1. **Henkilötiedot**

   Ilmari Lehtinen, 596624, Bioinformaatioteknologia, 23.4.2019

2. **Yleiskuvaus**

   Projektin aiheena on liikennesimulaatio. Simulaatiossa kartalle piirtyy viivoja, jotka esittävät teitä. Simulaation alussa teiden päihin luodaan autot, jotka käyttäjän käskystä lähtevät seuraamaan teitä, jotka ovat ennalta määrättyjä reittejä. Ajoneuvot yrittävät myös väistellä toisiaan, karkaamatta liikaa reitiltä. Tässä on vaihtelevia lopputuloksia. Ajoneuvot siis jarruttavat ja yrittävät vaihtaa suuntaa havaittuaan edessään toisen ajoneuvon. Ohjelmassa on graafinen käyttöliittymä, josta käyttäjä voi aloittaa ja sulkea simulaation. Lisäksi käyttäjä voi synnyttää simulaatioon uusia ajoneuvoja. Käyttäjä voi myös vaihtaa autojen käyttäytymistä kesken simulaation (follow/arrive/seek mouse). Näistä osa lähinnä testitarkoituksena. Alkuperäiseen suunnitelmaan verrattuna, ohjelmasta jäi puuttumaan monia asioita. Reitinhakualgoritmit, reittien luku tiedostosta, törmäysten rekisteröinti jäivät ainakin puuttumaan. Tämän takia projekti on suoritettu korkeintaan vaikeusasteella **keskivaikea**

3. **Käyttöohje**

   Ohjelman käyttöohje löytyy README-tiedostosta.

4. **Ulkoiset kirjastot**

   Olen käyttänyt projektissa PyQt grafiikkakirjastoa, sekä lisäksi pythonin sisäisiä math, sys ja random kirjastoja.

5. **Ohjelman rakenne**

   Ohjelmassa on kuusi keskeistä pääluokkaa, sekä lisäksi testiluokka muutamia yksikkötestejä varten. Pääluokat ovat World, Path, GUI, Vehicle, VehicleGraphics ja Vector

   World: Luokka mallintaa tilannetta koko simulaatiosta, pitää yllä listaa ajoneuvoista simulaatiossa sekä listaa reiteistä. Tämän luokan oliota helppo käyttää parametreinä muissa luokissa. Keskeisimmät metodit: addVehicle().

   Path: Luokka mallintaa ajoneuvojen seuraamaa reittiä koordinaattilistan avulla.

   GUI: Luokka, joka on vastuussa ohjelman käyttöliittymästä. Tämän avulla siis luodaan koko käyttöliittymä, piirretään tiet, autot, napit ja käytännössä alustetaan koko simulaation ulkoasu. Lisäksi luokkassa käsitellään ns eventlooppi, eli käsitellään ohjelman tapahtuma joka syklillä. Tärkeimmät metodit: Kaikki, mutta joitain nostoja: timerEvent(), InitUI(), updateVehicles()

Vehicle: Luokka, joka vastaa ajoneuvon simuloimisesta ja kaikista liikumisalgoritmeistä. Luokka siis mallintaa yksinkertaista ajoneuvoa(simple vehicle model). Käyttää luokkaa Vector lähes kaikkiin laskutoimituksiin. Luokan metodit mahdollistavat ajoneuvon simuloimisen. Näistä tärkeimmät: followPath(), move(), seek()

VehicleGraphics: Luokka vastaa ajoneuvon graafisesta esityksestä ruudulla. Tässä luokassa luodaan siis graafinen esitys luokan Vehicle simuloimille ajoneuvoille. Luokka luo siis kolmiot jotka esittävät autoja ja luokan metodit vastaavat kolmioiden sijainnin ja suunnan muuttamisesta. Tärkeimmät metodit: drawCar(), updatePos(), UpdateRotation()

Vector: Luokka, joka esittää kaksiulotteista vektoria. Luokassa on määritelty kaikki ohjelmassa tarvittavat vektorilaskutoimitukset. Näistä eniten käytettyjä: setMagnitude(), norm(), mult(), NormalPoint(), distance(), dot()

