Robottikäsi

22.4.2019

Eero Asikainen
595654 ENG/KJR

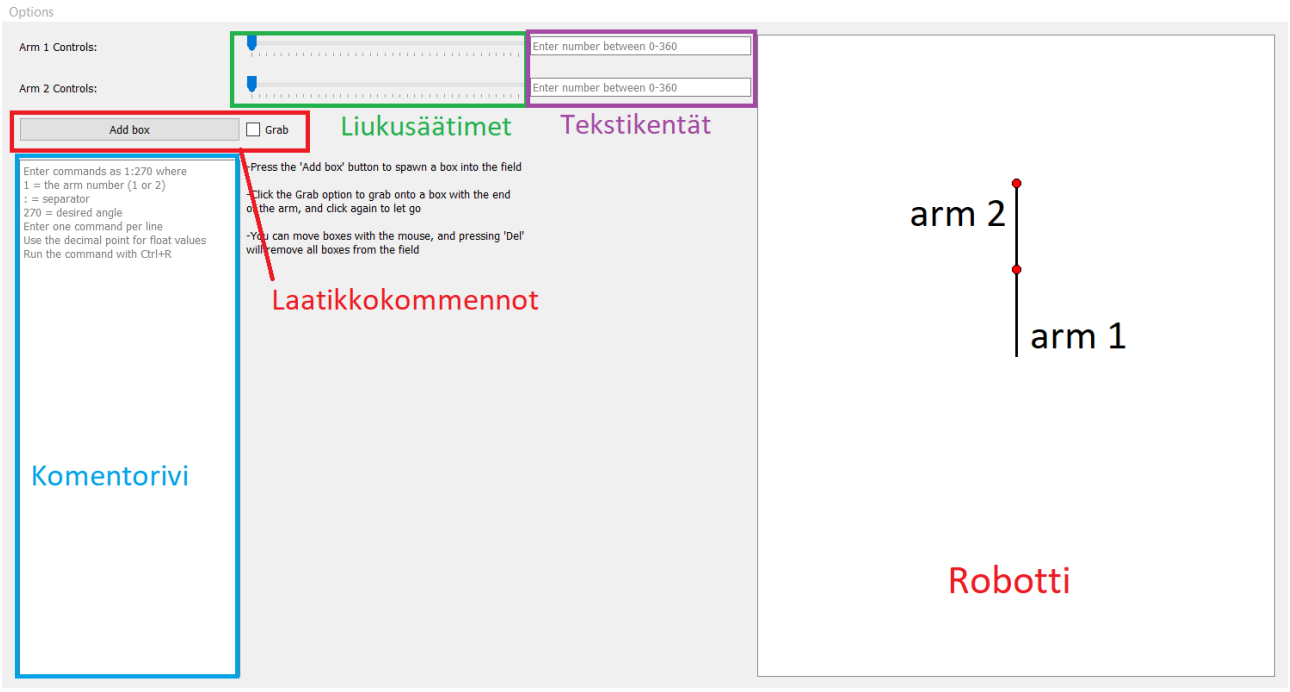
1. **Yleiskuvaus**

   Luotu ohjelma on keskivaikean vaatimustason mukainen, eli se simuloi robottikättä 2-ulotteisessa ympäristössä. Sen kulmien asetuksia voi säätää ohjelmasta joko liukusäätimillä, arvoja syöttämällä tai tekstinä ohjelmoiden. Sen liikkuminen on animoitua ja se pystyy tarttumaan sen ulottuvilla oleviin kappaleisiin. Vastoin alkuperäistä suunnitelmaa, siinä ei ole mahdollista muuttaa robotin käden nivelten määrää, tai niiden pituutta, vaan ne ovat kiinteät. Animaatioiden keston voi määrittää ohjelmasta.

