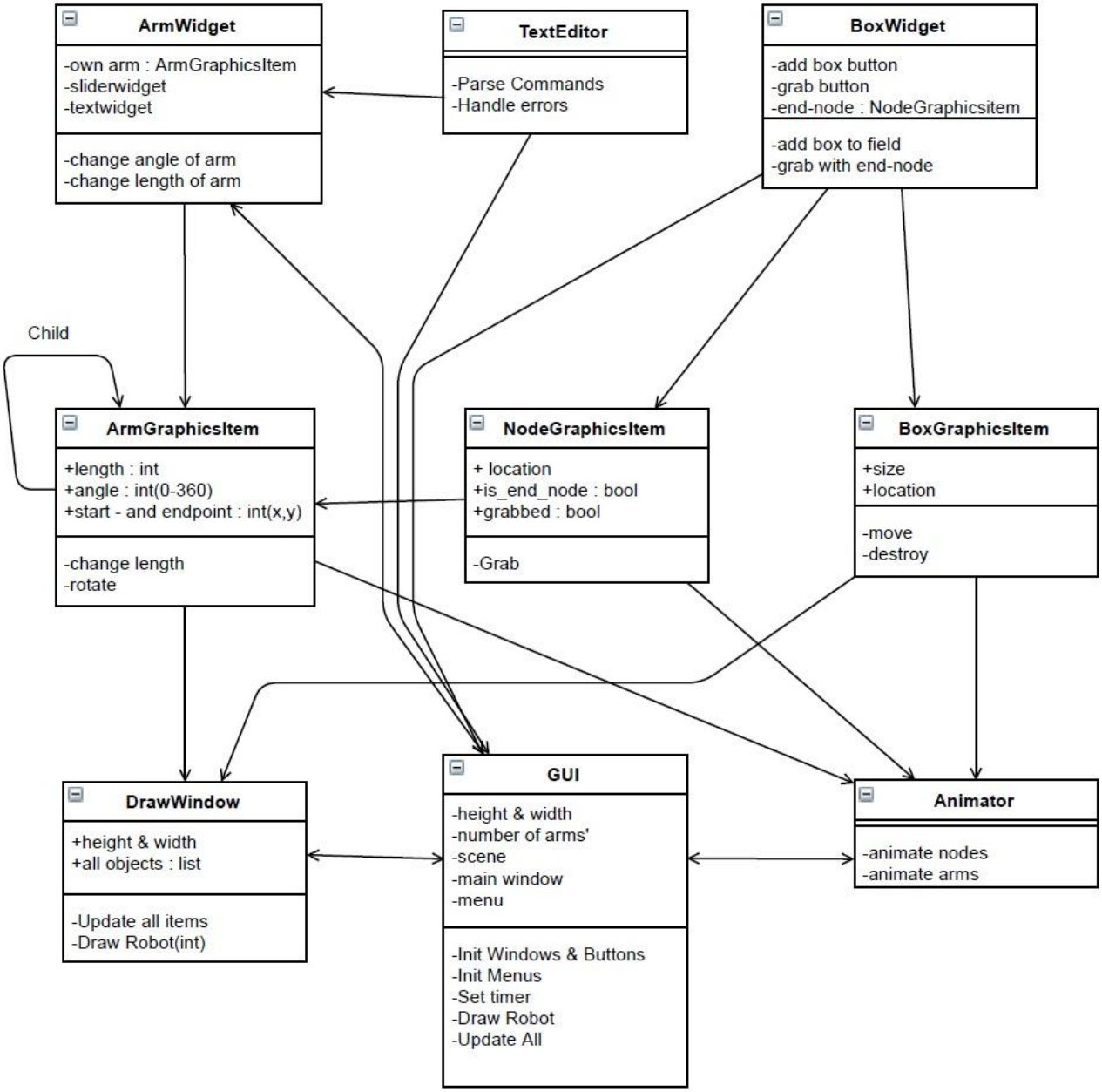
Tekninen suunnitelma, Robottikäsi
Eero Asikainen 595654

1. **Ohjelman rakennesuunnitelma**



Ohjelmaa ajetaan main.py tiedoston kautta, joka käynnistää tarvittavat prosessit ohjelman toiminnan kannalta. Toinen tärkeä osa on luokka GUI(QtWidgets.QtMainWindow) joka käsittelee käyttöliittymän toimintaa ja huolehtii piirtämisestä. Ajattelin rakentaa ohjelman suoraan Graafisten komponenttien varaan, enkä erikseen

sisäisen logiikan kautta, johon sitten lisäisin käyttöliittymän. Käsiteltävät objektit ja varsinkin niiden keskinäinen logiikka on melko yksinkertainen, joten uskon että tämä ratkaisu riittää ohjelman vaatimuksiin. Monimutkaisemmassa tapauksessa olisi perusteltua rakentaa ohjelman sisäinen logiikka enemmän erilleen sen muokkauksen helpottamiseksi.

