Autopeli

1. Henkilötiedot

Autopeli, Onni Komulainen, TIK, 2021, 23.04.2022

2. Yleiskuvaus

Projektini on formulapeli, jossa kamera seuraa autoa kaksiulotteisessa radassa. Pelissä on neljä pelimuotoa, joista kahdessa pelaaja voi pelata ja kaksi on vain sen takia, että voi katsoa ja nähdä miten AI ajaa itse jatoista AI:ta vastaan. Ne ovat kisa, aika-ajo, AI:n kisa ja AI:n aika-ajo.

Rata on tehty ottamalla kuvankaappauksia eräästä vanhasta ajosimulaattorista (Live for Speed) ja käytetty freeware ohjelmaa Pixelator, jolla radasta sai pixelartin näköistä, kuten tarkoituksena oli. Olen muokannut hieman sitä ruohontunnistus algoritmiä varten, jotta algoritmi voi tunnistaa, milloin pelaaaja on ja ei ole radalla. Autot on myös suoraan otettu reddit käyttäjältä u/Racing21187. Linkit kaikkeen on lähteissä.

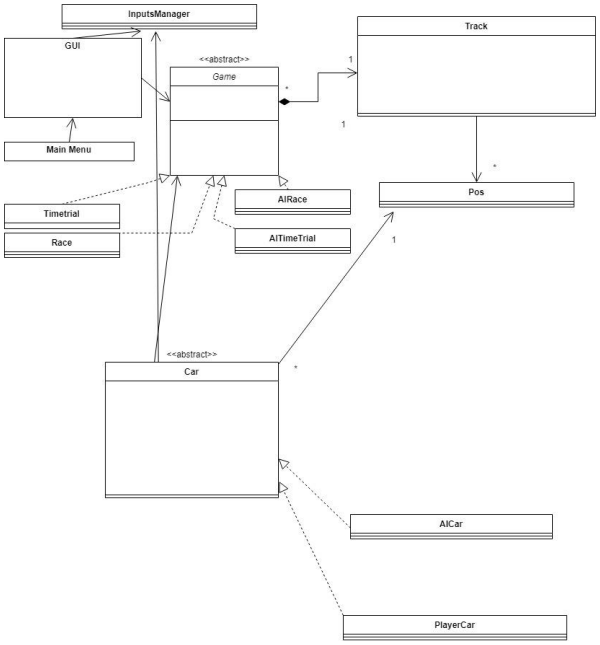
Pelissä formularatin asento on hiiri suhteessa ruudun keskilinjaan y-akselilla, sekä kaasu ja jarru W sekä S näppäimillä. Koska kaasulla ja jarrulla ei ole kuin true/false arvot, pelissä on ”täydellinen” ABS ja liukkaudenesto systeemi. Pelimoottori laskee auton maksimikiihtyvyyden kokoaika, jonka avulla lasketaan paljonko auto voi kääntyä, jarruttaa tai kiihtyä eteenpäin. Pelimoottori laskee moottorin tehon ja ilmanvastuksen aiheuttamat voimat, jolloin se olisi realistinen ja kitkan voisi laskea helposti.

Pelimoottori katsoo, onko auton rengas radan ulkopuolella (eli nurmella) katsomalla renkaan keskipisteen koordinaattien avulla onko radan pikseli siinä kohdassa vihreä vai ei. Samalla systeemillä katsotaan milloin pelaaja ylittää maaliviivan ja sektoriviivat. Pelin AI noudattaa samoja inputteja kun pelaaja.

3. Käyttöohje

Pelin käynnistäessä eteen tulee menu, josta voi valita neljästä pelimuodosta. Mikäli pelaaja valitsee muodon, jossa voi itse ajaa, hiiren X koordinaatti toimii ratin suuntana ja W sekä S kaasuna ja jarruna. C:llä voi vaihtaa kameran toiseen pelaajaan ja P:llä ”pausettaa” pelin. D:stä saa näkyviin AI:n reitin, mikäli semmoinen on siinä pelimuodossa ja Q:sta seuratun auton koordinaatit konsoliin, joka on hyödyllistä, jos tarvitsee selvittää uuden kartan aloitus- tai AI:n checkpoint paikkoja. Suosittelen katsomaan miten AI ajaa, jotta tietää suurin piirtein kartan ja miten ajaa nopeasti. Escape näppäimestä pääsee takaisin päävalikkoon.

