Python-projekti: Parvisimulaatio

Tekninen suunnitelma

Veera Itälinna
525608
Bioinformaatioteknologia, 2. vuosikurssi
9.3.2017

Ohjelman rakennesuunnitelma

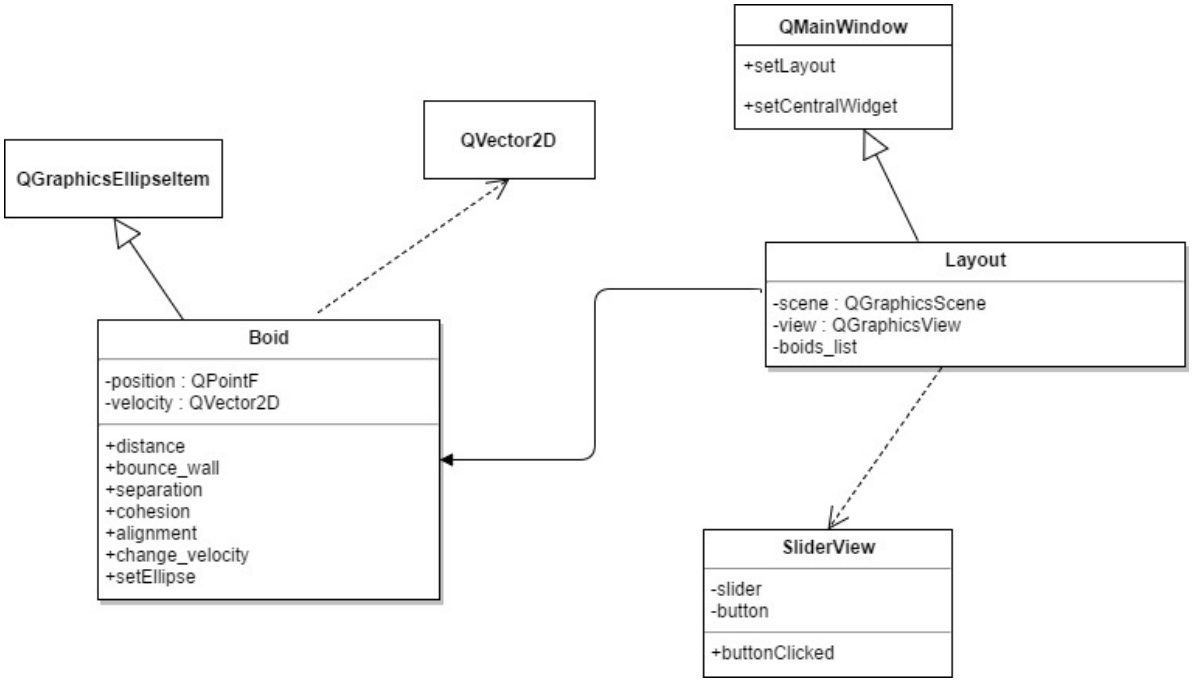
Ohjelman keskeisin luokka on parven yksilö eli Boid. __init__-metodissa alustetaan yksilöiden alku-sijainti ja –nopeus. Muita luokan metodeja voisivat olla ainakin metodi, joka laskee toisen yksilön etäisyyden, yksilön nopeusvektoria muuttavat metodit sekä eri liikkumissäännöt (Separation, Alignment, Cohesion) määrittävät metodit. Tarvitaan luultavasti myös metodi tai metodeja, jotka muuttavat sääntöjen painokertoimia simulaation kuluessa käyttäjän niin halutessa. Luokassa myös määritellään parven yksilöiden yksinkertainen graafinen toteutus QGraphicsEllipseItemin avulla.

Boid-luokka käyttää hyväkseen Qt:n QVector2D-luokkaa, jonka avulla voidaan ilmaista Boid-olion nopeus vektorina.

Graafinen käyttöliittymä luodaan PyQt-kirjaston avulla: luokista tarvitaan ainakin QApplication, QWidget ja QSlider. Käyttöliittymä kokonaisuutena tullaan myös luultavasti toteuttamaan omassa luokassaan. Myös slider, jossa määritetään parven koko ennen simulaation alkamista, tarvitsee ehkä oman luokan.

Lisäksi toteutetaan main-funktio, joka avaa graafisen käyttöliittymän ja käynnistää simulaation.

Alla hahmoteltuna alustavaa luokkajakoa UML-kaaviona.



