Luka Salonen
657893
Tietotekniikka
21.4.2019

Projektityön dokumentti

Yleiskuvaus

Projektina on toteutettu formula-peli, jossa pelaajat yrittävät kiertää annetun kilparadan mahdollisimman nopeasti läpi. Pelaajat siirtävät autojaan vuorotellen manipuloimalla auton suuntaa ja vaihteita. Auton nopeus ja kääntösäde riippuvat vaihteesta. Toisin kuin alkuperäisessä suunnitelmassa suunnittelin, päädyin tekemään peliin myös graafisen käyttöliittymän.

Sanoisin, että työ on lopulta toteutettu keskivaikean vaikeusasteen mukaisena.

Käyttöohje

Ohjelma käyynistetään ajamalla sen mukana tuleva RaceUI objekti. Sen jälkeen käyttäjällä on muutama vaihtoehto. Hän voi joko luoda uuden nimimerkin tai aloittaa uuden pelin. Nämä voi kummatkin tehdä työkalupalkin Game osiosta ja uuden pelin voi myös aloittaa ruudun keskellä olevasta New Game -nappulasta.

Kun aloitat uuden pelin, ohjelma pyytää sinua valitsemaan seuraavat asiat:

1. Millä radalla tahdot pelata
2. Kuinka mota pelaajaa tahdot (2-4)
3. Näiden pelaajien nimet lisättyjen pelaajien listasta

Tämän jälkeen ohjelma lataa kartan ja peli alkaa.

Jokaisella pelaajalla on erivärinen pallo joka kuvastaa heidän autoaan. Vasemmassa palkissa nimen kohdalla näkyy vastaava pallo, jotta kukin tietää mikä on oma auto. Vasemmassa palkissa näkyy myös kenen vuoro on, sillä vuorossa olevan pelaajan nelikulmio on tummemman värinen kuin muiden pelaajien.

Radalla näkyvät mustat ruudut kuvastavat seiniä tai muita esteitä, beigen väriset ovat ajotie, harmaat lähtöruutuja, violetti maaliviiva ja siniset ruudut ovat vuorossa olevan pelaajan mahdolliset lailliset siirrot.

Peli etenee siten, että jokainen pelaaja valitsee vuorollaan jonkin näistä vaihtoehdoista niin kauan, että kaikki paitsi yksi pelaaja ovat törmänneet tai jokin autoista on ylittänyt maaliviivan.

Pelin päätyttyä käyttäjältä kysytään vielä tahtooko hän pelata uudestaan tai lopettaa pelin.

Ohjelman rakenne

Ohjelmaan kuuluvat kaikki alkuperäisen suunnitelman luokat, sekä niiden lisäksi muutamia toteutuksen aikana hyödylliseksi kokemiani lisäluokkia. Kuvailen seuraavaksi niiden pääasiallisia vastuualueita sekä keskeisiä metodeja.

RaceUI

Pelin käynnistysolio, RaceUI piirtää käyttöliitymän ja ottaa vastaan käyttäjän käskyjä.

Keskeisiä metodeja:

- buildEverything() nimensä mukaisesti hoitaa ruudulle piirrettävien asioiden kokoamisen ja näyttämisen.
- takeTurn() huolehtii seuraavan pelaajan vuorosta. Se saattaa siirtää autoa eteenpäin tai ohittaa pelaajan vuoron kokonaan, riippuen onko auto kolaroinut.
- newGame() kysyy pelaajalta ennen pelin alkua Käyttöohje kohdassa mainitut kysymykset ja alustaa pelin niiden mukaisesti.

Race

Toimii ikään kuin siltana käyttöliittymän ja pelin varsinaisen logiikan välissä. Pitää myös kirjaa sellaisista asioista kuten kenen vuoro on tai onko joku voittanut pelin.

Keskeisiä metodeja:

- gameOver() tarkistaa onko jokin pelin päättävistä ehdoista täyttynyt.
- nextMove() käskee Track olion siirtää vuorossa oleva auto annettuun paikkaan ja antaa sitten vuoron seuraavalle pelaajalle.

Track

Pitää sisällään kaiken rataan ja autojen liikkumiseen liittyvän logiikan ja säännöt. Esim. laskee mahdolliset liikkeet, tarkistaa onko joku ylittänyt maaliviivan ja missä mikäkin auto on radalla.

Keskeisiä metodeja:

- moveOptions() palauttaa kaikki lailliset liikkeet mitä sillä vuorolla voi tehdä.
- initializeTrack() asettaa autot satunnaisesti lähtöruutuihin.

