Pienoisrautatie

Juho Aalto 651297

Informaatioteknologia 2. vsk

27.4.2020

1 Yleiskuvaus

Projekti *Pienoisrautatie* käsittelee ohjelmaa, jolla on mahdollista suunnitella pienoisrautateitä. Käyttäjällä on mahdollisuus luoda oma pienoisrautatiensä lisäämällä, poistamalla ja liikuttelemalla erilaisia paloja.

2 Käyttöohje

Ohjelma aukeaa suorittamalla *RailwayApp*-tiedosto. Ohjelma aukeaa tilaan, josta käyttäjä voi aloittaa vapaan suunnittelun tai ladata aikaisemmin tallennetun suunnitelman muokattavaksi. Käyttäjällä on käytössään ruudun oikeassa reunassa olevat painikkeet sekä paljon erilaisia näppäinkomentoja. Lisätyn palan voi valita klikkaamalla sitä, jolloin se muuttuu keltaiseksi. Vain valitut palat liikkuvat komentojen mukaan. Suunnittelualueen alareunassa on pieni tekstikenttä, jonka teksti kuvaa tapahtumaa, joka palalle on suoritettu tai minkänlainen pala on valittu.

Näppäin	**Tapahtuma**
A	vaihda kaarevan palan kulmaa 30/45 astetta
S	vaihda vaihteen asentoa
R	pyöritä myötäpäivään
E	pyöritä vastapäivään
+	pidennä palaa/kasvata kääntösädettä
-	lyhennä palaa/pienennä kääntösädettä
ylös	liikuta palaa ylöspäin
alas	liikuta palaa alaspäin
oikealle	liikuta palaa oikealle
vasemmalle	liikuta palaa vasemmalle
delete	poista pala

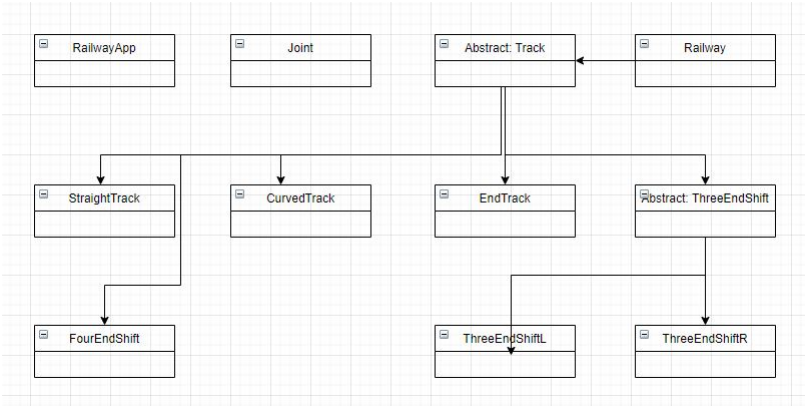
Oikean reunan painikkeet ovat seuraavat lueteltuna ylhäältä alas.

- Lataa tallennus
- Tallenna tämänhetkinen näkymä
- Lisää suora pala
- Lisää kaareva pala
- Lisää stopperi

- Lisää vaihde, jossa on kolme vapaata päätä ja joka kaartuu vasemmalle
- Lisää vaihde, jossa on kolme vapaata päätä ja joka kaartuu oikealle
- Lisää vaihde, jossa on nejä vapaatä päätä
- Poista valittu pala
- Poista kaikki palat

Ohjelma lopetetaan ylävalikosta File → *Exit*.

3 Ohjelman rakenne



Luokkajako Luokkajaon ylimpänä osana on luokka *Railway*, joka kuvastaa yhtä rautatiekokonaisuutta. Tämä luokka pitää sisällään palojen lisäämisen ja poistamisen, silmukoiden tutkimisen, erityyppisten palojen kuvaukset ja palan valitsemisen.

Abstrakti luokka *Track* esittää yhtä radanpalaa, jota tästä periytyvät muut luokat täsmentävät. Tämä luokka hallitsee metodeja, jotka ovat yhteisiä kaikille paloille mm. pyöritys ja siirtely.

