Määrittelydokumentti, Kalaha

Ohjelma pelaa käyttäjää vastaan kahden pelaajan strategiapeliä, Kalahaa. Pelilaudassa on molempien pelaajien puolella kuusi pienempää kuppia, sekä laudan päädyissä isommat kupit, mancalat. Peliä pelataan vastapäivään ja pelaajan oikealla puolella on oma mancala. Pelin alussa kaikissa pienemmissä kupeissa on neljä kiveä. Aloittaja arvotaan. Pelin aloittaja nostaa yhdestä kupista kaikki siinä olevat kivet ja pudottaa jokaiseen seuraavaan kuppiin yhden kiven. Omaan mancalaan pudotetaan kivi matkalla, mutta vastustajan hypätään yli. Jos pelaaja asettaa viimeisen kivensä omaan mancalaan, saa hän uuden vuoron. Jos taas viimeinen kivi osuu oman puolen tyhjään kuppiin, saa pelaaja sekä kyseisen kiven että vastapäisessä, vastustajan kupissa olevat kivet mancalaansa. Pelissä on tarkoitus kerätä omaan mancalaan mahdollisimman paljon kiviä. Peli päättyy, kun jomman kumman pelaajan kupit ovat tyhjiä, jolloin se pelaaja, jonka puolella on vielä kiviä, saa ne mancalaansa.

Ohjelma saa syötteenä pelaajalta numeron 1-6, joka kertoo, mistä kupista pelaaja tekee siirtonsa. Tulosteena pelilauta muuttuu tehdyn siirron mukaiseksi.
Esim. alkutilanne:

```
_____________
|0|4|4|4|4|4|4|
  |4|4|4|4|4|4|0|
```

Tietorakenteena käytetään puuta, jonka juuri on kyseisen hetken pelitilanne. Juuren lapset ovat pelitilanteet yhden siirron jälkeen, näitä lapsia on joka solmulla kuusi. Juuren lapsen lapset taas ovat pelitilanteet kahden siirron jälkeen jne. Puu on siis rakenteeltaan kuusihaarainen. Puun lehdet ovat tilanteita, joissa peli on päättynyt jomman kumman pelaajan voittoon tai tasapeliin.

Lehdille määritellään arvot arviointifunktion avulla lopputilanteen pisteiden perusteella. Esim. lopputilanne, jossa oman puolen kupit ovat tyhjiä, mutta vastustajalla on kupeissa ja omassa mancalassaan yhteensä enemmän kiviä kuin pelaajalla omassa mancalassa, on lopputilanteen arvo negatiivinen erotuksen verran.

Pelitilanne talletetaan 14-paikkaiseen listaan, jonka 1.-6. paikat ovat toisen pelaajan kupit ja 7. hänen mancala, 8.-13. toisen pelaajan kupit, sekä 14. hänen mancala.

Tietokoneen parhaan mahdollisen liikkeen löytämiseksi käytetään MinMax -algoritmia. Laskennan keventämiseksi, mahdollisesti huomattavastikin, käytetään alfa-beta karsimista. Tämä karsii laskennan edetessä ne solmut pois, joissa ei

kannata vierailla. Alfa-beta karsinta toimii erityisesti puissa, joiden alkiot ovat satunnaisessa järjestyksessä, kuten tässä kyseisessä pelipuussa.

Jos oletamme, että pelin jokainen siirto vie mantalaan keskimäärin yhden kiven, olisi pelissä vuoroja kivien lkm verran-1 (koska viimeinen kivi siirretään automaattisesti toisen mantalaan) eli 47.Tässä pelipuussa, jos oletetaan se yksinkertaisuuden vuoksi täydelliseksi puuksi (mitä se ei kyllä ole, koska osa peliliikkeistä johtaa nopeammin pelin päättymiseen) olisi 6^47 lehteä, sen korkeus olisi 47 ja solmuja 1+6^1+6^2+...6^47 (eli luokkaa 4,5*10^36).Aikavaativuus min-max algoritmilla on O(haarautumisaste^syvyys). Alfa-beta karsinnalla pyritään pienentämään ohjelman suoritusaikaa karsimalla turhat haarat pois.

Tilavaativuus on O(haarautumisaste*syvyys). Aina kun lasketaan solmulle arvo, tulee solmusta haarautua lapsisolmuun ja siitä sen lapsisolmuun jne kunnes ollaan lehdessä, josta voidaan sitten palauttaa arvo ylöspäin. Kun polun lehdet on tutkittu, ei ohjelma enää säilytä niiden tietoja.

Ohjelma kirjoitetaan Javalla. Käyttöliittymä rakennetaan **?ehkä?** Javan JFrame luokalla. Ikkunassa on näkyvillä kunkin hetken pelitilanne pelilaudalla, viimeisimmät siirrot, kenttä ja painike joilla pelaaja välittää ohjelmalle valintansa, sekä tekstikenttä, mikä kertoo jotain pelin tilanteesta.