Aineopintojen harjoitustyö: TiRa
Testausdokumentti

Juha Inkinen
Ohjaaja: Kristiina Paloheimo

07.12.2012

Sisällysluettelo

Yleistä

Shakki-pelin luonteen vuoksi yksikkötestaaminen olisi muodostunut ongelmalliseksi varsinaisen pelilohiikan osalta. Siksi päädyin testaamaan projektia pääosin lukuisilla testitulostuksilla ja debug-ominaisuuden avulla.

Luokat

1. Lauta

Lauta on varsin pitkä luokka, yli 200 riviä ja sen tehtävä on ylläpitää tietovarastoja pelin tilanteesta ja heurestisista ominaisuuksista.

Sen testaus alkoi yksinkertaisesti tarkastamalla, että laudan tulostusmuoto vastasi asetettua muotoa. Lisäksi ominaisuuksien toiminta on varmistunut lukuisin testiajojen aikana, koska tämä luokka toteutettiinn ensimmäisten joukossa.

Tulostuksilla [laudanTulostus()] testattin myös lähes kaikki muut luokan metodit vertailemalla tilannetta laudalla ja metodien tekemiä muutoksia ja päätelmiä.

Joissain luokissa käytettiin myös lukuisia aputulosteita, jotka olen siivonnut pois.

2. Lista

Lista on ArrayListaa muistuttava tietorakenne, joka säilöö myös omien maksimi- ja minimiarvojensa sijainnit saavuttaen O(1)-haut näille.

Tämän luokan testaus on suoritettu joillain testitulostuksilla, sekä Junit-testeillä.

3. Puu

Puu toimii puurakenteen juuren säilöjänä, joten sen avulla puuhun pääsee näppärästi käsiksi. Lisäksi Puu sisältää AlphaBeta-kruunausta käyttävän minmax-algoritmin.

AlphaBeta sekä minmax on testattu osittain yksikkötestauksella, mutta suuren osan testasin testiajamisella sekä tulostuksilla. Debug-ominaisuus oli myös hyödyllinen, koska se auttoi näkemään että algoritmi valitsee tuloksensa oikeiden arvojen mukaan.

4. Ruudut

Ruudut on taulukkorakenne joka säilöö ruutu-olioita ja osaa alustaa ne uutta peliä varten.

Tämän luokan testaus toteutui helposti testitulostuksilla, koska niistä näki heti alustuivatko nappulat oikein laudalle. Lisäksi getteri oli nopeasti testattu varmistamalla muiden algoritmien yhteydessä, että se antaa oikean tuloksen.

5. Ruutu

Ruutu tietää oman sijaintinsa sekä nappulan joka laudalla on. Lisäksi se säilöö ruudussa olevan nappulan siirrot.

Ruudun ominaisuudet luottavat siihen, että Lauta-syöttää ruudulle oikeita arvoja ja Ruutu toimiikin lähinnä tietovarastona, siksi virheet johtuisivat muista luokista.

6. Shakki

Shakki on main-luokka, jota käytetään tekoälyjen peluuttamiseen. Tämä luokka toimiikin siis testauksen apuvälineenä.

7. Siirto

Siirto-luokan oliot kuvastavat yksittäistä siirtoa. Siirrolla on lähtöpiste, loppupiste ja arvo.

Siirto on tietovarasto ja sen metodit ovat getter- ja setter-luontoisia, joten luokan testaaminen on triviaalia.

8. Siirtolaskuri

Siirtolaskuri laskee laudan nappuloille lailliset siirrot opetettujen ominaisuuksien mukaan. Tällä hetkellä esim kuninkaan shakki-tarkistukset on toteuttamatta.

Tämänkin luokan testauksen pääväline oli tulostukset. Testitulostuksilla havaittiin muun muassa, että tornit ja lähetit saivat aina nolla siirtoa, vaikka siirtoja olisi käytettävissä useampia. Testailun jälkeen kävi ilmi, että syynä oli siirtojen pituuksia kasvattavien for-looppien aloitus nollasta, jolloin nappulan ”törmäsi” itseensä ennen kuin päästiin tutkimaan kauempana olevia ruutuja. Ongelma korjattiin kasvattamalla aloitussiirron pituus yhteen.

Ongelmana oli alkuun myös laudan ulkopuolelle joutuminen. Tästä aiheutui taulukon indeksin virheitä. Korjauksena toimi loopin katkaiseva tarkistus, jolla varmistetaan, ettei uusia siirtoja enää lasketa, jos siirto päätyisi ulos laudalta.

9. Solmu

Solmut ovat puuhun tallennettavia olioita, joilla on tieto lapsistaan, vanhemmasta sekä omasta siirrosta.

Solmun testaus on toteutettu testitulostuksilla puurakenteen testailun yhteydessä. Jos puu toimii, on myös solmujen oltava kunnossa.

10. Tekoäly

Tekoäly-luokka vastaa tekoälyn ohjaamisesta. Sen valitseSiirto-metodi hyödyntää puurakenteen alphabeta-karsintaa, sekä tietenkin pelilaudalla vallitsevaa tilannetta ja mahdollisia siirtoja.

Tämän luokan testaus on toteutettu seuraamalla erilaisia pelitilanteita main-luokassa testitulostusten perusteella. Lisäksi tekoäly antaa monelle algoritmille syötteeksi apuarvoja, jotka turvaominaisuuksena rikkovat pelin, jos ne päätyvät valituksi. Tämän avulla voidaan olla melko varmoja siitä, ettei näitä arvoja voi tulla valituksi lopputilanteeseen, koska peli ei mene rikki.