Ohjelmointistudio 2
Lennonjohtopeli: Dokumentti

Oskari Mäkinen
477510
EST/ ELEC, 4. vuosikurssi
23.4.2019

Yleiskuvaus

Tehtävänanto:

Tee lennonjohtopeli. Pelissä on eri vaikeustasoja, joilla on eri määrä ”satunnaisesti” sijaitsevia kiitoratoja ja laitureita. Lentokoneiden saapumistiheys riippuu myös vaikeustasosta. Kiitoradat ovat erilaisia ja eri koneet tarvitsevat erilaisia kiitoratoja (radan minimipituus). Kiitoradat voivat olla myös ristikkäin.

Ohjelmaan voi lisätä jännitystä esim. sääolojen vaihtumisella sekä monilla muilla erilaisilla erityistilanteilla. Lentokoneita voi pitää kentän yläpuolella eri korkeuksilla yms.

Tavoitteena oli siis luoda peli, jossa saapuville ja lähteville lentokoneille annetaan käskyjä, jotta lentoliikenne pysyy sulavana. Ruuhkia, myöhästymisiä ja törmäyksiä tulee välttää. Lähtökohtanani oli mukailla Nintendo DS:n Air Traffic Chaos -peliä (https://en.wikipedia.org/wiki/Air_Traffic_Chaos) varsinkin pelillisen toiminnallisuuden suhteen. Lopputulos mielestäni mukailee inspiraation lähteenä ollutta peliä hyvin, vaikka onkin yksinkertaisempi.

Työ on toteutettu vaikeimmalla vaatimustasolla, sisältäen graafisen käyttöliittymän, useita ikkunoita, sekä mielekkäästi liikkuvat koneet.

Käyttöohje

Ohjelman käynnistämiseksi koneella tulee voida ajaa Scala-ohjelmia. Ohjelma tarvitsee toimiakseen Scala Swing-kirjaston. Seuraavia ohjeita noudattamalla peliä pitäisi pystyä ajamaan Eclipse Scala IDE:ssä. Oletuksena on, että Eclipse ja Scala toimii koneella.

1. Swing kirjaston käyttöönottamiseksi projekti hyödyntää O1Library-kirjastoa.
   https://grader.cs.hut.fi/static/O1_2018/projects/given/O1Library/
   Lataa O1Library.zip ja importtaa se projektina Eclipseen

2. Testit vaativat toimiakseen testikirjaston.
   https://grader-host.cs.hut.fi/static/studio2-2019/projektit/annetut/TrainTests/
   Lataa ja tuo projekti (vain TestingLibraries riittää) Eclipseen, kuten edellä.

3. Lopuksi tuo itse ”lennonjohtopeli” -projekti Eclipseen. Nyt pelin tulisi käynnistyä, kun ajat Main.scala. Testit käynnistyvät ajamalla Tests.scala.

Nämä ohjeet on testattu ja todettu toimivaksi Windows-koneella, jossa puhdas Eclipse Scala IDE asennus.

Peli koostuu neljästä eri ikkunasta:

Pääikkuna (Air Traffic Control)

New Game – käynnistää uuden pelin
Restart – aloittaa uuden pelin, kun käynnissä oleva peli on kesken tai hävitty

Lentokenttä (Airfield)

Tämä ikkuna näyttää visuaalisesti pelitilanteen. Keskellä on kiitoratoja, joille voi laskeutua. Kiitoratojen ympärillä on kaksi rengasta, joilla koneet kiertävät ennen laskeutumista. Oikealla on portit, joilla laskeutuneet koneet odottavat.

Saapuvien koneiden lista (Arrivals)

Painikkeet aktivoituvat tilanteen mukaan.
Dsc (Descend) – alenna korkeutta. Lentokone siirtyy sisemmälle kiertoradalle ja alentaa nopeuttaan.
Lnd (Land) – laskeudu. Avaa valikon, josta voi valita, mille kiitoradalle laskeudutaan.
Gat (Gate) – Määritä portti. Avaa valikon, josta voi valita portin, jolle kone siirtyy laskeuduttuaan.

Lähtevien koneiden lista (Departures)

Takeoff – Lähetä lentokone takaisin lentoon. Avaa valikon, josta voi valita kiitoradan nousua varten.

