Dokumentaatio

Parvisimulaatio

5.5.2015

Roope Savolainen 481030 AIT 2015

Yleiskuvaus

Projektin tarkoituksena oli luoda parvisimulaatio-ohjelma, joka simuloisi esimerkiksi lintu- tai kalaparven liikettä. Ohjelma käytti hyödyksi Craig Reynoldin *Steering behaviors for autonomous characters* -artikkelia (http://www.red3d.com/cwr/steer/gdc99/).

Käyttäjä voi ohjelman ajon aikana muokata ohjelman eri parametreja, kuten esimerkiksi parven toimijoiden sisäisten sääntöjen painokertoimia. Ohjelma käyttää graafista käyttöliittymää parven simuloimiseen sekä asetusten asettamiseen.

Käyttöohje

Ohjelma tarvitsee toimiakseen PyQt-kirjaston, jota käytetään graafisen käyttöliittymän piirtämiseen, sekä Python3-tulkin, jotta ohjelma saadaan käynnistettyä. Kun nämä esivaatimukset on asennettu, saadaan ohjelma käynnistettyä suorittamalla python-tulkilla src-kansio (esimerkiksi komennolla “python3 src”).

Ohjelma aukeaa konfiguraationäkymään, jossa käyttäjä voi säätää ohjelman eri parametreja. Jos ohjelma löytää suorituskansiostaan default.cfg-nimisen tiedoston, se lataa tiedostosta oletusasetukset. Lisäksi käyttäjä voi ladata asetustiedostoja ikkunan alareunan Load configuration -napilla. Voimassa olevat asetukset voi myös tallentaa haluamaansa tiedostoon Save configuration -napilla. Kaikki asetukset säädetään yksinkertaisilla liukusäätimillä. Asetukset saadaan voimaan klikkaamalla apply-nappulaa. Kaikki asetukset astuvat heti voimaan apply-nappulan painamisen jälkeen, paitsi toimijoiden lukumäärä, joka alkaa vaikuttaa vasta, kun simulaatio käynnistetään uudestaan.

Vaihtamalla välilehteä ikkunan yläkulmasta Simulation-välilehteen, käyttäjä voi tarkastella itse simulaatiota. Simulaatio saadaan päälle Start-nappulaa painamalla ja voidaan keskeyttää väliaikaisesti Pause-napilla. Simulaatio voidaan aloittaa alusta Stop-napilla.

Ohjelman rakenne

Ohjelman perusrakenne on melko yksinkertainen. Keskiössä on Simulation-luokka, joka kuvaa simulaatiota mahdollisimman self-contained kokonaisuutena. Simulation ei esimerkiksi koske lainkaan Qt-komponentteihin. Toinen merkittävä luokka on Boid, joka on peritty QGraphicsItemistä grafiikoiden piirtämisen helpottamiseksi. Simulation-olio sisältää listan simulaation boideista, ja kutsuu boidien päivittämismetodia joka simulaation timestepissä. Simulation-olio sisältää myös Parameters-

olion, joka sisältää simulaation parametrit, joita käyttäjä voi säätää configuration-näkymässä tai ladata tiedostosta. Parameters-luokka hoitaa kaiken ohjelman tiedosto-IO:n sekä asetustiedoston parsimisen.

Näiden luokkien lisäksi ohjelman UI:ta varten QMainWindow:sta on peritty ApplicationWidget-luokka, joka hoitaa käyttöliittymän toimintaa, sekä ajastusta, jolla ajetaan Simulation-olion timesteppien suoritusta. ApplicationWidget siis sisältää itse käyttöliittymän lisäksi myös itse simulaation, joskin Simulation-luokka ei millään tavalla vaadi tätä, vaan se voisi olla myöskin täysin globaali olio. Kuitenkin, koska Simulation pidettiin tarkoituksella mahdollisimman ulkomaailmasta agnostisena, oli sen timesteppiä ajettava ulkopuolilsella ajastimella, jolloin luontevinta oli antaa se ApplicationWidgetin omistukseen. ApplicationWidget lukee UI-layouttinsa QtCreatorilla luodusta .ui-tiedostosta aina ohjelman käynnistyksen yhteydessä.

Algoritmit

Ohjelmassa ei käytetä mitään erityisen monimutkaisia algoritmeja. Monimutkaisimmat käytetyt algoritmit olivat Reynoldin artikkelissa määritellyt kolme erilaista steering-sääntöä parven toimijalle; alignment, cohesion ja separation.

Cohesionin toteutin yksinkertaisesti laskemalla toimijan naapureiden keskimääräiset koordinaatit ja osoittamalla kiihtyvyysvektorin toimijasta keskipisteeseen.

Alignmentin toteutin laskemalla naapuritoimijoiden keskimääräisen nopeusvektorin. Tämän jälkeen suuntasin kiihtyvyysvektorin tarkasteltavan toimijan nopeusvektorin ja keskimääräisen vektorin erotuksen mukaisesti.

Separationin toteutin aluksi ottamalla suuntavektorin naapuritoimijoista tarkasteltavaan toimijaan ja normalisoimalla nämä suuntavektorit. Tämän jälkeen painotin ne termillä 1/r, jossa r on etäisyys tarkasteltavan toimijan ja naapurin välillä. Tämän jälkeen yksinkertaisesti summattiin nämä painotetut vektorit.

