Parvisimulaatio

Projektityön dokumentti

Veera Itälinna
525608 BIO
päiväys

Yleiskuvaus

Ohjelma simuloi graafisesti parven (esim. lintuparven) liikettä kaksiulotteisesti, ylhäältä päin kuvattuna. Parvessa liikkuu käyttäjän valitsema määrä yksilöitä, jotka liikkuvat ruudulla yksinkertaisten käyttäytymissääntöjen mukaisesti. Kun parven jokainen yksilö valitsee jatkuvasti kulkusuuntansa näiden sääntöjen perusteella, liike muistuttaa oikeaa parvikäyttäytymistä.

Parven käyttäytymissäännöt ovat seuraavat:

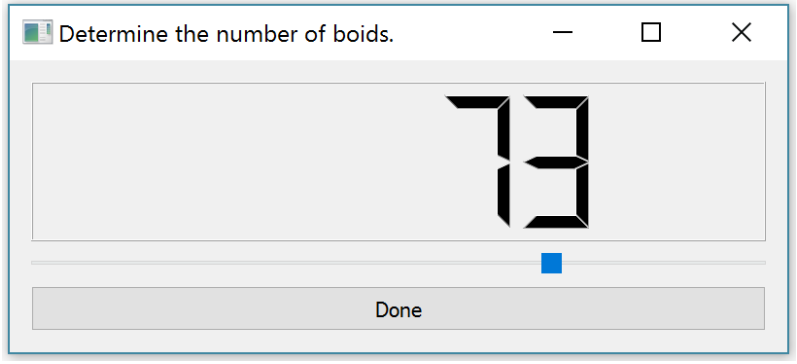
- Vältä törmäilyä muihin lintuihin (Separation)
- Lennä samaa nopeutta kuin muu parvi keskimäärin (Alignment)
- Pyri kohti parven keskipistettä (Cohesion)

Lisäksi parven yksilöt välttävät alueen seinämiin törmäämistä, ja niiden nopeuden liiallinen kasvu on myös estetty.

