OULUN SEUDUN
AMMATTIKORKEAKOULU

Janika Pasma

# 3D-POHJAISEN VERKKOPELIN TESTAUSMENETTELYN SUUNNITTELU

# 3D-POHJAISEN VERKKOPELIN TESTAUSMENETTELYN SUUNNITTELU

Janika Pasma
Opinnäytetyö
Kevät 2011
Tietojenkäsittely
Oulun seudun ammattikorkeakoulu

**TIIVISTELMÄ**

Oulun seudun ammattikorkeakoulu
Tietojenkäsittely

Tekijä: Janika Pasma
Opinnäytetyön nimi: 3D-pohjaisen verkkopelin testausmenettelyn suunnittelu
Työn ohjaaja: Tapani Alakiuttu
Työn valmistumislukukausi ja -vuosi: Kevät 2011 Sivumäärä: 36

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa testaus toimeksiantajan luomaan verkkopeliin nimeltä PolarHeroes. Työssä selvitettiin kuinka testaus voidaan liittää pelinkehitysprojektiin ja kuinka toimeksiantaja Fantastec Oy voi jatkossa jatkaa testausta pelinkehityksen aikana.

Ohjelmistotestauksen teoriaan tutustumalla selvitettiin testauksen tarkoitusta ohjelmistoprojekteissa ja mitkä testausmenetelmät sopisivat tähän tapaukseen. Teoriaa hyödyntämällä testauksen suunnittelussa, keskeisimmiksi käsitteiksi nousivat v-malli ja tutkiva testaus.

Tietoperustaa soveltaen työ aloitettiin tutkivalla testauksella, jossa opeteltiin pelin käyttöä ja samalla kirjoitettiin muistiinpanoja ja testitapauksia dokumentteihin. V-mallin osuus pysyi taustalla mukana, kun testitapauksia luotiin, mutta sen päätarkoituksena oli ohjeistaa toimeksiantajaa testauksessa ja sen jatkamisessa.

Tutkivassa testauksessa saavutettiin testijoukkoja ja –tapauksia, joilla mitattiin pelin toimivuutta. Pelissä ilmenneet virheet kirjattiin yrityksen käyttämään projektityökaluun, josta pelin muut kehittäjät, kuten ohjelmoijat, pystyivät virheet katsomaan ja korjaamaan. Testauksen aikana syntyi myös mielipiteitä ja kehitysehdotuksia peliin.

Asiasanat: testaus, verkkopelit, 3D, testausmenetelmät, testitapaukset

**ABSTRACT**

Oulu University of Applied Sciences
Degree Programme in Business Information Technology

---

Author: Janika Pasma
Title of thesis: Designing a testing procedure for a 3D-based online game
Supervisor: Tapani Alakiuttu
Term and year when the thesis was submitted: spring 2011
Number of pages: 36

---

The aim of the thesis was to design a testing process for an online game called PolarHeroes, and to execute the testing. It was investigated how this testing can be included in the game development project. In addition, it was studied how the client, Fantastec Oy, could continue testing during development of the game in the future.

The theoretical framework provided basis for determining the purpose of testing in general. With a help of this it was also possible to determine the suitable testing methods. V-model and exploratory testing were found as the most essential concepts in designed testing processes.

First, the theoretical framework was applied in exploratory testing which means that the game was learned at the same as notes were taken and test cases written. V-model was at the background when the test cases were created but its main purpose was to guide the client in testing.

Test suites and cases were created in exploratory testing and they helped to evaluate functionality of the game. The errors that occurred in the game were recorded in the project tool used by the client company. This way other developers such as programmers could find and correct the errors. As a result of testing, development suggestions were made also during the testing

---

Keywords: testing, online game, 3D, testing method, test cases

## SISÄLLYS

# 1 JOHDANTO

Työn tarkoituksena oli suunnitella ja toteuttaa testaus toimeksiantajan luomaan verkkopeliin. Työssä tarkastellaan mahdollisuuksia, miten testaus liitetään pelinkehittämisprojektiin, ja mitkä menetelmät olisivat käytännöllisiä. Haasteena liitettävyydessä on testauksen suunnitelmallisuus, koska testaajan on vaikeaa tietää ennalta pelaajien toimintoja ja liikkeitä pelissä. Tähän testaukseen otettiin mukaan v-malli ja tutkiva testaus. Nämä kaksi lähestymistapaa eroavat omalla tavallaan toisistaan, mutta joissakin tapauksissa niiden yhteiskäyttö on ollut hyödyllistä.

Testitapausten suunnittelu voi olla hankalaa etukäteen, kun ei tiedä testattavasta kohteesta paljon mitään. Ketterään testaukseen kuuluva tutkiva testaus on oiva tapa saada testaaja tutustumaan ja ehkä myös sattumanvaraisesti löytämään virheitä. Jottei testaus kuitenkaan jäisi vain sille tasolle, testaaja tutkii, kirjoittaa muistiinpanoja ylös, jatkaa testausta, luo ja toteuttaa testitapauksen, ja jatkaa taas testausta.

V-malli on prosessimalli, jossa ensin suunnitellaan testaus tarkasti ja luodaan testitapaukset. Tämän jälkeen testaus toteutetaan ja verrataan sen tuloksia määrittelyihin. Toiminnallisessa osuudessa pääpainoni oli toiminnallisuus- ja käytettävyystestauksessa, kun taas rajasin pois esimerkiksi yksikkö- ja integrointitestauksen, koska ne kuuluvat ohjelmoijille.

Toimeksiantajani Fantastec Oy on 3D-teknologiaan erikoistunut yritys. Se on osallistunut OAMK:n Kilpa-projektiin, jonka avulla opinnäytetyöaiheeni löysin. Opinnäytetyöstäni oli hyötyä toimeksiantajalleni, koska tarkoituksena oli varmistaa pelin laatu tehokkaalla ja suunnitelmallisella testauksella.

Suomen peliala on kasvanut ja saanut lisää huomiota. Vuoden 2010 parhaiksi peleiksi Time-lehti valitsi Angry Birdsin ja Alan Wake:in, jotka kummatkin ovat suomalaisia. Pelialalla on yleisen käsityksen mukaan liian vähän naisia. Alalle toivottaisiin enemmän naisia, koska tekijäporukka on liian miesvaltainen. (Juvo-

nen 2011, 28.) Uskaltaisin väittää Juvosen artikkelin mukaan, että opinnäytetyön aihe on ajankohtainen ja merkityksellinen minun ammatilliselle kehitykselleni, varsinkin kun kyseinen ala on herättänyt enenevässä määrin kiinnostusta.

Testauksen kohteena toimi toimeksiantajan luoma lapsille suunnattu PolarHeroes-peli. Pelin kohderyhmänä on 4-10-vuotiaat lapset. Peli pyörii 3D-pohjaisessa ympäristössä ja pelaaminen tapahtuu verkossa, johon tarvitaan Unity Web Player -sovellus. Peli on vielä beta-testausvaiheessa, ja peliin lisätään koko ajan eri minipelejä ja uusia toimintoja.

Sivustolle rekisteröitymisen jälkeen lapsi voi alkaa tehdä peliin kuuluvia tehtäviä Lapin maisemissa. Peli sisältää 26 tehtävästä koostuvan juonen, jossa seikkailevat televisiosta tutut Red Caps -hahmot. Tarkoituksena on tehtävä tehtävältä palauttaa Joulupukille Maagisen Kristallin palat, jotta joulu olisi pelastettu. Eri tehtävien välissä pelaaja voi seikkailla rauhassa eri kylien välillä tai rakentaa itselleen unelmakodin, keskustella muiden lasten kanssa, saada uusia kavereita, luoda oman pelihahmon, hoivata lemmikkejä, rakentaa erilaisia objekteja sekä pelata yksin- ja monipelejä. (Fantastec Oy, 2010. Hakupäivä 8.4.2011.)

Pelastuspartio Red Caps tulee tutuksi pelin lisäksi vuoden 2011 aikana esitettävistä kansainvälisestä TV-animaatiosarjasta ja 3D-elokuvasta. Animaatiosarjassa on 26 jaksoa ja jokaisen jakson seikkailuun liittyy opetuksellisuus. (sama.)

# 2 TESTAUKSEN TARKOITUS

Puhekielessä testauksella tarkoitetaan lähes mitä tahansa kokeilemista. Ohjelmistojen testauksella tarkoitetaan perinteisesti suunnitelmallista virheiden etsimistä ohjelman tai jonkin sen osan suorittamisella. Keskeisimmät sanat ovat suunnitelmallinen ja etsiminen, koska testaus tapahtuu useimmiten kokeilemalla ohjelmaa satunnaisesti jollain syöttöaineistoilla. Jos testaajana on ohjelman ohjelmoija, tavoitteena on usein ennemmin osoittaa ohjelman toimivuus kuin virheiden löytäminen. Testauksen määrä ei aina ole sama kuin testauksen tehokkuus: pienellä, huolellisesti suunnitellulla testitapausjoukolla voidaan saada parempia tuloksia muutaman tunnin testauksella kuin päiväkausien satunnaisella kokeilulla. (Haikala & Märijärvi 2006, 284.)

