Seuraa johtajaa -simulaatio

6.5.2015

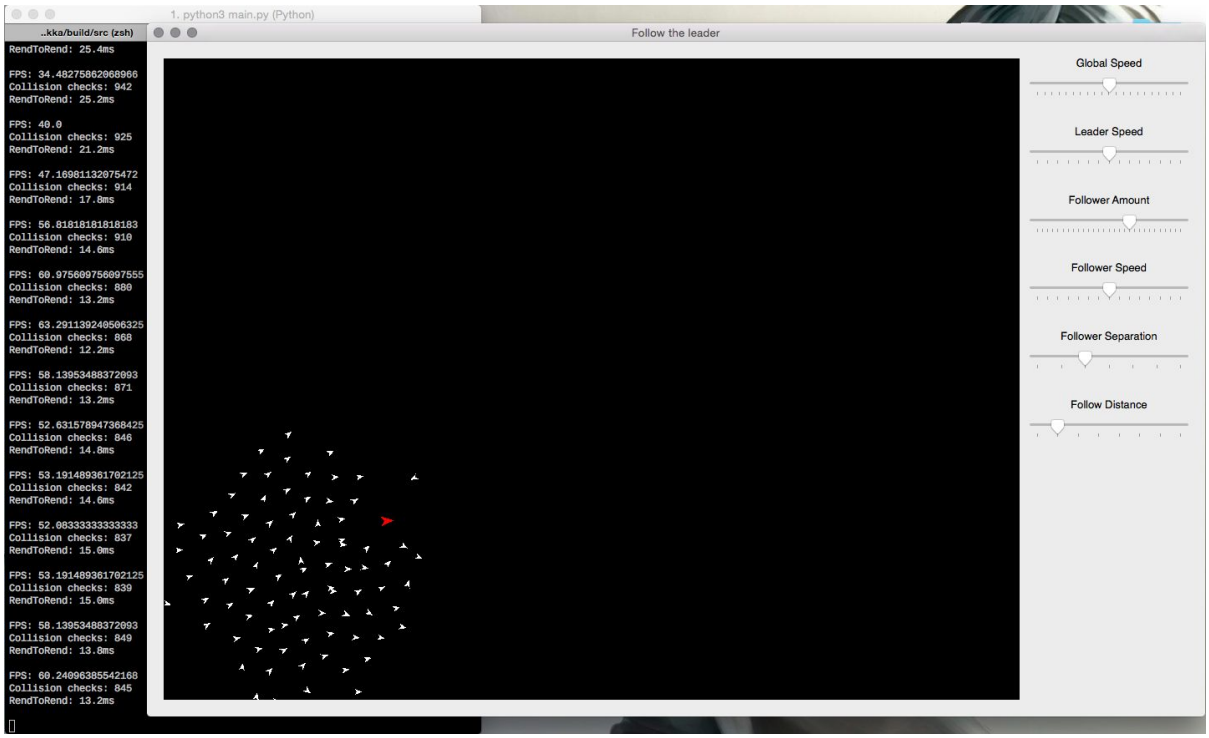
Santeri Salmijärvi
351885
AIT
2. vk.

Yleiskuvaus

Kyseessä on seuraa johtajaa -simulaatio, jossa vaihtelevan kokoinen lauma seuraa yhtä johtajaa annettujen etäisyyksien mukaan. Johtaja joko kuljeksii satunnaisesti, hakeutuu käyttäjän antamaan kohteeseen tai liikkuu käyttäjän ohjauksen mukaan. Projekti on mielestäni selvästi aiheessa määritetyn vaativan puolella.

Käyttöohje

Ohjelma käynnistetään ajamalla main.py python3-tulkilla.



Ikkunassa on vasemmalla OpenGL:llä piirretty simulaationäkymä ja vasemmalla simulaation parametreihin kytketyt slaiderit. Slaidereilla voi muuttaa parametrejä ajon aikana, jolloin muutos näkyy heti simulaation puolella. Komentoriville tulostetaan ruudunpäivitysnopeus, laskettujen “törmäysten” määrä sekä niihin kulunut aika millisekunteina.