6. **Algoritmit**

Ohjelman tärkeimmät algoritmit liittyvät ajoneuvojen liikkeeseen. Tärkeimmät näistä ovat followPath, Seek, ja Arrive

followPath: Ennustetaan auton tuleva paikka, jos pisteen projektio reitin keskikohtaan(normaali) on pienempi kun reitin ympärillä kulkevan sylinterin säde(”reitin varrella”), ei tarvitse tehdä mitään, auto on reitin varrella. Jos tulevan paikan projektio reitille on suurempi kuin säde, etsitään lähin piste reitillä(normaali), jonka jälkeen siirretään sitä hiukan eteenpäin ja asetetaan se targetiksi. Tämän jälkeen kutsutaan Seek() kohti uuttaa targettia. Repeat. Koska reitti on pitkä ja monimutkainen, pitää tutkia myös, että löydetään oikea normaalipiste reitiltä. Tämän takia reitti käydään for-loopilla läpi ja etsitään aina lähin normaalipiste.

Seek: Lasketaan haluttu nopeusvektori: desired = normalize (target – location) * max_speed
Lasketaan suuntavektori autolle: steer = desired – velocity
Algoritmi siis etsii halutun pisteen kartalta ja jää ajamaan sen läpi edes takas

Arrive: Sama kuin Seek, kun auto on kaukana kohteesta. Kun auto lähestyy kohdetta, piennetään nopeusvektoria lineaarisesti, jolloin auto pysähtyy kohteeseen. Vektorin suuntaa ei muuteta, ainoastaa pituutta.

7. **Tietorakenteet**

Ohjelmassa on käytetty lähinnä Pythonin omia tietorakenteita. Näistä käytetyin on pythonin oma List() tietorakenne. Pythonin list on muuttuva tietorakenne. Lisäksi toinen ohjelmassa käytetyin tietorakenne on Vector-luokan ilmentymä. Luokka kuvaa yksinkertaisesti paria numeroita(int/float) ja määrittelee näille sopivat laskutoimitukset. Ohjelmassa olisi voinut käyttää listan sijaan jonkinlaisia puurakenteita reittien esittämiseen ja hakualgoritmien

käyttöön. En kuitenkaan kerennyt toteuttaa hakualgoritmejä ollenkaan, niin pythonin list sopii reitin esittämiseen riittävän hyvin

8. **Tiedostot**

Ohjelma ei käsittele lainkaan tiedostoja. Ajoneuvojen reitit ovat alustettu main.py tiedostossa. Ohjelmaa voisi tosin laajentaa lukemaan reitit tekstitiedostosta. Ohjelma voisi myös lukea simulaation asetuksia tiedostosta, mutta tämä jäi toteuttamatta.

9. **Testaus**

Yksikkötestien käyttö jäi aikataulurajoitteiden takia erittäin vähälle. Yksikkötestejä on kirjoitettu ainoastaan Vector-luokan metodeille. Näistä kaikki menevät läpi. Testit löytyvät test.py tiedostosta. Koko ohjelman testaus on suoritettu lähinnä ajamalla ohjelmaa eri parametreillä ja tämän jälkeen havainnoimalla mitä tapahtuu…

10. **Ohjelman tunnetut puutteet ja viat**

Ohjelmassa on eniten ongelmia autojen liikkeessä. Autot eivät väistä sulavasti toisiaan, vaan ajavat usein toistensa läpi. Reittien seuraus ei onnistu riittävän tarkasti, vaan väistäessään muita autoja, ajoneuvot harhailevat pois reitiltä. Ajoneuvot löytävät kuitenkin usein takaisin reitille, pienen seikkailun jälkeen. Yksi reiteistä ei toimi ollenkaan(Path 7), ajoneuvot ajavat suoraan ulos ruudulta, ja pääsevät vain harvoin reitin loppuun asti. Ei havaintoa mistä johtuu. Ongelmat lienevät algoritmien hienosäädössä ja voimien skaalauksessa. Suurin puute tähän liittyen on ehkä kunnon valintafunktio, jossa ajoneuvo havaitsee ympäristöä ja tekee sen perusteella päätöksen seuraavasta liikkeestä. Lisäksi followPath algoritmissa on ongelmia normaalipisteen etsimisessä. Tätä pitäisi optimoida lisää.