Ohjelmistokehitykseen liittyy epäselvyyksiä, oletuksia ja puutteellista ihmisten välistä viestintää. Jokainen ohjelmistoon tehty muutos, uusi pala toiminnallisuudessa tai yritys korjata vikaa johtaa virhemahdollisuuteen. Nämä kaikki virheet kasvattavat riskiä, että ohjelmisto ei täytä käyttötarkoitusta sen kasvaessa. Testaus vähentää tätä riskiä. (Jenkins 2008, hakupäivä 8.4.2011.)

Laadunvarmistusprosessien käytöllä voidaan yrittää estää virheiden pääsy ohjelmistoon, mutta ainoa keino olemassa olevien virheiden vähentämiseksi on testata ohjelmistoa. Seuraamalla ja lisäämällä testauksen jaksoja, voidaan tunnistaa ongelmat ja ratkaista ne. Testaus myös auttaa määrittelemään riskin kokeilemattomassa palassa ohjelmistoa. Muutosten jälkeen ohjelmiston osaa voidaan suorittaa valvotuissa oloissa ja sen käytöstä havainnoidaan. Tämän tiedon pohjalta voidaan tehdä päätös siirtyä seuraavaan vaiheeseen projektissa tai yrittää korjata ongelma. (sama.)

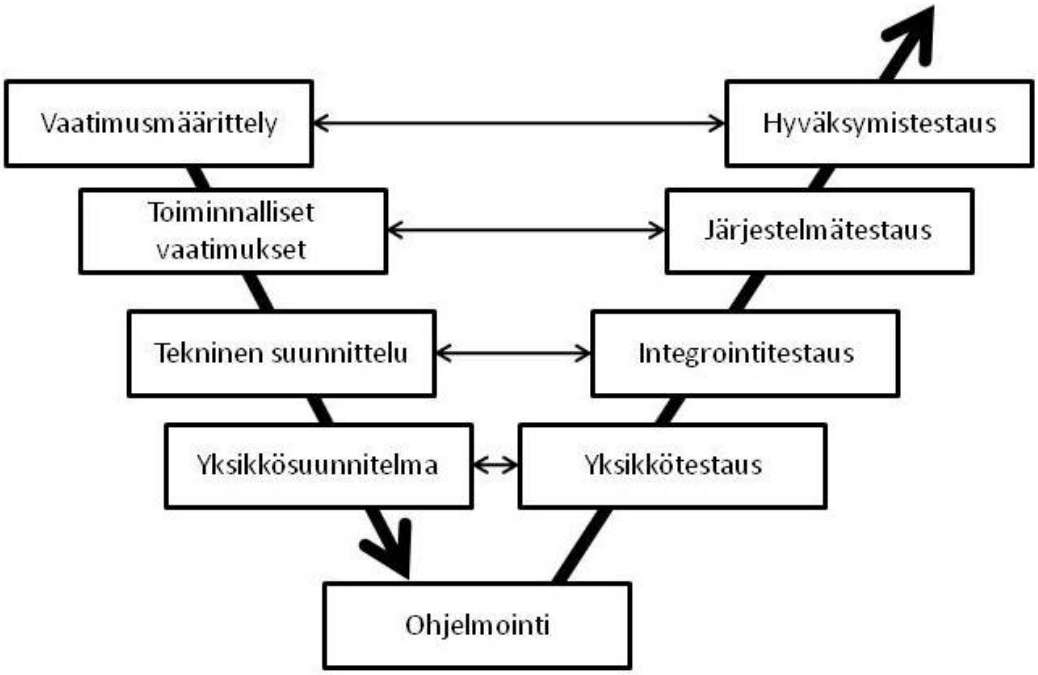
Blackin mukaan testauksen tarkoituksen luullaan usein olevan sen toteennäyttäminen, että ohjelmisto toimii täydellisesti. Usein luullaan myös, että kun ohjelmisto on testauksen ja tuloksien analysoinnin jälkeen hyväksytty tuotantoon, siinä ei voi olla enää virheitä, koska testauksessa on käyty läpi todennäköiset virhetilanteet. (2007, 6.)

Todellisuudessa tämä on täysin mahdotonta. Vähänkin kokeneet ohjelmoijat ja testaajat tietävät, ettei täysin virheetöntä ohjelmistoa ole enää mahdollista tehdä, sillä nykyään laitteissa on erittäin monimuotoiset ominaisuudet, ja ne käyttävät hyvin monimutkaisia ja suuria ohjelmistokokonaisuuksia. Virhetilanteita tulee väkisin vastaan, kun tiedon siirto, käyttö, tallennus ja uudelleen käyttö eri ohjelmistojen välillä sekä ohjelmistojen sisällä on niin monimutkaista. Näiden minimointiin auttaa virhetilanteiden ennakointi ja testaus, mutta kaikkien vikatilanteiden ennakointi on täysin mahdotonta. Ohjelmiston loppukäyttäjien syötteitä, toimintoja ja tilanteita ei voida ennustaa. Täydellistä kaiken kattavaa testausta ei olisi mahdollista toteuttaa, vaikka kaikki mahdolliset syötteet, toiminnot ja esiehtojen yhdistelmät testattaisiin ja vaikka budjetti, käytettävissä oleva aika ja resurssit olisivat äärettömät. (Black 2007, 6.)

# 3 TESTAUSMENETELMÄT

V-mallin mukaisesti testaustasoja ovat yksikkötestaus, integrointitestaus, järjestelmätestaus ja hyväksymistestaus. Järjestelmätestauksessa voi olla lisänä tarkempia testaustasoja. Muita käytettyjä termejä ovat alfa- ja betatestaus sekä regressiotestaus. (Haikala & Märijärvi 2006, 289.)

V-mallissa (katso kuvio 1) testauksen suunnittelu tapahtuu testaustasoa vastaavalla suunnittelutasolla. Hyväksymistestaus tehdään ja suunnitellaan vaatimusten pohjalta, järjestelmätestaus määrittelyvaiheessa, integrointitestaus teknisessä suunnitteluvaiheessa ja yksikkötestaus ohjelmointivaiheessa. Testauksen tuloksia voidaan näin vertailla niitä vastaaviin dokumentteihin. (Haikala & Märijärvi 2006, 289.) V-mallissa testausprosessi on mukana koko ohjelmistoprosessin alusta alkaen, joten projektin myöhästyessä siitä ei voida karsia sen enempää kuin muista prosessin osavaiheista (Taina 2004, hakupäivä 29.3.2011).




*KUVIO 1. V-malli (Toroi 2005, hakupäivä 29.3.2011)*

### 3.1 Yksikkötestaus

Yksikkötestauksessa (Unit Testing) testataan yksittäisiä yksiköitä. Yksikkö tarkoittaa yleensä noin 100–1 000 ohjelmariviä. Yksikön toimintaa verrataan yksikkösuunnittelun tuloksiin, mutta tavallisesti määrittelydokumenttiin. Testauksen suorittaa yleensä yksikön toteuttaja eli ohjelmoija. Testauksen suorittamiseksi voidaan joutua tekemään testipetejä, joilla yksikköä kokeillaan. Testipetiin voi kuulua sovelluksen ympäristöä simuloivia osia, kuten testiajureita ja tynkämoduuleita. Testiajurit mahdollistavat yksikön toteuttamien palveluiden kutsumisen ja tulosten katselun. Tynkämoduulit taas korvaavat testattavan yksikön tarvitsemat muut yksiköt, jos niitä ei ole vielä tehty. (Haikala & Märijärvi 2006, 289.)

Yksikkötestauksessa testaaja yrittää havaita vikoja, jotka liittyvät toiminnallisuuteen ja yksikön rakenteeseen. Yksiköt ohjelmoidaan monesti toisistaan irrallaan, jolloin ne voidaan myös testata toisistaan erillään. Yksikkötestaus on testauksen peruskivi, jonka vuoksi yksiköt on testattava huolella. (Toroi 2005, hakupäivä 29.3.2011.)

### 3.2 Integrointitestaus

Integrointitestauksessa (Integration Testing) yhdistellään yksiköitä tai yksikköryhmiä (osajärjestelmiä). Painopiste löytyy yksiköiden välisten rajapintojen toimivuuden tutkimisesta. Testauksen tuloksia verrataan yleensä tekniseen määrittelyyn. Integrointitestaus edistyy usein lähekkäin yksikkötestauksen kanssa. Testauksen kattavuuden kannalta yksikkötestauksella on helpompi saavuttaa hyvä testikattavuus, koska testattavan kokonaisuuden kasvaessa on koko koodin läpikäynti vaikeaa. Useimmiten integrointi etenee kokoavasti alimman tason yksiköistä ylöspäin. Jäsentävässä eli osoittavassa integroinnissa on päinvastainen etenemissuunta. (Haikala & Märijärvi 2006, 290.)

Burnsteinin mielestä integraatiotestauksen kaksi tärkeää tavoitetta on havaita vikoja, jotka tapahtuvat yksiköiden rajapinnoilla, sekä muodostaa yksittäisistä

yksiköistä alijärjestelmiä. Näin saadaan järjestelmä lopulta siihen kuntoon, että se on valmis järjestelmätestaukseen. (2003, 152.)

### 3.3 Järjestelmätestaus

Järjestelmätestaus (System Testing) vaatii paljon resursseja. Tavoitteena on varmistaa, että järjestelmä toimii vaatimusten mukaan. Lisäksi järjestelmän ja sen määrittelyn väliltä tulisi löytää ja korjata mahdollisimman paljon ristiriitoja. (Räsänen 2010, Hakupäivä 29.3.2010.)

