# Testausdokumentaatio

## Toiminnallisuus

Ohjelmaa on testattu kahdella tavalla: Automaattisin JUnit-testein, sekä manuaalisesti ajamalla ohjelma ja toivomalla parasta. Esimerkiksi käyttöliittymän toimintaa testattiin vain ajamalla se, ja katsomalla näyttääkö se oikealta. Junit-testit on helppo toistaa yksinkertaisesti ajamalla ne.

Tilanteissa, joissa oli vaikea määritellä, mitä automaattisia testejä kannattaa tehdä, ajettiin ohjelma ensin, ja sitten suunniteltiin automaattiset testit löydettyjen ongelmien diagnosointia varten. Tällä tavalla löydettiin ja korjattiin mm. heuristiikan laskennassa löytynyt huolimattomuusvirhe, jota kolme ihmistä ei huomannut suoraan koodista.

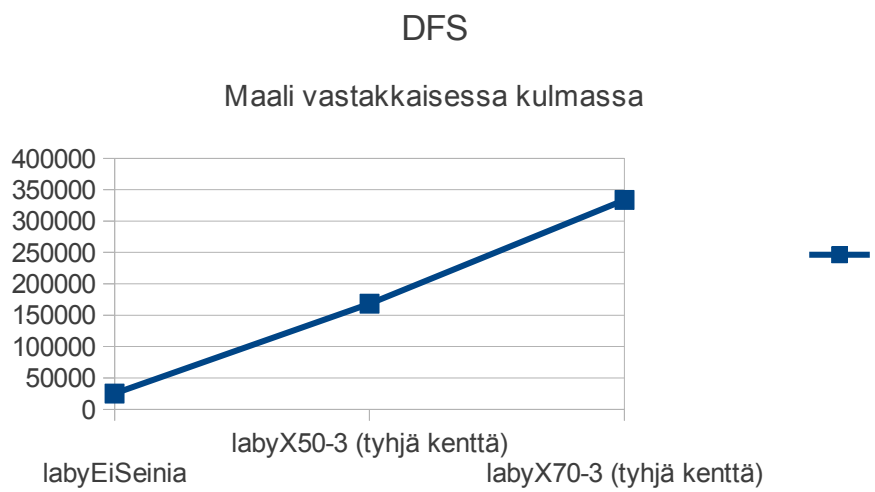
Ongelmia diagnosoitiin myös manuaalisesti lisäämällä ohjelmakoodiin aputulosteita, joista näkyi mm. jäätiinkö ikuiseen silmukkaan tai päästiinkö silmukkaan ylipäätään. Tämä selittää mm. miksi hakualgotitmin polunmerkinnässä on >= merkintä pseudokoodin vaatiman > merkinnän sijaan. Jälkimmäisessä tapauksessa kyseisen if-lauseen sisään ei oltaisi koskaan menty, ellei algoritmi olisi löytänyt sokkelosta silmukkaa.

Hakualgoritmin toiminnan testausta varten kehitettiin ohjelmaan myös ominaisuus, joka visualisoi algoritmin toimintaa samalla kun polunetsintä on käynnissä. Vaikka ominaisuus kehitettiinkin alun perin algoritmin toiminnan manuaaliseen tarkasteluun, se jätetään valmiiseen ohjelmaan vain siitä syystä, että se näyttää hyvältä ja tekee valmiista ohjelmasta astetta mielenkiintoisemman.