Luokka *Joint* huolehtii radanpalan liitoskohdista, joita on 1-4 kappaletta riippuen palan tyypistä.

Luokat StraightTrack, CurvedTrack, EndTrack kuvaavat nimiensä mukaisesti erilaisia radanpaloja ja periytyvät luokasta *Track*.

Tiedostossa *Shifts* on abstrakti luokka *ThreeEndShift*, josta perityy luokat oikealle *ThreeEndShiftR* ja vasemmalle *ThreeEndShiftL* kaartuvista, kolme vapaata päätä

sisältävistä vaihteista. Näiden lisäksi on luokka *FourEndShift*, joka kuvastaa puolestaan neljä vapaata päätä sisältävää vaihdetta.
Graafisen käyttöliittymän toteuttamseen käytetään seuraavia luokkia Button, Label, Panel, MenuBar, BorderPanel ja Boxpanel.

Track-luokan ja siitä periytyvien luokkien tärkeimmät metodit ovat:
connect Yhdistää palan toiseen, jonka lähellä se on. Ks. kohta Algoritmit.
*connectionPoints*Palauttaa pisteet, joissa palan päät ovat
*contains*Palauttaa totuusarvon siitä, onko piste alueen sisällä
*draw*Piirtää muodon.
*haveEnoughSpace*Tutkii voiko palan sijoittaa tähän. Ks. kohta Algoritmit.
*moveXX*Siirtää palaa 5 verran suuntaan XX.
*neighbours*Palauttaa listan palan naapureista.
overlaps Kertoo onko pala jonkun muun palan päällä.
rotate, inverseRotate Pyörittää palaa.
setPosition Asettaa palan haluttuun paikkaan.
trackToList Palauttaa merkkijonolistan palan tiedoista.

RailawayApp:in tärkeimät metodit ovat
setCommentary Vaihtaa infotekstin.
deletePiece Poistaa palan
containsOverlaps Tarkistaa, onko radassa päällekkäisyyksiä.
addX Lisää tyypin X palan rautatiehen.
neighboursOfPieces Palauttaa kaikkien palojen kaikki naapurit

4 Algoritmit

Ohjelman merkittävimmät algoritmit ovat palojen toisiinsa liittämiseen ja silmukoiden tutkimiseen tarvittavat algoritmit.

Palojen liittäminen toimii *Track*-luokan *connect*-metodilla, joka etsii käsiteltävästä rautatiestä kaikkien palojen liitospäät *(Joint)*. Näistä poistetaan käsiteltävän palan omat liitoskohdat. Tästä muodostetusta listasta etsitään ne liitoskohdat, jotka ovat kohdikkain ja samassa sijainnissa kuin jokin tämän palan liitoskohdista. Tässä apuna toimii liitoskohdan kohdalle luotu neliö (joka näkyy kun pala on valittuna), siten että palat ovat riittävän lähellä yhdistämisen toteuttamiseksi, kun nämä neliöt ovat toistensa päällä.

Silmukoiden etsimiseen käytetty algoritmi *checkLoops* sijaitsee luokassa Railway, joka etsii kaikki palat, joilla on vähintään 2 liitettyä päätä (=vaatimus palalle, jotta se voisi olla silmukassa). Tämän jälkeen tutkitaan äsken löydetystä paloista, ovatko kaikki palat toisillensa naapureita. Jos ovat, on silmukka muodostunut. Tätä metodia kutsutaan yhdessä *connect*-metodin kanssa, kun palaa liikutetaan tai käännetään.

Palojen piirtämiseen tiettyyn kulmaan ja sopivan kaaren piirtämiseen käytetään tavalllisia trionometrian funktioita $sin(x) = a/c$ ja $cos(x) = b/c$.