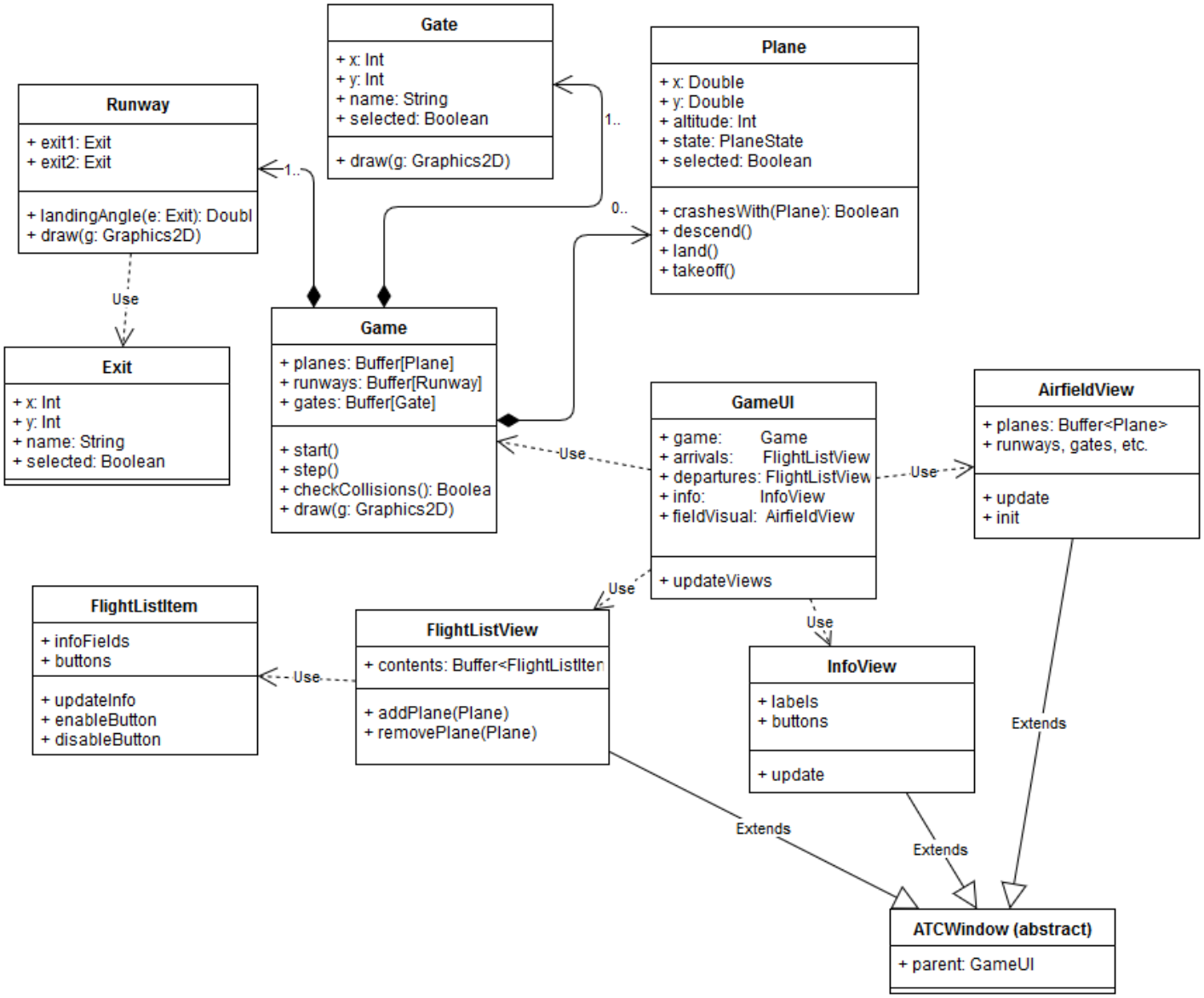
Käynnistettyäsi ohjelman aloita peli painamalla ”Start Game”. Jos koneet törmäävät, saat tästä ilmoituksen ja häviät pelin. Voit aloittaa uuden pelin tämän jälkeen painamalla ”Restart”. Pelissä saat pisteitä, kun kone on onnistuneesti poistunut lentoasemalta.

Ulkoiset kirjastot

Graafisten ominaisuuksien hyödyntämiseksi ohjelmassa on käytössä Scala Swing -kirjasto. Testaamiseen on käytetty scalatest-kirjastoa.

