KEESING GAMES

HOGESCHOOL UTRECHT

Quality Assurance

Het invoeren van testing bij Keesing Games

Geschreven door:	Ashley Warren
Studentnummer:	1518188
Cursus:	TCMT-AFST-09
Bedrijf:	Keesing Games
Datum:	03-08-2011

QUALITY ASSURANCE

Het invoeren van testing bij Keesing Games

Afstudeerscriptie

Verslag van de resultaten van mijn afstudeerstage bij Keesing Games te Hilversum in de periode februari 2011 – augustus 2011.

Gemaakt in opdracht van:
Hogeschool Utrecht,
Opleiding Mediatechnologie,
TCMT-AFST-09, Afstudeerproject.

Onder begeleiding van:
Cees de Waard (stagedocent)
Misja van Laatum (bedrijfsbegeleider)
Te Hilversum, 3 augustus, 2011

Ashley Warren, 1518188
Aantal woorden ~41.000 excl. bijlage
Versie 1.0

Samenvatting

Deze scriptie bevat het onderzoek en de resultaten van het afstudeerproject die uitgevoerd is bij en voor Keesing Games te Hilversum. Het betreft de periode van februari tot en met augustus 2011.

Keesing Games is een game developer dat onderdeel uitmaakt van de Telegraaf Media Groep. Het bedrijf heeft het probleem dat hun producten niet of nauwelijks getest worden. Deze scriptie bevat het onderzoek die verricht is die als doel heeft het overdragen van kennis en procedures betreffende het vakgebied testing voor het platform Yezzle.nl. De hoofdvraag van dit onderzoek luid:

- *Hoe kan Keesing Games de kwaliteitsbewaking verbeteren voor Yezzle?*

Het onderzoek is opgedeeld in twee delen, één deel dat gericht is op de playability van games (in essentie het leuk maken van games) en het ander deel is gericht op kwaliteitsbewaking.

Het playability onderzoek is in drieën gedeeld. De drie onderdelen zijn:

- Algemene informatie over casual game design. Waarbij de volgende onderdelen besproken worden: playtesting, player archetypes, casual level design en achievement design.
- Twee case studies van de populaire casual games Bejeweled Blitz en Microsoft Solitaire.
- Playtesten van de braintraining games van Yezzle.nl

De onderzoeksresultaten kunnen gebruikt worden om de kwaliteit van de speelbaarheid van de games van Yezzle.nl te verbeteren. Aan de hand van dit onderzoek is er geconstateerd dat er binnen Keesing Games *niet* genoeg aandacht besteed wordt aan het *fun* maken van hun games. Daarom is er een aanbeveling gemaakt om een game designer in te huren die fulltime bezig gaat met het *fun* maken van de games van Keesing Games.

Het tweede deel van het onderzoek is gericht op de verschillende processen, technieken en methodieken die er bij software testing komen kijken. Het bespreekt de tekortkomingen van het ontwikkelingsproces van Yezzle, daarbij worden er oplossingen of aanbevelingen gedaan naar onderwerpen als:

- Het OTAP proces
    - OTAP, staat voor ontwikkeling, test, acceptatie en productie. Het is de pad waar de ontwikkeling van software doorheen gaat. Deze scriptie beschrijft hoe dit proces efficiënt gebruikt kan worden.
- Bug reporting
    - Er ontstaan veel miscommunicaties en bugs doordat men niet effectief bugs weet te omschrijven. Hoe meer informatie over een bug gegeven kan worden, hoe minder tijd het zal kosten om de bug op te sporen en te fixen.
- Test documentatie
    - De verschillende type test documentatie wordt omschreven. Test documentatie bevordert de helderheid van de stand van zaken en creëert aansprakelijkheid.
- Bug tracking software als project management tool
    - Het is mogelijk om Mantis (bug tracking software) toe te passen als een project management tool. Wanneer men goed documenteert in Mantis dan wordt de database een centraal punt waarbij iedereen kan zien wie waar mee bezig is en welke deadlines er zijn.
- Black-box testing
    - Black-box testing is het testen van de software zonder dat men toegang heeft tot de code. Het is onder te verdelen in static en dynamic black-box testing, waarbij static testing het testen van de documentatie is en dynamic testing is het testen van het programma.

