Ruuhkasimulaatio

Mikko Hemilä, 428925

Tietotekniikka, 2013

26.4.2018

Yleiskuvaus

Projekti on ruuhkasimulaatio, joka mallintaa joukon liikehdintää sen pyrkiessä kulkemaan huoneessa olevan yhden oviaukon läpi.

Joukkojen liikkumista simuloidaan hyvin yksinkertaisesti antamalla kullekin joukon yksilölle yhteiset ehdot, joiden mukaan ne navigoivat.

Yksilöiden liikettä tässä ryysissimulaatiossa ohjaavat seuraavat periaatteet:

- Oven etsintä:
  - ovi ”vetää huoneessa olijoita puoleensa” (Seeking)
  - huoneessa olijat välttävät seiniä (Avoidance) (he eivät halua joutua seinää vasten eivätkä etenkään seinään, koska se on mahdotonta)
- Läpikulku:
  - Ihmiset pyrkivät kulkemaan oven läpi (Path following)
- Läpikulun jälkeen:
  - Jotta ihmiset eivät enää etsiydy ovelle suunnistavat ne eteenpäin, simulaatiossa vasemmalle
- Kaikissa yllämainituissa:
  - huoneessa olijat hiljentävät vauhtiaan kun he huomaavat lähellä edessään muita, jotka liikkuvat hitaammin kuin he itse (Braking)
  - huoneessa olijat välttävät törmäilyä toisiinsa muuttamalla suuntavektoriaan sopivasti. (Separation)

Yllä esitettyjä yksinkertaisia ehtoja käyttäen huoneessa olijat eivät törmäile toistensa päälle ainakaan kovin vahvasti. Seinän läpikulku on estetty erillisellä törmäyksen huomioinnilla. Projektia ei kuitenkaan ole hiottu täydelliseksi, joten ihmiset voivat olla jonkin verran seinän tai muiden ihmisten sisällä.

Projektin vaikeusaste on toteutettu keskivaikeana, sillä extra ominaisuuksia on vain muutamia.

Käyttöohje

Projekti saadaan käynnistettyä UI.scala tiedoston kautta. Projektin paketti täytyy purkaa eclipseen ja avata sitten eclipsessä. Tämän jälkeen täytyy etsiä UI.scala (pitäisi olla: the_ruuhkasimulaatio/ src/ ryysissimulaatio/ UI.scala). UI.scalaa täyty painaa oikealla hiiren painikkeella etsiä kohta Run As ja sieltä valita Scala Application.

Tämän jälkeen pitäisi avautua ensiksi kolme peräkkäistä ikkunaa, joissa voidaan valita huoneen leveys, korkeus ja ihmisten määrä. Näihin valikkoihin ei saa kirjoittaa muuta kuin numeroita tai ohjelma kaatuu. Huoneen minimi leveys ja korkeus on 200 ja jos jommalle kummalle annetaan tätä pienempi arvo nostetaan se takaisin 200.

Kun tämä alustus sarja on tehty pitäisi avautua kaksi ikkunaa. Yksi, jossa on yksinkertainen huone täynnä sinisiä pisteitä (ihmiset) ja asetusvalikko, jossa voidaan hallita käytettäviä ohjausalgoritmin osia (Wander, Seek, Wall Avoidance, Separation). Asetusvalikossa voidaan myös hallita ihmisten yleistä nopeutta ja kiihtyvyyttä sekä massaa, tosin massa ja kiihtyvyys ovat tässä käytännössä sama asia.

Ohjelman saa aloitettua pyörimään painamalla start nappulaa, joka muuttuu stop nappulaksi. Stop nappula pysäyttää väliaikaisesti ohjelman. Restart nappulalla voi satunnaistaa uuden huoneen, mutta se käyttää alussa annettuja parametrejä. Jos näitä parametreja haluaa muuttaa täytyy ohjelma käynnistää uudelleen. Quit nappula sekä yläkulman ruksi sammuttavat molemmat ohjelman.

Ohjelman rakenne

Ohjelma koostuu viidestä eri tiedostosta:

- UI
- Human
- SteeringAlgorithm
- Room
- Vector2D

UI

Ui hallitsee tässä projektissa aikaa ja käyttöliittymää sekä graafista esitystä. UI.scalan kautta voidaan käynnistää ohjelma, minkä jälkeen se pyytää alussa tarvittavia tietoja. Näitä tietoja käyttäen se luo uuden huoneen ja käyttää huoneen antamia tietoja piirtääkseen kuvan tilanteesta. Lisäksi se hallitsee ohjausalgoritmin käyttämiä osia asetuspaneelin avulla. Asetuspaneelin tilanteen muuttuessa muutetaan myös huoneen muuttujaa, jota käytetään ohjauspaneelissa.