4. Ohjelmanrakenne



Luokkarakenne pelissä on kohtuullisen yksinkertainen. On olemassa abstrakti auto luokka, johon AICar ja PlayerCar kuuluvat. Samoin Pelimuodoille on samanlainen systeemi eli että on olemassa abstrakti peliluokka ja pelimuodot ovat luokkia jotka ”extendaavat” abstraktiin luokkaan.

Car-luokan merkittävät metodit realTurn, joka laskee oikean kääntymissäteen mukaan lukien maksimikitkan, drive, jolle annetaan kääntymissäde ja kaasu- sekä jarrupolkimen inputit, ja se liikuttaa autoa ja muokkaa nopeutta oikean verran, sekä draw, joka piirtää auton oikeaan kohtaan ruutua. PlayerCar:in metodeissa on yksikertaiset metodit, jossa hiirestä ja W sekä S napeista saadan inputit autolle.

AICar:in lähes kaikki metodit ovat tärkeitä sen toiminnalle, joten tässä enemmän niistä. turningCirclellä saa kahden pisteen kautta menevän ympyrän säteen. turningPointForLate:lla AI saa kääntymisen aloituspisteen. stopTurningPointForEarly saadaan kääntymisen lopetuspiste. breakingPointForLate:lla saadaan jarrutuspiste, kun sille annetaan käännöksen aloituspiste ja kääntymissäde. calculateMiniCheckpointsilla saadaan seuraavan checpointtyppin avulla tarkat ohjeet AI:lle eli ”miniCheckpointit”. checkForOthersilla AI tarkastaa ja mahdollisesti antaa tilaa toiselle autolle, joka on sen lähellä. calculateInputs laskee AI:n inputit.

Track-luokassa ei ole mitään sen ihmeempää, kun tietoa radasta (AI:n reitti, aloituspaikat, kuva yms.). Pos luokassa on metodeja pos:iin liittyen, kuten angleBetween, joka antaa kulman toiseen Pos olioon, ja isBehind, joka antaa booleanin riippuen onko parametrina annettu pos nykyisen takana. GameApp:in metodeista tärkein on followCar. Tällöin GUI seuraa autoa, joka annettu parametreissä.

5. Algoritmit

Projektissa on paljon erilaisia algoritmeja, mutta oleellisimmat ovat Car- ja AICar-luokissa. Car luokassa ajamisalgoritmi (drive) toimii ottamalla kääntymissäteen ja kaasu- sekä jarrupolkimen inputit parametriksi ja muuttaa auton Pos oliota oikein (liikkuu ja kääntää) ympyrän geometrian avulla, sekä lisää tai vähentää nopeutta riippuen kaasu- ja jarrupolkimen inputista. Mikäli algoritmille antaa kääntymissäteeksi nollan, auto ajaa suoraan. Tässä algoritmissa on ”täydelliset” ABS ja liukkaudenestojärjestelmä, eli jos käyttää maksimimäärän kitkaa kääntymiseen, ei jarru- tai kaasupolkimen asennoilla ole väliä, koska niille ei jää yhtään kitkaa kääntymisestä. Olisi ollut mahdollista tehdä myös algoritmi muuten, mutta koska kaasu ja jarru on joko päällä tai pois, toimii tämä mielestäni paremmin, koska nyt pelaaja voi jarruttaa mutkaan, miten haluaa ja AI:lle ”täydellinen” luistonesto helpottaa sen kääntymisalgoritmeja huomattavasti.

Drive metodille apufunktioita ovat throttle, drag ja brake. Throttlelle annetaan kaasupolkimen asento Doublena väliltä 0-1 ja se palauttaa throttlen aiheuttaman voiman. Tämä saadaan lisäämällä kaavaan E=mv^2 400kJ/tickrate, ja laskemalla saatu voima kaavasta F=am, jossa a on uuden ja vanhan nopeuden ero. Brake toimii taas antamalla parametrinä jarrupolkimen asento samalla tavalla ja kertomalla sen jollain vakiolla. Drag toimii kertomalla v^1.7 jollain vakiolla. Oikeassa elämässä v^2 olisi oikea, mutta mielestäni oli mukavampi, että auto kiihdyttää pidemmän aikaan.