Het doel van dit document is om de werknemers de inzichten, technieken, methodieken en processen van het vak software testing mee te geven. Hiernaast wordt er de aanbevelingen gedaan om meer te gaan documenteren. Pas wanneer er vooraf het ontwikkelen helder en specifiek gedocumenteerd wordt kan het vak testen tot zijn volle potentie komen en ook echt een positief verschil in kwaliteit met zich mee brengen.

Inhoud

Voorwoord

Dit document en zijn bijlage vormen de scriptie van mijn afstudeerstage voor de opleiding Mediatechnologie aan de Hogeschool Utrecht. Deze afstudeerstage heeft plaatsgevonden bij Keesing Games te Hilversum.

Het was een moeizaam proces om er achter te komen welke richting ik wilde voortzetten binnen Mediatechnologie. Toen had ik het ineens, ik wil een game / software tester worden. En gelukkig kreeg ik van Keesing Games de mogelijkheid om mij hier in te verdiepen. Mijn dank hiervoor is erg groot!

Het was een erg leerzame semester, ik heb een hoop nieuwe informatie tot mij genomen wat zeer relevant is voor mijn toekomst.

Ik wil hierbij iedereen van Keesing Games, in het bijzonder de Yezzle crew, bedanken voor hun begeleiding en hun gezelligheid.

Ashley Warren,
Hilversum, 3 augustus 2011

Inleiding

Keesing Games is een bedrijf dat onderdeel uitmaakt van het “Telegraaf Media Groep” (TMG) gevestigd op het mediapark in Hilversum. TMG is aan het zoeken naar andere mogelijkheden om de steeds verminderende hoeveelheid mensen die de traditionele krant lezen op te vangen. Daarom wordt er nu hard gewerkt aan het uitbreiden van de producten en diensten van TMG naar verschillende digitale media. Zo heeft TMG de stap genomen om sociaal netwerk Hyves over te kopen. En is Keesing Games in opdracht van TMG bezig om verschillende gaming ervaringen neer te zetten op het web en op verschillende mobiele platformen.

Keesing Games is hard bezig met het creëren van verschillende speel ervaringen voor verschillende doelgroepen, waarbij alle projecten wel één overeenkomst hebben, namelijk de projecten zijn allemaal gericht op de casual gamer. De casual gaming industrie is in de laatste jaren een serieus tak geworden van de traditionele gaming industrie, waarbij er tegenwoordig enorm veel geld te verdienen valt. Dus met het bereik van de Telegraaf Media Groep achter zich hoopt Keesing Games bestaande merken als Denksport en Bingo als digitaal spelervaring neer te kunnen zetten.

Het creëren van digitale games kan een langdradig en complex proces zijn. En om er voor te zorgen dat de games ook nog eens gespeeld worden door betalende klanten is het essentieel dat de kwaliteit hiervan erg hoog is. Het ontwikkelingsproces van Keesing Games mist een belangrijk schakel, namelijk het controleren van de kwaliteit van de games op fouten en speelbaarheid. En hier wil Keesing Games dus iets aan doen. De procedure voor het testen op kwaliteit van de games moet neergezet worden.

Bij het opstellen van de test procedures is het ook belangrijk dat kennis over testing wordt overgedragen aan de werknemers van Keesing Games. Daarom wordt er naast het opstellen van de test procedures ook gekeken naar het verkrijgen en overdragen van kennis van testing en playability.

In dit document zal het verloop van het onderzoek en de resultaten hiervan beschreven worden. Het zal eerst beginnen met algemene informatie en de probleemstelling waarna de verschillende tekortkomingen van Yezzle besproken worden en hoe deze opgelost kunnen worden. Tot slot worden de resultaten van deze fases besproken waarbij er conclusies getrokken worden en/of aanbevelingen gegeven zullen worden.