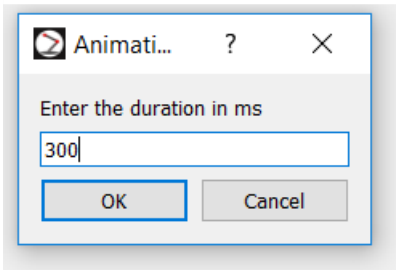
2. **Käyttöohje**

   Ohjelman tekstimuotoinen käyttöohje löytyy README-tiedostosta. Tässä kuitenkin muutama havainnollistava kuva:

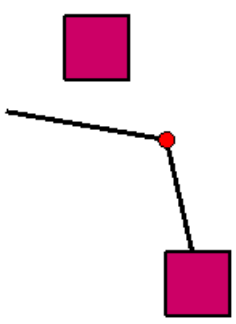
   Ohjelma perustilassaan käynnistyksen jälkeen:



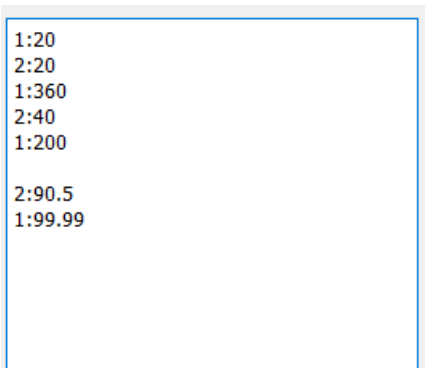
Animaatioajan asettaminen:



Tarttuminen laatikoihin:



Tekstikomentoesimerkkejä

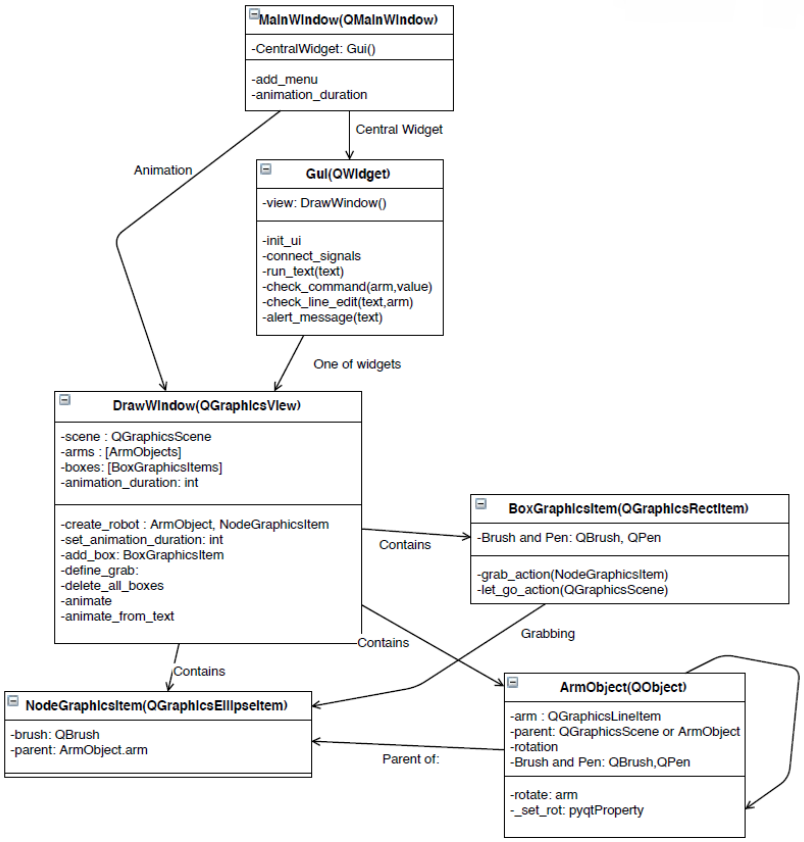


3. **Ulkoiset kirjastot**

   Ainoa käytetty ulkoinen kirjasto on PyQt5, jonka komponenttien varaan lähes koko ohjelma pohjautuu. Lisäksi käytetään ”sys” kirjastoa ohjelma ikkunan näyttämiseen käyttöjärjestelmässä, ja ”random” kirjastoa lisättyjen laatikoiden sattumanvaraisen sijainnin aikaansaamiseksi.