Toinen car:in tärkeä funktio on checkForGrassAndSectors. Tämä funktio hankkii auton renkaiden positiot wheelPlaces funktiolta ja tarkastaa jokaisen renkaan alta kartan pikselin, jonka päällä renkaan keskikohta on. Mikäli se on joku sektoriviivan väri, funktio tarkistaa onko se juuri oikean ja jos on, se päivittää tiedot. Jos pikseli on vihreä, lisää se sen count nimiseen kokojaan, jonka funktio palauttaa. PlayerCar olioilla se palautuu metodille tpIfGrass, joka kaikkien renkaiden ollessa vihreellä muuttaa nykyisen position sekunti sitten olleeseen ja ottaa nopeutta pois. AI:n metodit eivät ole täydellisiä, jonka takia totesin, että on parempi antaa sen käydä nurmella, jos se sinne menee, koska se ei ”tahalleen” hyödynnä sitä ja silloin kisa pysyy mukavampana.

realTurn on Car-luokan funktio, joka ottaa parametrikseen pelaajan tai AI:n antaman renkaan kääntymiskulman ja palauttaa kääntymissäteen, jossa on otettu huomioon maksimikitka. Tässä

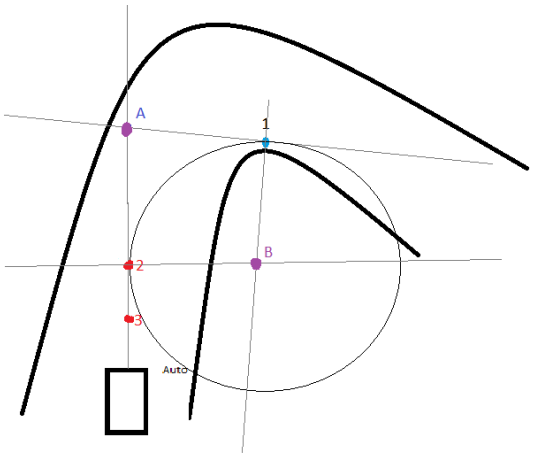
$$\frac{v^2}{\frac{\left(m \cdot 9.81 + v^2 \cdot d\right) \cdot n}{m}}$$

maksimikitka on , jossa v on nopeus, m massa, d downforce kerroin (vakio) ja n kitkakerroin. Slipstream saadaan ratkaistua katsomalla, onko auton paikka muiden autojen edellisten paikkojen lähellä.

AICar:in funktioita on paljon, jota käyn läpi, jotta sen ymmärtää. AI toimii siten, että radan olioon on laitettu AI:n reitti, jossa on kokoelma neljänpituisia Tupleja, jotka ovat (Pos, Int, Boolean, Double). Pos kertoo checkpointin koordinaatit ja suunnan. Int kertoo checkpointin tyypin, Boolean vaihtoehtoisen

suunnan (esim. checkpointin ollessa vasemmalla puolella rataa, booleaniksi pitää laittaa true ja näin AI tietää, ettei kiilaa vasemmalla puolella olevaa autoa pois radalta, eli siirtää omaa checkpointtia oikealle kolmen metrin päähän) ja Double, joka on kerroin sille, kuinka kaukana vaihtoehtoinen checkpoint on (3m * Double).

Checkpointteja on kolmen tyyppisiä ja ne jaetaan pienimmiksi ”miniCheckpointeiksi”, joiden mukaan AI ajaa. Ensimmäinen on tyyppi, jossa sen alussa auto ajaa aluksi suoraan, jarruttaa, jos sen tarvitsee, ja sitten kääntyy. Alla havainnollistava kuva.

