Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Tietorakenteiden harjoitustyö

Ohjaaja: Pekka Mikkola
Harjoitustyön tekijä: Aku Peltonen

Toteutusdokumentti

Johdanto

Aiheena tietorakenteiden harjoitustyöksi valitsin yksinkertaisen tekoälyn ohjelmoimisen ristinolla-peliin. Työn toteutin C-kielellä, jonka opettelin projektin ohessa.
Ohjelmassa käytettiin negamax-algoritmia, joka on yksinkertaisempi muunnos yleisesti pelien tekoälyissä käytössä olevasta minimax-algoritmista.

Ohjelmakoodi

Ohjelman koodi on hajautettu kolmeen tiedostoon: main.c, helpers.c ja negamax.c. Näille on luonnollisesti myös header-tiedostot. Lisäksi ohjelmakoodin kääntämisen helpottamiseksi on kirjoitettu makefile-tiedosto.

Main.c-tiedostossa on toteutettu ohjelman komentorivipohjainen käyttöliittymä, jonka avulla käyttäjät voivat pelata ristinollaa tietokonetta vastaan. Käyttöliittymässä on aluksi yksinkertainen valikko, josta voi valita ruudukon koon, jolloin näytölle tulostetaan peliruudukko, josta valitaan numeronäppäimillä seuraava peliruutu. Tämän jälkeen tietokonevastustaja suorittaa oman siirtonsa ja peli jatkuu niin pitkään kunnes toinen voittaa, tai peli päättyy tasapeliin.

Helpers.c-tiedostossa toteutettiin ohjelmassa käytettävät apufunktiot peliruudukon tulostamiseen ja pelitilanteen tarkistamiseen.

Negamax.c-tiedostossa on toteutettu ristinollan tietokonepelaajan käyttämä tekoäly. Käytetyn negamax-algoritmin toiminta perustuu siihen, että peliä simuloidaan siirto kerrallaan funktiota rekursiivisesti kutsuen ja muodostuneesta pelipuusta valitaan sellainen haara, joka johtaa pelin minimituloksen maksimointiin. Pelipuun lehtisolmuille annetaan arvot sen mukaan, mikä on lopputulos pelaajan kannalta. Solmu saa arvon -1 mikäli vastustaja on voittanut, 0 jos peli päättyi tasan ja 1 jos voittaja on pelaaja itse. Negamax eroaa minimaxista siten, että kun minimaxissa kutsutaan vuorotellen min- ja max-funktioita, hyödynnetään negamaxissa sitä tietoa, että kun vastustaja maksimoi oman pelituloksensa, se samalla minimoi pelaajan tuloksen, sillä ristinolla on nollasummapeli. Negamaxissa siis algoritmi kutsuu itseään ja palauttaa tuloksen negatiivisena jos kyseessä on vastustajan pelivuoro.

Pseudokoodi algoritmille on seuraavanlainen:

```
int NegaMax(Board b, int depth, int color) {
    if (GameOver(b) or depth>MaxDepth)
        return sign[color]*Analysis(b)
    int max = -infinity
    for each legal move m in board b {
        copy b to c
        make move m in board c
        int x= - NegaMax(c, depth+1, 1-color)
        if (x>max) max = x
    }
    return max
}
```

Omassa toteutuksessani funktio negamax() palauttaa tietueen move, jossa on kentät int slot ja int move. Edellä mainittu sisältää tiedon parhaan siirron paikasta pelitaulukossa ja jälkimmäinen taas kyseisen siirron pelipuun haaran minimituloksen.

Testaus

Testausta varten kirjoitin erilliseen tiedostoon test.c muutaman testifunktion ohjelman apufunktioiden toiminnan ja funktion negamax() oikeellisuuden testaamiseksi. Testauksessa ei käytetty apuna mitään varsinaista testauskirjastoa tai -frameworkia, vaan testit laadittiin käyttäen hyväksi sitä ominaisuutta, että testattavat apufunktiot palauttavat kokonaislukuarvon jolloin on helppoa todeta jos arvo on väärä.
Negamax-algoritmin oikean toiminnan testaamiseksi kirjoitin funktion test_negamax() joka pelaa 50 peliä ristinollaa kahden tietokonevastustajan välillä ja raportoi pelien tulokset. Mikäli algoritmi toimii oikein, tuloksena pitäisi olla 50 tasapeliä, sillä täydellisen strategian ristinollassa ei pääse parempaan tulokseen, ellei vastustaja tee virhettä.

Käyttöohje

Ohjelma käännetään komentorivillä ajamalla ohjelman juurihakemistossa make-ohjelma, jonka tulostiedostona saadaan OXO-niminen binääritiedosto sekä test-niminen testiohjelma. Mikäli halutaan tulostiedostona vain toinen, ajetaan komento make [OXO|test].