Kun näistä kolmesta säännöstä oli saatu kolme eri komponenttia ohjaamaan toimijaa, painotin niitä vielä käyttäjän syöttämillä painokertoimilla. Lisäksi funktioiden sisällä painotin arvoja testaamalla löydetyillä vakioilla, jotka hieman tasoittivat eri sääntöjen tuottamia kiihtyvyysvektoreita.

Tietorakenteet

Ohjelmaa varten ei tarvinnut toteuttaa erityisiä tietorakenteita. Käytännössä kaikki tiedon varastointi hoitui listoilla.

Tiedostot

Ohjelman asetustiedoston säilöttiin json-muotoiseen tekstitiedostoon. Tiedostoa luettiin Qt:n tiedosto-IO -funktioilla ja itse json-data muodostettiin pythonin json-kirjastolla. Json-tiedosto oli hyvin yksinkertainen, koska sinne tallennettiin vain vakiomäärä ohjelman asetuksia.

Testaus

Ohjelman yksikkötestaus hoidettiin pythonin unittest-kirjastolla. Ohjelma testaa pääasiassa Boidin eri funktioita. Testaus voidaan ajaa suorittamalla python-tulkilla tiedosto src/test.py.

Ohjelman tunnetut puutteet ja viat

Ohjelma täyttää tehtävänannossa asetetut speksit, mutta toteuttamatta jäi muutamia ominaisuuksia, joita olin ajatellut sisällyttää ohjelmaan.

Suurimpia ongelmia mielestäni on toimijoiden liikkumismekanismi. Olisi varmaankin ollut järkevämpää, jos toimijat olisivat muokanneet suuntaansa kääntymällä, toisin kuin tämänhetkisessä projektin versiossa, jossa ne pystyvät muuttamaan nopeuttaan vapaaseen suuntaan. Tosielämässähän esimerkiksi linnut eivät pysty liikkumaan näin. Tämä olisi pitänyt huomioida jo projektin alkumetreillä jotta sen olisi saanut toimivasti mukaan projektiin.

Olisin lisäksi halunnut toteuttaa käyttäjän valittavaksi eri parametreja, jotka olisivat vaikuttaneet alkutilanteeseen. Esimerkiksi käyttäjä olisi voinut valita toimijoiden alkusijoitteluun vaikuttavia tekijöitä; missä muodostelmassa ne ovat alussa, minkälaisia nopeuksia ne saavat simulaation alussa…

Lisäksi en ehtinyt tarkemmin selvittää Qt:n tuottamia päänsärkyjä. Esimerkiksi kameran seuratessa parvea, parvi jää usein ruudun laidoille. Tämä johtuu siitä, että Qt:n grafiikka-stackin QGraphicsScene laajentaa itseään sitä mukaan, kun objekteja liikkuu sen aiempien reunojen ulkopuolelle. Tämän vuoksi QGraphicsView ei pysty scrollaamaan Scenen ulkopuolelle.

Parhaat ja heikoimmat kohdat

Olen tyytyväinen siihen, että simulaation parametreja pystytään muuttamaan ajon aikana. Tämä johtuu pitkältikin siitä, että otin ominaisuuden huomioon jo luokkia suunnitellessa ja parametrien

muuttaminen kesken ajon toimi käytännössä ilman ylimääräisiä säätöjä. Todennäköisesti jos ominaisuutta olisi alkanut miettiä vasta projektin loppuvaiheilla, olisi ollut hyvin työlästä toteuttaa se.

Projektin heikkoja kohtia onkin jo käyty yllä kohdassa *Ohjelman tunnetut puutteet ja viat*.

Poikkeamat suunnitelmasta

Projekti noudatti suunnitelmaa mielestäni hyvinkin tarkasti. Kuitenkin jäi muutamia ominaisuuksia, jotka olisin halunnut mahdollisesti toteuttaa, mutta aika loppui kesken. Näistä enemmän ym. Kohdassa *Ohjelman tunnetut puutteet ja viat*.

Toteutunut työjärjestys ja aikataulu

Aikataulu venähti loppua kohden ja en saanut tämän vuoksi toteutettua kaikkia haluamiani ominaisuuksia, mutta kuitenkin pääosin pysyin aikataulussa. Olin onneksi varannut aikataulun loppupuolelle vähemmän konkreettista toteutettavaa, joten aikataulusta myöhästyminen projektin alkupuolella ei ollut onneksi niin vakavaa.

Arvio lopputuloksesta

Vaikka ohjelmasta puuttuikin ominaisuuksia, joita olisin siihen halunnut sisällyttää, mielestäni kokonaisuutena ohjelma on ihan hyvä. Se vastaa speksejä ja simulaatio näyttää jossakin määrin realistiselta parvelta.

Jos alkaisin näillä tiedoilla työstämään projektia alusta, ottaisin todennäköisesti huomioon vaihtoehtoiset toimijoiden liikkumistavat, kuten mainitsinkin jo yllä. Simulaatio voisi näyttää huomattavasti järkevämmältä, jos toimijoiden liike muistuttaisi enemmän lintujen tai kalojen liikkumista.

Viitteet

Craig Reynolds:

http://www.red3d.com/cwr/steer/gdc99/

Pythonin dokumentaatio:

https://docs.python.org/3.4/library/json.html

Qt:n dokumentaatio:

http://doc.qt.io/qt-5/index.html

Liitteet

Ohjelman lähdekoodi sekä hyvät asetukset sisältävä asetustiedosto löytyvät gitistä.