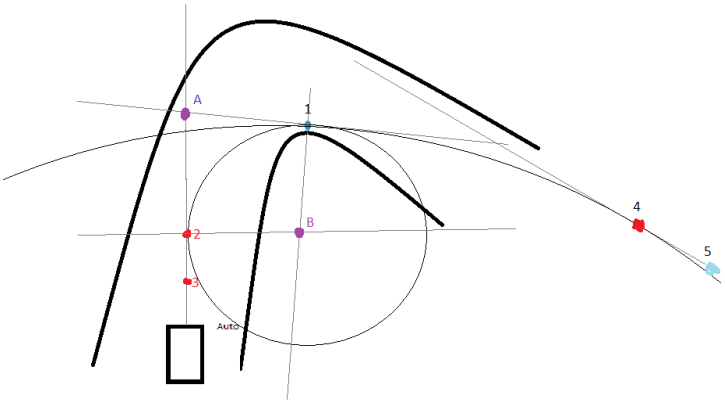


Kuvasta alun perin tiedentään auton Pos olio ja Pos 1, joka on etukäteen määritelty checkpoint, jonka tyyppi on yksi. Tekemällä suoran y = kx + c, jossa y = tan(rotation) * x – tan(rotation) * x_0 + y_0, kummastakin tiedetystä pisteestä, voidaan selvittää piste A. A:n ja pisteen 1 välisellä etäisyydellä voi selvittää kakkospisteen, joka on kääntymispiste. Tällä algoritmilla toimii turningPointForLate. Piste 3 voidaan selvittää fysiikan kaavoista v = v_0 + at => t = max ((v_0 - v) / a, 0), jossa a kiihtyvyys eli F/m (F on voima ja m massa), jotka kummatkin löytyy Constants oliosta. v on kulman maksiminopeus eli

$$\left|\text{solve}\left(v^2=r\cdot\frac{\left(m\cdot g+v^2\cdot d\right)\cdot t}{m},v\right|\right.$$
$$\left.v=-\sqrt{\frac{-g\cdot m\cdot r\cdot t}{d\cdot r\cdot t-m}}\ \text{and}\ \frac{g\cdot m\cdot r\cdot t}{d\cdot r\cdot t-m}\le 0\ \text{or}\ v=\sqrt{\frac{-g\cdot m\cdot r\cdot t}{d\cdot r\cdot t-m}}\ \text{and}\ \frac{g\cdot m\cdot r\cdot t}{d\cdot r\cdot t-m}\le 0\right|$$

, t tässä on kitkakerroin ja r ympyrän säde. Ympyrän säteen saa selvittämällä pisteen B, jonka voi selvittää samalla lailla, kun kääntymispisteenkin, mutta kulmakerroin on -1/tan(rotation). Pisteestä B etäisyys pisteisiin yksi tai kaksi on ympyrän säde. Toinen tarvittava kaava on x = x_0 + vt + ½*at^2. Trigonometrian avulla tästä saa pisteen 3. Tämän pisteen laskee brakingPointForLate. (Huom. Ilmanvastusta tai maksimikitkaa ei oteta huomioon jarruttaessa, jotta jarruttaessa kiihtyvyys on vakio)

Checkpointtyyppi kaksi on samankaltainen, mutta siinä käännös tapahtuu ennen suoraan ajoa. Siihen on laitettu tietynlainen ”hätäjarru”, koska AI:n kisatessa reitit voivat olla hieman arvaamattomia, mutta yleisesti nopeuden pitää checkpointiin tultaessa olla pienempi, kuin käännöksen maksiminopeuden. Alla on seuraavasta kuvasta jatkettu versio.



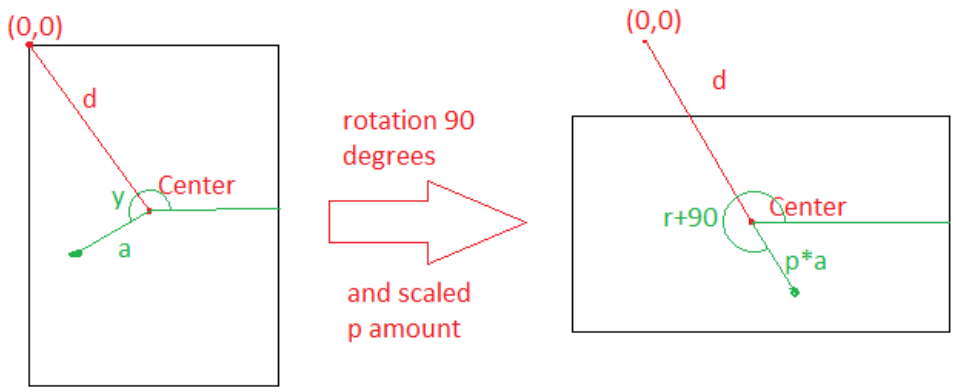
Tässä piste 5 on ihmisen laittama ”isompi” checkpoint ja 4 piste, jossa auto lopettaa kääntymisen. Pisteet lasketaan samalla tavalla, kun tyypissä yksi. AI:n kääntyessä kaasu on aina täysillä, koska täydellisen liukkaudeneston takia nopeus ei voi kasvaa käännöksen maksiminopeutta isommaksi sitä tehtäessä.

Kolmas ”checkpointtyyppi” on vain piste, jota kohti AI ajaa. Yllä olevassa kuvassa pisteet kolme ja viisi toimivat samalla tavalla, eli AI ohjaa niitä päin, mikäli se ei syystä tai toisesta ole suoraan niitä päin.