Human

Human on ihmisolio eli simulaation kulkuväline. Se pitää kirjaa sijainnistansa sekä nopeudestansa ja muuttaa näitä move-metodin avulla joka kierros. Move-metodi kutsuu getAcceleration-metodia, joka kutsuu ohjausalgoritmin vastaavaa metodia. Ohjausalgoritmin metodin antamaa vektoria käytetään nopeuden muuttamiseen ja nopeusvektoria käytetään paikan muuttamiseen. Human

luokalla on myös kumppaniolio, jolla hallitaan maksimi nopeutta, maksimi kiihtyvyyttä sekä kiihtyvyys-, nopeus-, ja massakertoimia, joita voidaan muuttaa asetuspaneelin kautta.

SteeringAlgorithm

Ohjausalgoritmi on koko projektin ydin. Se vaikuttaa ihmisten liikkumiseen antamalla niille kiihtyvyysvektorin ja siten liikuttaa heitä. Tämä kiihtyvyys vektori kootaan getAcceleration metodissa ja koostuu useammasta osasta. Tärkeimmät osat ovat: Seeking/PassThrough, WallAvoidance, Wander ja Separation. Seeking on vektori suoraan ovea kohti ja siten ohjaa ihmisiä ovea päin. PassThrough on vektori, joka kytkeytyy päälle ihmisten päätyessä tarpeeksi lähelle oviaukkoa. Se pyrkii saamaan ihmiset kulkemaan oven läpi mieluusti sen keskeltä. WallAvoidance vektori etsii sopivan lähellä olevat pysty- ja vaakasuorat seinät ja niiden seurauksena ohjaa ihmisiä suoraan poispäin niistä. Wander on muuttuva, mutta muistissapysyvä vektori, joka saa ihmiset kulkemaan satunnaisesti huoneessa. Separation vektori pyrkii ohjaamaan ihmisiä poispäin muista ja siten hidastamaan heidät paikallensa tungoksessa.

Room

Huone on simulaation maailma. Sillä on seinät ja yksi oviaukko. Kun luodaan uusi huone annetaan sille leveys, korkeus ja ihmisten määrä. Ihmisten alkusijainti satunnaistetaan huoneen sisälle. Lisäksi huoneessa pidetään kirjaa ihmisistä ArrayBufferina, koska alussa oli tarkoitus poistaa ihmiset, kun he kulkisivat oven läpi. Huoneessa myös tallennetaan ohjausalgoritmin asetukset. Nämä asetukset pitäisivät todellisuudessa olla ohjausalgoritmin kumppaniluokassa, mutta en ajatellut asiaa ajoissa. Huoneessa on myös metodi runRound, joka kutsuu jokaisen ihmisen move-metodia.

Vector2D

Vectori luokka toimii apuna vektorien hallinnassa. Sen pohjana oli asteroid-projektin vektori luokka, mutta olen tehnyt siihen monia muutoksia. Luokka tarjoaa erilaisia metodeja vektorien manipuolintiin, kuten lisäksen ja kerto sekä jakolaskun. Vektoreita käytetään useissa eri osuuksissa, kuten ihmisten sijainnissa ja ohjausalgoritmissa.

Algoritmit

Algoritmeja on jonkin verran käytössä erityisesti ohjausalgoritmissa.

Seeking tapahtuu poistamalla ihmisen sijainti oven sijainnista. Tästä saadaan vektori, jonka suunta on ihmisestä oveen, Tämän jälkeen sen pituus rajoitetaan ihmisen maksiminopeudeksi, jotta sen vaikutus olisi vakio.

Wander perustuu tallessa pidettävään vektoriin, joka pidetään rajattuna halutun laisen ympyrän ulkopinnalla. Sen liikkuu satunnaisesti vähän, mutta palautetaan aina tämän ympyrän ulkopinnalle. Tämän avulla saadaan jatkuvia käännöksiä. Tässä simulaatiossa voidaan sanoa, että sen vaikutus on liian suuri, mutta halusin myös tämän näkyvän.

Separation on todennäköisesti simulaation raskain osuus, sillä se tarkistaa kaikkien muiden ihmisten sijainnin (kasvaa siis N^2) ja jos joku on liian lähellä antaa vektorin poispäin tästä ihmisestä yleensä hidastaen kulkua.