Johtaja on mahdollista kaapata suoraan ohjaukseen painamalla välilyöntiä ja tämän jälkeen vasen/oikea nuoli kääntää sitä. Välilyönnin uudelleen painaminen vapauttaa johtajan suorasta ohjauksesta. Simulaatiossa klikkaamalla voi asettaa johtajalle kohteen, johon se liikkuu ennen vaeltelun jatkamista. Suoralla ohjauksella on kuitenkin korkein prioriteetti.

Ohjelman rakenne

Ohjelman ylimmän tason muodostaa ikkunaolio, jonka sisälle luodaan simulaatiosta vastaava OpenGL-konteksti ja tarvittavat QT-slaiderit. OpenGL-kontekstin sisältä löytyvät Leader-, Follower-, Grid sekä simulaation parametriluokat. Seuraajat ovat listassa piirtoa ja laskentaa varten, mutta törmäyslaskenta tapahtuu optimointisyistä Grid -kaksiulotteisen listan läpi. Pelkällä kaksiulotteisellakin listalla voisi operoida, mutta priorisoin koodin selkeyttä muistioptimoinnin yli.

Leader-luokassa on omat metodit yleiselle liikkumiselle, joka kutsuu tilan mukaan kunkin liikkumismoodin omaa metodia. Follower-luokalla on erotettu törmäyslaskenta- ja liikkumismetodi. Edellisillä luokilla on myös omat piirtofunktiot, joita kutsutaan pääikkunan OpenGL-luokasta.

Laskenta tapahtuu osittain käyttäen itse kirjoitettua vektorikirjastoa.

Algoritmit

Leader

Johtajalla on kaksi eri algoritmia kääntävän voiman laskennalle. Kohteen haku toteutetaan laskemalla johtajan ja kohdepisteen välinen vektori, katkaisemalla tämä määritetyn maksimivoiman pituiseksi.Vaeltelu on toteutettu laskemalla edelliseen liikkumis suuntaan vakio vektorin päähän toinen vektori, jonka kulma on satunnainen sopivalla välillä. Näiden yhteenlaskettu ja maksimivoimaan rajoitettu vektori on edellisen tapaan kääntävä voima.

Tästä saadaan “kiihtyvyys” jakamalla se massalla ja kertomalla simulaation nopeuden määrittävällä vakiolla. Kiihtyvyys lisätään vanhaan nopeuteen katkaisemalla maksiminopeudesta uuden nopeuden laskemiseksi ja uusi paikka on yksinkertaisesti vanha paikka lisättynä uudella nopeudella.

Lisäksi kaikille reunoille on vaeltelussa määritetty tietty etäisyys, jolla johtajaan kohdistetaan takaisin keskelle työntävä voima kääntäen verrannollisena etäisyyteen reunasta. Näin johtaja ei vaella rajojen ulkopuolelle.

Vaeltelun olisi voinut totetuttaa myös suoraan kääntämällä liikevektoria, mutta käytetty tapa tuottaa pehmeämmän lopputuloksen. Vastaavaan olisi päästy integroimalla suoraa kääntöä, mutta tämä menisi epäselvemmäksi koodin kannalta, eikä varsinaista etua saavutettaisi käytökseen tai tehokkutteen.

Follower

Seuraaja noudattaa johtajasta tuttua kohteen haun algoritmia sopivin laajennuksin. Ensinnäkin kohde lasketaan parametreissa määritetylle etäisyydelle johtajan taakse. Lisäksi seuraaja alkaa hidastaa asetetulla etäisyydellä seurattavasta pisteestä ja pysähtyy, mikäli saavuttaa pisteen (voima *= etäisyys / määritettyEtäisyys).