4. **Ohjelman rakenne**

   UML mallikuva ohjelman luokista:



   Ohjelman pääikkuna on MainWindow luokan luoma QMainWindow. Tähän ikkunaan asetetaan PyQt5 käytäntöjen mukaisesti CentralWidget, joka on luokan Gui instanssi. Gui luokka luo kaikki muut ohjelman widgetit (Sliderit, tekstikentät, ja kytkimet) ja asettaa ne osaksi omaa Layouttiaan. Näistä tärkein on DrawWindow, joka on se ikkuna, jonka sisällä robotti ja laatikot sijaitsevat. DrawWindow huolehtii mm. animaatioiden luonnista, robotin ja sen osien piirrosta ja laatikon grab-toiminnon suorittamisesta. DrawWindow luo create_robot metodillaan varsinaisen robotin. Robotin varret ovat ArmObject instansseja ja nivelet NodeGraphicsItem instansseja. ArmObject luo itselleen varren, joka on QGraphicsLineItem, ja tämän olion rotaatiota muutetaan ohjelmassa.

   DrawWindow vastaa käytännössä QGraphicsView luokkaa, ja se sisältää oman QGraphicsScene:n, johon eri graafiset elementit tulee lisätä. Tämän ymmärtäminen on keskeistä Qt:n piirtymisen ymmärtämiseksi. GraphcisView on ikkuna, josta katsellaan Sceneä, jossa oliot teoriassa sijaitsevat. Scene on myös kaikkien niiden olioiden parent-item, joilla se ei ole toinen graafinen olio.

   Ohjelman lopullinen luokkajako poikkeaa melko paljon alkuperäisen suunnitelman luokkajaosta, lähinnä koska vasta projektia toteuttaessa opin paremmin ymmärtämään millaisia luokkia kannattaa luoda PyQt5:ttä käyttäessä. Luokkajakoa voisi yksinkertaistaa enemmänkin, sillä tällä hetkellä esimerkiksi NodeGraphicsItem on melko turha omana luokkanaan. Tosin sen

olemassaolo mahdollistaa ohjelman kehittelyn jatkossa, jolloin sille voisi lisätä omia metodejaan, kuten vaihtaa väriä tms.

5. **Algoritmit**

   Ohjelman keskeisin toiminto on robotin varsien kulmien muuttaminen. Tämä onnistuu suoraan PyQt:n metodeilla, ja lisäksi, PyQt välittää muutoksen kaikille kyseessä olevan objectin child-objekteille. Robotti on luotu siten, että sen toinen varsi on ensimmäisen varren child, ja varsien nivelet ovat varsien child-objekteja. Tällöin ei tarvitse erikseen miettiä, miten yhden varren rotaatio vaikuttaa muiden osien sijaintiin, vaan PyQt laskee sen itse.

   Robotin Grab-toimintaa varten tulee selvittää, mitkä laatikot osuvat robotin uloimpaan niveleen. Tämä onnistuu myös suoraan PyQt:n metodilla, joka tarkistaa päällekkäisiä graafisia olioita. Kun tarttuminen tapahtuu, niin se on toteutettu siten, että uloin nivel lisätään laatikon parent-objektiksi, jolloin niveliä liikuttamalla laatikkokin siirtyy. Tämän olisi voinut toteuttaa myös siten, että niveltä liikuttaessa se päivittää jatkuvasti sijaintinsa laatikolle, joka sitten vuorostaan siirtää itsensä samaan pisteeseen.

   Tekstikenttään annettavat komennot muutetaan erillisiksi käskyiksi Pythonin omilla listaoperaatioilla, joissa syötetty komento pilkotaan osiin, tarkistetaan kunkin osan oikeellisuus ja välitetään käskyt eteenpäin.

   Animaatiot on luotu PyQtn omalla animaatioluokalla. Peräkkäisten käskyjen animoinniksi käskyt sijoitetaan ensin listaan, ja animaatiolle annetaan lähtö- ja loppuarvoiksi listan ensimmäinen ja viimeinen alkio. Listan muut arvot annetaan animaatiolle väliarvoiksi, ja niiden paikaksi animaatiossa annetaan niiden sijainti listassa normitettuna välille 0-1.
   1 / ((listan pituus) + 1) * ((listan indeksi) + 1)
   Kullekin robotin varrelle on oma animaationsa, joka mahdollistaa niiden ajamisen samanaikaisesti.

6. **Tietorakenteet**

   Ohjelmassa esiintyvät arvot tekstikentissä siirretään asetuksia muuttaville metodeille String muodossa, jonka Python muuttaa joko int tai float muotoiseksi. DrawWindow sisältää kaksi listaa, joista toinen sisältää käden ArmObject:it ja toinen kentälle lisätyt laatikot. Kaikki muutoksia aiheuttavat syötteet saadaan käyttäjältä, eikä mitään dataa tarvita etukäteen. Ohjelma ei myöskään tallenna mitään mihin pääsisi käsiksi sen sulkemisen jälkeen.