Toinen suuri ongelma liittyy ohjelman suorituskykyyn. Algoritmit eivät ole ollenkaan optimoituja nopeiksi. Tähän ratkaisuksi olisi jakaa koko kartta ruudukkoon, ja pitää kirjaa missä ruudussa ajoneuvot ovat. Tällöin liikkumisalgoritmien ei tarvitsisi tutkia listaa kaikista ajoneuvoista simulaatiossa, vaan ainoastaan lähimpien ruudukkojen ajoneuvot. Tämä olisi luultavasti valintafunktion jälkeen seuraava kehityskohde. Ohjelma alkaa siis hidastelemaan huomattavasti, kun ajoneuvoja on ruudulla useita kymmeniä.

Lisäksi ohjelmasta jäi puuttumaan kokonaan reitinhakualgoritmit. Ajoneuvot seuraavat tiukasti etukäteen määriteltyjä reittejä, jolloin simulaatioista jää puuttumana tietynlainen satunnaisuus ja realistisuus. Risteyksiin olisi voinut määrittää sääntöjä, esim. liikennevaloja jne. Nyt autot jäävät usein hidastelemaan risteyksiin ja ajavat myös toistensa läpi.

Käyttäjä ei pysty myöskään vaikuttamaan ajoneuvojen parametreihin(maksiminopeus, maksimivoima jne.) muuttamatta koodia. Tämä rajaa erilaisten simulaatioiden määrää huomattavan paljon…

11. **3 parasta ja 3 heikointa kohtaa**

Parhaat:

- Graafinen käyttöliittymä: Selkeä ja yksinkertainen, toimii ainakin oman koneen näytöllä halutusti
- Vector-luokka: Selkeät ja toimivat metodit vektorilaskutoimituksille.

Heikot:

- Ajoneuvojen liike
- Algoritmien hitaus
- Vähäinen konfiguroitavuus/säätely, kaikki kohdat selitetty tarkemmin kohdassa 10.

12. **Poikkeamat suunnitelmasta**

Useimmat poikkeukset selitetty jo aikasemissa kohdissa. Suurimmat poikkeamat ovat siis hakualgoritmien puuttuminen, vähäinen konfiguroitavuus, ruudukon puuttuminen(epäotimoitu), sekä ajoneuvojen epäsulava liikkuminen. Poikkeamat johtuvat aikataulurajotteista.

13. **Toteutunut työjärjestys ja aikataulu**

- Toimivan graafisen käyttöliittymän luonti: 15h
  - autojen piirtäminen ruudulle, valikot jne.
- Autojen liikuttaminen ruudulla ja yksinkertaisia ohjausalgoritmejä: 20h
  - miten esim kaksi autoa reagoi keskenään jne.
  - Vektorilaskenta kuntoon
- Useampien autojen liikennekäyttäytyminen keskenään: 20h
  - path following usealla autolla
- Testaus/muuta hienoa: 5h

Projektin tekeminen sujui pääosin suunniteman mukaan, mutta en kerennyt käyttää vaiheisiin läheskään niin paljon aikaa, kun olin alun perin suunnitellut. Projektista jäi myös puuttumaan muutama vaihe, esimerkiksi hakualgoritmien koodaus.

14. **Arvio lopputuloksesta**

Pääosin projekti sujui oikein mukavasti, lukuunottamatta aikataulurajoitteita. Niiden takia monia ominaisuuksia jäi tekemättä, mutta mielestäni projektin pääpiirteet ovat kassassa, ja niiden tulisi mielestäni riittää hyväksyttyyn arvosanaan projektista. Projekti ei ole läheskään täydellinen liikennesimulaatio, vain pintaraapaisu siihen, mitä voisi tehdä jos aikaa olisi ollut enemmän. Eniten jäi harmittamaan autojen epäsulava liike. Autot eivät väistä toisiaan läheskään aina ja eksyvät reitiltä. Tämä lieneekin projektin heikoimpia puolia. Autot kuitenkin liikkuvat lähes reitin mukaan, ja joissain tilanteissa risteyksissä on ihan sulaviakin jarrutuksia jne. Graafinen käyttöliittymä toimii ainakin omalla läppärin näytöllä lähes halutulla tavalla ja eri napit sivupalkissa toimivat oikein. Projektiin olisi ollut kiva saada jonkinlainen ominaisuus lukea eri dataa tiedostosta, jolloin

erilaisten simulaatioiden määrä olisi kasvanut. Nyt autojen liike riippuu lähinnä siitä, kuinka useasti käyttäjä lähettää autoja matkaan. Lisäksi kaikki reitit eivät toimi kunnolla ja muutamia autoja jää aina harhailemaan kartalle. Ohjelmaa on mielestäni suhteellisen helppo lähteä laajentamaan jatkossa, jos niin tahtoo. Uusia liikkumisalgoritmejä on mahdollista toteuttaa vahvan Vector-luokan avulla.