Järjestelmätestauksessa arvioidaan sekä toiminnallista käyttäytymistä että laatuvaatimuksia, kuten luotettavuutta, käytettävyyttä, suorituskykyä ja turvallisuutta. Nämä palaset testauksessa ovat erityisen hyödyllisiä havaittaessa ulkoisten laitteiden ja ohjelmistojen rajapintavikoja. Rajapintavikoja voivat olla esimerkiksi ne, jotka aiheuttavat kilpailun olosuhteet, umpikujia, ongelmia keskeytyksissä ja lukuun ottamatta käsittelyä, ja tehotonta muistin käyttöä. Järjestelmätestauksen jälkeen ohjelma käännetään käyttäjien arvioitavaksi hyväksymistestauksen ajaksi. Järjestelmätestaus on hyvä harjoitusskenaario hyväksymistestaukseen. (Burnstein 2003, 163.)

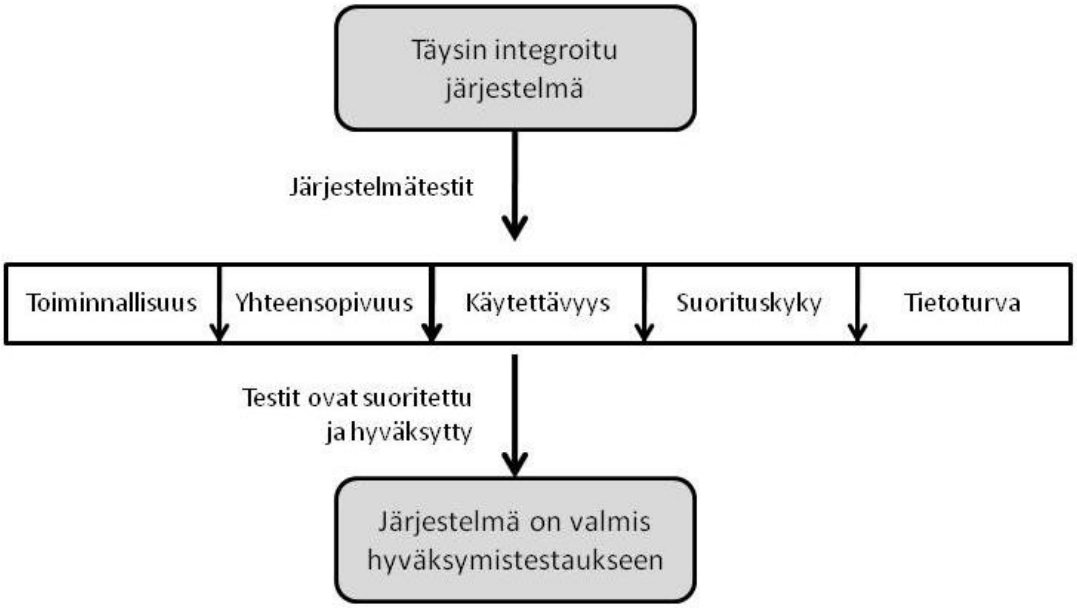
Kaikkia ohjelmistojärjestelmiä ei tarvitse käydä läpi kaikilla järjestelmätestauksen lajeilla. Testauksen suunnittelijoiden tarvitsee päättää mitä testityyppiä sovelletaan mihinkin ohjelmistojärjestelmään. Päätökset riippuvat järjestelmän ominaisuuksista ja saatavilla olevista testausresursseista. (sama.)

Burnstein toteaa, että järjestelmätestauksen suunnittelu on monimutkainen tehtävä. Suunnitelmassa on monia osia, joihin täytyy osata valmistautua. Nämä liittyvät esimerkiksi testauksen lähestymiseen, kustannuksiin, aikatauluihin, testitapauksiin ja testausmenetelmiin. (2003, 163.)

Koska järjestelmä koostuu osista, monet tämäntyyppiset testit on toteutettu komponenttien osille ja alijärjestelmille. Kuitenkin järjestelmätestauksen aikana testaajat voivat toistaa näitä kokeita ja muotoilla lisätestejä koko järjestelmässä.

Toistattaessa testejä voidaan joissain tapauksissa pitää regressiotestaus, koska ohjelmassa on todennäköisesti tapahtunut muutoksia vaatimuksiin ja itse järjestelmään projektin aloittamisesta alkaen. Järjestelmätestauksessa tunnollinen työ on oleellista korkealle ohjelmiston laadulle. Kunnolla suunniteltu ja toteutettu järjestelmätestaus on erinomainen valmistautuminen hyväksymistestaukseen. (sama.)

Pressmanin mukaan järjestelmätestauksessa on useita erilaisia testejä, joiden ensisijaisena tarkoituksena on harjoitella tietokonepohjaista järjestelmää. Vaikka jokaisella testillä on erilainen tarkoitus, jokainen testi varmistaa, että järjestelmän elementit on kunnolla integroitu ja suoritettu jaettuihin tehtäviin. (2005, 408.) Kuviossa 2 havainnollistetaan järjestelmätestauksen tyyppejä, jotka suoritetaan integraatiotestauksen jälkeen. Järjestelmätestaustyyppien jälkeen voidaan tehdä tarvittava regressiotestaus tai siirtyä suoraan hyväksymistestaukseen.




*KUVIO 2. Järjestelmätestauksen tyypit (Burnstein 2003, 165)*

### 3.3.1 Toiminnallisuus

Järjestelmän toiminnallisuustestauksessa (Functional Testing) on paljon päällekkäisyyttä hyväksymistestauksen kanssa, koska molemmissa todistetaan jär-

jestelmän toimivuutta. Toiminnallisuustestausta käytetään sen varmistamiseksi, että ohjelma noudattaa vaatimusmäärittelyä. Kaikki toiminnalliset vaatimukset täytyy olla saavutettavissa järjestelmässä. Pääpaino on syötteiden ja kunnon tulosteen jokaisessa toiminnossa. Järjestelmän on osattava käsitellä väärää ja laitonta tulostetta. Järjestelmän käyttäytyminen jälkimmäisessä asianhaaran testauksessa on huomioitava. Kaikki toiminnot on testattava. (Burnstein 2003, 166.)

Burnsteinin mukaan testien olisi keskityttävä seuraaviin tavoitteisiin.

- Ohjelman täytyy hyväksyä kaikki lailliset syötteet tyypeissä tai luokissa
- Kaikki luokat laittomissa tuotoksissa on hylättävä (järjestelmän pitäisi jäädä käytettäväksi)
- Kaikki mahdolliset luokat järjestelmän tuotoksessa pitää olla harjoiteltuja ja tutkittuja
- Kaikki tehokkaat järjestelmätilat ja tilasiirtymät pitää olla harjoiteltuja ja tutkittuja
- Kaikkien toimintojen pitää olla harjoiteltuja. (2003, 167.)

Kuten edellä mainittiin, määriteltyä ja dokumentoitua muotoa tulisi käyttää kuvaamaan testituloksia toiminnallisessa ja kaikissa muissa järjestelmätesteissä. Jos vika on havaittavissa, muodollinen testi häiriöraportissa tulisi olla valmiina ja palautettuna (kuten myös testiloki) koodin korjaajalle. (sama.)

### 3.3.2 Yhteensopivuus

Yhteensopivuustestaus (Compatibility Testing) varmistaa, että sovellus toimii oikein monilla selaimilla ja kokoonpanoilla. Yhteensopivuustestaus voidaan suorittaa paikassa, joka sisältää erilaisia alustoja. (Perry 2006, 808–809.)

Perry toteaa, että web-pohjaisen sovelluksen on kyettävä toimimaan oikein monenlaisilla järjestelmäkokoonpanojen kanssa, kuten selaimilla, käyttöjärjestelmillä ja laitejärjestelmillä. Web-sovelluksien ulkonäköön ja toimintoihin vaikuttavat käyttöjärjestelmät ja valittu selain. Jokaisessa selaimessa on kokoonpanovaih-

toehdot, jotka vaikuttavat siihen, kuinka tiedot näkyvät. Järkevin testausstrategia on määritellä optimaalinen kokoonpano, joka löytyy tavallisimmista selaimista ja testata niiden pohjalta luotua kokoonpanoa. (2006, 804-805.) Tässä tapauksessa on hyvä ottaa huomioon audio-, video- ja multimediatuet, muisti (RAM) ja kovalevytilat, uudelleenlataus ja välimuisti.

### 3.3.3 Käytettävyys

Käytettävyystestauksella (Usability Testing) pyritään varmistamaan, että käyttäjä voi käyttää ohjelmaa mahdollisimman hyvin ja selviää tehtävistä, joiden suorittamiseksi järjestelmää ollaan rakentamassa. Käytettävyystestaus on useimmiten järjestelmän käyttöliittymän testausta, ja sitä tehdään usein jo määrittelyvaiheessa käyttöliittymäprototyypillä. Käytettävyystesteihin pyritään usein valitsemaan pieni otos tulevista käyttäjistä, jolloin he suorittavat erilaisia tehtäviä valvotussa koetilanteessa. Tavallinen tapa koetilanteessa on käyttää videointia, jossa käyttäjä kertoo ääneen millaiseen pohdintaan hänen toimintansa perustuu. Ensimmäisillä kerroilla testin tulokset ovat usein suuria hämmästyksen aiheita järjestelmän kehittäjille. Käytettävyystestauksen lisäksi voidaan käyttää myös ohjelmistojen käytettävyyden arviointiin erikoistuneita ammattilaisia. (Haikala & Märijärvi 2006, 291.)

