NUMEERISEN DATAN VISUALISOINTIKIRJASTO

Projektityön dokumentti

Tuomas Manninen 480109 (EST 2. vk)

6.5.2016

Yleiskuvaus

Ohjelmalla voidaan visualisoida kaksiulotteista numeerista dataa .csv tiedostosta luettujen koordinaattien perusteella viivadiagrammien, pylväsdiagrammien tai piirakkadiagrammin avulla. Viiva- ja pylväsdiagrammeja voi olla kuvaajassa ja tiedostossa mielivaltainen määrä.

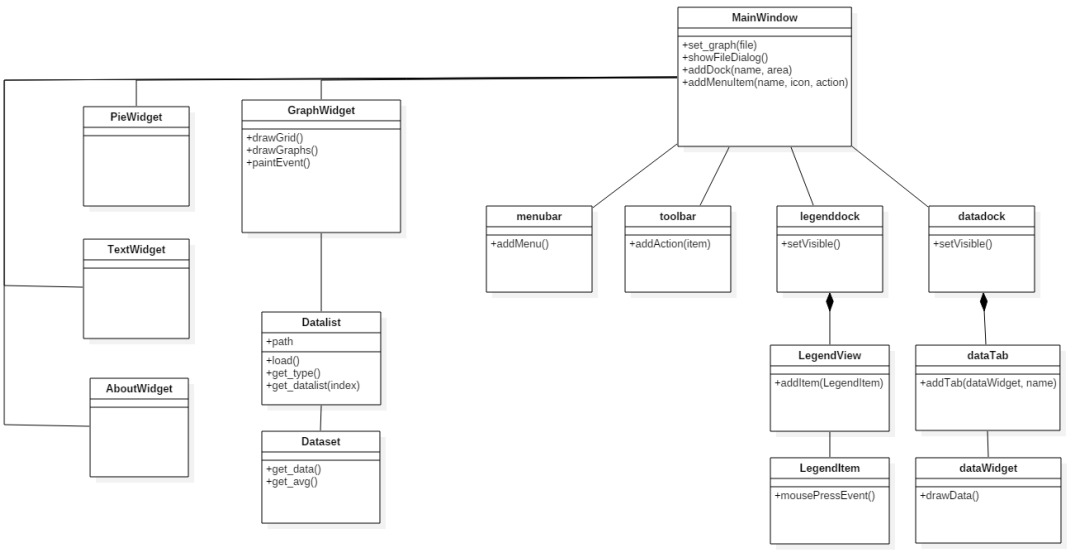
Käyttäjällä on mahdollisuus ladata kuvaajatiedostoja, navigoida kuvaajassa, muuttaa otsikoita ja vaihtaa asetuksia.

Käyttöohje

Ohjelma käynnistetään suorittamalla main.py tiedosto pääkansiosta, josta ohjelman käyttöliittymä käynnistyy puhtaalta pöydältä.
Kuvaajan voi ladata ”File” valikosta ”Load file” valinnalla, joka avaa tiedostovalintaikkunan. Täällä voidaan etsiä ja avata haluttu .csv tiedosto. Ohjelma määrittää tiedon sisällön automaattisesti ja latautuu pääikkunaan kaikkine datasta saatavine oheistietoineen.
Valikkopalkista voidaan sulkea ohjelma, sekä muuttaa akseleiden nimet ”X-Title” ja ”Y-Title” painikkeilla, sekä ottaa vaaka-ja pystyruudukko pois ”X-Grid” ja ”Y-Grid” painikkeilla.
Window-valikosta voidaan piilottaa Data ja Legend telakat.
Käyttöliittymän hiirikontrollit löytyvät About>Help valikosta.