Ohjelman rakenne

Tässä toteutuneen ohjelman UML-luokkakaavio. Malli on yksinkertaistettu todellisesta, näyttäen vain luokkien keskeiset muuttujat ja funktiot.



Game – Pelilogiikan pohjaluokka. Pitää kirjaa koneista, porteista ja kiitoradoista. Pitää huolta pelitilanteen edistämisestä step-funktion avulla.

Plane – Huolehtii lentokoneiden liikkumislogiikasta ja piirtämisestä

Gate – Yksinkertainen luokka, joka huolehtii porttien piirtämisestä

Runway – Luokka, joka pitää huolta kiitoratojen piirtämisestä ja lähestymiskulman laskemisesta.

Exit – Kiitoradan pääty.

GameUI – Käyttöliittymän pohjaluokka. Huolehtii eri ikkunoiden luomisesta ja päivittämisestä.

ATCWindow – Yliluokka pelin ikkunoille, huolehtii kaikille ikkunoille yhteisestä toiminnallisuudesta

GameInfoView – Näyttää pelin nykyiset pisteet, ja sisältää painikkeet uuden pelin aloittamiseen.

AirFieldView – Esittää graafisesti pelitilanteen. Päivitetään 10ms välein

FlightListView – Näyttää allekkain FlightListItemeita, ja huolehtii näiden lisäämisestä ja poistamisesta.

FlightListItem – Näiden painikkeilla koneille voi antaa käskyjä. Alarivillä kuvaus koneen tilasta, sekä tekstikenttä, johon tulee ajoittain ilmoituksia.

Toiminnot määräytyvät sen mukaan, onko tämä Arrivals- vai Departures-ikkunassa.

Algoritmit

Törmäyksentarkistus: checkCollisions() (Game-luokassa) käy läpi jokaisen koneparin, ja kutsuu parille crashesWith-metodia. Jos koneet ovat samalla korkeudella, ja niiden etäisyys x- ja y-suunnissa on alle 32, on törmäys tapahtunut ja palautetaan true, muussa tapauksessa palautetaan false.

Vaihtoehtoinen tapa tarkistaa törmäykset olisi ollut järjestää koneet ensin x- ja y-koordinaattien perusteella. Tämän jälkeen vain kahta vierekkäistä konetta olisi tarvinnut verrata, mikä olisi mahdollisesti voinut vähentää laskenta-aikaa. Koneita on kuitenkin suhteellisen vähän, joten en nähnyt tarpeelliseksi muuttaa toteutustani ainakaan tässä vaiheessa.

Pistelasku: – Kun kone on noussut ja poistunut pelistä, lisätään pelaajan pistesaldoon koneesta saatavat pisteet. Eri konetyypeistä saa eri määrän pisteitä.

Kiitoradan kiertokulma: Lasketaan kaavalla: atan(päätyjen x-koordinaattien etäisyys / päätyjen y-koordinaattien erotus). Tätä lukemaa käytetään kiitoratojen piirtämiseen, ja se auttaa myös koneiden tulokulman laskemisessa.

Tietorakenteet

Ohjelmassa on käytetty vain muuttuvatilaisia tietorakenteita tiedon varastointiin. Tärkein näistä on Buffer, jota käytetään monessa kohtaa koneiden ryhmittämiseen. Bufferit ovat hyvin kätevä tähän tarkoitukseen, sillä niitä on helppo iteroida ja muokata. FlightListView käyttää myös HashMap-tietorakennetta, joka yhdistää koneen ja siihen liittyvän FlightListItemin.

Joissain tapauksissa olisi ollut järkevämpi käyttää muuttumattomia tietorakenteita (esim. lista kiitoradoista), mutta en ehtinyt tehdä näitä muutoksia.

Tiedostot

Tällä hetkellä ainoat pelin käyttämät tiedostot ovat png-kuvia, joita hyödynnetään koneiden ja kiitoratojen piirtämisessä. Jos olisin ehtinyt edetä pidemmälle, olisin myös käyttänyt äänitiedostoja ääniefekteille, sekä luonut yksinkertaiset tekstitiedostoformaatit ennätyspisteiden tallentamiseen ja erilaisten lentokenttien esittämiseen.

