Shakkipeli

Testausdokumentti

Tuukka Paukkunen
tuukka.paukkunen@cs.helsinki.fi

Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit
27.8.2015
Helsingin yliopisto, tietojenkäsittelytieteen laitos

Sisällysluettelo

1 Testauksen laajuus

Koska erilaisia shakkilaudan pelitilanteita arvioidaan olevan 10^120 kappaletta, ja näiden välillä erilaisia siirtovaihtoehtoja n. 10^3 kpl [Wik15]. Tämän perusteella lienee selvää, että kattavan testitapauslistan muodostaminen on mahdotonta, joten tutkivalla testauksella sovelluksen toimiminnan varmistamisessa on olennainen osa.

Sovelluksen testauksessa on käytetty laajasti toki myös jUnitia. Testaus jUnitilla on keskittynyt etenkin siihen, että pelinappuloita voi siirtää shakkilaudalla ainoastaan shakin sääntöjen mukaisesti.

2 Testausaineisto

Testausaineisto koostuu 172 erillisestä yksikkötestistä, jotka testaavat perusasiat mm. nappuloiden liikuttelusta, pelilaudan toiminnasta ja käyttäjän siirtokäskyjen vastaanotosta. Lista yksikkötesteistä löytyy jUnit-testien javadoceista: https://github.com/tuukkapa/Shakki-TIRALabra2015/blob/master/test/javadoc/index.html

Tärkein rooli kuitenkin on tutkivalla testauksella, joka yksinkertaisesti tarkoittaa sitä, että peliä pelataan mahdollisimman monella erilaisella siirrolla ja samalla kaikkia nappuloita pyritään liikuttamaan kaikilla mahdollisilla tavoilla. Samalla kirjataan ylös, miten tietokone vastaa käyttäjä siirtoihin. Mikäli jotain sääntöjen vastaista tai poikkeavaa tapahtuu, tällöin otetaan ylös, minkälaisessa pelitilanteessa poikkeus tapahtuu, ja pyritään toistamaan tilanne esimerkiksi aloittaen peli suoraan poikkeuksen aiheuttavasta tilanteesta ja seuraamalla sovelluksen toimintaa rivi riviltä.

Kuten kappaleessa 1 mainittiin, kattavan listan luominen käsin tehtävistä testitapauksista on mahdoton tehtävä.

3 Miten sovellusta testataan

Projektihakemiston test-hakemistossa on luokat jUnit-testausta varten sovellusta vastaavissa pakkauksissa. Yksikkötestit (jUnit) kattavat sovelluksen perustoiminnallisuuden, kuten nappuloiden sääntöjen mukaisen liikuttelun ja käyttäjän syöttämien komentojen tarkistamisen.

Käyttöliittymälle ei ole tehty jUnit-testejä, vaan käyttöliittymän oikeellisuus todetaan visuaalisesti vertaamalla ohjelman tulostusta määrittelydokumenttiin.

Tämän lisäksi sovellusta on testattu pelaamalla peliä tietokonetta vastaan, siirtämällä nappuloita laudalla sekä seuraamalla tarvittaessa sovelluksen toimintaa debug-moodissa etenemällä koodia rivi riviltä.

4 Testaustulokset

Kaikki 172 testitapausta menevät läpi. Nämä varmistavat sovelluksen perustoiminnan. Tämän lisäksi olen pelannut onnistuneesti monta peliä läpi ilman virhetilanteita.

5 Suorituskykytestaus

Oheisessa taulukossa näkyy sovelluksen suorituskyky erilaisilla pelipuun syvyyksillä ilman alpha-beta -karsintaa sekä alpha-beta -karsinnan kera. Tuloksista selviää, että alpha-beta -karsinnalla on suorituskykyyn olennainen vaikutus.

5.1 Suorituskyky ilman alpha-beta -karsintaa

Pelipuun syvyys	**Tietokoneen siirtoon kulunut keskimääräinen aika**
1	
2	
3	
4	
5	
6	
7	
8	

5.2 Suorituskyky alpha-beta -karsinnan kera

Pelipuun syvyys	**Tietokoneen siirtoon kulunut keskimääräinen aika**
1	
2	
3	
4	
5	
6	
7	
8	

6 Lähteet

Wik15 Shannon number, Wikipedian artikkeli, https://en.wikipedia.org/wiki/Shannon_number [27.8.2015]