AI:lla on metodi checkCheckpoint(), joka tarkistaa, ettei nykyinen miniCheckpoint ole auton takana, mikäli on, se siirtyy seuraavaan ja yms. Kun auto menee ihmisen laittaman ”isomman” checkpointin ohi checkCheckpoint() vaihtaa routeCalculated booleanin false:ksi ja tällöin calculateMiniCheckpoints() laskee uudet minicheckpointit.

AI jättää toiselle autolle tilaa, jos se on tarpeeksi lähellä auton sivupuolilla. Sen lisäksi, mikäli auto on edessä ja nykyinen auto lähestyy sitä yli ½ m/s, auto valitsee vaihtoehtoisen reitin yrittääkseen ohitusta. Tämä algorimi on checkForOthers funktiossa, joka laittaa AICar olion alt booleanin päälle, mikäli se muuten kiilaisi auton radalta ulos. Algoritmi toimii tekemällä uusia Pos oliota auton kummallekkin puolelle ja katsomalla, mitä niistä toinen auto on lähimpänä. Mikäli se on tarpeeksi lähellä tämmöistä pistettä (3.5 m), katsoo algoritmi tarviiko AI:n käyttää väistävää reittiä vai ei. Mikäli sen pitää antaa tilaa toiselle autolla, laitaa se alt booleanin päälle ja routeCalculated pois päältä, jolloin AI laskee miniCheckpointit uudestaan.

Viimeisin mielestä merkittävä algoritmi on GameApp-luokassa oleva followCar, joka hoitaa auton seuraamisen. Koska pelissä kartta kääntyy auton suunnan mukaan ja koko muuttuu nopeuden mukaan, on kyseinen algoritmi aika monimutkainen. Alla on kuva, joka kertoo algoritmin toiminnasta.



Kuvassa d, r ja a on vakioita, joten uuden pisteen voi laskea trigonometrian avulla tiedettäessä vanhan pisteen koordinaatit.

6. Tietorakenteet

Projektissani on käytetty muuttumattomia vektoreita esim. radan tietojen pitämiseen, koska helppo ja mukava, sekä arrayta GUI:hin koska scalafx vaatii. Muuttuvia kokoelmia ovat esim. Buffer AI:n reitin näyttämiseen ja autojen viimeisten Pos-olioiden pitämiseen sekä mutable.Set InputManagerissa, jotta montaa smanlaista keyCodea ei ole samaan aikaan samassa kokoelmassa.

7. Tiedostot ja verkossa oleva tieto

Ohjelmassa on koodin lisäksi vain kuvatiedostoja GUI:ta varten.

8. Testaus

Ohjelmaa testattiin projektisuunnitelman mukaisesti alussa ennen GUI:ta REPL:iä käyttäen ja GUI:n tullessa sen ja lähinnäkin println kanssa. Testaus sujui hyvin ja varsinkin AI:n toimivuutta oli hyvä testata GUI:ssa toista AI:ta vastaan (AI Race) ja näin oli helppo katsoa, minkälaista säätöä algoritmit vaativat.

9. Ohjelman tunnetut puutteet ja viat

Isoin puute on selkeästi törmäysmekaniikan puuttuminen. Osaksi sen puuttuminen johtuu siitä, että oikeanlaisen törmäysfysiikan tekeminen voisi sekoittaa AI:n, jos sen reitti muuttuu merkittävästi esim. käännöksessä.

Huomasin vasta AI:ta tehdessä, että pelin Y ja R koordinaatit ovat väärin päin eli Y positiivinen alaspäin ja R positiivinen myötäpäivään. Siihen asti kaikki metodit oli toteutettu toimivaksi tästä huolimatta, joten ajattelin että on parempi jatkaa niin.

AI:n algoritmi ei ole täydellinen, jonka takia kaksi AI:ta voi ”törmätä” toisiinsa eli mennä toisiensa sisään, tähän nopea korjaus on Car-luokan checkCollisions(), mutta se käytännössä estää vain sen, että auto ei voi ajaa suoraan toisen läpi. Muutenkin, jos alt menee päälle väärässä kohtaa, saattaa auto käydä hieman nurmikolla, jonka takia AI ei käytä tpIfGrass metodia.

Joku voisi myös sanoa sitä viaksi, että kääntyessä ei välttämättä voi jarruttaa, jos kitka ei riitä kääntymisen jälkeen. Tämä on kuitenkin täydellisen ABS-systeemin ominaisuus.