Käytettävyystestaus voidaan toteuttaa eri tavoilla. Erilaisia tapoja voivat olla suora havainnointi ihmisten käyttäessä web-sovelluksia, käytettävyystutkimukset ja betatestit. Päätavoitteena käytettävyystestauksessa on varmistaa, että ohjelmaa on helppo ymmärtää ja navigoida. (Perry 2006, 808.)

### 3.3.4 Suorituskyky

Järjestelmän suorituskykytesteissä on tavoitteena nähdä, kattavatko ohjelmiston suorituskyvyn vaatimukset. Testaajat oppivat myös suorituskykytestissä onko laitteisto- tai ohjelmistotuottajilla ollenkaan vaikutusta järjestelmän suorituskykyyn. Suorituskykytestaus (Performance Testing) sallii testaajien virittää järjestelmän, eli optimoida järjestelmän resursseja. Testaajat voivat esimerkiksi

todeta, että järjestelmä täytyy jakaa muistialtaisiin tai muuttaa prioriteettitaso tiettyyn järjestelmän toimintoon. (Burnstein 2003, 167-168.)

Suorituskyvyn tavoitteet on ilmaistava selkeästi käyttäjien ja asiakkaiden vaatimusasiakirjassa ja todettava selvästi järjestelmän testaussuunnitelmassa. Tavoitteiden on oltava määriteltyjä. Testaussuunnitelmassa ei pidetä hyväksyttävänä vaatimuksena sitä, että järjestelmä palauttaa vastauksen kyselystä "kohtuullisessa ajassa". Aikavaatimus on määriteltävä kvantitatiivisesti. Suorituskyvyn testauksen tulokset ovat laskettavissa. Käytetyissä jaksoissa todellinen vastausaika määräytyy varsinaisesti tapahtumien käsittelyssä ajanjaksoa kohden eikä sekunneissa (minuuteissa ja niin edelleen). Näitä voidaan arvioida vaatimuksen tavoitteisiin. (sama.)

Pressmanin mielestä suorituskykytestauksen tarkoituksena on testata ohjelmiston suorituskykyä integroidun järjestelmän puitteissa. Suorituskykytestaus esiintyy kaikissa testausprosessin vaiheissa, kuten ihan alimmalta tasolta asti yksittäisen moduulin suorituskyvystä voidaan tehdä arviotestaus. Todellinen suorituskyky järjestelmässä voidaan kuitenkin todeta vasta sitten, kun kaikki järjestelmän osat ovat täysin integroituneet. (2005, 410.)

Suorituskykytestaukset ovat usein yhdistettävissä stressitestauksen kanssa, ja yleensä ne edellyttävät sekä laitteiston että ohjelmiston asentamista. Tämä on usein tarpeen mitattaessa resurssien käyttöasteen vaativuutta. Ulkoista asennusta voidaan valvoa toteuttamisvälein, lokitapahtumilla (keskeytykset) tapahtumahetkillä ja näytteellä koneen tilasta säännöllisin väliajoin. Järjestelmää asentaessaan testaaja voi huomata tilanteita, jotka johtavat suorituskyvyn heikentymiseen ja mahdollisesti järjestelmän epäonnistumiseen. (Pressman 2005, 410.)

Kuormitustestaus (Load Testing) kuuluu suorituskykytestaukseen ja siinä selvitetään testattavan järjestelmän kykyä suoriutua tehtävistään kuormitettuna vaaditussa ajassa. Tavoitteena on myös varmistua ennen käyttöönottoa, että käyttöönotto ja tuotantokäyttö onnistuvat tarkoitetulla tavalla. Tarve kuormitustestaukselle syntyy esimerkiksi silloin, kun uudentyyppisissä sovelluksissa mahdolli-

nen käyttäjäkunta on lähes rajaton. (Conformiq Software Ltd 2006, hakupäivä 9.3.2011.)

Kuormitustestaus on mahdollista toteuttaa manuaalisesti, mutta se on työlästä ja heikosti toistettavaa, joten työkalujen avulla on helpompaa luoda kuormaa ja mitata vasteaikoja. Haasteena voidaan pitää testauksen aloitusta, koska se voidaan aloittaa vasta kun järjestelmä on pääosin rakennettu. Myös järjestämistä voidaan pitää haasteena, koska tuotannonkaltaista ympäristöä voi olla vaikeaa järjestää vain testausta varten. Tyypillisimpiä virheitä ilmaantuu, kun generoitu kuorma on liian yksipuolinen, jolloin se ei ole todenmukainen, tai kun suorituskykyä rajoittavia ilmiöitä ei löydetä tai niitä etsitään vääristä paikoista. (Conformiq Software Ltd 2006, hakupäivä 9.4.2011.)

Monikäyttäjätestauksessa (Multiuser, käytetään myös nimeä Multiplayer Testing) altistetaan järjestelmä suurelle määrälle samanaikaisia virtuaalikäyttäjiä. Tällä tavalla yritetään todistaa, että se hyväksyy tarpeellisen määrän yhtäaikaisia käyttäjiä. (HiQ 2011, hakupäivä 9.4.2011.) Tätä testausmenetelmää käyttävät paljon isot pelitalot, jolloin verkkoon laitetaan pelistä beta-versio, jota loppukäyttäjät pääsevät testaamaan ja pelaamaan.

### 3.3.5 Tietoturva

Jokainen järjestelmä, joka hoitaa arkaluonteisia tietoja tai aiheuttaa sellaisia toimenpiteitä, jotka voivat sopimattomasti vahingoittaa yksilöitä, voi olla laittoman tai asiattoman tunkeutumisen kohteena. Tunkeutuminen käsittää laaja-alaisen toiminnan: hakkeri, joka yrittää tunkeutua järjestelmään huvikseen, tyytymättömät työntekijät, jotka yrittävät päästä kostamaan, epärehelliset henkilöt, jotka yrittävät saada laittomasti henkilökohtaista hyötyä. (Pressman 2005, 409.)

Hyökkäykset voivat olla satunnaisia tai systemaattisia. Vahinkoa voi tapahtua eri tavoin, kuten viruksilla, troijalaisilla, takaovilla ja laittomilla jakeluilla. Tietoturvaloukkausten vaikutukset ovat laajoja ja niistä voi aiheutua esimerkiksi tietojen

menetystä, lahjottua tai väärää tietoa, yksityisyyden loukkauksia ja palvelunestoa. (sama.)

Pressmanin mukaan annettaessa riittävästi aikaa ja resursseja, hyvä tietoturvatestaaja pääsee lopulta tunkeutumaan järjestelmään. Ohjelmistokehittäjän roolina on tehdä tunkeutumisesta paljon kalliimpaa, kuin saatavissa olevan tiedon arvo. (2005, 409.)

### 3.4 Regressiotestaus

Regressiotestaus (Regression Testing) ei ole testaustaso, vaan ohjelmiston uudelleentestaamista. Uudelleentestaus tapahtuu kun tehdään muutoksia varmistaakseen, että uusi ohjelmistoversio on säilyttänyt ominaisuuksia vanhasta versiosta ja ettei uusia virheitä ole ilmestynyt muutosten vuoksi. Regressiotestausta voi esiintyä kaikilla testauksen tasoilla. Esimerkiksi yksikkötestausta ajettaessa yksikkö voi päästä läpi näistä testeistä, kunnes yksi kohta testeistä paljastaa vian. Yksikön korjaamisen jälkeen se uudelleentestataan kaikilla vanhoilla testitapauksilla taatakseen, että muutokset eivät ole vaikuttaneet sen toimivuuteen. (Burnstein 2003, 176.)

Burnsteinin mukaan regressiotestaus on erityisen tärkeä silloin, kun moninkertaisen ohjelmiston versiot ovat kehittyneet. Käyttäjät haluavat uusia ominaisuuksia viimeisemmästä versiosta, mutta silti odottavat vanhojen ominaisuuksien pysyvän paikallaan. Testitapaukset, testausmenetelmät ja muut testaukseen liittyvät kohteet aikaisemmista versioista pitäisi olla saatavilla niin, että nämä testit voidaan suorittaa ohjelmistossa uusien versioiden kanssa. Automatisoidut testaustyökalut tukevat testaajia tässä hyvin paljon aikaa vievässä tehtävässä. (2003, 176.)

### 3.5 Hyväksymistestaus

Hyväksymistestaus (Acceptance Testing) on erittäin tärkeä virstanpylväs kehittäjille (Burnstein 2003, 177). Ohjelmistokehittäjän on melkein mahdotonta enna-

koida, miten asiakas todella käyttää ohjelmaa. Käyttöohjeet voidaan tulkita väärin, outoja yhdistelmiä voidaan käyttää säännöllisesti, tuotokset jotka näyttivät testaajan silmissä helpoilta, voivat olla vaikeaselkoista loppukäyttäjälle. Jos ohjelmisto on kehitetty monille asiakkaille, on epäkäytännöllistä tehdä hyväksymistestiä jokaiselle. Useimmiten ohjelmistojen kehittäjät käyttävät prosesseja nimeltään alfa- ja betatestaus, joissa paljastetaan virheitä, jotka vain loppukäyttäjät kykenevät löytämään. (Pressman 2005, 406-407.)