Palojen sijoittamisen sellaiseen kohtaan, jossa sen päähän ei mahtuisi sijoittamaan enää edes päätöspalaa estää *Track*-luokassa sijaitseva metodi *haveE-*

noughSpace, joka hyötyy *Joint*-luokan metodista *enoughSpace*. Palan väri muuttuu punaiseksi, mikäli liitoskohtaan luotu neliö osuu toiseen palaan, joka ei ole kuitenkaan liitetty tähän.

5 Tietorakenteet

Tietorakenteita projektissa käytetään palojen hallitsemiseen rautatiessä ja palan liitoskohtien sijainteihin ja asentoihin. Myös yllä mainintut algoritmit käyttävät väliaikaisesti tietorakenteita sopivien palojen hallintaan. Tietorakenteista otin käyttöön sellaisia, joisden tila on muuttuva, jolloin ne palvelevat tarkoitustaan paremmin tässä projektissa.

6 Tiedostot

Ohjelma käyttää tietojen tallentamiseen csv-tiedostoja. Tiedoston ensimmäiselle riville on tallennettu eri tyyppisten palojen kuvaukset, joilla ns. oletuskuvaukset korvataan, kun tallennus ladataan. Loput rivit sisältävät kaiken tiedon paloista, jotka rakennelmaan kuuluu.

Ensimmäisellä rivillä tunniste RAILWAY, jonka jälkeen paljon kuvaukset järjestyksessä suora, kaareva, stopperi, vasemmalle kaartuva 3 vapaan pään vaihde, oikealle kaartuva 3 vapaan pään vaihde, 4 vapaan pään vaihde

Lopuilla riveillä tiedot ovat: palan tyyppi, x-suuntainen sijainti, y-suuntainen sijainti, pyörityskulma, mahdollinen pituus/kaartuvuus säde.

Merkki	**Selitys**
S	suora
3	30 asteen kaarre
4	45 asteen kaarre
E	stopperi
TL	vasemmalle kaartuva vaihde kolmella vapaalla päällä
TR	oikealle kaartuva vaihde kolmella vapaallaa päällä
F	vaihde neljällä vapaalla päällä

7 Testaus

Sunnitelmasta poiketen en luonut erillistä testausohjelmaa, vaan kokeilin ohjelmaa graafisen käyttöliittymän avulla, jonka työstämisen aloitin jo aikaisessa vaiheessa. Pyrin käymään läpi kaikki sellaiset tilanteet, joita voisi syntyä ohjelman käytössä erikoisimmat tilanteet mukaan lukien. Graafinen käyttöliittymä oli suuri apu etenkin erilaisten palojen luomisessa sekä liitoskohtien testaamisessa, sillä se auttoi hahmottamaan ongelman nopeasti.

8 Puutteet ja viat

Neljäpäisen vaihteen toimimimaan saaminen silmukan tunnistuksen kanssa osoittautuu oletettua hankalammaksi, joten se ei katkaise silmukkaa, vaikka vaihde olisi käännetty toiseen asentoon. Mikäli tekisin vaihteen alusta uudelleen, jakaisin sen todennäköisesti neljään erilliseen radanpalaa, joista aina kerrallaan kaksi olisi näkyvissä.

Luokan *Track* metodi *setPosition* ei osaa käsitellä virheellsiä arvoja, joita vikaantuneesta tiedostosta saattaa tulla. Tämän voisi korjata asettamalla jonkinlaisia ehtoja arvoille ja muokata metodia sellaiseksi, että se osaisi muokata niitä sopiviksi.

Tiedostoa, josta luetaan tai johon kirjoitetaan ei käyttäjä pysty vaihtamaan. Tämän vian voisi korjata kysymällä asiaa joka kerta esimerkiksi tekstikentällä kun käyttäjä haluaa ladata tai kirjoittaa tiedostoa.

Silmukoiden tutkiminen on toteutettu kömpelöllä tavalla. Sen voisi muokata sujuvammaksi jonkinlaisella rekursiivisella metodilla ja se voisi tallentaa listaa jokaisen löytämänsä silmukan, jolloin niiden lukumäärän voisi laskea. Tällä hetkellä saadaan ainoastaan tieto, mikäli sellainen on syntynyt.