GUI(QGraphicsMainWindow): Piirtää ikkunan, ja asettelee sinne oikeat QtWidget komponentit, joista käyttöliittymä muodostuu.
-height & width
-no of arms
-main window
-scene
-menu

- Init windows – asettaa ikkunat
- Init Buttons – asettaa käytettävät näppäimet ja niiden toiminnot
- Init menus – asettaa ikkunan yläpalkin valikon ja sen toiminnot
- Set timer – luo sisäisen kellon, joka määrittää update funktion taajuuden
- Draw robot – käynnistyy valittaessa robotin varsien lukumäärä, ja piirtää robotin perusasennossa
- Update all – päivittää kaikkien komponenttien piirron

TextEditor(): Qwidget, johon käyttäjä voi kirjoittaa valittuja komentoja tekstinä, ja siten ohjelmoidusti muuttaa robotin asetuksia. Tämän tarkempi toteutus vaatii vielä opiskelua Qt:n dokumentaatiosta, mutta tarkoitus olisi, että widget olisi command-line tyyppinen tekstikenttä, jonne käyttäjä voi syöttää valmiiksi ohjelmoituja komentoja, jotka sitten muuttavat ArmWidgetin arvoja, ja sitä kautta liikuttavat kättä halutulla tavalla. Komennot olisivat luokkaa: arm1.rotate(30), joka muuttaisi ensimmäisen varren kulmaa 30 astetta vastapäivään.

DrawWindow(QGraphicsScene): Qt-widget, johon piirretään robotti. Sijoitetaan MainWindown oikeaan reunaan. Sisältää ikkunan, jonka koordinaatiston mukaan robotin osat piirtyvät.
-height & width

- Update all items – päivittää(liikuttaa) kaikki robotin graafiset osat
- Draw Robot – luo robotin kenttään

ArmGraphicsItem(QGraphicsItem): Käden varsia kuvaava luokka. Sisältää alku- ja päätepisteen, joiden väliin varsi piirretään. Varren rotaatiopiste tulee olemaan sen alkupää, jolloin sitä kierrettäessä se kiertyy sen pisteen kautta. Ensimmäisen varren alkupiste tulee aina olemaan origo, ja muut varret ovat aina edellisen lapsia PyQt mielessä, jolloin yhtä vartta kääntäessä myös sen lapset siirtyvät suhteessa isäntävarren koordinaatistoon.
-pituus
-kulma
-alkupiste
-päätepiste

- Muuta pituutta – pitää alkupisteen paikoillaan, ja siirtää loppupistettä varren suunnassa muutoksen verran
- Muuta kulmaa (rotate) – kääntää vartta alkupisteen suhteen

NodeGraphicsItem(QGraphicsItem): Piirtää ympyrän origoon, ja jokaisen Arm-varren päätyyn. Kaikki Nodet ensimmäistä lukuun ottamatta perivät PyQt:n kautta varsiobjektin, jotta niiden sijainti siirtyy myös varren siirtyessä.
-sijainti
-viimeinen node (on/ei) – vain robottikäden viimeinen node pystyy tarttumaan esineisiin

- Grab – tarkistaa, onko noden kohdalla Box-oliota. Näin ollessa tarttuu siihen, ja muuttaa sen sijaintia aina kun node itse liikkuu

BoxGraphicsItem(QGraphicsItem): Piirtää neliön muotoisen objektin, joita voidaan lisätä DrawWindow:n kenttään.
-size
-position

- Move
- Destroy

Animator(QPropertyAnimation): luokka, joka suorittaa muuttuvien objektien animoinnit aina kun arvoja muutetaan. (En ole vielä varma miten tämä kokonaisuus käytännössä toimii, mutta tämä tullaan suorittamaan joko GUI:n UpdateAll tick:ien mukaan, tai sitten erikseen kun jotain arvoja muutetaan.)