7. **Tiedostot**

   Ohjelma ei käsittele tiedostoja.

8. **Testaus**

   Keskityin ohjelmaa luodessa lähinnä siihen, että sen toiminta vastaisi graafisesti sitä, mitä olin ajatellut. Testaus tapahtui siis siten, että ajoin ohjelmaa itse ja tutkin miten eri osa-alueet piirtyvät, ja miten eri arvoja muuttamalla ohjelma käyttäytyy. Projektia tehdessä haasteellisinta oli ymmärtää, miten saisin PyQt5:n luokat ja metodit toimimaan niin kuin halusin. Yksikkötestauksesta ei olisi ollut hyötyä, kun en vielä osannut sanoa mitä käytännössä olisin halunnut testata. QTest luokalla on mahdollista simuloida käyttöliittymän yksinkertaisia toimintoja, kuten nappien tai hiiren painalluksia, mutta en oikein löytänyt hyvää tapaa testata sitä, että onko ruudulle piirtyvä käden asento juuri se, mikä sen sillä hetkellä pitäisi olla. Tämän vuoksi kirjoitin testejä sisältävän moduulin oikeastaan vasta ohjelman ollessa melko valmis.

   Lopullinen ohjelma sisältää testing.py luokan, missä on hieman PyQt:n QTest ja Pythonin Unittest integraatiota. Ohjelma läpäisee tällä hetkellä kaikki siinä olevat testit, jotka ovat lähinnä käden varren kulmien ja niiden arvoja syöttävien widgettien välistä vastaavuutta tutkivia. Olisin halunnut testata enemmän laatikoiden tarttumisen ja siirtelyn toimivuutta, mutta en osannut kirjoittaa niille sopivia testejä. En myöskään osannut kirjoittaa testejä animaatioille.

9. **Ohjelman tunnetut puutteet ja viat**

Ohjelmassa ei tällä hetkellä ole mahdollista muuttaa robotin varsien lukumäärää, tai niiden pituutta. Olin ajatellut alun perin lisääväni nämä ominaisuudet, mutta päädyin lopulta jättämään ne pois. Teoriassa sen toteuttaminen olisi hyvinkin mahdollista, sillä jokainen kohta, jossa muutetaan jonkin varren arvoa voitaisiin muokata muotoon missä se toimii n-kappaleelle varsia ja metodeille määriteltäisiin varren numero parametrina. Samoin ikkunaa luodessa voitaisiin määritellä sen luovan n-määrän liukukytkimiä ja tekstikenttiä jne.

Ohjelma toteuttaa tällä hetkellä varsien pyörittelyn PyQt:n setRotation() metodilla, joka kääntää kyseisen graafisen olion sisäistä koordinaatistoa. Ajattelin aluksi tämän tavan olevan kätevä, sillä se siirtää rotaation sen lapsiolioille. Nyt PyQt:n toimintaa paremmin ymmärtävänä toteuttaisin ohjelman kenties kuitenkin hyödyntämällä paint() ja update() metodeja, enkä kääntäisi koordinaatistoja ollenkaan. Se olisi selkeämpi vaihtoehto, ja olisi paremmin muokattavissa. Nykyisessä toteutuksessa on hieman epäselvää, miten päin minkäkin olion koordinaatisto milläkin hetkellä on.

Edellä mainittu ongelma vaikuttaa laatikoiden grab toimintaan. Sen toteutus ei tällä hetkellä ole kovin elegantti. Koska varren nivel liitetään laatikon vanhemmaksi, laatikko lisätään ikään kuin samaan koordinaatistoon vanhempansa kanssa. Koska laatikoiden origo on niiden keskipisteessä, vartta pyörittämällä laatikon origo pyörii nivelen origon ympärillä, jolloin laatikko liikkuu suhteessa tarttuvaan niveleen. Tämän voi huomata erityisesti tarttumalla laatikkoon aivan sen reunasta. Ongelma ratkeaisi todennäköisesti sillä, että koko ohjelmassa olisi vain yksi koordinaatisto, ja sekä varsien kääntyminen että niiden sijainti toteutettaisiin siinä. Tällöin kulmien muuttaminen tulisi rakentaa kokonaan eri tavalla, joten tyydyin siksi tähän versioon, sillä minulla ei ollut aikaa muuttaa ohjelmaa niin paljon. Grab toiminnan toteutuksen toinen heikko puoli on, että laatikoiden parent-objektin muuttaminen muuttaa myös niiden piirtojärjestystä GraphicsScenissä, jolloin monta laatikkoa samaan aikaan siirtäessä saattaa niiden päällekkäisyys muuttua.