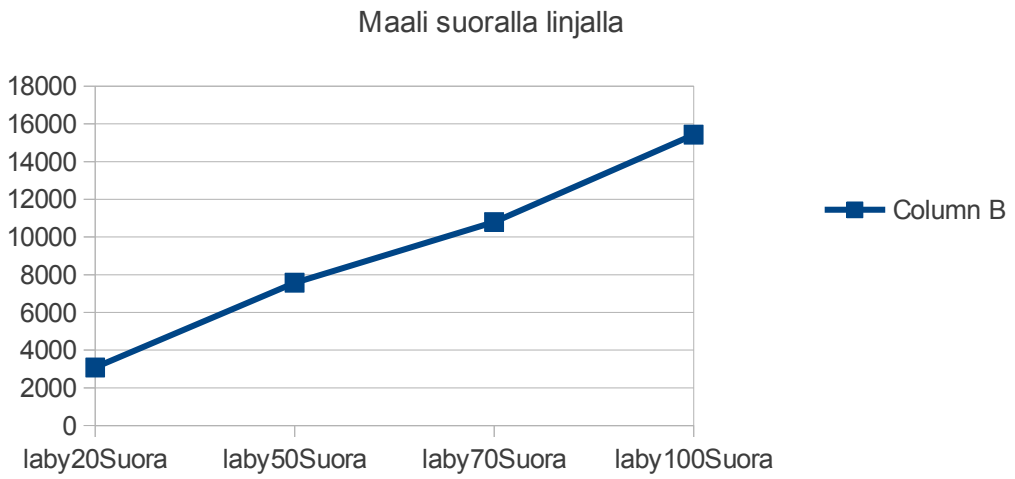
## Suorituskyky

Suorituskykytestauksessa otettiin aikaa, ja katsottiin kuinka kaunan ohjelman ajaminen kestää erilaisilla syötteillä. Ajanotto automatisoitiin ohjelmakoodiin, ja se on toistettavissa poistamalla kommentoinnit niiden edestä ja ajamalla ohjelma. Ajanotto on ohjelmoitu siten, ettei se ota huomioon piirtämiseen kuluvaa aikaa, eikä jokaisen käsiteltävän solmun kohdalla ajettavaa 70 millisekunnin viivettä, joka mahdollistaa polun piirtämisen silmälle miellyttävän animoinnin.

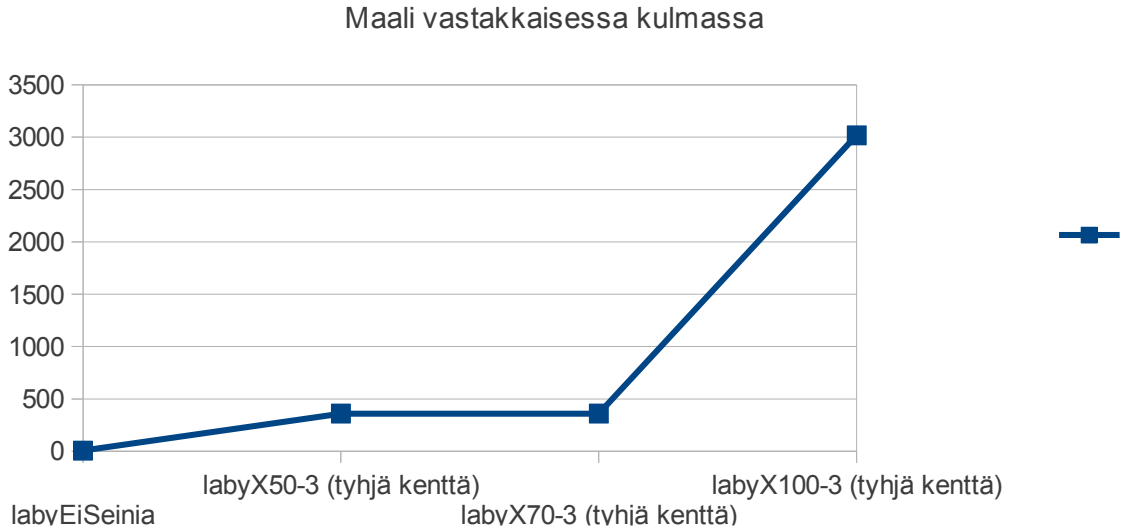
Testitulokset kaavioina:
(Kaaviot otettu vertailukelpoisista labyrinteista. Vastakkaisissa kulmissa sijaitsevien kohdepisteiden tapauksessa vertailtiin tyhjällä (esteettömällä) kentällä tapahtuvaa etsintää, ja kohdepisteiden sijaitessa samalla vaaka-akselilla labyrintti oli muuten tyhjä, mutta pisteiden välissä sijaitsi viiden pikselin kokoinen este, joka on kaikissa labyrinteissa saman muotoinen, ja sijaitsee samassa kohtaa pystyakselia.)



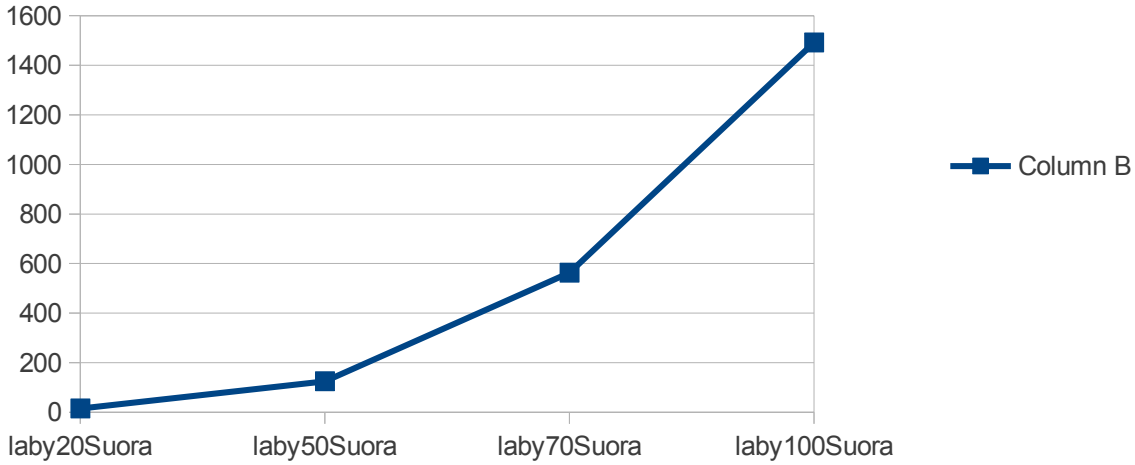
DFS


Maali suoralla linjalla
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
laby20Suora
laby50Suora
laby70Suora
laby100Suora
Column B