- Animate Nodes
- Animate Arms

BoxWidget(QWidget): sisältää laatikoiden operaatiot käsittelevät nappulat, ja hoitaa niiden kytkemisen tapahtumat
-koko
-location
-end_Node

- add_box_to_field – napin avulla voi hiirtä klikkaamalla lisätä BoxGraphicsItem olion kentälle
- grab_with_end_node – nappi siirtää käskyn Grab viimeiselle Nodelle, ja yrittää tarttua laatikkoon

ArmWidget(QWidget): sisältää tekstikentät ja sliderit varsien liikuttamista varten. Jokaiselle varrelle on oma ArmWidget, joka säätää vain sen varren toimintaa.
-own_arm
-slider_widget
-text_widget

- change_angle_of_arm – kutsuu kyseisen ArmWidgetin oman varren rotate funktiota
- change_length_of_arm – kutsuu kyseisen ArmWidgetin oman varren change_length funktiota

2. **Käyttötapauskuvaus**

Käyttäjä käynnistää ohjelman ajamalla main.py tiedoston. Tämä avaa ohjelman ikkunan kaikkine painikkeineen, mutta robotin piirtoon varattu ruutu on tyhjä. Käyttäjä valitsee käden varsien määrän valikosta, jonka jälkeen ohjelma piirtää varsien määrän mukaisen robotin siten, että varret ovat suoraan ylöspäin, ja niiden pituus on jokin vakio. Varsien lukumäärän määrittävä nappi käynnistää GUIssa piirtofunktion.

Tämän jälkeen käyttäjä voi säätää nappien avulla robotin asetuksia, esimerkiksi säätää ensimmäisen varren kulman 90 astetta myötäpäivään. Tämä tapahtuu ArmWidgetin säätimen avulla, joka muuttaa halutun varren kulmaa. Kulman muutos tapahtuu Arm-objektin sisällä, joka muuttaa kulman halutuksi, ja käskee tämän jälkeen Animator-oliota suorittamaan animaation, joka aikaansaa käden liikkumisen. Käyttäjä voi myös muuttaa varsien pituuksia, jolloin sama ArmWidget muuttaa Arm-objektin pituutta, joka päivitetään DrawWindowiin.

Käyttäjä voi myös muuttaa varsien asetuksia TextEditor:in avulla, jolloin hän voi suoraan kirjoittaa haluamansa kulmien muutokset. TextEditor käsittelee annetut syötteet, ja määrittää tarvittavat käskyt, joilla se muuttaa ArmWidgetin arvoja, mikä puolestaan aikaansaa käden liikkumisen.

Painaessaan Add Box to field nappia BoxWidgetistä, käyttäjä voi asettaa laatikon DrawWindow:n kentälle hiiren avulla. Tämän jälkeen hän voi siirtää käden asentoon, jossa viimeinen Node on laatikon päällä, ja tarttua tähän. Tämän jälkeen käden asentoa muutettaessa laatikko kulkeutuu mukana, kunnes käyttäjä vapauttaa laatikon samasta napista, ja laatikko jää paikoilleen.

Jos käyttäjä haluaa muuttaa varsien lukumäärää, hän valitsee menu:sta eri määrän. Tämä käynnistää GUI:ssa piirtofunktion uudella lukumäärällä, joka luo kentän uudelleen alkutilanteeseen eri määrällä varsia.

3. **Algoritmit**

Ohjelma on simulaatio, jossa käyttäjä muuttaa itse suoraan kaikkia parametreja, joita ovat käytännössä käden varsien pituus ja kulma, sekä kentän laatikoiden sijainti ja lukumäärä. Laatikoiden suhteen ei tarvita mitään algoritmeja, sillä laatikoiden sijainti tulee suoraan joko hiiren painalluksesta, tai liikkuvan varren end_node:n perusteella. Varsien ja nivelten suhteen lähinnä niiden koordinaattien ilmoittaminen pääikkunan koordinaatistossa vaatii hieman laskemista, sillä yhden objektin koordinaatisto on riippuvainen sen isäntäobjektien koordinaatistoista. Tässä voisi soveltaa ”Forward Kinematics” matikkaa, joka keskittyy juuri tähän ongelmaan mutta Qt5 valmiiksi rakennetuissa metodeissa on scenePos(), joka palauttaa kyseisen objektin sijaintipisteen koordinaatit ikkunan scene koordinaateissa. Eli riittää pitää huolta, että jokaisen varren kulmat ja pituudet ovat oikein, niin niiden avulla sijainti on aina saatavilla.

