuttaa esimerkiksi jakamalla ikkuna puolivälin

kohdalta selvästi x- ja y-akseleihin. Jakovälit olisi myöskin tullut todella luontevasti nykyisen algoritmin kanssa, ainoa muutos olisi ollut luoda jakovälit negatiiviselle puolelle, eli käyttää nykyisen kaavan sijaan skaalaa mittaavan arvon vastalukua sekä negatiivisia kertoimia. Itse datapisteiden piirtäminen ja niitä yhdistävien viivojen piirtämisessä olisi ollut suurin ajatustyö. Tämäkin ongelma olisi ratkennut tosin vain muuttamalla hieman viivan piirtämisen algoritmia. Vaikka QMainWindow toimiikin positiivisessa koordinaatistoneljänneksessä, ei se tarkoita sitä, etten voisi keinotekoisesti jakaa tätä ikkunaa haluamallani tavalla. Tässä olisi vain tarvittu voitu vain jakaa ikkunan leveys ja korkeus puoliksi ja sopia se yhteiseksi origoksi.

Toinen puute ohjelmassa on marginaalitila koordinaatiston ympärillä. Tämän varaaminen olisi ollut järkevää, jottei esimerkiksi selitteet, valintapalkit tai jakovälien numerot tulisi graafien kanssa päällekkäin. Ohjelmassa saattaa törmätä tällä hetkellä hieman epämukaviin tilanteisiin, joissa datapiste on esimerkiksi yläpalkin alla peitossa. Pyrin kiertämään tätä ongelmaa lisäämällä mahdollisuuden piilottaa yläpalkki, jolloin kaikki piirretyt pisteet löytyvät.

Kolmanneksi, toiseen puutteeseen liittyen, ohjelmaa käyttäessä voi päätyä tilanteeseen, jossa selitteet eivät näy kokonaan, koska niille ei ole varattu tarpeeksi tilaa. Tätä ongelmaa onkin hieman monimutkaisempaa ratkaista, sillä mahdollisesti varattu marginaalikaan ei riittäisi pitkille kuvaajaselitteille. Ongelmaa lieventää se, että kuvaikkunaa voi venyttää ja asettaa vaikka koko ruudun kokoiseksi, jolloin teksti todennäköisemmin mahtuu kokonaisuudessaan paikalleen.

Yksi parhaista ominaisuuksista ohjelmassani on sen helppokäyttöisyys. Vain hetken tutustuttuaan syötetiedoston rakenteeseen käyttäjä luultavasti osaa toimia ohjelman kanssa melko intuitiivisesti. Kaikkea mahdollista, mitä käyttäjä voi tehdä, kysytään selvästi erikseen avautuvissa ikkunoissa.

Toinen parhaista ominaisuuksista ohjelmassa on sen yksinkertainen rakenne. Olen tyytyväinen valintaani QMainWindow-pohjaiselle ratkaisulle, koska uskon, että se on todella tehokas ja elegantti. Graafiin liittyvä data siirretään suoraan olioiden välillä dynaamisena listana, eikä jokaisesta pisteestä tarvitse välttämättä tehdä omaa luokkaa.

Kolmas parhaista ominaisuuksista on tietorakenteiden yksiselitteisyys. Ohjelmassa ei luoda turhia tietorakenteita tai luokkia, vaan todella pitkälle pärjätään samalla luokalla ja usealla metodilla. Tämä mahdollisesti tekee luettavuudesta hieman hankalampaa, mutta toteutuksesta suoraviivaisempaa.

12. Poikkeamat suunnitelmasta

Luokkajako päättyi olemaan paljon yksinkertaisempi kuin suunnittelin. Ajankäyttö ei toiminut parhaalla mahdollisella tavalla muun työmäärän ja vapaa-ajan puutteen vuoksi. Koodin toteutusjärjestys noudatti täysin suunnitelman mukaista.