Ohjelman rakenne

Ohjelman data kuuluu Datalist-luokkaan, joka sisältää kaikkien kuvaajien Dataset-luokan instanssit. Datalist-luokasta yksittäisten kuvaajien instanssit saadaan get_datalist(index)-metodilla. Näistä Dataset instansseista voidaan noutaa tiedot get_data()-metodilla, joka palauttaa listan kuvaajan arvoista.
Lisäksi molemmat luokat sisältävät metodeja erilaisten numeeristen tietojen palauttamiseen. Datalist-luokan metodit palauttavat kaikkien kuvaajien kombinaatiodataa, kun taas Dataset-luokasta saadaan yksittäisten kuvaajien data.
Käyttöliittymä rakentuu MainWindow-luokan pohjalle, johon voidaan addX metodeilla lisätä valikot, työkalupalkit ja telakat. Luokkien Qt-perinnät näkee helpoiten koodista.
Luokkakaavio:



Algoritmit

Värien generointi tapahtuu lineaarisesti HSV värikartan avulla, jossa värin arvo sijaitsee yhdellä akselilla. Tämän avulla väriakseli voidaan jakaa tasavälein sen mukaan kuinka monta erilaista väriä tarvitaan, ilman että päällekkäisiä värejä esiintyy.
Gridin piirtoalueen laskeminen tapahtuu jakamalla näyttöalueen koko ruutujen koolla. Ruutujen koko lasketaan ruudun skaalauskertoimen(x/yscale) ja ruudun kokokertoimen(x/ygridsize) tulosta. Tällä vältetään ylimääräisten viivojen piirtäminen joka hidastaisi ohjelmaa.
Gridin viivoitus lasketaan tasavälein ruudukon koon ja skaalauskertoimen tulosta, johon lisätään offset-kerroin.
Viivoituksen karsiminen lasketaan käänteisesti verrannollisesti ruudukoiden kokoon nähden. Karsiminen suoritetaan jakojäännöksen avulla.

Piirakkadiagrammin laskenta perustuu data-alkioiden normalisoimiseen kokonaissumman avulla, jolloin kaikkien normalisoitujen alkioiden summasta saadaan 100 % täyteen. Tällä algoritmilla tietoja ei tarvitse syöttää prosentteina.

Tietorakenteet

Ohjelman lukema data säilötään sarakkeittain, eli kuvaajakohtaisesti, listoihin ja nämä listat yhteen kaiken datan kattavaan listaan. Listoja käytetään koska näitä on helpoin manipuloida, eikä säilöttävän tiedon tyypistä tarvitse huolehtia. Jos datapisteitä tulisi mallintaa miljoonia, voisi kevyempi taulukkorakenne olla niille parempi vaihtoehto.

Tiedostot

Numeerinen data luetaan CSV muodossa, jonka myös Excel tunnistaa. Sarakkeilla jaotellaan data jokaisen yksittäisen kuvaajan nimen alle ja riveillä sijaitsee tämän kuvaajan yksittäiset datapisteet. Sarakkeita lisäämällä voidaan lisätä useampia kuvaajia samaan tiedostoon. Piirtokoordinaatistossa ensimmäinen sarake määrää kuvaajan X-akselin arvot, joka on esimerkiksi aika.

Samaa tiedostorakennetta voidaan lukea myös pylväs- ja piirasdiagrammeja varten. Viimeisellä rivillä ensimmäisen sarakkeen elementti ilmaisee datan tyypin, joita on tässä ohjelmassa LINE, BAR ja PIE. Esimerkki, jossa neljä XY datapistettä kahdella eri kuvaajalla:

```
Time;Temp;Humidity
0;54;45
1;45;34
2;34;45
3;23;54
LINE;;
```

Testaus

Datan käsittelyssä on tärkeää että data tulee luetuksi oikein, tunnistaa viat tiedostossa sekä olla sotkematta dataa väärän otsikon alle. Ohjelman tulee sietää whitespacea ja tunnistaa täysin tyhjät ja virheelliset lohkot.