Ohjelmaan tulisi jatkossa kirjoittaa enemmän testejä, jotka tutkivat ajoneuvojen liikettä. Näin olisi helpompi löytää, mistä ongelmat oikeasti johtuvat.

15. **Viitteet**

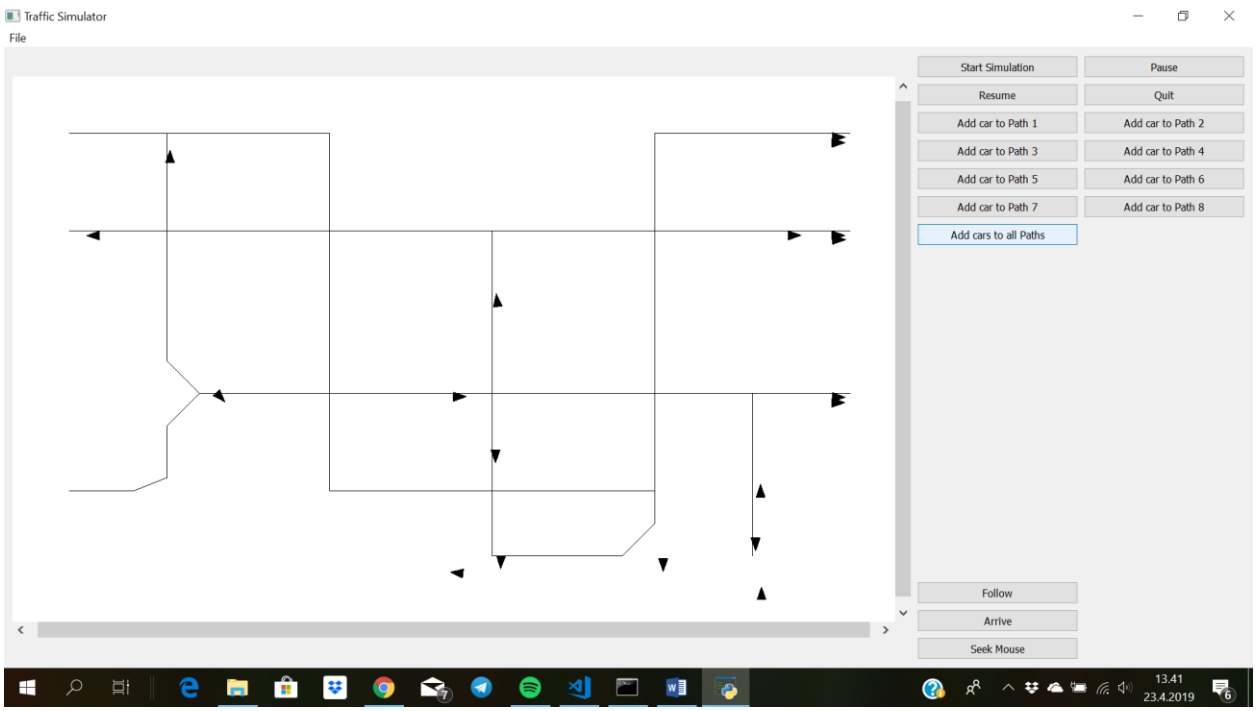
Nature of Code, Daniel Shiffman,
https://natureofcode.com/book/

Steering Behaviors For Autonomous Characters, Craig W. Reynolds,
http://www.red3d.com/cwr/steer/gdc99/

Python,
https://docs.python.org/3.5/reference/index.html#reference-index
https://docs.python.org/3.5/library/index.html

PyQT5,
http://pyqt.sourceforge.net/Docs/PyQt5/index.html
http://zetcode.com/gui/pyqt5/
https://doc.qt.io/qt-5/

16. **Liitteet**


Traffic Simulator
File
Start Simulation
Pause
Resume
Quit
Add car to Path 1
Add car to Path 2
Add car to Path 3
Add car to Path 4
Add car to Path 5
Add car to Path 6
Add car to Path 7
Add car to Path 8
Add cars to all Paths
Follow
Arrive
Seek Mouse
13.41
23.4.2019