Car

Mallintaa autoa. Car olio pitää kirjaa kuinka monta vuoroa se on ajanut sekä siitä, onko se ajokuntoinen, eli ei kolaroinut.

Driver

Pitää kirjaa yksittäisen pelaajan nimestä.

IO

Tämä yksittäisolio pitää huolta tietojen kirjoittamisesta tiedostoihin ja niiden lukemisesta.

Keskeisiä metodeja:

- readTrack() lukee annettua tiedostonimeä vastaavan radan tiedot. (Radan nimi, sen ruudut ja sen huippuajat)
- writeNewDrivers() kirjoittaa annetun kuskin nimen drivers.txt tiedostoon.

Coordinates

Kuvastaa yksittäistä pistettä pelikartan (x,y) koordinaatistossa. Sisältää myös metodeja koordinaattien vertailuun.

Keskeisiä metodeja:

- moveTheDiffrence luo uuden Coordinates olion joka on siirtynyt tästä pisteestä annetun pisteen suuntaan niiden erotuksen.

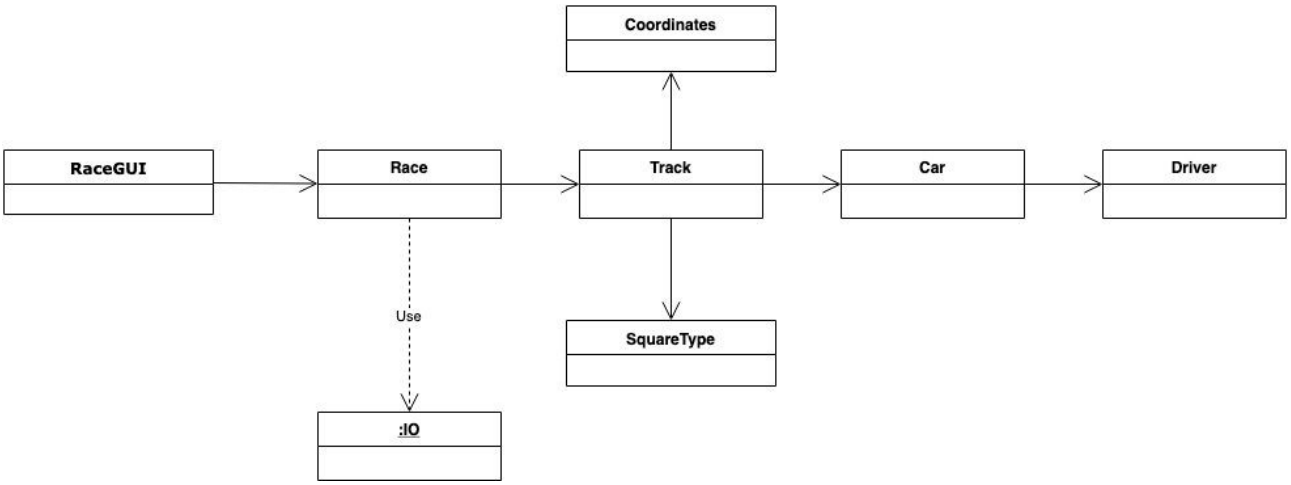
SquareType

Abstrakti luokka jonka aliluokat kuvaavat kaikkia mahdollisia pelilaudan ruutuja. (Este, ajotie, maaliruutu, lähtöruutu ja checkpoint ruutu)

SquareType oliot pitävät kirjaa onko niissä jokin auto ja voiko niiden läpi ajaa.

Exceptions

Muutama oma Exception luokan aliluokka. Eivät oikeastaan ole kovin keskeisiä ohjelman toiminnan kannalta mutta halusin testata niitä kuitenkin.



Algoritmit

Algoritmi, joka etsii kaikki lailliset siirrot toimii jotakuinkin näin.

1. Lasketaan auton edellisen ja tämä hetkisen sijainnin erotus. Esim. jos edellinen sijainti oli pisteessä (5,8) ja nyt ollaan pisteessä (6,10) on erotus (1,2).
2. Liikutaan tämän hetkisestä pisteestä erotuksen verran eteenpäin. Näin saadaan ns. keskipiste mahdollisille liikkeille.
3. Lisätään mahdollisiin liikkeisiin keskipistettä ympäröivät kahdeksan pistettä. Esim. jos keskipiste on (7,12) niin ympäröivät pisteet ovat (6,11), (7,11),(8,11),(6,12),(8,12),(6,13), (7,13) ja (8,13).
4. Poistetaan näistä mahdollisista liikkeistä kaikki, joissa on auto tai jokin muu läpäisemätön este tai joihin päästäkseen auton on kuljettava esteen läpi.

