Testausdokumentti.

Automaattista testausta on toteutettu Javan JUnit työkalulla, joten ne ovat helposti toistettavissa.

Käyttöliittymätesteissä testataan pääasiassa sitä, että käyttöliittymän Noden toimivat oikein ja ne maalataan oikeilla väreillä oikeassa kohdassa. Käyttöliittymätesteissä on myös testejä, joissa hyödynnytetään Javan Robot oliota simuloimaan käyttäjän syötteitä graafisessa käyttöliittymässä. Tässä testissä kokeillaan siirtää robotin kursori tiettyyn pisteeseen ruutua ja piirretään seinää, jonka jälkeen tarkistetaan AssertTrue metodilla, että Noden väri muuttui oletetuksi. Muita keskeisiä testejä käyttöliittymälle ovat kaikkien getterien testit, joilla varmistetaan, että metodit antavat oikeat paluuarvot, ja että nodet ovat oikein synkronoituna itse käyttöliittymän taulukkoon.

Astarin automatisoiduissa testeissä annetaan algoritmille erilaisia tilanteita. Osassa joissa tilanne on melko normaali, ja toisissa esimerkiksi on alkupiste loppupisteen päällä. Tietyissä Astar testeissä on myös timeout, jonka sisällä algoritmin on pakko suorittaa tehtävä. Tämä pääasiassa sitä varten, että algoritmi ei ajautuisi ikuiseen silmukkaan, joihin se testauksen aikana aina välillä pääsi erikoisemmissa testeissä.

JumpPointSearch:in automatisoiduissa testeissä testataan pääasiassa JPS:n eri metodien toimintaa yksittäin metodi kerrallaan. Metodit testataan pääasiassa ohjelman oletussyötteillä, joiden kuitenkin tulisi päteä yleisellä tasolla.

Tietorakenteita on testattu pääasiassa yksinkertaisilla getteri ja setteri testeillä, joilla varmistetaan, että näiden arvot saadaan asetettua ja haettua oikein. Testeissä, kuten aiemmissakin hyödynnetään monien metodien boolean palautusarvoa, jonka avulla tiedetään, että metodi on suorittanut loppuun asti. Gettereitä ja settereitä yleensä testataan yhdessä, jolloin asetetaan arvo setterillä ja haetaan se getterillä. Tämän jälkeen verrataan getterin palauttamaa arvoa alkuperäiseen arvoon.

Iso osa testeistä liittyen siihen, kuinka ohjelma todellisuudessa toimii, on tapahtunut kuitenkin empiirisesti. Algoritmien toimintaa on pääasiassa testattu maalamalla käyttöliittymään seiniä ja sen jälkeen ajamalla algoritmi etsimään polku maaliin. Tämä on toistettavissa melko yksinkertaisesti käynnistämällä ohjelma ja piirtämällä tietyn tyyliset seinät ruudukkoon ja ajamalla ohjelmisto. Testauksen perusteella ohjelma tuottaa aina saman tuloksen samankaltaisilla arvoilla.

Testaus on ollut melko kattavaa ja algoritmeilla on empiirisesti testattu niin yksinkertaisia kuin monimutkaisempiakin kenttiä, esimerkiksi Astar algoritmilla on testattu rakentaa mahdollisimman monimutkainen ”labyrintti” ruudukkoon ja tämän jälkeen ajamalla itse algoritmi. Tämän jälkeen on tutkittu polku, jota algoritmi päätti käyttää, ja selvitetty minkä takia kyseinen polku on valittu.

Testit ovat muunmoassa paljastaneet sen, että JPS algoritmi ei ensimmäisissä versioissa nähnyt ollenkaan tiettyjen seinien taakse, vaan nämä seinät jäivät niin sanottuihin sokeisiin pisteisiin. Tämä ongelma on kuitenkin saatu jokseenkin ratkaistua jälkeenpäin.

Testejä on myös suoritettu vertailutesteinä, jolloin samalle kentälle on ajettu molemmat algoritmit ja verrattu, kuinka ne eroavat toisistaan.

Suorituskykytestauksessa JumpPointSearch tuntuu voittavan Astar algoritmin aika reippaasti jokaisessa tilanteessa. Ajoin Molemmat algoritmit kymmenen kertaa samalle kentällä ja tyhjällä kentällä Astar sai keskiarvoksi 6.2ms ja JPS 0.6ms. Kun kentälle lisättiin seiniä, niin Astar sai keskiarvoksi 12.3ms ja JPS 1.4ms. Tässä kuitenkin tulee ottaa huomioon graafinen käyttöliittymän päivittämiseen liittyvä hitaus, ja se kuinka Astar joutuu kokonaisuudessaan värittämään useamman Noden kuin JPS. Koodin tulisi lisätä ominaisuus ajaa algoritmit päättömänä, tai muuten vain ilman graafisia ominaisuuksia testauksen tarkentamiseksi.