Seuraajille on lisäksi määritetty “laumakuri” eli etäisyys, joka niiden tulisi pitää muista lauman jäsenistä. Tämä on toteutettu hakemalla aiemmin mainitusta kaksiulotteisesta listasta kaikki kunkin seuraajan lähinaapurit. Erilliseen kääntövoimaan lisätään kustakin naapurista poispäin työntävä voima (jälleen etäisyyden ja määritetyn välimatkan suhteessa), mikäli etäisyys on tarpeeksi pieni. Tämän lisäksi johtaja ja sen eteen määritetty piste toimivat väistettävinä kohteina, mikäli etäisyys on alle johtajan seuraamiseen määritetyn etäisyyden. “Väistövoima” lisätään jo katkaistusta kääntövoimasta, jotta väistön vaikutus kasvaisi. Lopputuloksena saatu voima katkaistaan toki uudestaan määritetylle maksimivoimalle.

Johtajan väistämisen olisi voinut toteuttaa elegantimmin toteuttamalla johtajan liikesuunnan väistämisen jonkinlaisen vektorin suhteen, mutta käytetty ratkaisu on tehokkaampi ja riittävän tarkka.

Vector2f

Vektorikirjasto määrittää ohjelman käyttämät algoritmit vektorin pituuden laskemiselle, normalisoinnille, katkaisemiselle ja kääntämiselle sekä kahden vektorilla määritetyn pisteen etäisyydelle toisistaan. Lisäksi luokka sisältää määrittelyt pythonin +, -, * ja / -operaattoreille.

Grid

Törmäyslaskentaan käytettyyn kaksiulotteiseen listan koordinaatit saadaan jakamalla seuraajan paikka 32:lla ja lisäämällä se saatavan kokonaisluvun osoittamille indekseille.

Tietorakenteet

Projekti käyttää kaksiulotteisena listana määritettyä Grid-luokkaa törmäyslaskentaan ja c-structin omaisia parametriluokkia simulaation parametrien asettamiseen. Gridille en keksinyt vaihtoehtoja ja parametriluokat tuntuivat erillistä asetustiedostoa tehokkaammalta ratkaisulta näin pienen ohjelman kanssa. Niissä kuitenkin määritetään vain oletusarvot, joiden pohjalta on asetettu myös slaiderien arvot.

Gridin lisäksi merkittävä oma tietorakenne on Vector2f, joka on nimensä mukaisesti kahden liukuluvun (x, y) sisältävä luokka, jota käytetään paikkadatan ja liikkumiseen liittyvien vektorien pyörittelyyn.

Testaus

Ohjelmaa on testattu pääasiassa simulaatiota ajamalla sekä komentorivitulosteilla. Näiden lisäksi näin parhaaksi kirjoittaa yksikkötestit vektoriluokalle, joka on keskeinen varsinaisten simulaatioalgoritmien toiminnan kannalta. Nämä testit menevät läpi. Lisätestejä olisi ollut vaikea kirjoittaa, enkä kokenut niitä tarpeellisiksi, sillä tämän simulaation kanssa pienet epätarkkuudet ovat huomaamattomia

Tunnetut puutteet ja viat

Toisinaan ohjelma heittää suljettaessa OpenGL-virhekoodin 1286 tai segmentation faultin, mitkä johtunevat QT:n kanssa käytetystä ajastinrakenteesta.

Seuraajat päätyvät toisinaan toistensa läpi jääden hetkeksi sisäkkäin. Tämä jäi kompromissiksi, jotteivät seuraajat pyörisi hervottomasti toisiaan väistellessä.

Törmäyslaskenta vie vieläkin hieman turhan paljon aikaa, minkä takia se tapahtuukin puolet simuloinnin liikettä toteuttavaa funktiota hitaammin. Tämä ei näy enää simulaation nopeudessa, mutta ruudunpäivitys muuttuu epävakaammaksi suurilla seuraajamäärillä.

Komentorivitulosteen käyttäminen on purkkaratkaisu, koska QT:n oman labelin päivittäminen oli kohtuuttoman raskasta, enkä löytänyt parempaa vaihtoehtoa ikkunaan piirtämiselle tekemättä omaa OpenGL-pohjaista ratkaisua, joka olisi vaatinut suhteettomasti lisää koodia.