13. Toteutunut työjärjestys ja aikataulu

Projektin alku oli hidasta, sain Checkpointtiin mennessä juuri ja juuri luotua datankäsittelyluokan guireaderin valmiiksi. Oletin, että tähän kuluisi paljon vähemmän aikaa. Suurimman osan työstä olen tehnyt huhtikuun ensimmäisen ja toisen viikon aikana. Tällöin sain rakennettua järkevän kuvaikkunan, johon aloin lisäilemään käyttäjäystävällisyyttä sekä miettimään pisteiden ja viivojen laskenta-algoritmeja. Suunnitelmasta poikettiin siinä määrin, että työ pääsi vauhtiin vasta checkpointin jälkeen.

14. Arvio lopputuloksesta

Ohjelman parasta kolmea ja huonointa kolmea kohtaa kuvatessa tuli kaikkein parhaiten ja selviten ilmi asiat, jotka vaikuttavat lopputuloksen arvioon. Olen kokonaisuudessaan tyytyväinen, sillä vaikka lopputulos jäikin suunnitellusta vajaaksi, on se todella hyvällä pohjalla ja siihen on helppo lisätä uusia ominaisuuksia tai laajentaa sitä esimerkiksi vaikean vaikeustason vaatimuksille. Ohjelmaa voisi tulevaisuudessa parantaa yksinkertaisesti ottamalla huomioon paremmin asioita, joihin törmää luultavasti tulevaisuudessa ohjelman kehittyessä. Ohjelman koodi sisältää melko paljon kommentoituja osia tai kokonaisia metodeja, jotka jouduin jättämään lopullisesta versiosta pois bugien tai tuntemattomien ongelmien takia.

15. Viitteet

Olen käyttänyt paljon PyQt tutoriaaleja, joita löysin netistä, joitain opetusvideoita Youtubesta ja melko paljon stackoverflow:n keskustelufoorumeja.

Käytin esimerkiksi seuraavia lähteitä:

<https://pythonspot.com>

<https://wiki.python.org/moin/PyQt/Tutorials>

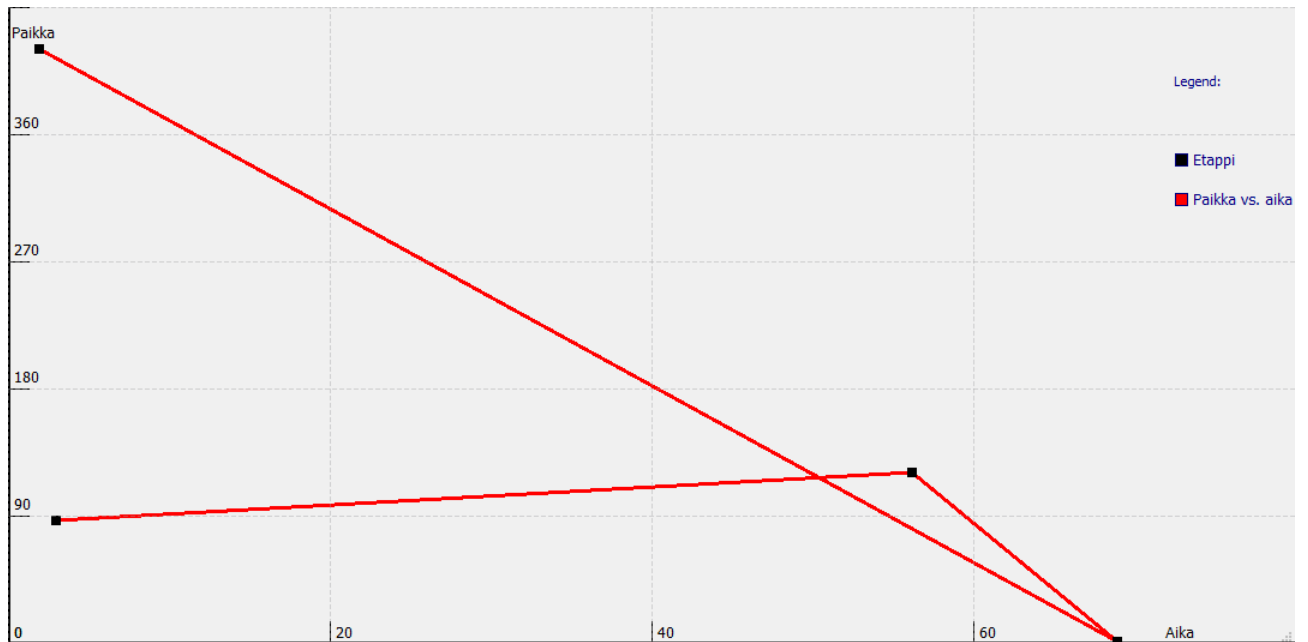
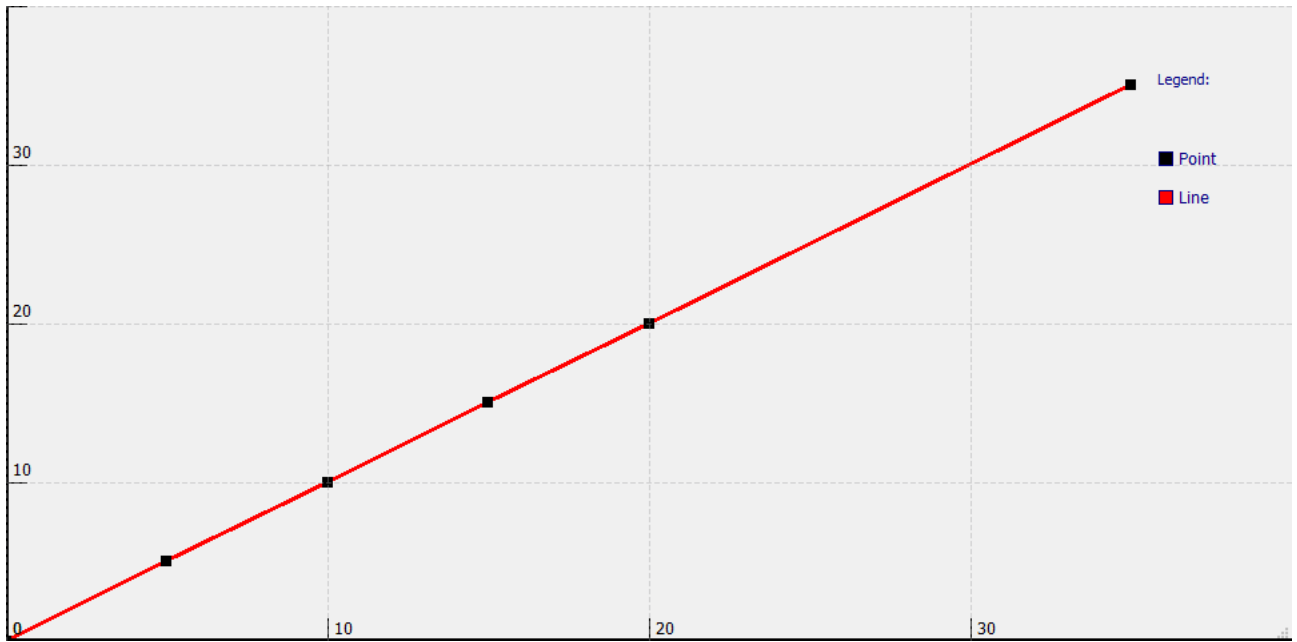
<https://www.tutorialspoint.com/pyqt/>

<https://stackoverflow.com>

<https://sourceforge.net/projects/pyqt/>

16. Liitteet

Alla esimerkkiajo muutaman datapisteen yhdistämisestä tapauksesta sekä tavallisen lineaarisen suoran pätkä.



Esimerkki mielenkiintoisesta tilanteesta, jossa mitataan kappaleen paikkaa ajan suhteen ohjelman avulla. Kuvan kaikki selitteet on luotu valintapalkin valinnoilla ja kuva otettu Save-toiminnolla.