Virheellisen datan automaattinen testaus tehdä datatiedostolla jossa on:

1. Kirjaimia/merkkejä
2. Tyhjiä rivejä
3. Puuttuvia lohkoja
4. Ylimääräisiä lohkoja
5. Ei mitään, eli tyhjä tiedosto
6. Puuttuva tiedosto

Datan lukemisen automaattinen testaus tehdään syöttämällä virheelliseksi muotoiltu tiedosto kustakin tyypistä, jolloin Datalist-luokan load-metodin tulisi havaita virhe ja asettaa luokan instanssin datatyypin arvoksi 0.

Numeeristen arvojen lukemisen testaus suoritetaan tiedostolla jossa on kolme yksinkertaista kuvaajaa joista tiedetään minimi, maksimi ja keskiarvo. Ohjelman tulee palauttaa tämä ennalta tunnettu arvo kunkin laskutoimituksen metodilla. Ohjelman automaattinen testausrutiini voidaan ajaa suorittamalla test_units.py skripti.

Visuaalista testaamista varten on sisällytetty kuvaajia kansioon Test/Visual:

1. Yhden kuvaajan diagrammi
2. Yhden pylväsjoukon diagrammi
3. Piirakkadiagrammi
4. Useiden kuvaajien diagrammi
5. Useiden pylväsjoukkojen diagrammi
6. Viivadiagrammin automaattiskaalaus suurilla arvoilla

Ohjelman tunnetut puutteet ja viat

Useiden pylväsdiagrammin pystyakselin nimi määräytyy ensimmäisen pylvään nimen mukaan, joka ei välttämättä ole sama muille pylväille jos tätä ei ota huomioon. Nimen voi kuitenkin muuttaa, mutta muutosta ei tallenneta tiedostoon.
Desimaalilukuja ei voi käyttää, vaan ne tulkitaan virheelliseksi tiedostoksi.
Viivadiagrammi ei tarkista x-akselin lukujen etenemistä pienemmästä suurempaan, vaan hyväksyy arvoja myös taaksepäin jos tiedostossa näin on.
Skaalaus vaikeaa isojen lukujen ääriarvoilla ja voi olla hankalaa eri hiiren sensitiivisyyksillä.
Jos telakan irrottaa ja lataa sen jälkeen uuden kuvaajan, irrotettu telakka katoaa mutta Window valikossa näkyvä rasti näyttää yhä telakan olevan esillä.

3 parasta ja 3 heikointa kohtaa

Kuvaajien navigointijärjestelmä onnistui toimimaan yllättävän hienosti, eikä suurempia kompromisseja tarvinnut tehdä. Yksittäisten kuvaajien piilotus ja välilehtiin nostaminen(hiiren painikkeella legend-telakasta) ovat myös hyödyllisiä ja hyvin toimivia ominaisuuksia käyttöliittymässä. Lukujen ja pylväsdiagrammien nimien adaptoituminen toimii myös hyvin, eikä arvot pääse päällekkäin ainakin Windowsin oletusfonttikoolla.

Heikkoina kohtina voidaan pitää kuvaajan suurennoksen keskipistettä, joka ei ole näytön keskellä vaan sen alakulmassa. Jos tätä ei ota huomioon, se usein aiheuttaa joskus kuvaajan sinkoutumisen näyttöalueen ulkopuolelle zoomatessa.

Muutamissa tapauksissa tuli myös ajan säästämiseksi käytettyä kertoimina kokeilupohjaisia hattuvakioita, jotka eivät skaalaudu automaattisesti esimerkiksi jos fonttien kokoa tai asetelmia haluaisi muuttaa. Dynaamisten kertoimien puuttuminen toisaalta yksinkertaistaa ohjelmakoodin lukemista.

Poikkeamat suunnitelmasta

Muutin alkuperäisestä suunnitelmasta poiketen PyQt:n 4 versioon, koska PyQt5 kaatui jostain syystä ilman virheilmoituksia, joten kehitys oli lähes mahdotonta. Vaihto ei tosin ollut ongelma, sillä mitään varsinaista tarvetta Qt:n 5 versiolle ei ollut.
Päätin myös hylätä kokonaan kuvaajat joissa on yksikin virheellinen elementti, jotta ohjelma ei vahingossakaan näyttäisi väärää dataa. Virheellisen datan tulkinta on aina arvaus ja ongelmat on luotettavampia selvittää manuaalisesti.
Lisämetodeja numeerisen datan laskentaan en tehnyt, koska tämä olisi ollut puhtaasti rutiinikoodausta eri kaavoilla, vaan käytin ylimääräisen ajan käyttöliittymän ominaisuuksiin.