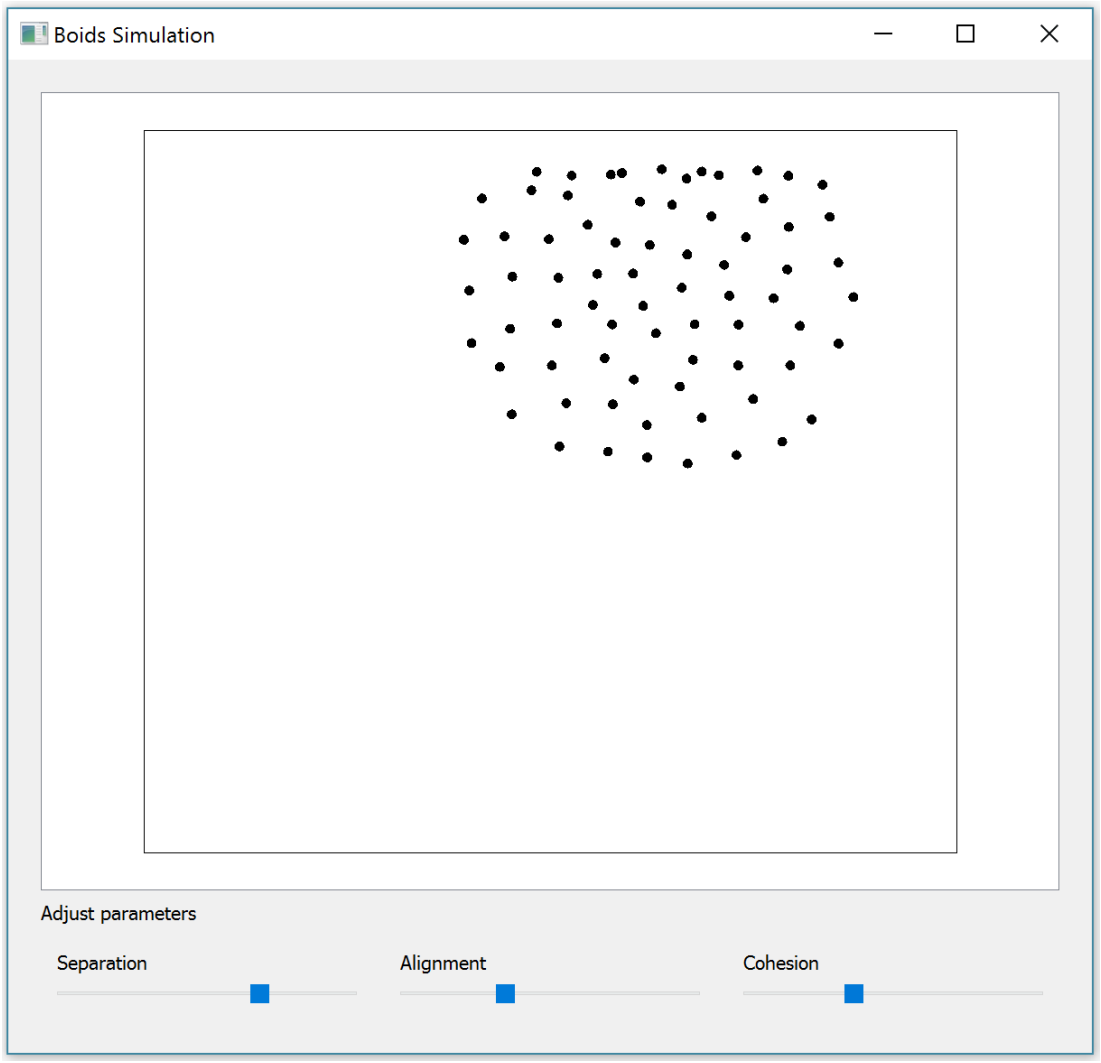
Käyttöohje

Kun ohjelma avataan, käyttäjälle avautuu ikkuna, jossa liukusäätimellä valitaan parven yksilöiden lukumäärä väliltä 5-100 (kuva 1). Haluamansa parven koon määritettyään käyttäjä painaa ”Done” -nappia, jolloin ensimmäinen ikkuna sulkeutuu ja varsinainen simulaatio käynnistyy uudessa ikkunassa (kuva 2). Ruudulle ilmestyy haluttu määrä parven yksilöitä edustavia pisteitä, joiden alkukoordinaatit sekä -nopeus on arvottu.

Parven liikettä voidaan muokata kolmella ikkunan alareunassa olevalla liukusäätimellä, jotka muuttavat edellä mainittujen käyttäytymissääntöjen painokertoimia reaaliaikaisesti. Säätimet on asetettu oletusarvoisesti puoliväliin, jolloin painokertoimia voi muuttaa joko pienemmäksi tai suuremmaksi alkuperäisiin arvoihin verrattuna. Ohjelma sulkeutuu, kun käyttäjä sulkee ikkunan.



Kuva 1.



Kuva 2.

Ohjelman rakenne

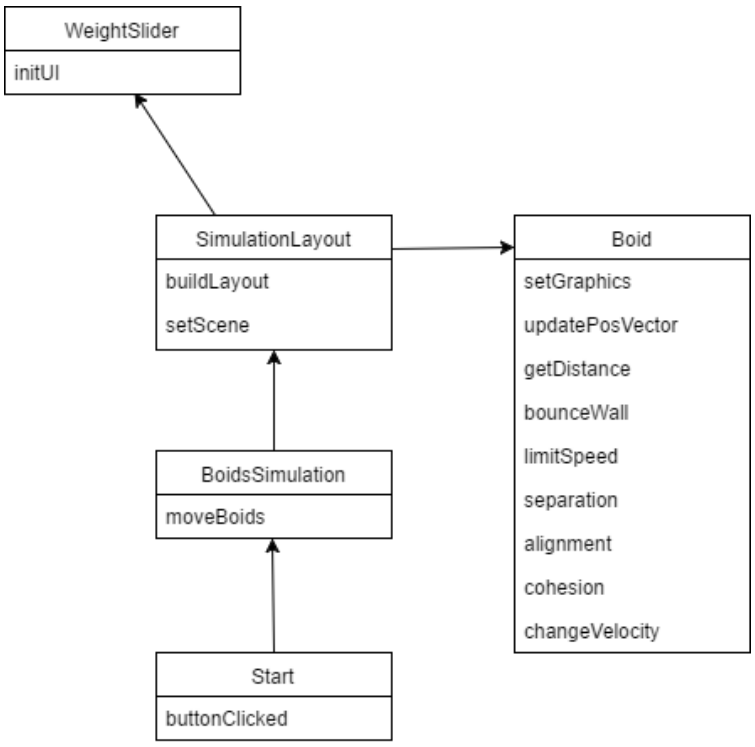
Lintuparven yksilöitä kuvaa luokka Boid. __init__-metodissa alustetaan yksilöiden alkusijainti ja –nopeus. Luokan setGraphics-metodissa luodaan lintujen graafinen ulkoasu, eli määritellyn kokoinen piste. Boid-luokassa on määritelty myös kaikki lintujen nopeuteen vaikuttavat metodit, eli parvikäyttäytymisen säännöt, seinämien välttely, nopeuden suuruuden rajoittaminen, nopeuden muutosvektorin laskeminen sekä muutama apumetodi edellisten tueksi.