Alfatestaus suoritetaan kehittäjän tiloissa loppukäyttäjän testatessa ohjelmaa. Tyypillinen käyttäjä käyttää ohjelmaa perusasetuksilla nauhoittaen virheet ja käytettävyysongelmat samalla kun kehittäjä ”tarkkailee yli olkapään”. Alfatestit tehdään valvotussa ympäristössä. (Pressman 2005, 407.)

Betatestaus hoidetaan loppukäyttäjän ympäristössä. Toisinkuin alfatestaus, kehittäjä ei ole yleensä läsnä. Siksi betatestaus on suoraa soveltamista ohjelmiston tulevassa käyttöympäristössä, ja tällöin kehittäjä ei voi ohjata sitä. Loppukäyttäjä nauhoittaa kaikki ongelmat (oikeat tai kuvitellut), jotka ovat ilmenneet betatestauksen aikana ja raportoi niistä kehittäjälle säännöllisin väliajoin. Koska betatestauksen aikana havaitut ongelmat on raportoitu kehittäjälle, ohjelmistokehittäjä tekee muutoksia ja sen jälkeen valmistautuu julkaisemaan ohjelmistotuotteen koko asiakaskunnalleen. (Pressman 2005, 407.) Burnstein (2003, 178) väittää, että monissa tapauksissa korjaukset myöhästyvät ennen seuraavaa julkaisua.

# 4 TUTKIVA TESTAUS

“Exploratory testing is simultaneous learning, test design, and test execution” (Bach 2003, hakupäivä 10.4.2011).

Tutkivassa testauksessa (Exploratory Testing) opitaan, kuinka ohjelmaa käytetään. Se on sivistynyt ja harkittu lähestymistapa testaukseen ilman muistiinpanoja, ja sen avulla voidaan ohittaa ilmiselvät variaatiot, jotka on jo testattu. Tutkivassa testauksessa yhdistyvät oppiminen, testin suunnittelu ja testauksen toteuttaminen yhdeksi testauksen lähestymistavaksi. Testauksen aikana voidaan oppia lisää testattavasta järjestelmästä ja käyttää tätä tietoa hyväksi uusien testien suunnittelussa. (Crispin & Gregory 2009, 195.)

Tutkivassa testauksessa ei istuta näppäimistön eteen ja kirjoiteta jatkuvasti. Se alkaa perussäännöistä, jotka määrittävät, mitä puolia toiminnallisuudesta tutkitaan. Se edellyttää kriittistä ajattelua, tulosten tulkintaa ja niiden vertaamista odotuksiin tai vastaaviin järjestelmiin. Tutkivan testaustapahtuman aikana tehdään muistiinpanoja, jotta testit voidaan toistaa ongelmitta ja tehdä tarvittaessa lisää tutkimuksia. (Crispin & Gregory 2009, 198.)

Varsinaista testaussuunnitelmaa ei tutkivasta testauksesta löydy, mutta siinä on tiettyjä asioita, jotka vaikuttavat testaukseen. Tällaisia ovat esimerkiksi testausprojektin tarkoitus, testaajan rooli ja hänen taitonsa ja lahjakkuutensa, käytettävissä olevat työkalut ja palvelut, käytettävissä oleva aika ja testiaineisto, saatavissa oleva muiden ihmisten apu, nykyinen testausstrategia, tuote ja sen käyttöliittymä, käyttäytyminen, puutteet, testattavuus ja tarkoitus, ja se mitä testaaja tietää tai haluaisi tietää tuotteesta. Nämä tekijät muuttuvat jatkuvasti koko testausprojektin tai jopa testi-istunnon aikana. (Bach 2003, hakupäivä 10.4.2011.)

Tutkivan testauksen tehoa voidaan suurentaa koko testausprosessin ajan, kun taas suunnitelmallisella testauksella on taipumus muuttua koko ajan vähemmän tehokkaaksi, koska se ei ikinä muutu. Suunnitelmallinen testaus heikkenee monista syistä, mutta suurin syy on, että kun se on kerran suoritettu eikä ongelmia

löytynyt, useimmissa tapauksissa sen jälkeen on huomattavasti pienempi mahdollisuus löytää ongelma toisella suorittamisella, kuin jos sen sijaan suoritettaisiin uusi testi. (sama.)

Vapaamuotoinen tutkiva testaus sopii moniin tapauksiin. Sitä voidaan käyttää esimerkiksi silloin, kun tarvitaan nopeaa palautetta uudesta tuotteesta tai ominaisuudesta, tai testaajan täytyy oppia tuote nopeasti, tai kun halutaan löytää yksittäinen tärkeä vika lyhyessä ajassa. Aina kun palaute on silmukassa heikko tai kun silmukka on erityisen pitkä, hidas tai kallis, tutkiva testaus menettää tehoa. Silloin on hyvä turvautua huolellisesti suunniteltuihin testitapauksiin. (sama.)

Bach toteaa, että tutkiva testaus ei kokonaan korvaa suunnitelmallista testausta. Joissakin tapauksissa testauksesta voidaan saavuttaa paremmin läpimentävä, kun noudatetaan suunnitelmallista tapaa, mutta joissakin tapauksissa testaustapahtuma hyötyy enemmän kyvystä luoda ja kehittää testejä niitä suoritettaessa. Useimmissa tapauksissa on ollut hyödyllistä käyttää kumpaakin lähestymistapaa. (2003, hakupäivä 10.4.2011.)

Tutkivaa testausta voidaan hoitaa joko delegoimalla tai osallistumalla siihen, mutta Bachin mukaan on erittäin tehokasta tehdä se tiiminä. Yksi tapa järjestää tiimit on laittaa testaajat pareihin jakamaan keskenään tietokone, jolla he sitten testaavat. Toinen tapa on, että yksi testaaja toimii näppäimistöllä ja hiirellä, kun muut katselevat ja kommentoivat. Jos kyseinen toimija huomaa ongelman tai hänellä on kysymys, jota täytyy tutkia, voi yksi katsojista irrottautua ja yrittää tutkia tätä kysymystä muulla testausalustalla. Tämä vapauttaa toimijan ja hän voi jatkaa päätestausta ilman suurempia häiriötekijöitä. Tämä menetelmä toimii erityisen hyödyllisenä keinona, kun halutaan saada testaaja kokeilemaan toisenlaista testaustapaa eikä rutiininomaista, tai auttaa koulutettavaa testaajaa ymmärtämään tuotteen tekniikka tai testaussuunnitelman menetelmät. (2003, hakupäivä 10.4.2011.)

Tutkiva testaus etenee usein testi-istuntosarjoittain. Jokainen istunto kestää tunnista kahteen tai kolmeen tuntiin. Istunnon täytyy kestää tarpeeksi pitkään,

jotta prosessi voidaan toistaa muutaman kerran. Tällöin testaukseen ja sen tuloksiin ei pystytä koskaan täysin uppoutumaan ja tekemään samanlaisia teräviä ja älykkäitä arvauksia siitä, missä virheet ovat. (Black 2007, 271.)

Tutkivaa testausta on vaikea automatisoida, koska mikään ohjelma ei osaa olla luova. Kohlin mukaan automatisointia on kuitenkin mahdollista saada mukaan. Automatisointia käytetään, kun asennetaan testaukseen tietojen tuottaminen tai toistuvia tehtäviä. Sen jälkeen käytetään testaajan taitoja ja kokemusta todella hankalien vikojen etsimiseen, jotka salakavalasti jäävät muuten vähälle huomiolle. Automatisoitua testausta voidaan käyttää myös tutkimiseen: muutetaan sitä hieman, katsotaan tuloksia kun sen työskenteleyä, muokataan sitä uudelleen ja katsotaan mitä tapahtuu. (2007, hakupäivä 10.4.2011.)

# 5 TESTIJOUKOT

Testitapaus on yksityiskohtainen menettelytapa, jossa testataan koko ominaisuus tai osa ominaisuudesta. Testaussuunnitelma kuvaa mitä testataan, kun taas testitapaus kertoo kuinka suorittaa tietty testi. Testitapaus sisältää: testin tarkoituksen, kuvauksen siitä mitä testissä tehdään ja odotetut tulokset tai onnistumisen kriteerit testissä. (Microsoft 2003, hakupäivä 5.5.2011.)

Yksityiskohtaisessa testitapauksessa kuvataan, miten testi tehdään. Kuvailevassa testitapauksessa testaaja päättää testauksen keston, miten testi tehdään ja mitä tietoja käytetään. Useimmat organisaatiot pitävät parempana yksityiskohtaisia testitapauksia, koska määritelmät *hyväksytty* tai *hylätty,* ovat useimmiten helpoimpia. Tämän lisäksi yksityiskohtaiset testitapaukset ovat toistettavissa ja ne ovat helpompi automatisoida kuin kuvailevat testitapaukset. Yksityiskohtaiset testitapaukset vievät enemmän aikaa kehitykseltä ja ylläpidolta. Toisaalta, tulkinnanvaraiset testitapaukset eivät ole toistettavissa, eikä niistä pystytä vaatimaan virheenkorjausta. Tällöin olisi parempi käyttää aika itse testaukseen. (sama.)