Testaus

Alkuperäisistä suunnitelmista:

> Yksi keskeinen testattava asia ohjelmassa on törmäyksentunnistus. Ohjelman tulee pystyä tunnistamaan, mikäli kaksi konetta ovat liian lähellä toisiaan. Tätä voi testata simuloimalla erilaisia tilanteita kentällä ja katsoa, tunnistetaanko törmäys. Virheellisiin tai puutteellisiin tiedostoihin tulee myös reagoida, tätä voi helpoiten testata syöttämällä virheellistä dataa. Pelin yleistä toimivuutta on luonnollista testata pelaamalla itse peliä. Tässä testausta voi vauhdittaa esimerkiksi nopeutustoiminto, tai mahdollisuus luoda ja tallentaa tilanteita.

Ohjelman testaus toteutettiin pääasiassa käyttämällä ohjelmaa. Virheellistä toimintaa havaitessani paikansin, missä kohdassa koodia virhe ilmenee. Virheen juurisyy ja ratkaisu löytyi usein println-komentoja ja Eclipsen debuggeria hyödyntämällä. Joskus taas nettihaut ja Swingin tai Scalan dokumentointi toivat ratkaisun ongelmaan. Uusia ominaisuuksia luodessani testasin niitä usein myös erillään muusta ohjelmasta, ja vasta toimivuuden varmistuttua integroin ominaisuudet ohjelmaan. Tämän jälkeen testasin vielä, että uudet osat toimivat hyvin yhteen muun ohjelman kanssa.

Koneiden liikkumista oli helpoin testata visuaalisesti, sillä kentällä liikkuvista koneista näki nopeasti, jos liike ei ollut halutunlaista.

Tein yksinkertaiset yksikkötestit törmäyksentunnistusalgoritmille, ja ohjelmani läpäisi ne. Joitain muitakin keskeisiä metodeja olisin halunnut testata, mutta tähän ei aikani riittänyt.

Ohjelman tunnetut puutteet ja viat

Ei erilaisia kenttiä tai vaikeustasoja – Pelissä on tällä hetkellä vain yksi kiinteästi koodattu lentokenttä. Suunnitelmanani oli varastoida kentät tekstitiedostoihin, jolloin uusia olisi helppo lisätä koodiin koskematta. Mielestäni peli on kuitenkin rakenteeltaan sellainen, että tämän toiminnallisuuden lisääminen kävisi suhteellisen vaivattomasti. Vaikeustasoja en ehtinyt myöskään toteuttaa. Vaikeutta olisi voinut helposti muuttaa lisäämällä koneiden saapumistiheyttä, tai rajoittamalla koneiden ilmassa oloaikaa.

Pelin lopetus – Pelin lopetuksen toteutus on vielä hyvin yksinkertainen. Suunnitelmana oli toteuttaa lopetusanimaatio, kun koneet törmäävät. Tämän jälkeen olisin toteuttanut myös huippupisteiden tallennuksen.

3 parasta ja 3 heikointa kohtaa

+Lentokoneiden käyttäytyminen – Mielestäni olen saanut lentokoneet liikkumaan melko aidontuntuisesti.

+Luokkajako – Luokkajako on mielestäni järkevä, ja toiminnallisuudet on jaettu siten, että ne on suhteellisen helppo löytää ja ohjelman laajentaminen ei ole kovin vaivalloista. Olen pyrkinyt luomaan luokat asioista, joita tarvitsee monistaa paljon, sekä asioista, joiden kanssa on helpoin vuorovaikuttaa metodien avulla. Olen myös pyrkinyt mahdollisimman hyvin erottamaan peli- ja käyttöliittymälogiikan.

+Scalan ominaisuuksien hyödyntäminen – Mielestäni ohjelmassa on hyödynnetty monipuolisesti ja järkevästi Scalan toiminnallisuuksia, kuten foreach- ja filter-metodien putkittamista, sekä erilaisia suunnittelumalleja, kuten State (Lentokoneen tilaa kuvaava PlaneState on keskeinen osa Plane-luokkaa), tai tehdasmetodeja eri konetyyppejä ja FlightListItemeita luotaessa.