10. **3 parasta ja 3 heikointa kohtaa**

    Hyvät:
    -Tekstikenttään kirjoitettavien käskyjen animaatio toimii mielestäni kauniisti. Sain siitä juuri sellaisen kuin olin sen kuvitellut, ja robotin pyörittäminen sen kautta on kaikkein hauskinta. Lisäksi se, että se osaa suorittaa animaation useamman vaiheen kautta on mielestäni hienoa.

    -Virheellisten syötteiden tunnistaminen toimii hyvin, ja ohjelma ilmoittaa niistä käyttäjälle.

    -Animaation keston muuttaminen on mielestäni hyvä lisä

    Heikot:
    -Kuten kohdassa 10 todetaan, robotin pyörittämisen toteuttaminen toisella tavalla olisi jatkossa muokattavuuden kannalta parempi vaihtoehto, ja sen vuoksi myös laatikoiden siirtely Grab toiminnolla ei ole kovin toimiva.

    -Tutkimalla PyQt:n layout ja style vaihtoehtoja enemmän ohjelman ulkonäöstä voisi saada kauniimman ja käytön kannalta selkeämmän. Se on nykyisenään minulle järkevä, kun tunnen sen toiminnan, mutta uudelle käyttäjälle esimerkiksi mikä säädin liikuttaa mitäkin vartta ei ole intuitiivisesti selvää, vaan sitä pitää kokeilla.

    -Animaatiot toimivat hyvin, jos niiden antaa pyöriä rauhassa, mutta jos pitkän animaation aikana säätää muita säätimiä, niin robotti liikkuu oudosti. Tämän voisi korjata esimerkiksi lukitsemalla kaikki muut säätimet animaatioiden ajaksi.

11. **Poikkeamat suunnitelmasta**

    Alkuperäistä luokkajakoa tehdessäni en ymmärtänyt PyQt:n toiminnasta tarpeeksi, että se olisi ollut millään tasolla järkevä. Siksi ohjelman lopullinen luokkajako poikkeaa siitä melko lailla. Tarvitsin esimerkiksi pääikkunalle erillisen luokan, ja animointia varten riittivät DrawWindow:issa olevat metodit. En myöskään toteuttanut varsien lukumäärää tai pituutta säätäviä vaihtoehtoja.

Suunnitelman toteutusjärjestys vastasi melko hyvin toteutunutta. Totesin tosin, että PyQt:n toiminnan vuoksi olisi järkevämpää aloittaa suoraan Gui:n luomisesta, jolloin voisi kirjoittaa suoraan metodeja siinä oleville widgeteille, ja niiden pohjalta kokonaiskuva hahmottuisi paremmin. Toisin sanoen jälkiviisaasti suunnitelmani luoda ensimmäisenä robotin graafiset osat aiheutti ylimääräistä työtä, kun jouduin jälkeenpäin kirjoittamaan niiden ominaisuuksia uudelleen widgettien vaatimusten mukaisesti. Konkreettinen esimerkki tästä on, että QPropertyAnimatiion luokkaa käytettäessä animoitava olio on oltava luokan QObject instanssi, joten jouduin sen vuoksi kirjoittamaan ArmObject luokan uudelleen.

Ajankäytön suhteen arvioni osuivat mielestäni hyvin kohdalleen. Aika jakautui eri osien kirjoittamiseen tosin hieman eri tavalla kuin olin ajatellut, mutta kokonaismääräisesti se oli melko samaa luokkaa kuin käyttämäni aika.

12. **Toteutunut työjärjestys ja aikataulu** Kerro tässä yleisellä tasolla missä järjestyksessä projekti lopulta toteutettiin (mielellään myös päivämäärät). Missä poikettiin suunnitelmasta?

