Määrittelydokumentti, Eclipse calculator

Ongelma

Ohjelman tehtävä on laskea voittotodennäköisyys erässä Eclipse lautapelin taisteluvaihetta.

Määritelläkseen voittotodennäköisyyden, algoritmi selvittää jokaisen aluksen todennäköisyyden selvitä taistelusta hengissä ja tarvittaessa toistaa pelikierroksia kunnes on todennäköistä että jompi kumpi pelaajista on hävinnyt. Tarkat rajat joita voiton määrittelyyn käytetään voidaan tarkentaa ohjelman toimintaa testattaessa.

Ohjelma on rajattu sisältämään alukset, jotka pystytään rakentamaan pelin core versiolla, ilman lisäosien aluksia ja aluspäivityksiä. Ohjelma liitetään valmiiseen aiemmin tekemääni käyttöliittymään osoitteessa:

http://darth.kipsu.fi/EclipseCalculator/

Ohjelman käyttö

Ohjelmaa käytetään nettikäyttöliittymän kautta. Käyttöliittymästä saa tällä hetkellä ulos listan taisteluun osallistuvista aluksista JS objekteina ja tarkoitus on siirtää nämä JSON muodossa ohjelman käsiteltäviksi.

Ohjema muuttaa JSON objektit käyttämäänsä tietomuotoon. Ainakin aluksi mapiksi, mutta mikäli map pitää toteuttaa itse niin joksikin omaksi vektoreita hyödyntäväksi tietorakenteeksi.

Kun ohjelma on suorittanut todennäköisyyslaskelmat, se palauttaa datan jälleen JSON muotoon käännettynä käyttöliittymälle, joka näyttää tulokset käyttäjälle.

Toteutettavat algoritmit ja tietorakenteet

Ohjelma sisältää vielä ennalta nimeämättömän algoritmin nopanheittojen todennäköisyyksien laskemiseen. Yritin löytää valmista tarkoitukseen soveltuvaa algoritmia, mutta en ainakaan vielä löytänyt sellaista, joten lähden kehittämään algoritmia binomitodennäköisyyslaskuihin perustuen.

Algoritmin saamassa syötteessä alukset on valmiiksi järjestetty hyökkäysjärjestykseen. Näin ollen algoritmin tehtävä on toteuttaa hyökkäystoiminto vuorotellen jokaiselle alukselle. Aluksen tulee tunnistaa kohteensa ja kirjata todennäköisyys osumaan jokaiselle aluksen kilvelle. (Jokainen aluksen tykeistä vastaa yhtä d6 noppaa.) Näistä pystytään myös

määrittelemään todennäköisyys jolla alus ei saa lainkaan osumaa tai ammutaan alas.

Algoritmi huomioi myös hyökkääjän mahdollisen tietokoneen jolla nopan osumatarkkuutta voi parantaa tai kohteen suojat, jolla tietokoneen vaikutuksia voi pienentää.

Alukset säilytetään vektorissa ja jokainen alus koostuu mapista, joka sisältää aluksen ominaisuudet: aseet, kilvet, suojat ja tietokoneet, sekä osumatodennäköisyydet jokaiselle kilvelle.

Toteutettavia tietorakenteita voisi olla jonkinlainen kekorakenne, jossa vihollisaluksia voisi säilyttää niin että paras kohde olisi aina helposti saatavilla. Alusten tiedot tallentavan mapin korvaus ei kuulosta Clojurella järkevältä koodin luettavuuden kannalta, mutta lienee myös mahdollista toteuttaa.

Aika- ja tilavaativuudet

Osumatodennäköisyyksiä lasketaan alusten aseiden määrä, sekä mahdollisuus 0 osumalle. Jokainen alus hyökkää jokaisella tykillään, eli aikavaativuuteen vaikuttaa ainakin luvut n = alusten määrä ja k = yhteenlaskettu tykkien määrä.

Koska alukset on tallennettu mappiin tai vastaavaan rakenteeseen, on aluksen tietojen (kuten kilpien määrä) saaminen O(1) operaatio.

Jokaisen aluksen on ennen hyökkäystä tunnistettava hyökkäyksen kohteet. Koska kohteet on sijoitettu kekoon, on myös kohteen haku O(1) operaatio.

Näin ollen yhden kierroksen aikavaativuus on O(n+k).