Wall avoidance toimii separaation kaltaisesti, mutta seinille. Seinän ollessa liian lähellä aiheutuu siitä seinästä poispäin oleva vektori. Lisäksi läpikulkua seinistä estää myös ihmisissä oleva törmäyksen huomaus. Tämä algoritmi ajetaan joka liike kierros ja se tarkistaa, että jos ihminen ei ole tarpeeksi lähellä ovea tai oven vasemmalla puolella niin ihmisen sijainti ei voi kulkea seinän läpi.

Tietorakenteet

Tietorakenteet eivät ole tässä projektissa mielenkiintoisessa asemassa. Ihmiset tallennetaan ArrayBufferiin, koska oli tarkoitus voida muuttaa heidän määrää, mutta tämä jäi toteutumatta. Merkittäväin kysymys ohjelman tehokkuuden kannalta olisi separation algoritmi. Tämän voisi parantaa esim rajoittamalla eri tavoin etsittäviä ihmisiä, mutta näitä ei ohjelmassa toteutettu.

Testaus

Ohjelmaa testattiin vähän ohjelmoimisen yhteydessä. Projektia varten ei kirjoitettu testejä ja ongelmakohdat korjattiin pääosin vain, kun ne tulivat vastaan. Ohjelmaa ei myöskään ole testattu ultra pienellä huoneella ja isolla ihmismäärällä tai liukusäätimien ääritapauksissa, joten on mahdollista (tosin epätodennäköistä) että outoja asioita tapahtuisi.

Bugit

Merkittävin puute on se, että aloitus laatikkoihin voi kirjoittaa vain numeroita. Jos niihin kirjoittaa muuta kaatuu ohjelma todennäköisesti. Lisäksi ohjelmaa ei ole hiottu täydelliseksi, joten ihmiset voivat päätyä jonkin verran seinän sisään tai toistensa päälle. Seinän läpi niiden ei kuitenkaan pitäisi päätyä (On mahdollista, että ihminen voisi päätyä seinän läpi, jos hän kulkee tarpeeksi hitaasti, mutta tämä on teoreettinen oletus). Huoneen aloituksessa ihmisten sijoittaminen on täysin satunnaista, joten on mahdollista että he syntyvät päällekäin.

Poikkeamat suunnitelmasta (ja heikkouksia)

Seurasin suunnitelmaa melko vähän, sillä en ilmeisesti ollut tehnyt sitä kovinkaan kattavasti. Lisäksi kandidaatintyöni aiheutti ongelmia ajankäytön osalta, joten en käyttänyt projektiin riittävästi aikaa toteuttaakseni hienompia ominaisuuksia. Erityisesti en ole tyytyväinen projektini lopulliseen tiedon kulkuun, esimerkiksi huone tallentaa muuttujia, jotka pitäisivät olla ohjausalgoritmissa. Lisäksi UI oli melko tuntematonta aluetta, joten toteutin esimerkiksi ohjauspaneelin erillisenä ikkunana enkä osana huone ikkunaa. Ajankäyttö arvio oli ylimitoitettu, sillä olin nopea ohjelmoimaan, mutta käytin myös liian vähän aikaa. Toteutusjärjestys oli suurinpiirtein oikeassa.

Arvio lopputuloksesta

Ohjelma toteuttaa perusosuuden, mitä siltä pyydettiin eli ohjausalgoritmin. Se myös antaa graafisen käyttöliittymän ja mahdollisuuden kontrolloida ohjausalgoritmin osia ohjelman pyöriessä. Ohjelmaa ei ole testattu kovinkaan laajasti ja projektiin käytetty aika oli tarkoitusta pienempi, koska tärkeämpi kurssi satutti päätä. Ohjelmakoodi on melko selkeää, mutta tiedonkulku ja jotkut rakenteet voi olla

paremmin toteutettu. Koodia on kommentoitu sopivasti ja muuttuja on nimetty selvästi. Ohjelmaa voisi parantaa tästä koodista, mutta olisi suositeltavaa ohjelmoida se uudelleen, koska tämä mahdollistaisi paremman suunnitelman ja ohjelman rakentamisen. Ohjelma on melko hyvä.

Viitteet

A+ tiedot: https://plus.cs.hut.fi/studio_2/k2018

Craig. W. Reynolds, Steering Behaviors For Autonomous Characters: http://www.red3d.com/cwr/steer/gdc99/

Stack Overflow: https://stackoverflow.com