Testitapaukset muodostavat testijoukon. Testijoukko kuvaa millaisia testitapauksia se sisältää tai mihin toimintoihin testitapauksissa keskitytään. Testijoukkoja luotiin toimeksiantajan peliin. Testijoukkojen luonti oli vaikeaa, koska on mahdotonta tietää miten loppukäyttäjät pelaavat peliä. Testijoukoiksi otettiin eri kyliin siirtymiset, hahmoeditorin käyttäminen, pelin juoneen sisältyvät tehtävät ja hahmon liikkuminen pelimaailmassa ja sieltä löydetyt läpimenot.

Pelistä löytyy tällä hetkellä viisi eri kylää, jotka ovat Joulupukin kylä, Iso-Syöte, Levi, Kemi ja Taikametsä. Näiden kylien välillä hahmo voi kulkea junalla tai näytöllä näkyvän maapallon avulla. Maapallon kautta liikuttaessa hahmo voi olla joko poron tai moottorikelkan kyydissä, mutta tämä ei onnistu junalla kulkiessa. Eri kylien välillä liikkuminen sujui moitteettomasti molemmilla liikkumisvaihtoehdoilla. Tästä testijoukosta ei löytynyt lainkaan virheitä.

### 5.1 Hahmoeditori

Hahmon ulkonäköä pääsee muokkaamaan hahmoeditorissa, jossa voi valita valikosta esimerkiksi ihonvärin, sukupuolen, hiukset, vaatteet ja kengät. Muutokset näkyvät heti hahmossa, ja hahmon ulkonäköä voi tarkastella eri suunnista painamalla hahmon ympärillä olevia nuolia.

Kuviossa 3 on osa hahmoeditorin testitapauksista. Hahmoeditorissa ovat omat painikkeet mistä muutokset tapahtuvat. Muutosten hylkäämiseen on olemassa oma painike, mutta se ei toiminut. Valikkojen sivut eivät aina ilmestyneet oikein. Testitapaus numero 6 on virhe valikon sivujen toimivuudesta, jossa ylimääräinen sivu ilmestyi, vaikka sellaista ei ollut toisessa valikossa.

| Testijoukko: Hahmoeditori | | |
|---|---|---|
| **Testi** | **Testitapaukset** | **Testin tulos** |
| 1 | Hyväksymispainikeen toimivuus | OK: Toimii |
| 2 | Muutosten hylkäämis(peruutus)painike | Virhe: Painikkeesta ei tapahdu mitään |
| 3 | Poika ja Tyttö painikkeiden muutokset | OK: Toimivat |
| 4 | Hahmon pyörimispainikkeet | OK: Toimivat |
| 5 | Valitaan päähineosiosta, toiselta sivulta avatarille hiukset | OK: Valittu muutos toimii |
| 6 | Valitaan paitaosio esille päähineosion jälkeen. | Virhe: Paitaosiossa ei löydy kuin yksi sivu, mutta se menee toiselle tyhjälle sivulle, koska hiusosiossa oli kaksisivua. Tapahtuu myös jos valitsee tarvikeosion toiselta sivulta jotain. Toistuu vain ensimmäisellä kerralla kun tullaan hahmoeditoriin. |

*KUVIO 3. Hahmoeditorin testitapaukset*

### 5.2 Juoneen kuuluvat tehtävät

Pelin juoneen sisältyy erilaisia tehtäviä. Uuden tehtävän voi aloittaa aina edellisen tehtävän päätyttyä. Uuden tehtävän voi aloittaa silloin, kun käy Poron luona juttelemassa. Poro vie Red Capsin päämajalle, jossa kerrotaan mikä on seuraava tehtävä. Peliruudussa pelaajalla on käytössä matkapuhelin, josta voi tarkistaa annetut tehtäväinfot. Matkapuhelin pysyy ajan tasalla ja neuvoo mitä pitää tehdä päästääkseen tehtävissä eteenpäin.

Kuviossa 4 on esimerkki juoneen sisältyvien tehtävien testitapauksista. Loppupään tehtävissä alkoi enemmän löytyä virheitä, koska niitä on todennäköisesti vähemmän testattu. Tehtäviä tulee olemaan 26 kappaletta, mutta tällä hetkellä tehtävä 22 oli viimeinen.

Tehtävissä 13 ja 22 huomasi lumipalloefektin. Kun pelistä oli löytänyt yhden virheen, niin siitä myös pystyi löytämään vielä pari lisää. Virhetilanteen jälkeen ilmeni paljon muitakin virheitä, joita oli vaikea arvioida tulivatko ne vain aiemmista virheistä vai olivatko ne yksistään oma virhe.

| | | |
|---|---|---|
| 12 | Tehtävä 12. Yhdistä viivat | OK: Ei virheitä |
| 13 | Tehtävä 13. Moottorikelkkakisan läpimenot | Virhe: Kelkka menee lelulaattikkojen välistä läpi, suuntaa puille joista pääsee myös yli. Matka päättyy kun kelkka tippuu siniseen maailmaan. |
| 14 | Tehtävä 13. Sinisen maailman jälkeen peli alkaa uudestaan | Virhe: Hahmolla ei ole enää kelkkaa alla (leijuu ilmassa). Päästeessään maaliin peli ei pääty. |
| 15 | Tehtävä 14. Säydänaukiolta sydämiä | OK: Ei virheitä |
| 16 | Tehtävä 15. Kirjeen palasten kerääminen | Virhe: Kirjeiden palasia on oikeasti 9 (yksi on näkymätön) |
| 17 | Tehtävä 15. Palasten keräämisen jälkeen | Virhe: Laskiessa mäestä ulos patja katoaa alta eikä peli pääty. Palaset voi käydä noukkimassa kävellen. |
| 18 | Tehtävä 16. Kokoa kirjeen palaset | OK: Ei virheitä |
| 19 | Tehtävä 17. Nurin-kurin | OK: Ei virheitä |
| 20 | Tehtävä 18. Suksikarusellin kyydissä | Virhe: Suksikarusellissa heiluttaessa kättä tai ollessaan surullinen, hahmo jatkaa vain toimintaansa |
| 21 | Tehtävä 19. Suksien lainaaminen | Virhe: Annettuasi sukset lainaan, käyt välillä toisessa kylässä ja palaat hakemaan suksia. Silloin mies kysyy saman kysymyksen kuin alussa, vaikka pitäisi kiittää. Painamalla "kyllä" saat jalokiven |
| 22 | Tehtävä 20. Robolaisten lassoaminen | OK: Ei virheitä |
| 23 | Tehtävä 21. Tehtävän annossa väärin | Virhe: Tehtävän annossa puhutaan satumetsästä vaikka lääke piti viedä taikametsään. |
| 24 | Tehtävä 22. Ei saanut roboja kiinni huskylla, paina: OK | Virhe: Vaikka pelaaja ei saanut roboja kiinni huskylla, pelaaja silti sai jalokiven |
| 25 | Tehtävä 22. Paina: uudestaan | Virhe: Pelaajalla ei ole enää huskya vaan se lentää ilmassa |
| 26 | Tehtävä 22. Paina: uudestaan, pallot ja välille jäänyt robo | Virhe: Heitetyt pallot menevät limittäin, väri sekamelskaa. Myös edellinen robo on jäänyt paikoilleen |
| 27 | Tehtävä 22. Paina: uudestaan, saa robot kiinni | Virhe: Vaikka palloihin osuu monta kertaa, peli väittää että robot saatiin kiinni |
| 28 | Tehtävä 22. Paina: uudestaa, peli onnistuu | Virhe: Pelin onnistuessa pelaaja jää peli paikalle eikä palaa alkuun |

*KUVIO 4. Tehtävien testitapauksia*

Tehtäviä tehdessä oli hyvä aina välillä seurata matkapuhelinta pysyykö se ajan tasalla. Useimmiten se osasi antaa oikeat neuvot oikeissa kohdissa, mutta virheitäkin ilmeni. Tavallisesti virheet ilmenivät silloin, kun hahmo oli vasta nostanut jalokiven maasta, mutta matkapuhelin neuvoi vielä ottamaan sen. Virheen toistaminen samassa kohtaa oli hankalaa, joten kyseessä oli todennäköisesti vain matkapuhelimen päivitysnopeus eri tilanteissa.

Testitapausten 26 ja 28 mukaan Robo edellisestä pelistä jäi paikoilleen ja pelaaja ei palannut aloituspaikalle. (katso kuvio 5). Kuviossa 5 näkyy tielle jätetyt esteet eli joulupallot, joita minipelissä oli tarkoitus väistää. Moni joulupallo oli jäänyt edellisestä pelistä, jonka vuoksi niitä oli nyt liikaa. Saatuaan jalokiven, hahmon täytyi kävellä takaisin aloituspaikalle.


*KUVIO 5. Esimerkki kuva testitapauksista 26 ja 28*

### 5.3 Liikkuminen ja läpimenot

Pelissä voidaan liikkua joko kävellen, poron selässä tai kelkalla. Porolla ja kelkalla pääsee kulkemaan nopeampaa kuin kävellen, mutta kelkkaa on vaikeampi kääntää ja se helposti nousee pystyyn. Läpimenolla tarkoitetaan pelialueen rajojen tai vaikka rakennuksien läpi menemistä.