1 Organisatie en werkwijze

1.1 Keesing Games

Keesing Games is een casual game developer, bestaande uit ongeveer 35 mensen, wat onderdeel uitmaakt van het Telegraaf Media Groep, gevestigd in Hilversum. Het is een bedrijf dat zich richt op het maken van online puzzel gaming platformen en de puzzel / skilll games die daarbij horen. Ze hebben als core business dus game development.

Keesing Games heeft op het moment 5 projecten lopende.

- Ziggiz.com
  - Een online spelletjes portal
- Yezzle.nl
  - Het afstudeerproject zal zich voornamelijk richten op Yezzle.nl. Dit is een spelletjes platform die gericht is op braintraining spelletjes.
- Bingo.nl
  - Dit wordt een online variant van het spel Bingo
- Stratego
  - Dit wordt een online variant van het bordspel Stratego.
- Trivialis
  - Hier wordt nog geen informatie over vrijgegeven

Elk van deze projecten heeft zijn eigen projectgroep, die voornamelijk gericht is op het realiseren van het desbetreffende product. Hierbij is het wel zo dat er externe hulp is ingehuurd voor sommige projecten. Zo besteed “team yezzle” een deel van haar taken uit naar een bedrijf genaamd NinTec, en wordt Bingo.nl technisch gerealiseerd door een Nederlands bedrijf genaamd WhiteBear Studios.

1.2 Yezzle.nl

Alle op te leveren eindproducten worden gemaakt voor Yezzle.nl. Dit is een online spelletjesportaal, waar op het moment 11 mensen van Keesing Games aan werken. Wat Yezzle onderscheid van andere spelletjesportalen zoals spelletjes.nl, is dat het zich volledig richt op de zogenoemde “Braintraining” spelletjes. Dit zijn spelletjes waar de nadruk niet op skill ligt, zoals bij de meeste games, maar op het gebruik van je hersenen. Het zijn gewoonlijk puzzels waarbij je hard na moet denken wil je deze oplossen. De website biedt op het moment 32 spelletjes aan, waaronder de bekende woordzoekers van het “Denksport” merk. Deze spelletjes worden opgedeeld in vijf verschillende competenties:

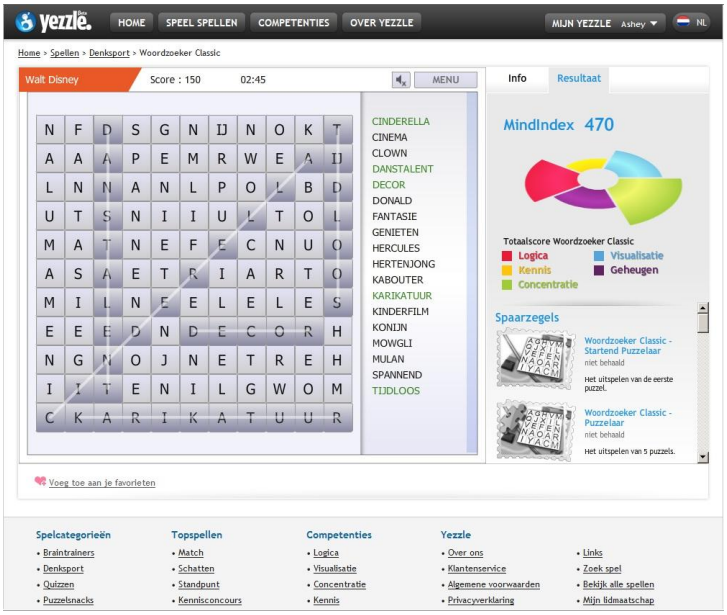
- Logica
- Visualisatie
- Concentratie
- Kennis
- Geheugen

Figuur 1-1 Voorpagina van Yezzle.nl

Waarbij in de zogenoemde “MindIndex” deze competenties worden bijgehouden. De MindIndex is een score waaraan je kunt zien hoe goed je bezig bent. Je kunt er precies zien wat je score is per competentie en hieruit kan je concluderen welke competenties je goed of minder goed in bent. Zo kun je je juist richten op de spelletjes die goed zijn voor het trainen van een competentie waar je niet goed in bent, om zo je hersenen goed en volledig te “trainen”. Om dit proces te verbeteren zijn er trainingsprogramma’s waar je elke dag specifieke games moet spelen.