SimulationLayout-luokan avulla toteutetaan simulaation graafisen käyttöliittymän layout ja lisätään haluttu määrä Boid-yksilöitä simulaatioon. Käyttöliittymän ominaisuuksissa hyödynnetään PyQt5-kirjastoa. Luokan buildLayout-metodissa asetetaan tarvittavat widget-elementit paikoilleen, ja setScene-metodissa luodaan QGraphicsScene, johon parven yksilöt lisätään. Painokertoimia muuttavat liukusäätimet on toteutettu omassa WeightSlider-luokassaan, jossa määritellään säätimen alkuasetukset ja otsikko.

Simulaation pääikkunan luominen ja ohjelman jatkuva pyörittäminen tapahtuvat BoidsSimulation-luokassa, joka perii PyQt5:n QMainWindow-luokan. Pääikkunaan lisätään layoutiksi SimulationLayout-luokan instanssi. moveBoids-metodissa muutetaan lintujen nopeusvektoreita ja liikutetaan ruudulla näkyviä pisteitä while-silmukassa, joka päättyy, kun ohjelma suljetaan.

Start-luokassa toteutetaan simulaation alussa näkyvä liukusäädin, jolla valitaan parven koko. Säätimen Done-nappi yhdistyy buttonClicked-metodiin, joka tallentaa lintujen valitun lukumäärän, luo BoidsSimulation-luokan instanssin ja kutsuu simulaation moveBoids-metodia.

Alla graafinen hahmotelma ohjelman luokkajaosta ja keskeisimmistä metodeista.



Algoritmit

Ohjelman olennaiset algoritmit perustuvat tehtävänannossa mainittuun Craig Reynoldsin artikkeliin ”Steering behaviours for autonymous characters”.

Parven yksilö eli Boid-olio määrittää jatkuvasti, mitkä muut parven yksilöt ovat yksilön ympyränmuotoisen havaintoalueen sisäpuolella, käymällä toistuvasti läpi kaikki linnut sisältävää listaa. Näistä lähellä olevista yksilöistä saadaan tietää niiden etäisyys sekä nopeuden suuruus ja suunta, ja näiden tietojen avulla yksilö muuttaa oman nopeutensa suuruutta ja suuntaa kolmen käyttäytymissäännön (Separation, Alignment, Cohesion) mukaisesti. Lintu määrittää jatkuvasti myös etäisyytensä alueen seinämiin, ja pyrkii välttämään törmäystä.

Separation-sääntö toteutetaan niin, että jokainen yksilön havaintoalueen sisällä oleva lintu aiheuttaa itsestään poispäin suuntautuvan työntövoiman, jonka suuruus on kääntäen verrannollinen lintujen välisen etäisyyden neliöön. Kokonaisvektori saadaan yksittäisten voimavektorien summana.

Cohesion-säännön toteuttamiseksi havaintoalueen sisällä olevista lintuolioista lasketaan keskimääräinen sijainti (massakeskipiste), jota kohti yksilöä ohjaava voima suuntautuu.

Alignment-käyttäytymistä varten lasketaan havaintoalueen sisällä olevien lintujen keskimääräinen nopeus (suuruus ja suunta). Yksilöä ohjaava vektori saadaan tämän ”tavoitenopeuden” ja yksilön oman nopeusvektorin erotuksesta.