    Loin ensimmäisenä graafiset elementit, eli ArmObject, NodeGraphicsItem ja DrawWindow luokat. Tämän jälkeen tajusin melko nopeasti tarvitsevani Gui luokan, johon voin lisätä kaikki widgetit (sliderit, tekstikentät yms), joten kirjoitin sen. Tässä vaiheessa suurin osa ajasta meni widgettien signaalien liittämiseen varsien kulmiin, ja niiden piirtymisestä oikein. Kirjoitin sen jälkeen BoxGraphicsItem luokan, ja niiden kentälle lisäävät toiminnot. Laatikoiden siirtämisen toteuttamiseen meni paljon enemmän aikaa, kuin mitä olin ajatellut, eikä lopputulos edes ollut kovin onnistunut. Saatuani sen valmiiksi, siirryin tekstimuotoisten widgettien pariin, ja niiden hiominen oli myös aluksi hidasta. Viimeinen osa oli animaatioiden kirjoittaminen, ja useamman syötteen tekstikentän luominen. Tämän jälkeen tein vielä testiluokan.

13. **Arvio lopputuloksesta**

    Kuten olen edellisissä kohdissa todennut, ohjelman tämänhetkinen tapa määrittää varsien kulmien muutos niiden koordinaatistoa käyttämällä toimii ohjelman nykyisessä tilassa hyvin, mutta sitä laajennettaessa toteuttaisin sen eri tavalla. Tämä mahdollistaisi myös laatikoiden siirtämisen tarkemmin ja järkevämmin. Ohjelma nojaa nykyisellään hyvin paljon PyQt5:n omiin luokkiin, ja ohjelman toiminnan ymmärtäminen vaatii ymmärrystä PyQt:n toiminnasta. Tämä ei välttämättä ole huono asia, mutta aiheuttaa sen, että moni toiminnan kannalta oleellinen koodi (kuten osien piirtyminen parent-child suhteiden kautta) on melko näkymättömissä, ja sitä laajennettaessa mahdollisten virheiden havaitseminen olisi monimutkaista.

    Jos ohjelmaa arvioi sen suhteen, mikä sen tavoite oli sitä suunnitellessa, niin se on mielestäni oikein onnistunut. Olen tyytyväinen siihen, että kaikki vaaditut osa-alueet toimivat vähintään hyvin, ja mielestäni koodi itsessään on selkeää. Ohjelma keskittyy olennaiseen, ja se on loppujen lopuksi melko yksinkertainen. Sitä käyttäessä se toimii kevyesti ja responsiivisesti. Tämä oli ensimmäinen kirjoittamani graafinen ohjelma, joten opin sen tekemisestä valtavasti ja olen myös ylpeä saatuani sen toimimaan.

14. **Viitteet**

    Kaikista eniten käytin Qt:n omaa dokumentaatiota, jonka tulkkaaminen PyQt:n kielelle tuotti välillä haasteita. Muut tärkeät lähteet olivat PyQt:n oma dokumentaatio (tosin vain PyQt4:lle) ja erilaiset StackOverflow:n langat PyQt:n käytöstä. Katsoin muutamaa playlistiä Youtubesta, joissa selitettiin PyQt:n käyttöä. QTest:in ja Pythonin unittestin yhteiskäyttöä varten löysin myös hyvän ohjenettisivun. Jonkin verran käytin kurssin omia ohjeita, ja kierroksen 5 RobotWorld-tehtävän lähdekoodia, josta yritin tutkia PyQt:n logiikkaa.

    Lähteitä, joita olen käyttänyt:

    https://doc.qt.io/
    https://www.youtube.com/watch?v=JBME1ZyHiP8&list=PLQVvvaa0QuDdVpDFNq4FwY9APZPGSUyR4
    https://docs.python.org/3.2/index.html
    https://www.riverbankcomputing.com/static/Docs/PyQt5/

http://johnnado.com/pyqt-qtest-example/
https://wiki.python.org/moin/PyQt/Tutorials
http://zetcode.com/pyqt/qpropertyanimation/

En listaa tähän linkkejä kaikkiin käyttämiini stackoverflow:n vastauksiin, sillä en niitä kaikkia edes muista, ja ne vastasivat yleensä aina johonkin yksittäiseen ongelmaan, joka minulla oli yrittäessäni ymmärtää PyQt:n käyttöä.