Toteutusdokumentti

Ohjelman yleisrakenne

Ohjelma käynnistää pelin Mylly -luokan main -metodissa. Peli aloitetaan luomalla käyttöliittymä ja Peli -olio, johon luotu käyttöliittymä annetaan konstruktorin parametrina. Peli -olion konstruktorille annetaan parametreina myös Pelaaja -rajapinnan toteuttavat oliot, jotka edustavat kutakin pelaajaa. Peli käynnistyy, kun kutsutaan Peli -olion pelaa -metodia.
Pelimerkit ja niiden sijainnit on tallennettu erilliseen Lauta -luokan ilmentymään, joka tarjoaa erilaisia metodeja pelimerkkien siirtelyyn ja tarkasteluun. Siirrot laudalle tehdään kutsumalla Pelaaja -rajapinnan toteuttaville olioille metodeja jotka puolestaan kutsuvat Lauta -luokan metodeja. Nämä merkitsevät laudalla olevaan sijaintiin pelimerkkiä edustavan kokonaisluvun. Luku 1 edustaa mustaa ja luku 2 valkoista. Luku 0 viittaa tyhjään sijaintiin. Laudan sijainnit on esitetty 3 rivisenä kokonaisluku matriisina, jossa on kahdeksan paikkaa per rivi. Ensimmäinen rivi edustaa myllylaudan ulointa neliötä ja viimeinen, eli rivi 2 laudan sisintä neliötä.
Pelaaja rajapinta määrittelee kolme metodia, siirraLaudalle, siirraLaudalla ja lenna, pelin eri vaiheita varten, sekä poistaLaudalta metodin vastustajan nappien poistamiseen myllytilanteita varten. AIPelaaja -luokka tarjoaa toteutukset näille metodeille. AIPelaaja –luokan olioissa metodit kutsuvat ensin Tekoaly –luokan metodeja, jotka laskevat min max –algoritmia hyväksikäyttäen optimaalisimman sijainnin pelaajan siirrolle. Sen jälkeen laskettu siirto tehdään kutsumalla Lauta –luokan metodeja. Pelaaja –rajapinnan määrittelemät metodit palauttavat paluuarvoinaan pelimerkkien uudet sijainnit Peli –luokkaan, josta ne päivitetään Kayttoliittyma –rajapinnan toteuttavan TKayttoliittyma -luokan (tekstikäyttöliittymän) hallinnoimaan näkymään.
Tekstikäyttöliittymässä pelilauta on esitetty merkkijonona, johon tehdään muutoksia sitä mukaa, kun siirtoja laudalle tehdään. Kayttoliittyma tarjoaa metodin myös pelilautaesityksen tulostamiseen ja se tulostetaan käyttäjän näkyville jokaisen tehdyn muutoksen yhteydessä.
Peli –luokassa peli etenee vaiheittain. Ensin siirretään 9 nappia laudalle (per pelaaja). Tämän jälkeen laudalla olevia nappuloita siirrellään niin kauan, kunnes toisella pelaajalla on enää kaksi nappulaa laudalla taikka toinen pelaajista ei voi enää liikuttaa nappejaan. Kun tähän lopputilanteeseen on päästy, selvitetään pelin voittaja ja ilmoitetaan voittajan väri käyttöliittymän kautta käyttäjälle.
Pelissä pelilaudan tyhjiä sijainteja, mustien merkkien sijainteja ja valkoisten merkkien sijainteja säilytetään kolmessa eri puussa ja sijainteja kokeillessa min max -algoritmi käy jotain näistä puista läpi. Kun pelaajat tekevät laudalla siirtoja, siirrellään sijainteja avaiminaan sisältäviä solmuja puusta toiseen. Toinen toteuttamani tietorakenne pelissä on yksisuuntainen linkitetty lista, jota käytetään mm. parhaiden sijaintien listaamiseen.

Aika ja tilavaativuudet

Sekä aika- ja tilavaativuus siirtojen laskemiselle pelkällä min max –algoritmilla on eksponentiaalinen (O(n^n)). Tämä johtuu itse min max –algoritmista ja siitä, että algoritmissa läpikäytävät pelilaudan sijainnit on talletettuna puu –rakenteeseen, josta on luotava uusi kopio jokaisella pelipuun generoinnin rekursiotasolla, jotta ylemmän tason puu ei menisi rikki alemman tason sijainteja laskettaessa.
Suorituskykytestauksessa aikavaativuus osoittautui oletettua vastaavaksi, mutta alpha-beta –karsinnalla päästiin tulokseen, jossa aikaa laskentaan kului alle puolet siitä mitä pelkällä min max –algoritmilla kuluu, kun syvyys oli yli kolme. Kahden ja kolmenkin syvyydellä karsinnasta oli etua. Pelissä alpha-beta –karsintaa aletaan käyttää oikeastaan vasta toisella rekursiotasolla, jotta tekoälyn tekemiin siirtoihin tulisi vaihtelevuutta, mutta testauksessa käytin sitä myös ensimmäisellä rekursiotasolla. Tästä saatiin lisäetua vain parillisilla syvyyksillä, jolloin kokeilukertojen määrä ensimmäisellä rekursiotasolla puolittui.

Puutteet ja parannusehdotukset

Heuristiikassa on vielä parantamisen varaa, koska tekoäly ei esimerkiksi aina osaa blokata vastustajan myllyä ajoissa. Pelissä ei ole myöskään ratkaistu mahdollista tasapeli-tilannetta, jossa pelaajat jäävät vain siirtelemään merkkejä edes takaisin laudalla. Pelaaja –rajapinnalle voisi kirjoittaa vielä toisen toteuttavan luokan, jotta myös käyttäjä pääsisi pelaamaan peliä. Pelissä käytetyt binäärihakupuut eivät ole tasapainotettuja. Tasapainotuksella joitakin operaatioita olisi voinut ehkä vielä nopeuttaa. Toisaalta puun koko ei kasva koskaan suuremmaksi kuin 24 ja peli käyttää siirtojen valinnassa jonkin verran satunnaisuutta, joten puiden pitäisi olla jossain määrin tasapainoisia.