-Visuualinen ilme – Peli näyttää vielä paikka paikoin melko karulta ja alkeelliselta. Ilmettä voisi melko helposti kohottaa lisäämällä kuvakkeita ja tekstuureita, ja esimerkiksi osan painikkeista voisi näyttää kuvakkeiden avulla. Käyttöliittymäkomponenttien asettelu kaipaa myös viilausta

-Kommenttien vähyys ja koodin viimeistelemättömyys – Koodista saisi ymmärrettävämpää runsaammalla kommentoinnilla. Muutenkin koodia voisi siistiä enemmän ja esimerkiksi muuttujien näkyvyys pitäisi tarkistaa.

-Pelimäisyys – Ks. puutteet ja viat. Myös lentokoneista olisi voinut tehdä erilaisempia, nyt ne eroavat vain pistemäärän ja hidastuvuuden osalta.

Poikkeamat suunnitelmasta

Ominaisuuksia kehittäessä uusia luokkia syntyi ja joitain ei lopulta tarvittukaan. Tämä oli kylläkin odotettavissa, sillä usein vasta toteutusvaiheessa havaitsee tarpeen uusille luokille. Uudet luokat vähensivät koodin toisteisuutta ja estivät muiden luokkien paisumisen valtaviksi. Esimerkiksi Flight-luokan totesin tarpeettomaksi, kun taas ATCWindow oli todella hyödyllinen ikkunoiden pohjaluokaksi.

Aikataulu myös muokkaantui hieman. Lähdin lopulta toteuttamaan peliä käyttöliittymä edellä (eikä logiikka), mikä osoittautuikin hyväksi ratkaisuksi, sillä väärin liikkuvat koneet oli helppo havaita nopeasti.

Aikaa kului suunnilleen saman verran kuin suunnitelmassa. Koodin kirjoittamiseen kului ehkä suunniteltu noin 140 tuntia, mutta ainakin saman verran aikaa kului testaamiseen, tiedonhakuun ja oikeiden Swingin komponenttien löytämiseen. Monessa tilanteessa ei ollut heti ilmeistä, mitä halutun ominaisuuden toteuttamiseen kannattaisi käyttää. Loppua kohti tämä kuitenkin helpottui, kun Swingin ominaisuudet kävivät tutummiksi.

Toteutunut työjärjestys ja aikataulu

GitLabista näkee tarkemmin toteutuksen eri vaiheet.
<u>Helmikuu</u>
Swingiin tutustumista ja ensimmäisiä kokeiluja 20h

<u>Maaliskuu</u>
Lisää kokeiluja, käyttöliittymän pohjan luominen, sekä pelilogiikan aloittaminen 50h

<u>Huhtikuu</u>
Pelin loppuun hiominen 60h
Dokumentointi: 8h

Arvio lopputuloksesta

Kaiken kaikkiaan projekti onnistui mielestäni varsin hyvin, ja kaikki keskeiset pelaamiseen liittyvät ominaisuudet on toteutettu loogisesti ja pelaajaystävällisellä tavalla. Luokkajako on järkevä ja Scalan ja Swingin ominaisuuksia on hyödynnetty laajasti. Johdonmukainen rakenne edesauttaa myös muutosten ja laajennusten tekemistä. Peruspelaaminen toimii, koneet liikkuvat järkevällä tavalla. Puutteitakin on: Pelin lopetusta ei ole vielä hiottu loppuun asti, ja visuaalinen ilme kaipaisi myös hieman kohennusta, erilaisilla kuvakkeilla ja tekstuureilla. Nämä eivät kuitenkaan mielestäni ole kovin suuria puutteita ottaen huomioon, että muuten ohjelma toimii miltei moitteettomasti. Enemmän puutteista ja niiden ratkaisuehdotuksista edellisellä sivulla.

Ajankäyttö olisi pitänyt suunnitella huolellisemmin. Lentokoneiden liikkumisen toteuttaminen vei yllättävän kauan, jolloin loppuhiomiseen jäi turhan vähän aikaa. Projektin alkupuoliskolla pysyin kuitenkin mielestäni hyvin aikataulussa. Vaikka muutama pelillisesti tärkeä ominaisuus (mm. vaikeustasot, kentät) jäi toteuttamatta, nämä saisi mielestäni melko nopeasti lisättyä peliin myöhemmin.

Viitteet

Ainakin stackoverflow ja stackexchange ovat olleet useaan otteeseen tietolähteenä. Myös Scalan ja Swingin dokumentaatio.

png-kuvakkeet: www.iconarchive.com

Liitteet

Projektin koodi on GitLabissa.

Alla vielä pari kuvakaappausta pelistä.