Tietorakenteet

Niin kuin alkuperäisessä suunnitelmassa ajattelin, käytin projektissa tietorakenteina pääasiallisesti yksi- ja kaksiulotteisia taulukoita. Minusta valinta oli hyvin oikeutettu, sillä tietorakenteisiin kohdituvat vaatimukset eivät olleet kovin ihmeellisiä ja yksinkertainen on kaunista.

Tiedostot ja verkossa oleva tieto

Ohjelma lukee ja kirjoittaa kahdenalaisia tekstitiedostoja: sellaisia, jotka sisältävät radan tiedot ja sellaista joka sisältää pelaajien nimet.

Käydään läpi tiedostojen formaattit.

Pelaajien tiedot sisältävä tiedosto (drivers.txt)

Formaatti on hyvin yksinkertainen. Jokaisella rivillä on yhden pelaajan nimi eikä mitään muuta.

Radan tiedot sisältävä tiedosto (london.txt, paris.txt)

Tiedostossa on viidenlaisia rivejä:

1. nameOfTrack:radanNimi
2. lapTime1:pelaajanNimi,kierrosaika
3. lapTime2:pelaajanNimi,kierrosaika
4. lapTime3:pelaajanNimi,kirrosaika
5. Rivi sisältää ainoastaan merkkejä # ¤ g s c

Neljällä ensimmäisellä mainitulla rivityypillä kaksoispisteeseen päättyvä rivin alku on rivin tunniste. Puolipisteen jälkeen tulee kirjoitettu tieto, tässä tapauksessa radan nimi tai parhaat kierrosajat ja ne ajanut kuski. Rivien järjestyksellä ei ole merkitystä mutta jokainen niistä on löydyttävä tiedostosta.

Viidennen kaltaiset rivit taas kuvaavat itse rataa. Niillä ei ole varsinaista tunnistetta vaan niihin lasketaan mukaan kaikki rivit jotka eivät ole tyhjiä ja sisältävät vain yllämainittuja merkkejä. Toisin kuin muiden tyyppisillä riveillä, näiden järjestyksellä on merkitystä. Rivit tulevat pelikarttaan samassa järjestykessä kuin ne ovat tiedostossakin. Kaikkien radan rivien on oltava keskenään saman pituisia. Sama myös sarakkeilla.

Radan rakenteesta vielä sen verran, että maaliviivan (g) ja lähtöviivan (s) on oltava lähellä toisiaan (neljän ruudun sisällä), jotta peli tietää mihin suuntaan autojen kuuluu lähteä ajamaan. Radan noin puoleen väliin tulee myös laittaa ns. ”checkpoint” viiva (c). Tämä viiva on ylitettävä ennen maaliviivaa, jotta pelaajat eivät voi vain kääntyä ympäri lähtöruudussa ja ajaa maaliin. Seiniä (#) ja ajotietä (¤) voi asetella radalle miten huvittaa, niillä ei ole rajoituksia.

Testaus

Ohjelman testaus koostui pääasiassa pelin pelaamisesta ja taktisesti asetelluista println komennoista. Silmätestin mukaan alkuperäisen suunnitelman mukaiset ja muut keskeiset toiminnot kyllä näyttäisivät toimivan, mutta kieltämättä testauksen olisi voinut tehdä paljon paremmin ja johdonmukaisemmin.

Ohjelman tunnetut puutteet ja viat

1. Yksi puute liittyy ratojen parhaisiin kierrosaikoihin. Ohjelma kyllä lukee ja kirjoittaa ne oikein mutta niillä ei oikeastaan tehdä mitään hyödyllistä tällä hetkellä, eikä niitä voi mistään nähdä. Tämän voisi korjata vaikka laittamalla ne sivupalkkiin näkyviin kun radalla pelataan.
2. Välillä ohjelma jäätyy pelin loputtua jos koittaa aloittaa uutta peliä saman tien. En tiedä mistä tämä johtuu tai miten sen voisi korjata.

2 parasta ja 2 heikointa kohtaa

Omasta mielestäni vahvimpia kohtia:

1. Pidän siitä, miten olen käsitellyt poikkeukset. Ne luovat ruudulle ison ja selkeän ikkunan, joka kertoo jonkin mennen pieleen ja näyttää stack tracen vian selvittämiseksi. Minusta ne ovat oikein käteviä ja helpottavat testausta.
2. Tiedostojen lukeminen ja kirjoittaminen toimii mielestäni hyvin. Niillä ei ole ollut ongelmia hoitaa hommiaan kunhan tiedostot ovat jotakuinkin oikeassa muodossa.