A*


Maali vastakkaisessa kulmassa
3500
3000
2500
2000
1500
1000
500
0
labyEiSeinia
labyX50-3 (tyhjä kenttä)
labyX70-3 (tyhjä kenttä)
labyX100-3 (tyhjä kenttä)

A*


Maali suoralla linjalla
1600
1400
1200
1000
800
600
400
200
0
laby20Suora
laby50Suora
laby70Suora
laby100Suora
Column B

Kaikki tulokset :

## Suorituskykytestaus A*

| | Algotitmi (ms) | Polun muodostus (ms) |
|---|---|---|
| **Kun maali on sijoitettu diagonaalille, vastakkaiseen kulmaan** | | |
| **20*20 pikseliä** | | |
| laby20x20 | 7 | 0 |
| laby2 | 1 | 0 |
| laby3 | 13 | 0 |
| labyEiSeinia | 6 | 0 |
| labyMetsa | 13 | 0 |
| **50*50 pikseliä** | | |
| labyX50 | 315 | 0 |
| labyX50-2 | 50 | 0 |
| labyX50-3 (tyhjä kenttä) | 360 | 0 |
| labyX50-4 (satunnaiset kaaret) | 942 | 0 |
| labyX50-5 (metsä) | 1420 | 0 |
| **70*70 pikseliä** | | |
| labyx70 | 20 | 0 |
| labyX70-2 | 45 | 0 |
| labyX70-3 (tyhjä kenttä) | 360 | 0 |
| labyX70-4 (satunnaiset kaaret) | 9791 | 0 |
| labyX70-5 (metsä) | 358 | 0 |
| **100*100 pikseliä** | | |
| labyX100 | 761 | 1 |
| labyX100-2 | 623 | 0 |
| labyX100-3 (tyhjä kenttä) | 3018 | 0 |
| labyX100-4 (satunnaiset kaaret) | 3192 | 1 |
| labyX100-5 (metsä) | 3151 | 0 |
| | | |
| **Kun maali on suoralla linjalla vastakkaisella sivulla, ja välissä on vain yksi, viiden pikselin kokoinen este** | | |
| laby20Suora | 15 | 0 |
| laby50Suora | 125 | 0 |
| laby70Suora | 563 | 0 |
| laby100Suora | 1492 | 0 |

Polun muodostukseen kuluva aika ei muuttunut tarpeeksi, jotta siitä voitaisiin tehdä kaavio. Se on kuitenkin koodin perusteella lineaarinen solmujen määrän suhteen. Polun muodostamisen pseudokoodiesitys:

```
Lista<Solmu> muodostaPolku()
    Solmu s = maali;
    lisää solmu s polkuun;

    while (on olemassa s.getPolku()) //s.getPolku on solmuun tallennettu tieto seuraavasta lyhimpään polkuun kuuluvasta solmusta
        lisätään kekoon s.getPolku();
        s = s.getPolku;

    while (keko ei ole tyhjä)
        lisää solmu s polkuun
```

# Suorituskykytestaus DFS

| | Algotitmi (ms) |
|---|---|
| **Kun maali on sijoitettu diagonaalille, vastakkaiseen kulmaan** | |
| **20*20 pikseliä** | |
| laby20x20 | 11994 |
| laby2 | 2665 |
| laby3 | 16267 |
| labyEiSeinia | 25103 |
| labyMetsa | 22298 |
| **50*50 pikseliä** | |
| labyX50 | 49939 |
| labyX50-2 | 44530 |
| labyX50-3 (tyhjä kenttä) | 168128 |
| labyX50-4 (satunnaiset kaaret) | 112506 |
| labyX50-5 (metsä) | 165982 |
| **70*70 pikseliä** | |
| labyx70 | 40607 |
| labyX70-2 | 258026 |
| labyX70-3 (tyhjä kenttä) | 333565 |
| labyX70-4 (satunnaiset kaaret) | 318916 |
| labyX70-5 (metsä) | 315127 |
| **100*100 pikseliä** | |
| labyX100 | 77431 |
| labyX100-2 | 398140 |
| labyX100-3 (tyhjä kenttä) | STACK OVERFLOW |
| labyX100-4 (satunnaiset kaaret) | STACK OVERFLOW |
| labyX100-5 (metsä) | STACK OVERFLOW |
| | |
| **Kun maali on suoralla linjalla vastakkaisella sivulla, ja välissä on vain yksi, viiden pikselin kokoinen este** | |
| laby20Suora | 3085 |
| laby50Suora | 7574 |
| laby70Suora | 10800 |
| laby100Suora | 15428 |