Käyttäytymissääntöjen perusteella muodostetut vektorit summataan yhteen painotettuna omilla kertoimillaan, ja muodostettu kokonaisvektori summataan linnun nopeusvektoriin. Jos linnun vauhti eli nopeusvektorin pituus kasvaa liian suureksi, vektori normalisoidaan ja kerrotaan ennalta määrätyllä maksiminopeutta kuvaavalla vakiolla.

Jos lintu päätyy liian lähelle alueen seinämää, linnun nopeuteen kohdistuu seinästä poispäin suuntautuva voima, joka on kääntäen verrannollinen etäisyyden neliöön.

Tietorakenteet

Parven yksilöt eli Boid-oliot on tallennettu listaan, tarkemmin sanottuna Pythonin list-rakenteeseen. Listaan järjestyksessä lisääminen sekä sen läpikäyminen ”for var in list” -tyyppisellä käskyllä onnistuvat tehokkaasti. Ohjelman ei tarvitse missään vaiheessa poistaa olioita listasta, lisätä olioita keskelle listaa eikä hakea tiettyä yksilöä, joten ei ole tarvetta käyttää Pythonin listan tilalla esimerkiksi linkitettyä listaa, sanakirjaa tai hakupuuta.

Tiedostot

Ohjelma ei hyödynnä tiedostoja. Mahdollinen lisäominaisuus ohjelmalle olisi voinut olla simulaation alkuasetusten lukeminen tiedostosta, mutta se ei vaikuttanut luontevalta ohjelman käyttökokemuksen kannalta.

Testaus

Ohjelman testaaminen koko kehitystyön ajan tapahtui pääasiassa tarkastelemalla visuaalisesti lintujen ilmestymistä kuvaruutuun ja niiden liikettä. Myös painokertoimia muuttavien liukusäädinten oikeanlainen toiminta on hyvin erotettavissa silmämääräisesti. Lisäksi ajoittain rakennusvaiheessa esimerkiksi lintujen sijaintien koordinaatit sekä nopeusvektorien arvot tarkastettiin sopivilla print-komennoilla.

Lisäksi tein ohjelman rakennuksen loppuvaiheessa muutamia yksikkötestejä, jotka testaavat QGraphicsSceneen lisättyjen itemien lukumäärää, painokertoimien oletusarvoja sekä painokerroin-liukusäädinten kiellettyjä arvoja. Nämä kaikki testit ohjelma läpäisee.

Ohjelman tunnetut puutteet ja viat

Ainoa huomaamani varsinainen vika tapahtuu silloin, jos parven kooksi valitaan 1: tällöin lintu vain välähtää ruudulla ja katoaa. En saanut selville, mistä tämä ongelma johtuu, joten kiersin sen asettamalla parven minimikooksi 5 (mikä tosin tuntuu muutenkin suhteellisen luonnolliselta, kun ohjelman ideana on simuloida parvikäyttäytymistä eikä yksittäisen linnun liikettä).

Ohjelmassa ei ole mahdollisuutta valita muita alkuasetuksia kuin lintujen lukumäärä. Toisaalta mahdollisuus muuttaa parametreja simulaation kuluessa mielestäni korvaa tämän puutteen kohtalaisen hyvin.

Parhaat ja heikoimmat kohdat

Projektissa on hyödynnetty paljon PyQt-kirjaston erilaisia widgettejä, layouteja ja muita ominaisuuksia. Toiminnallisuuksiin on käytetty mm. signaaleja sekä lambda-funktioita, joita kurssimateriaaleissa ei juurikaan ole käsitelty. Ansioksi voisi periaatteessa lukea myös perehtymisen Pythonin säikeisiin, vaikka niiden toteutus ei ohjelman lopulliseen versioon päätynytkään.

Parven liikkuminen hidastuu jonkin verran, kun parven koko on useita kymmeniä yksilöitä, mikä onkin odotettavissa, sillä laskennan kompleksisuus on verrannollinen lintujen lukumäärän neliöön ($O(N^2)$). Tästä johtuen parven maksimikoko on suhteellisen pieni, 100 yksilöä.

Tietyillä painokertomien ääriarvoilla (Separation-kerroin maksimissa, Alignment-kerroin minimissä) lintujen liike muuttuu hieman epävakaan näköiseksi värähtelyksi. Tämän saisi halutessaan korjattua muokkaamalla säädinten arvojen sallittuja vaihteluvälejä, mutta toisaalta ilmiö on tavallaan kiinnostava.