Toteutunut työjärjestys ja aikataulu

Aikataulu toteutui suurin piirtein, paitsi viikkojaon osalta. Ohjelma tuli tehtyä suurimmilta osin kahden viikon sisällä. Tämä on toisaalta parempi tapa, jolloin koodia tarvitse tulkita aina taukojen jälkeen uudelleen. Kuvaajan piirto-ominaisuuksiin tuli käytettyä hieman enemmän aikaa kuin suunnitelmassa.
Työjärjestys oli jokseenkin samanlainen, vaikkakin käyttöliittymää tuli kasattua rinnakkain piirto-ominaisuuksien yhteydessä.

Arvio lopputuloksesta

Mielestäni ohjelman käyttöliittymästä tuli varsin kattava ja robusti. Lisäksi käyttöliittymän ominaisuuksien viimeistely oli odotettua mukavampaa, joten siihen tuli käytettyä aikaa.

Pääohjelman jako oli ihan hyvin toteutettu. Ohjelman edetessä Qt:hen tutustuttuani enemmän tulin havainneeksi että olisin voinut pohjata kuvaajan piirtämisen kokonaan QGraphicsScene-luokkaan jossa olisi ollut erittäin käteviä piirto-ominaisuuksia, mutta en nähnyt syytä muokata koko piirtofunktiota alusta sen pohjalle. Pohjasin Legend-telakan kuitenkin tähän QGraphicsScene -luokkaan.
Koodin yleislaatu on juuri sellaista mitä ensimmäiseltä yrittämiseltä uuden käyttöliittymäkirjaston kanssa voi odottaakin. Mitään erityisen nokkelia metodeja tai rakenteita ei ole käytetty, vaan koodi on muodostunut lähinnä yrityksen ja erehdyksen pohjalta kasautuen. Funktioiden ja luokkien perusrakenne on siistitty kuitenkin jonkinlaiseen muotoon.
Datan luokat olisi voinut ehkä jakaa jokaiselle datatyypille erikseen. Sinnittelin saman luokan käytön kanssa niin pitkään ettei sitä ollut järkeä kirjoittaa uudelleen, koska lopputulos sen avulla kuitenkin toimii. Dataset-luokka on toisaalta muuten järkevästi toteutettu ja erilaisten numeerista laskentaa suorittavien metodien lisääminen onnistuu siihen helposti.

Viitteet

Sain esimerkit PyQt:n käyttämisestä:
http://zetcode.com/gui/pyqt4/
http://www.tutorialspoint.com/pyqt/