Käyttötapauskuvaus

Ohjelman käynnistyessä avautuu ikkuna, jossa käyttäjä voi slidereilla määrittää lintujen lukumäärän sekä mahdollisesti myös alkunopeuden tason. Simulaation käynnistyessä listaan tallentuu käyttäjän valitsema määrä Boid-luokan instansseja, joille on arvottu alkupiste (x- ja y-koordinaatit) sekä alkunopeus. Avautuu ikkuna, jossa pisteistä muodostuva parvi liikkuu määritettyjen käyttäytymissääntöjen mukaisesti.

Ikkunassa on myös kolme slideria, joilla käyttäjä säätää eri sääntöjen painokertoimia. Standardiarvoina ovat kokeilemalla selvitetyt kertoimet, joilla parvi vaikuttaa liikkuvan luonnollisimmin. Jokaisella ajanhetkellä ohjelma tarkistaa, onko asetuksia muutettu, ja tekee tarvittaessa määrätyt muutokset Boid-luokan metodien avulla.

Käyttäjä haluaa keskeyttää simulaation, jolloin hän painaa ikkunassa olevaa nappia. Tällöin ikkuna sulkeutuu ja avautuu jälleen alkuperäinen ikkuna, jossa käyttäjä voi jälleen säätää haluamansa alkuasetukset uutta simulaatiota varten.

Algoritmit

Jokaisella parven yksilöllä (lintuoliolla) on tieto muiden yksilöiden sijainnista. Yksilöllä on myös ympärillään oma ympyränmuotoinen ”havaintoalueensa”. Yksilö määrittää tietyn, lyhyen ajanjakson välein, mitkä muut parven yksilöt ovat tämän alueen sisäpuolella. 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. Vasteen voimakkuus on kääntäen verrannollinen yksilöiden välisen etäisyyden neliöön, jolloin saadaan toivottavasti aikaan luontevan näköinen liike.

Separation-sääntö toteutetaan niin, että yksilön ”havaintoalueen” sisällä olevien lintujen sijaintien perusteella lasketaan jokaisen linnun aiheuttama ”työntövoima”. Voimat summataan yhteen, ja saadun kokonaisvoiman perusteella muutetaan yksilön suuntaa.

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.

Eri sääntöjen aiheuttamat nopeusvektorit summataan yhteen omilla painokertoimillaan (voidaan muuttaa simulaation aikana), ja saadun summavektorin avulla muutetaan lopulta yksilön suuntaa. (Nopeus ei saa kuitenkaan ylittää vakiona määrättyä maksiminopeutta.)

Craig Reynoldsin teksteissä mainitaan myös tapoja, joilla Boid-olion ei tarvitsisi olla tietoinen jokaisen muun parven yksilön sijainnista, ja joilla siten voisi tehostaa laskentaa. Ne vaikuttivat kuitenkin vaikeilta toteuttaa, joten parempi ratkaisu vaikuttaa olevan yksinkertaisesti parven maksimikoon pitäminen riittävän pienenä. Toki tähänkin voin perehtyä, jos aikaa riittää.

Tietorakenteet

Käyttäjän valitsema määrä parven yksilöitä eli Boid-luokan instansseja tallennetaan listaan, sillä se on helppo toteuttaa ja käydä ohjelman aikana läpi. Parven maksimikoko pidetään sen verran pienenä, että listan läpikäyminen ei vie liikaa laskentatehoa.

Aikataulu

Aloitin parvisimulaatioprojektin jo viime vuoden Y2-kurssilla. Sain projektia jo melko pitkälle tehtyä, esimerkiksi graafinen käyttöliittymä on käytännössä valmis. Parvi ei kuitenkaan vielä liiku oikein, ja aion myös toteuttaa luokkajaon hieman järkevämmin. Tuntimäärää debuggaamiseen on luonnollisesti hyvin vaikea arvioida.

Kokonaisuudessaan uskon ajan riittävän tällä kertaa hyvin projektin vaatimusten mukaiseen toteutukseen. Jos aikaa jää yli, yritän kehitellä simulaatioon vielä lisäominaisuuksia tai panostaa sen visuaalisuuteen.

Yksikkötestaussuunnitelma

Animaation toimivuuden näkee pääosin silmämääräisesti, joten yksikkötestit tuskin ovat välttämättömiä. Simulaation kuluessa muuttuville numeerisille arvoille (lintujen nopeudet ja sijainnit sekä sääntöjen painokertoimet) voidaan kehittää lisäksi jonkinlaisia testejä. Olennaisimpia testimetodeja niille tulevat kuitenkin luultavasti olemaan assertEqual, assertGreater ja assertLess.

Kirjallisuusviitteet ja linkit

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