Poikkeamat suunnitelmasta

Tehtävänannossa sanottiin reaaliaikaisen simulaation tarvitsevan säikeitä. Sain ohjelman osittain toimimaan Pythonin Threading-moduulin säikeiden avulla, mutta ohjelma ei sulkeutunut oikein eivätkä säikeet tehneet laskennasta tehokkaampaa. Jätin lopulta säikeet kokonaan pois.

Ohjelman luokkajako poikkeaa suunnitelmasta osittain. Tärkeimmät luokat, eli Boid ja SimulationLayout, on kuitenkin toteutettu pitkälti suunnitelman mukaan. Simulaation alun valintaikkunan avaava Start-luokka vastaa teknisen suunnitelman SliderView-luokkaa. Simulaation pääikkunaa hallinnoivaa BoidsSimulation-luokkaa ei ole mainittu suunnitelmassa.

Ohjelman ulospäin näkyvä toteutus vastaa pääosin alkuperäistä suunnitelmaa. Lopullisesta toteutuksesta puuttuu suunniteltu nappi, jolla simulaation voisi aloittaa alusta eri säädöillä.

Toteutunut työjärjestys ja aikataulu

Suurin osa koodaustyöstä keskittyi ajallisesti maaliskuun loppuun ja huhtikuun alkupuolelle. Olin jo viime keväänä, kun osallistuin kurssille ensimmäistä kertaa, toteuttanut pääpiirteittäin graafisen käyttöliittymän sekä parven liikkumisalgoritmit. Näitä jouduin tosin vielä jonkin verran korjailemaan myöhemmin.

Eniten aikaa kului säikeiden toteutuksen miettimiseen ja ohjelman kaatumisongelman korjausyrityksiin. Suurimmat yksittäiset muutokset tein 11.4. kun toteutin painokertoimia reaaliaikaisesti muuttavat liukusäätimet ja 12.4. kun poistin säikeet ohjelmasta kokonaan. Tämän jälkeen ohjelma oli käytännössä valmis pientä koodin siistimistä lukuun ottamatta, eli aika riitti lopulta hyvin.

Arvio lopputuloksesta

Ohjelman lopputulos on tehtävänannon mukainen ja toimiva, eikä siinä ole silmiinpistäviä vikoja tai puutteita. Graafinen käyttöliittymä on helppokäyttöinen ja selkeä, tosin tutustumalla syvemmin PyQt:n eri mahdollisuuksiin muokata widgettien ulkoasua, ohjelmasta olisi voinut saada jonkin verran

visuaalisesti miellyttävämmän. Mahdollisuus muokata säätöjen painokertoimia reaaliaikaisesti tekee ohjelmasta mielenkiintoisen, kun muutosten vaikutukset näkee välittömästi.

Ohjelman luokkajako on kohtalaisen yksinkertainen ja looginen. Luokkien metodit on muodostettu mielestäni järkevästi, eivätkä ne ole liian pitkiä tai toisteisia.

Simulaatioon olisi ollut mahdollista toteuttaa enemmänkin toimintoja, kuten maksiminopeuden ennalta määrittäminen tai muuttaminen simulaation aikana. Uskoisin vastaavien laajennusten onnistuvan ohjelmaan hyvin. Esimerkiksi juuri maksiminopeutta ja seinämien ”työntövoimaa” voisi säädellä hyödyntäen samoja ratkaisuperiaatteita kuin nykyisissä painokertoimien säätimissä.

Viitteet

Craig Reynoldsin sivusto
http://www.red3d.com/cwr/boids/

Craig Reynolds: Steering Behaviors for Autonymous Characters
http://www.red3d.com/cwr/steer/gdc99/

Craig Reynolds: Flocks, Herds, and Schools: A Distributed Behavioral Model (1987)
http://www.cs.toronto.edu/~dt/siggraph97-course/cwr87/

The Qt Company
http://doc.qt.io/qt-5/index.html

PyQt5-tutorial
http://zetcode.com/gui/pyqt5/

PyQt5 Reference Guide
http://pyqt.sourceforge.net/Docs/PyQt5/index.html

Python Software Foundation: Data Structures
https://docs.python.org/2/tutorial/datastructures.html