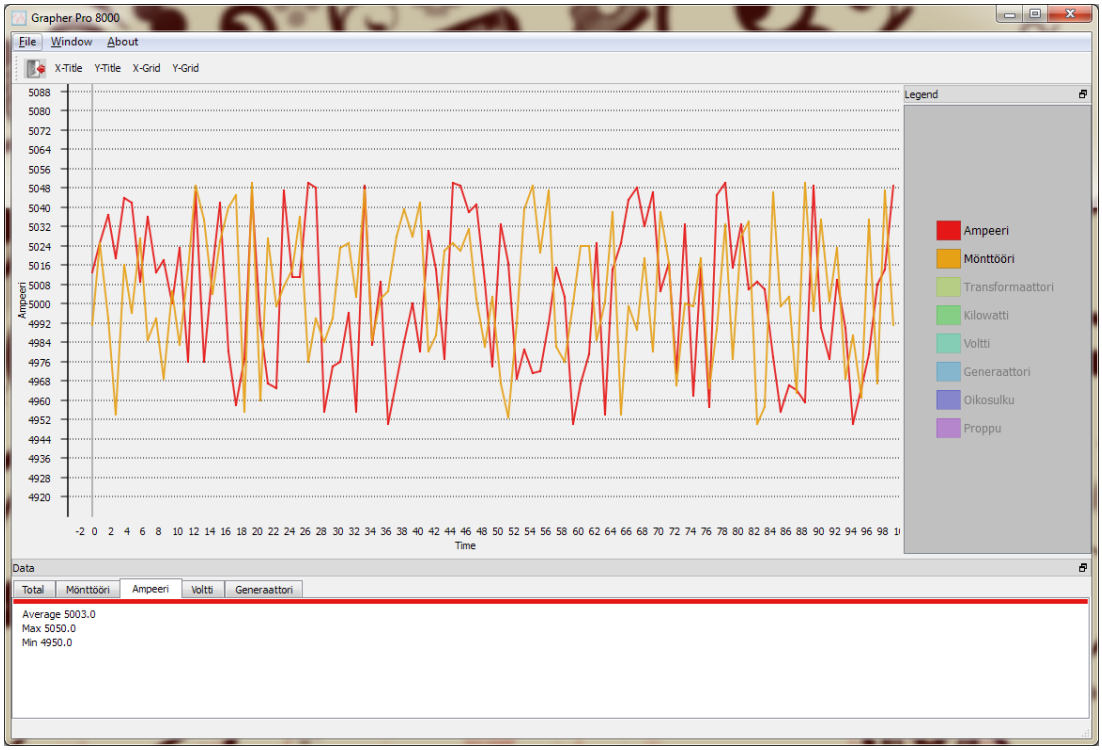
Luokkien ja metodien tiedot:
http://pyqt.sourceforge.net/Docs/PyQt4/
http://doc.qt.io/qt-4.8/index.html

Vastaukset yleisiin kysymyksiin ja PyQt:n syntaksien selvitykseen:
http://stackoverflow.com/