3 parasta / 3 heikointa

Onnistuneimmat osat ovat mielestäni pelkistetyn toimiva ulkoasu, parametrimuutosten välitön näkyminen simulaatiossa sekä kohtalaisen laaja toimiva skaala kyseisissä parametreissa.

Suurimpina heikkouksina ovat törmäyslaskennan optimointi, vieläkin hieman liikaa tärisevät seuraajat sekä niihin käytetty kaksinkertainen muistirakenne. Optimointiin olisi voinut koittaa etsiä vielä uutta tapaa gridin kilpailijaksi ja täriseviä seuraajia voisi kokeilla rauhoitella viilaamalla parametreja tai lisäämällä jonkin asteista integrointia tai ennustusta väistöalgoritmiin.

Poikkeamat suunnitelmasta

En kirjoittanutkaan testejä slaidereille, koska niiden toteutus osoittautui täysin triviaaliksi ja virheet tunnisti heti käsin testaamalla. Johtajan yhdistelmäkäytös muuttui oletusvaelteluksi, käsketyksi kohteen hauksi ja käyttäjän ohjaukseksi. Lisäksi luokkien metodit muotoutuivat hieman suunnitelmasta poikkeaviksi ja reset-toiminto jäi toteuttamatta. Ajankäyttöarvio ja suunniteltu työjärjestys toteutuivat suurimmalta osalta, vaikka aikaa tulikin loppuvaiheessa käytettyä yleiseen viilaamiseen.

Toteutunut työjärjestys ja aikataulu

Gitistä siistitty päälisäysten loki näyttää seuraavalta:

Fri May 6 10:27:14 2016 +0300 - Add comments and split collisions for optimization
Fri Apr 29 14:31:05 2016 +0300 - Add seeking to mouse clicks
Fri Apr 29 14:18:33 2016 +0300 - Add mouse click capture
Thu Apr 7 16:54:32 2016 +0300 - Add render/compute info printing
Wed Mar 23 17:40:46 2016 +0200 - Add working but unoptimized grid collisions
Mon Mar 21 21:43:30 2016 +0200 - Enable stopping time entirely
Sun Mar 6 00:06:26 2016 +0200 - Add window layout and implement sliders
Sat Mar 5 21:40:23 2016 +0200 - Add leader avoidance
Sat Mar 5 21:07:08 2016 +0200 - Add multiple followers and separation
Sat Mar 5 15:30:15 2016 +0200 - Add a single follower
Sat Mar 5 00:44:24 2016 +0200 - Add wandering for leader
Fri Mar 4 22:52:21 2016 +0200 - Remove redundant vector functions and finish tests
Fri Mar 4 12:55:16 2016 +0200 - Add vector calculations and start on tests
Wed Mar 2 17:07:02 2016 +0200 - Add Leader and Parameters -classes
Wed Mar 2 16:37:45 2016 +0200 - Add ogl-context and draw an arrow
Tue Mar 1 14:40:41 2016 +0200 - Initial commit

Tästä nähdään, että arvio kahdesta viikosta raakaversioon alittui slaiderien ollessa valmiina kuutisen päivää ensimmäisen commitin jälkeen. Tämän jälkeen käytin satunnaisia päiviä projektin hiomiseen. Järjestyksestä poikkesin tekemällä johtajaluokan ennen Vector2f:ää ja toteuttamalla vaeltavan käytöksen ensin.

Arvio lopputuloksesta

Projekti sujui odotusten mukaisesti, vaikka pari suunnittelemaani pikkuominaisuutta jäivätkin toteuttamatta luultuani, että lopullinen dedis on vasta hieman myöhemmin. Ohjelma on suorituskykyvalitteluistani huolimatta melko sulava myös täydellä 160:llä seuraajalla ja kohtalaisen helposti jatkokehitettävä. Ensimmäinen parannuskohde olisi varmaankin törmäysten lisäoptimointi ja seuraajien liikkeiden “pehmentäminen”.

Viitteet

-Python3 API
-http://gamedevelopment.tutsplus.com/series/understanding-steering-behaviors--gamedev-12732
-http://www.red3d.com/cwr/steer/gdc99/

Liitteet

-ohjelman lähdekoodi