4. **Tietorakenteet**

Ohjelman suoritus ei tallenna mitään. Ohjelman luokat pitävät sisällään kaikki ohjelman suorittamiseen tarvittavat muuttujat, joita ovat esimerkiksi objektien sijainnit, listat lisätyistä objekteista ja graafisten elementtien parametrit. Tämän kaiken toteuttamiseen riittävät pythonin ja PyQt5n omat tietorakenteet.

5. **Aikataulu**

Suunnittelen rakentavani ohjelman seuraavassa järjestyksessä:

-Aloitan DrawWindowin ja siihen liitettävien elementtien ArmGraphicsItemin ja NodeGraphicsItemin luomisella. Tämä on ohjelman tärkein osa, ja näiden luokkien toiminta ja piirtyminen oikein rakentaa pohjan muulle toiminnalle. (**10** h)
-Seuraavaksi aion liittää DrawWindowin GUI luokkaan, ja lisätä ohjelmaan ArmWidgetit, ja saada varsien säätämisen toimimaan käyttöliittymästä. (**8** h)
-Lisään mahdollisuuden muuttaa varsien lukumäärä, ja luon valikon GUI:iin (**6** h)
-Luon BoxGraphicsItemin ja BoxWidgetin (**3** h)
-Rakennan ohjelmaan mahdollisuuden lisätä laatikoita, ja siirtää niitä kentällä (**6** h)
-Luon TextEditor luokan, ja rakennan siihen liitettävät käskyt (**6** h)
-Lisään TextEditorin käskyt muokkaamaan ArmWidgettien arvoja (**4** h)
-Luon Animator luokan, ja liitän siihen varren liikkumisen animoinnin (**10** h)
-Tässä vaiheessa ohjelman kaikki komponentit on ohjelmoitu, ja siirryn testaamaan ja hiomaan sen toimintaa. (**10** h)

6. **Yksikkötestaussuunnitelma**

Ohjelman alkuvaiheessa luodaan robotin keskeiset graafiset osat ja mahdollisuus liikuttaa niitä. Testauksen ydintavoitteena olisi varmistaa, että robotin osat piirtyvät aina oikein ja niiden liikuttaminen tuottaa halutun lopputuloksen. Tämän testaus hoituu käytännössä luomalla robotti ikkunaan, antamalla sille lähtöarvot ja liikuttamalla tämän jälkeen varsien pituuksia ja kulmia. Vertaamalla objektien muuttuneita sijainteja oletettuihin arvoihin voidaan tarkistaa niiden olevan oikeassa paikassa. Esimerkiksi robotin ollessa suoraan y-akselin suuntaan, kääntämällä robottia 90 astetta myötäpäivään tulisi sen pään palauttama sijainti olla origosta robotin pituuden verran negatiivisen x-akselin suuntaan. Testausta varten on luotava metodit, jotka palauttavat kunkin objektin sijainnin, jotta niitä voidaan verrata oikeisiin arvoihin.

Objektikohtaisia testejä voi kirjoittaa esimerkiksi return_position tai return_angle metodeille, sekä testata DrawWindow:n sisällä olevien objektien tyypin ja lukumäärän olevan oikein.

7. **Kirjallisuusviitteet ja linkit**

Olen käyttänyt, ja tulen käyttämään erityisesti Qt:n ja pythonin omaa dokumentaatiota, yrittäessäni ohjelmoida minulle uusia graafisia elementtejä. Olen myös tutkinut hieman Forward Kinematics teoriaa netistä, ja yrittänyt sen kautta ymmärtää robottikäden koordinaattien laskemista. Youtubesta olen löytänyt hyvän PyQt perustutoriaalin, josta uskon olevan hyötyä. Aion myös hyödyntää Stackoverflow foorumia siltä osin, kun löydän sieltä omaan tapaukseeni liittyviä neuvoja.

https://doc.qt.io/qt-5/
https://wiki.qt.io/Category:HowTo
https://docs.python.org/3/contents.html
https://stackoverflow.com/
https://cs.gmu.edu/~kosecka/cs685/kinematic-chains.pdf
https://en.wikipedia.org/wiki/Forward_kinematics
https://www.youtube.com/playlist?list=PLQVvvaa0QuDdVpDFNq4FwY9APZPGSUyR4