9 3 parasta ja 3 heikointa kohtaa

Ohjelman vahvuudet

- Joustavuus
- Selkeys
- Monipuolisuus

Ohjelman heikkoudet

- Tallennusten hallinta
- Silmukat
- Rajat

Tarkennukset:

Luomani *constants.scala*-tiedosto auttaa säätämään ohjelmien ominaisuuksia, ympyräsektoreista muodostetut kaarteet ja radanpalojen pidennys- ja lyhennystoiminnot tekevät ohjelmasta hyvin joustavan muokata. Esimerkiksi ohjelmakoodin muokkaaminen sellaiseksi, että 90-asteen kaarteet olisivat mahdollisi on hyvin helppoa, kuin myös värien muuttaminen *contants*-tiedostoon. Nämä ominaisuudet ovat tuoneet ohjelmaan monipuolisuutta ja joustavuutta. Selkeyden puolesta olen tyytyväinen ratkaisuun, että paloja voi pyörittää ja siirrellä 5 pikselin välein.

Radan tallennusta ja lukemista olisi voinut parantaa siten, että rikkoutuneen tiedoston lukeminen olisi varmempaa. Myös käyttäjälle olisi voinut tehdä toiminnon tiedoston valintaa varten.

Silmukoita olisi voinut tarkastella siten, että jokaisen muodostuneen silmukan olisi listannut ja sitten laskenut silmukoiden määrän.

Palojen mahdollisia liikutusrajoja olisi voinut tarkentaa entisestään.

10 Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Poikkesin suunnittelemastani aikataulusta muiden kurssien tuoman työmäärän vuoksi. Vaikka työn tekeminen jäi koko annetun ajan loppuvaiheeseen, ei se mielestäni vaikuttanut huomattavasti projektin etenemiseen/valmistumisen. Laatimani aikatalun viikkoaikataulutus ei onnistunut näin ollen, mutta tein asiat kuitenkin siinä järjestyksessä, kuin ne oli aikataulussa mainittu.

Aikaisemmin mainitun mukaisesti yksikkötestauksessa toimittiin suunnitelmaan nähden eri tavalla. Teknisen suunnitelman UML-kaavio on mielestäni onnistunut hyvin, kun sitä vertaa lopputulokseen: Luokat ja metodit ovat melko hyvin toteutuneet. Muutamat käyttöliittymän toiminnallisuudet ovat muuttuneet suunnitelmasta, mutta ne selkeentyivät itselleni vasta projektin aikana.

Projektin aikataulutus olisi pitänyt suunnitella paremmin muiden kurssien ohelle eikä niiden jälkeen.

11 Kokonaisarvio

Tässä työssä kehityin paljon ohjelmoinnissa ja opin paljon sovellusprojektin vaiheista. Työn aikana huomasin, että valitsemani aihe oli itselleni melko vaikea, mutta jälkeenpäin olen tyytyväinen, että haastoin itseni.

Jatkaisin projektin kehittämistä paikkailemalla aikaisemmin mainittuja vikoja: Vaihdeonelma, tiedostojen tallennus, siirtelyrajat ja silmukoiden tunnistaminen.

Ohjelman rakenne on mielestäni hyvin joustava ajatellen tulevaisuudessa tehtäviä muutoksia, mihin olen hyvin tyytyväinen.

Jos nyt alottaisin projektin, suunnittelisin muut kurssit ja tämän projektin aikataulutuksen eri tavalla. Ohjelman rakenteen puolesta keskittyisin hieman enemmän toimivuuteen, karsien muokattavuuden ominaisuuksista.

12 Viitteet

1. https://www.overleaf.com/learn/
2. https://plus.cs.aalto.fi/o1/2019/
3. https://www.stackoverflow.com
4. https://www.scala-lang.org/api/2.13.2/index.html
5. https://docs.oracle.com/javase/7/docs/api/overview-sum
6. https://www.maerklin.de/en/products/gauge-h0/tracks/

13 Liitteet

1. Lähdekoodi
2. Ruutukaappaus