Kuviossa 6 on testitapauksia pelissä liikkumisesta ja läpimenoista löydetyistä virheistä. Testijoukkoon ei kirjattu kuin virheitä, koska pelin reunojen määrittäminen testitapauksiin oli vaikeaa. Hahmon ei kuuluisi päästä rajojen yli, jottei tipahtaminen pelistä onnistuisi. Peli-alueen rajat eivät ole tarkoin määriteltyjä, jolloin rajapintoihin pääsy aiheuttaa vaikeuksia liikkumisessa. Kun hahmo menee rajapinnalle, useimmiten puiden päälle, hahmo jää jumiin. Hahmon jumiutuessa, aina ei ole muuta vaihtoehtoa kuin aloittaa peli alusta.

Pelistä löytyy tehtävien lisäksi myös minipelejä. Minipelejä löytyy jokaisesta kylästä kiikareina, mutta ainoastaan Joulupukin kylässä, Levillä ja Iso-Syötteellä ovat omat pelihallit. Kiikareita lisääntyy joidenkin tehtävien jälkeen, jolloin on mahdollisuus kokeilla tehtävässä suoritettu peli uudelleen.

Red Caps Ralli on nimensä mukaisesti rallipeli jäällä, jossa ajellaan pelin omalla kulkupelillä. Red Caps Rallissa oli inhottavaa jäädä puihin kiinni, jos ei osannut ajaa tarpeeksi varovasti mutkissa. Menopeli syöksyi puiden päälle ja jäi sinne junnaamaan. Joissain tapauksissa menopeliä ei saatu kuusista irti ja ralli täytyi lopettaa tai aloittaa uudelleen.

| Testijoukko: Liikkuminen & Läpimenot | | |
|---|---|---|
| **Testi** | **Testitapaukset** | **Testin tulos** |
| 1 | Joulupukin kylässä pääsee kulkemaan rinnettä ylös, päämajan vierelle | Virhe: Hahmon ei pitäisi päästä kulkemaan rinteelle. Rinnettä pääsee kulkemaan pitkälle, kunnes tippuu siniseen maailmaan ja peli alkaa alusta |
| 2 | Iso-Syötteellä, suksikarusellin takana oleva metsä | Virhe: Metsän läpi pääsee kulkemaan helposti, jonka jälkeen tippuu siniseen maailmaan. |
| 3 | Moottorikelkalla junan alle | Virhe: Moottorikelkka jää jumiin. Pitää odottaa että juna lähtee |
| 4 | Levin rinnettä alas reunoja pitkin | Virhe: Tietystä kohdasta pääsee liitelemään ylös taivaalle |
| 5 | Porolla kulkiessa Kemissä | Virhe: Tertun vieressä olevaan sikaveistokseen jäi jumiin. |
| 6 | Minipeli: Red Caps Ralli | Virhe: Jää kuusiin jumiin. Jos menee mutkista liian lujalla vauhdilla, menopeli jää liian helposti jumiin rajalla oleviin kuusiin |

*KUVIO 6. Testitapauksia liikkumisesta ja läpimenoista*

Kuviossa 7 näkyy kun hahmo kulkee porolla Levin yläpuolella. Kuvio 7 on esimerkki kuva edellisen testijoukon testitapauksesta 4. Porolla kulkiessaan Levin rinnettä alas, se lähti kulkemaan taivasta kohti omaa näkymätöntä tietä pitkin. Jonkun matkaa kulkiessaan se tipahtaa rinnemökin katolle.

Kuviossa 7 näkyvät myös perusruudun painikkeet eli maapallo, matkapuhelin, reppu, hymiöt ja lapanen. Repun vieressä olevaan laatikkoon pelaaja pystyy

kirjoittamaan jotain ja näyttämään sen hahmon yläpuolella. Vihreä pokaali tarkoittaa kerättyjä pisteitä ja kultainen tammenterhokolikko näyttää omistetun rahamäärän.


*KUVIO 7. Hahmo kulkee Levin yläpuolella*

Muita testijoukkoja olisi voitu tehdä minipeleistä tai pelihalleista. Aika oli rajallinen eikä näistä löytynyt suuria virheitä, jotka olisivat inspiroineet tekemään niille testitapauksia. Testitapaukset, joista ei tämän testauksen aikana löytynyt virheitä ei tarkoita ettei niistä silti voisi löytyä vielä virheitä. Virheelliset testitapaukset kirjattiin toimeksiantajan ostamaan raportointityökaluun Hansoftiin.

Virheiden toistettavuus oli hankalaa. Virheen huomatessa, piti miettiä mitä tein, mitä klikkasin, mistä kohdasta hahmo kulki ja mikä voisi olla virheen syy. Joissakin tapauksissa virheet olivat ilmiselviä ja helposti toistettavia, mutta löytyi niitäkin joissa virhe ilmestyi vain kerran. Useiden kokeilujen jälkeen virhettä ei saatu toistettua, joten sitä ei voitu kirjata virheelliseksi testitapaukseksi.

# 6 TULOKSET JA JOHTOPÄÄTÖKSET

Aikaisemmin käytettävyystestausta toimeksiantaja on tehnyt muutaman kerran yhteistyössä koulujen kanssa. Lapset ovat tulleet kehittäjien tiloihin pelaamaan peliä samalla, kun kehittäjät ovat seuranneet heidän toimintojansa. Mielestäni tämä keino on erityisen hyvä käytettävyystestauksessa, kun otetaan mukaan pieni otos loppukäyttäjistä. Toivottavasti tätä menetelmää tullaan vielä jatkossakin jatkamaan.

Pelissä pystyy liikkumaan nuolinäppäimillä ja hiirellä. Hiirellä valitaan kohteita eli esimerkiksi puhuttaessa henkilöille, valittaessa moottorikelkan tai menemällä pelihalliin sisälle. Hiirellä pystyy myös liikkumaan, kun klikkaa johonkin suuntaan tietyin väliajoin. Yhdellä hiiren klikkauksella henkilö liikkuu monta askelta.

Aikuisena ihmisenä, enkä pelin kohderyhmänä, minun mielipiteeni voivat vaihdella käytettävyydestä erillä tavalla. Ensimmäisinä kertoina kun aloitin testauksen, pelin käyttäminen tuntui välillä haasteelliselta ja turhauttavalta. Pelkästään hiirellä liikkuessa hahmo tuntui aina törmäilevän joka paikkaan, jatkavan kävelyään seinää pitkin tai jäävän johonkin jumiin. Aikansa nuolinäppäimillä pyöritellessä, hahmo pystyi taas liikkumaan, mutta joskus jumiutuminen aiheutti myös sen, että peli piti aloittaa uudelleen.

Mitä enemmän peliä pelasi, sitä luontevammalta liikkuminen tuntui. Oli selvästi helpompaa liikkua pelkästään nuolinäppäimillä ja ainoastaan hiirellä valita kohteita. Jokaisen pelin käytäntöjen ja liikkumisen opetteluun menee oma aikansa, niin kuin myös tässäkin.

Pelissä olevat ohjeet ja opasteet ovat ymmärrettäviä ja oikeasti neuvoa antavia. Sivuston leveys on ehkä turhan suuri. Peliä ja chattia voi olla vaikea seurata yhtä aikaa, jos omistaa pienen näytön. Selaimen sisältöä zoomatessa pelin resoluutio pysyy hyvänä, mutta painikkeet pysyvät samankokoisina ja samoilla paikoilla. Pienennettäessä sivua pelistä häviää kaikki oleelliset napit ja toimin-

not. Suurennettaessa taas pelin toiminnot kuten matkapuhelin, reppu ja hymiöt, pysyvät paikoillaan ja näin ollen toiminnot ovat keskellä peli-ikkunaa.

Pelin yhteensopivuudentestaaminen eri selaimilla jäi vähäiseksi. Peli ei toimi kokonaan kaikilla selaimilla. Minipelien pelaaminen onnistuu ainoastaan Google Chrome -selaimella, joka on myös kehittäjien tiedossa. Lähitulevaisuudessa olisi hyvä paneutua myös yhteensopivuustestaukseen.

Tietoturva on myös oleellinen asia tässä pelissä, koska pelaajat rekisteröityvät ja antavat omia tietojaan. Rekisteröitynyt käyttäjä pääsee pelaamaan omilla tunnuksilla kirjautumalla sisälle ja näin ollen tulevaisuudessa pelaajien tileille tallentuu heidän toiminnot pelissä. Pelaajan kirjautuessa ulos hän olettaa, että enää kukaan muu ei pääse hänen tunnuksillaan sisälle. Näin ei kumminkaan ole, kun painamalla selaimen ”edellinen sivu” -nappia, peli alkaa uudelleen pyöriä eikä vaadi sisäänkirjautumista niin kuin sen pitäisi. Tällä hetkellä tästä ei koidu suurta vahinkoa, koska peli ei tallenna pelaajien toimintoja pelissä.

Peliä avatessa omalla koneella, lataus tuntui kestävän liian kauan. Tätä ongelmaa ei ollut kuitenkaan toimeksiantajan tiloissa, johtuen ehkä huomattavasti nopeammasta Internet-yhteydestä. Peliä aloittaessa Windowsin palomuuri kysyy pelin sallimisen, mutta jos koneella on jokin muu palomuuri, kuten F-Secure, mitään sallimisilmoitusta ei tule eikä peli käynnisty. Tähän tilanteeseen pitäisi miettiä jonkinlaista ratkaisua tai laittaa jotain ohjeisiin.