Omasta mielestäni ei niin vahvoja kohtia:

1. Koko Track luokka on mielestäni vähän heikko. Se kyllä ajaa asiansa kohtuu hyvin mutta koodi on ajoittain todella todella sekavaa ja vaikeaselkoista. Tämä tekee tietysti laajentamisesta tai bugien korjaamisesta hankalampaa kuin sen ehkä pitäisi olla.
2. RaceUI luokka pitää muistissa kaikki käyttämänsä muodot, sen sijaan että se vain piirtäisi ne ja muistaisi piirretyn kuvan. Se on resurssien käytön kannalta tarpeettoman tuhlaavainen.

Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Projektin toteutusjärjestys oli lopulta hyvin lähellä suunniteltua. Tietysti työstin luokkia osin päällekkäin ja siksi tarkkaa järjestystä (tai päivämääriä) on paha antaa mutta se meni suurin piirtein näin.

1. Driver, Car
2. SquareType, Coordinates, Track
3. Race, IO
4. RaceUI, Exceptions

Alunperin arvelin koko homman vievän noin 50 tuntia ja arviosin sen olevan hyvinkin lähellä lopullista aikaa, ehkä hieman yläkanttiin. Aloitin kuitenkin projektin varsinaisen toteutuksen myöhemmin kuin oli tarkoitus, mikä johti tuntien tiivistymiseen pienemmän ajan sisään.

Yleisesti ottaen työn aikataulutuksen ja työjärjestyksen suunnitelmat onnistuivat odotettua paremmin ja olen niihin hyvin tyytyväinen.

Kokonaisarvio lopputuloksesta

Pienenä kertauksena yllämainituista vahvuuksista ja heikkouksista. Suurimpana heikkoutena pidän koodin vaikealukuisuutta ja monilta osin puutteellista kommentointia. Hyviä puolia ovat mm. se, että suuri osa asioista toimii juuri niin kuin tahdon niiden toimivan. Graafisesta käyttöliittymästäkin tuli lopulta kohtalaisen hyvä vaikka aluksi suunnitelinkin tekeväni pelin ainoastaan komentoriville. Suurin puute on ehdottomasti se, että parhailla kierrosajoilla ei tehdä mitään muuta kuin kirjoitetaan tiedostoon ja luetaan siitä. Jos jatkaisin projektia, ensimmäisenä parannusten listalla olisi tämän

muuttaminen vaikka laittamalla kierrosajat sivupalkkiin esille. Tekisin myös käyttöliittymästä kevyemmän ja kauniimman. Käyttöliittymäsuunnitelu ylipäätänsä alkoi edes jollain tasolla sujua vasta projektin lopussa, joten olisi kiva katsoa mitä saisi aikaan jos sen aloittaisi tyhjältä pöydältä.

Ohjelman rakenteeseen liittyvät seikat ovat mielestäni kohtuullisen hyvin perusteltuja. Luokkien jako sekä tietorakenteiden valinnat toteuttavat ohjelman tarpeet hyvin. Eivät ne ehkä täydellisiä ole mutta ottaen huomioon oman kokemattomuuteni tämän kokoisten projektien kanssa, olen niihin tyytyväinen. Mitä tulee muutoksiin ja laajennuksiin, rakenne tuskin on ongelma. Sen sijaan ylempänä mainittu koodin epäselvyys saattaa tehdä niistä hankalampia kuin niiden tarvitsisi olla.

Jos aloittaisin projektin nyt uudelleen alusta tekisin tekisin monia mainittuja asioita toisin. Ensimmäinen asia olisi tehdä enstistä selkeämpi ja konkreettisempi suunnitelma. Kun joitakin osia projektista joutui vetämään hatusta, tuli niistä usein sekavampia kuin osista, jotka olin suunnitellut paremmin. Keskittyisin myös enemmän koodin laatuun ja luettavuuteen sekä kommentointiin. Tuntuu, että jos en olisi itse kirjoittanut ohjelmaa olisi siitä hankala saada selkeää kokonaiskuvaa koodin perusteella.

Yleisesti olen hyvinkin tyytyväinen aikaansaannokseeni.

Viitteet

https://www.youtube.com/user/DrMarkCLewis

https://www.scalafx.org/api/8.0/index.html#package

https://www.scalafx.org/docs/dialogs_and_alerts/

https://www.scala-lang.org/api/current/scala/Array.html

https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/

https://en.wikipedia.org/wiki/Racetrack_(game)

https://harmmade.com/vectorracer/

https://www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2011/rapport/olsson_robert_OCH_tarandi_andreas_K11062.pdf

https://plus.cs.hut.fi/studio_2/k2019/k15/osa03/