10. 3 parasta ja 3 heikointa kohtaa

Parhain asia projektissa on AI:n toiminta ja se, että sille voi laskea miniCheckpointteja fysiikan lakien mukaan, jonka on lisännyt. Esimerkiksi AI:n jarrutuspiste lasketaan kaavoja v = v_0 + a * t sekä x=x_0+v*t+½*a*t käyttäen.

Toiseksi paras on onnistunut fysiikkamoottori. Tykkäsin lukiossa kaikista fysiikan kursseista, jotka liittyivät mekaniikkaan, ja se on yksi syy, miksi halusin kokeilla tehdä tämmöisen pelin. Kolmas on mielestäni mukava pelaamiskokemus.

Heikoimmat kohdat ovat huono menu, johon ei ollut aikaa tai motivaatiota, AI:n törmäileminen muihin pelaajiin, törmäyssysteemin puuttuminen, koska peli voisi olla parempi sen kanssa ja Y sekä kääntymiskoordinaattien ”väärin päin oleminen”, koska se hankaloitti algoritmeja paljon. Viimeisenä lisäyksenä vielä pelin pyöriminen ei niin tehokkailla tietokoneilla. Kehitin tämän koneella, jolla on kuusi 4.8GHz ydintä ja demotessani nyt viimeisenä päivänä kaverille läppärillä peli ei pyörinyt kovin hyvin.

11. Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Projektin toteutumisjärjestys oli aika lailla se, mitä ajattelin alusta asti. Aluksi toteutettiin perus luokkarakenne ja ensimmäisenä isompana Pos-luokan add metodi, jonka testaus tapahtui REPL:issä. Tämän jälkeen ensimmäinen GUI paintilla tehdyn testikartan kanssa ja sen toimiessa oikean kartan teko. Sitten tuli bugin korjaus ja lopulta AI:n lisääminen. Ajankäyttöarvio oli suurin piirtein oikein, mutta mielestäni käytin hieman enemmän aikaa kuin arvioitu 80 tuntia, varsinkin jos laskee algoritmin miettimisen muualla, kuin koneen äärellä. Kahden viikon välinen aikataulu oli todella väärin laitettu, koska minulla oli todella paljon kursseja nelosperiodissa ja en jaksanut tai ehtinyt tehdä, joten projekti jäi melkein täysin kahdelle viimeiselle viikolle. Opin sen, että AI:n testaamiseen voi mennä oikeasti todella paljon aikaa, jos sitä haluaa hienosäätää.

12. Kokonaisarvio lopputuloksesta

Mielestäni projektista tuli todella hyvä ja on mukava kisata AI:n kanssa. Myös AI:n tekeminen onnistuneesti fysiikan ja analyyttisen geometrian kaavoilla oli mukavaa. Työssä isoin puute on vieläkin

törmäysmekaniikan puute. Sen tekeminen ei olisi kovin helppoa ja AI:n pitäisi olla este törmäysmekaniikan mukaan, jotta sen algoritmi ei mene täysin sekaisin. Sen lisäksi peliin voisi lisätä uusia karttoja. Tämän ei pitäisi olla kovin vaikeaa. Siihen kehittäjän pitäisi lisätä valinta kartalle menuun ja tehdä kartta, aloituspaikat, reitti AI:lle, ja sektoriviivojen värit. Mielestäni ohjelma on helposti laajennettava, koska pyrin jo tekemisvaiheessa jättämään turhan toiston pois välistä.

Jos aloittaisin projektin alusta uudelleen, tekisin koordinaatiston Y ja R koordinaatit oikein päin ja Pos-olion hieman erillä lailla, koska suunnitelulla tavalla se lähti aina muuttamaan radan Pos-oliota, mutta muuten olen todella tyytyväinen projektiini.

13. Viitteet

Pixelator http://pixelatorapp.com/

Autot https://www.reddit.com/r/formula1/comments/b1ui1c/pixelart_of_2019_grid/

Kartta https://www.lfs.net/

MAOL, 2019

Scala Standard Library 2.13.8 https://www.scala-lang.org/api/current/

ScalaFX API https://javadoc.io/doc/org.scalafx/scalafx_2.13/latest/index.html

14. Liitteet

Esittely kisasta: https://www.youtube.com/watch?v=NkljyhS-h8Y

Lähdekoodi: https://version.aalto.fi/gitlab/komulao3/os2-autopeli