Pelin ollessa kehitysvaiheessa on erityisen tärkeää, että testausta tapahtuu koko ajan. Testitapaukset auttavat kehittäjiä tulevaisuudessa muistamaan mitä testejä tulisi korjausten jälkeenkin vielä tehdä. Korjaukset voivat myös aiheuttaa lisää virheitä ja ne voivat ilmentyä eri paikoissa mihin korjauksia on tehty.

# 7 POHDINTA

Työn tavoitteena oli suunnitella ja toteuttaa testausmenettely toimeksiantajan luomaan peliin. Opinnäytetyöprosessin alussa suunniteltiin alustavasti tietopohjaa, joka rakentuisi testauksen tarkoituksesta ja v-mallista. V-malli lähestyy prosessin näkökulmasta ja sen mukaan ennen testausta pitäisi tehdä yksityiskohtainen testaussuunnitelma, jossa kerrottaisiin mitä testataan, millä menetelmillä ja millä testitapauksilla. Testaussuunnitelma olisi ollut vaativa tehdä ja ehkä sen käyttäminen tämäntyylisessä testauksessa olisi ollut hieman hankalaa.

Prosessin edetessä tutustuin tutkivaan testaukseen, joka vaikutti hyvältä tavalta lähteä minun tapauksessani testausta tekemään. Toimeksiantajan luoma peli ei ollut minulle lainkaan tuttu ja perehdytystä ei juuri annettu. Perehdytyksen pois jättämisen syynä pidettiin sitä, että haluttiin kuulla minun mielipiteeni uutena pelin käyttäjänä. Olin kerinnyt jo suhteellisen paljon kirjoittamaan tietopohjaa v-mallista, joten sitä ei enää alettu ottamaan pois. Eräiden lähteiden mukaan tutkivan testauksen ja v-mallin yhdistäminen on jopa hyödyllistä.

Tutkivan testauksen ja v-mallin yhdistämistä pohdittiin ja tultiin siihen tulokseen, että jos vertailtaisiin näitä testaustapoja toisiinsa. Kiireisen aikataulun vuoksi vertaileminen jätettiin pois ja sitä myötä myös testaussuunnitelman laadinta. Testaukseen lähdettiin paneutumaan tutkivan testauksen teorian mukaan eli oppimaan pelin käyttöä, kirjoittamaan muistiinpanoja ja luomaan testitapauksia.

Testaus tapahtui päänsääntöisesti omalla koneellani. Alussa kävin toimeksiantajan tiloissa tekemässä testausta, mutta yhdessä tultiin siihen tulokseen että se olisi helpompaa tehdä kotona. Näin säästin ajassa ja sain tehdä testausta silloin kun minulle sopi eikä toimeksiantajan aikataulun mukaan. Ongelmia syntyi heti, kun yritin ensimmäistä kertaa pelata kotona. Peli ei käynnistynyt, joten kysyin apua toimeksiantajalta, mutta siellä epäiltiin että palvelin oli ollut vain poissa käytöstä. Lopulta tultiin siihen tulokseen, että palomuuri esti pääsyn peliin. Tämä hieman hidasti testauksen aloittamista.

Päästessäni kotona testaamaan, työ sujui paljon helpommin. Testitapausten kirjoittaminen tuotti välillä päänvaivaa ja vaikeuksia, kun kyseessä ei ollut mikään tavallinen järjestelmä. Vähitellen testitapauksia alkoi syntyä ja samalla mielipiteitä käytettävyydestä kehittyä. Yritin testausprosessin aikana ottaa huomioon v-mallissa ilmenneitä testauksen lajeja ja löytää niiden ohjeiden mukaan uusia virheitä.

Virheiden etsiminen eli pelin pelaaminen oli koko opinnäytetyöprosessin mielekkäintä työtä. Testauksessa saattoi vierähtää parikin tuntia huomaamatta ajankulua ja sitä mitä pelistä oikeasti etsin. Testauksen jälkeen oli jouheaa kirjoittaa raporttia, kun olin ahkerasti kirjannut muistiinpanoja ja tiesin mistä ja mitä kirjoittaisin. Olin todella iloinen, että sain opinnäytetyöaiheekseni näin hauskan ja mielenkiintoisen työn. Prosessin aikana kiinnostuin koko ajan enemmän testauksesta ja sen menetelmistä, jonka myötä sain uutta potkua opiskelulle.

Aikatauluni oli todella tiukka ja kiireinen. En aina ollut varma aiheesta tai tekemisistäni, jolloin epävarmuus iski ja sen myötä myös luovuttamisen tunne. Sain sitten onneksi opastusta ja iskettyä motivaatiota itseeni, jotta työ taas lähti vauhtiin. Työ eteni ripeästi ja olisin ehkä toivonut enemmän aikaa, jotta olisin voinut paneutua paremmin työhön, mutta oma aikatauluni ei antanut myöden. Jos jotain tekisin toisin, niin se olisi paneutuminen tietoperustaan. Paneuduin omasta mielestäni liikaa tietoperustaan ja varsinkin v-malliin, jolloin toiminnallinen osuus jäi vähäisemmälle osalle. Liiallinen paneutuminen tietoperustaan johtui enimmäkseen omasta uteliaisuudesta ja kiinnostuksesta aihetta kohtaan, jonka vuoksi en osannut tehdä tarpeellista rajausta aiheelleni. Olen silti ylpeä saavutuksestani, vaikka aika sääteli sitä paljolti.

Toimeksiantajani voisi jatkossa jatkaa yhteistyötä opiskelijoiden kanssa. Omaan opinnäytetyöhöni perustuen toinen opiskelija voisi tehdä opinnäytetyön testauksesta, vaikka suorituskyvystä tai tietoturvasta, tai yhdistelemällä jotain muita testausmenetelmiä. Näihin osa-alueisiin syventyessä opiskelijalta pitäisi vaatia tietämystä ja mielenkiintoa aiheesta tai sitten läheistä yhteistoimintaa toimeksiantajan yhteistyökumppanin kanssa. Toinen yhteistyömahdollisuus voisi olla monikäyttäjätestausmenetelmän kokeileminen koulun testauksen opintojaksolla.

Testauksen opintojaksolla jokainen opiskelija voisi yhtä aikaa testata peliä ja pelialustaa. Toimeksiantajani on myös hyvä jatkaa testausta pelin kohderyhmän eli lasten kanssa. Tähän asti lasten kanssa yhteistyö on ollut käytettävyystestausta toimeksiantajan tiloissa, mutta tulevaisuudessa lapset voisivat myös yrittää testata kotona yksikseen.

# LÄHTEET


Bach, J. 2003. Exploratory Testing Explained. Hakupäivä 10.4.2011 http://www.satisfice.com/articles/et-article.pdf.

Black, R. 2007. Pragmatic Software Testing. Indianapolis: Wiley Publishing, Inc.

Burnstein, I. 2003. Practical Software Testing. New York: Springer-Verlag, Inc.

Conformiq Software Ltd. 2006. Kuormitustestaus. Hakupäivä 9.3.2011 http://users.jyu.fi/~kolli/testaus2006/materiaali/6.3_Kuormitustestaus_v1.pdf.

Crispin, L. & Gregory, J. 2009. Agile Testing. Boston: Pearson Education, Inc.

Fantastec Oy. 2010. Mikä PolarHeroes? Hakupäivä 8.4.2011 http://www.polarheroes.com/info/parents.

Haikala, I. & Märijärvi, J. 2006. Ohjelmistotuotanto. Hämeenlinna: Karisto Oy.

HiQ. 2011. Kuormitustestaus. Hakupäivä 9.4.2011 http://www.hiq.se/fi/Tata-tarjoamme/Palvelut/Laadunvarmistuspalvelut/Testauksen-verifiointi--ja-validointipalvelut-/Ei-toiminnallinen-testaus/Suorituskykytestaus--/Kuormitustestaus/.

Jenkins, N. 2008. A Software Testing Primer. Hakupäivä 8.4.2011 http://www.nickjenkins.net/prose/testingPrimer.pdf.

Juvonen, J. 20011. ”Naisia vain tuotepakkausten kansissa”. Markkinointi&Mainonta 18.2.2011, 28.

Kohl, J. 2007. Man and Machine. Hakupäivä 10.4.2011 http://www.stickyminds.com/BetterSoftware/magazine.asp?fn=cifea&id=103.

Microsoft. 2003. Designing Test Cases. Hakupäivä 5.5.2011 http://technet.microsoft.com/en-us/library/cc782852(WS.10).aspx.

Perry, W. E. 2006. Effective Methods for Software Testing. Indianapolis: Wiley publishing, Inc.

Pressman, R. S. 2005. Software engineering; A Practitioner's Approach. New York: McGraw-Hill.

Räsänen, S. 2010. Ohjelmiston testaus ja laatu. Hakupäivä 29.3.2011 http://webd.savonia.fi/home/ktrasse/muut/testaus_laatu/testaus_3.pdf.

Taina, J. 2004. Ohjelmistojen testaus, syksy 2004 – luentomateriaali (testaus-prosessi). Hakupäivä 29.3.2011 http://www.cs.helsinki.fi/u/taina/ohte/s-2004/luennot/.

Toroi, T. 2005. Tietojärjestelmän testaus. Hakupäivä 29.3.2011 http://his.uku.fi/avointa/julkaisut/TestausluentoESHI111005.ppt.