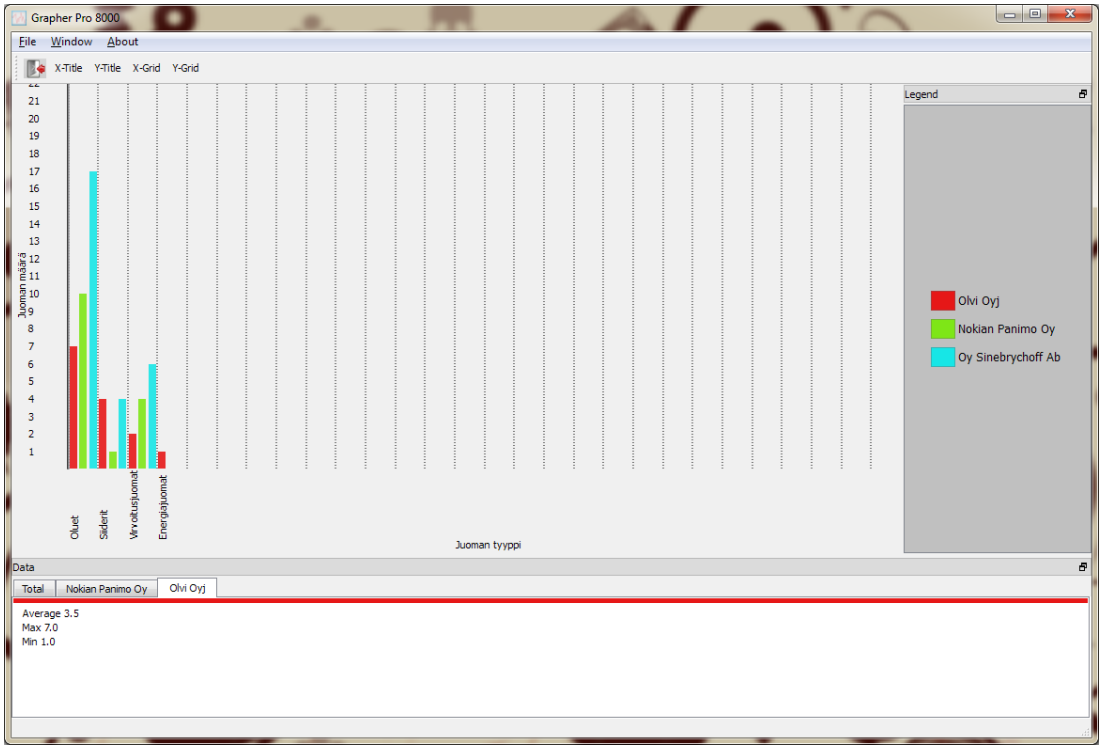
Liitteet

Kuvankaappauksia erilaisista käyttötapaustilanteista:

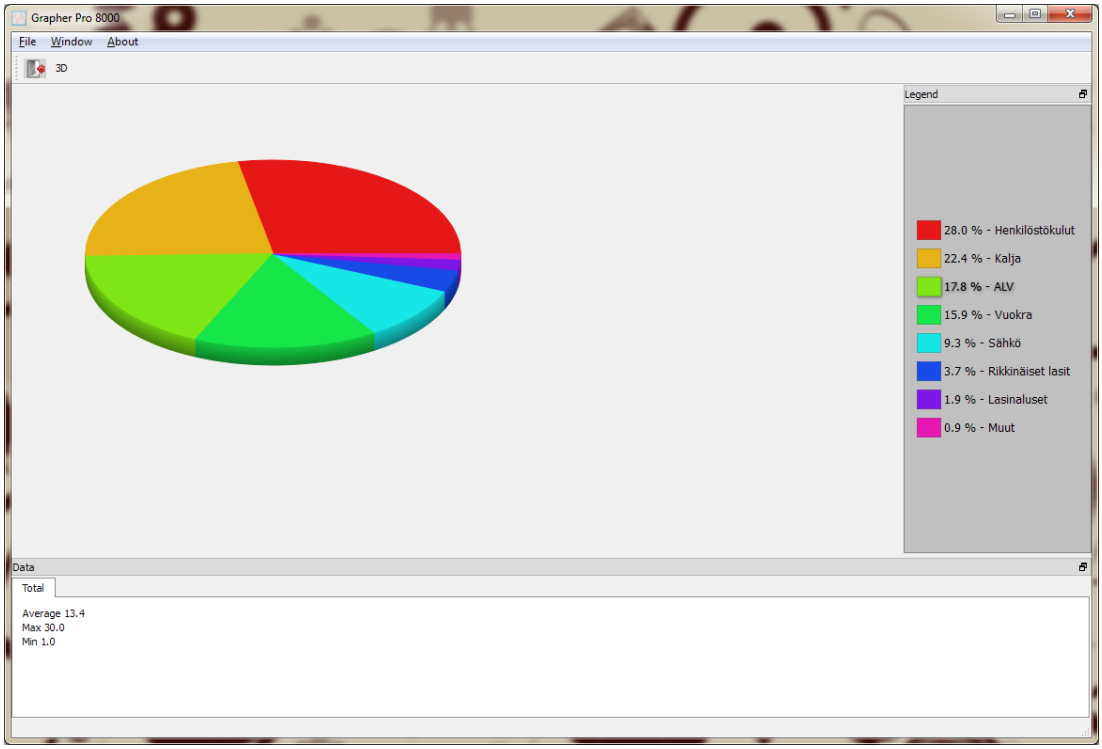
Viivadiagrammi jossa on tehty kuvaajien piilotus, välilehteen nosto ja pystyruudukon piilotus:



Useiden pylväiden pylväsdiagrammi jossa on tehty skaalaus:



Piirakkadiagrammi 3D-näkymässä:



Projektin lähdekoodi on Gitissä ja sisältää main.py, test_units.py, graph.py, widget.py, loadfile.py skriptit.