Je kunt twee weken lang Yezzle.nl gratis uitproberen. Wil je de games na deze periode blijven spelen dan zijn er verschillende abonnementen waaruit gekozen kunnen worden. 4.95 euro voor een maand, 13.95 euro voor drie maanden, 24.95 euro voor zes maanden, 39.95 voor een jaar abonnement. Voor dit geld kan je ongelimiteerd de spelletjes spelen en krijg je voor de Denksport games elke dag drie nieuwe puzzels.

Keesing Games gaat in de toekomstig regelmatig nieuwe games toevoegen aan Yezzle en wordt er gesproken over het “sociaal” maken van Yezzle. Ook kan er in de toekomst prijzen gewonnen worden door de mensen die hun spaarboekjes vol spelen. Het spaarboekje is een collectie van achievements die te behalen zijn.



Figuur 1-2 Woordenzoeker van Yezzle.nl

1.3 Organisatie

Bedrijf

Opdrachtgever:	Keesing Games
Straat + huisnr.:	Sumatralaan 45
Postcode + Plaats:	1217GP, Hilversum
Tel. Nummer:	00 31 35 677 51 79

Student

Naam:	Ashley Warren
Studentnr.:	1518188
Tel. Nummer:	06-15367669
Rol:	Afstudeerder

Begeleiding

Stagedocent:	Cees de Waard
Bedrijfsbegeleider:	Misja van Laatum

1.4 Werkwijze

Werktijden:	8:30 – 17:00
Stagevergaderingen:	Wekelijks op vrijdag om 11:00
Werkoverleg:	Wekelijks op donderdag om 14:00

2 Project / Opdracht

2.1 Probleemstelling

Keesing Games is bezig met verschillende projecten, waarbij delen van sommige projecten ge-outsourced worden naar verschillende bedrijven. Bij Yezzle, het project waar het onderzoek mee te maken heeft, wordt een deel van de taken in India uitgevoerd. Door het gebrek aan testing en de verspreiding van de taken wordt het voor Keesing Games erg lastig om de kwaliteit van de producten te kunnen waarborgen. Het bedrijf is nu zo aan het groeien dat ze er niet meer mee weg kunnen komen om niet aan Quality Assurance(QA) / Testing te doen. Er is dus iemand nodig die dit proces gaat implementeren voor Keesing Games.

2.2 Onderzoeksvragen

Gedurende deze afstudeerstage werden er enkele vragen gehanteerd als richtlijn voor het onderzoek. Het beantwoorden van deze vragen stond centraal tijdens de stageperiode.

2.2.1 Hoofdvraag

Om het probleem van Keesing Games op te lossen, is de volgende vraag als hoofdvraag van het onderzoek opgesteld:

- *Hoe kan Keesing Games de kwaliteitsbewaking verbeteren voor Yezzle?*

Omdat deze vraag op veel verschillende manieren te beantwoorden valt, zijn er ook deelvragen opgesteld die het onderzoek specifiek en gericht houden.

2.2.2 Deelvragen

- *Deelvraag 1: Hoe zou een effectief testplan eruit zien voor de in-house development?*

Het zou niet mogelijk zijn om binnen één semester alles over quality assurance te onderzoeken en te implementeren voor Keesing Games. Hierdoor heeft het project zich gericht op het neerzetten van een testplan voor Yezzle die als handboek kan dienen om de website en games te testen.

- *Deelvraag 2: Hoe zou je de playability en replayability van Yezzle kunnen verbeteren?*

Naast het focussen op het maken van een testplan voor Yezzle, is er ook gekeken worden naar manieren om de speelbaarbaarheid van de games van Yezzle te verbeteren met het oog op de doelgroep van Yezzle.

Door middel van het beantwoorden van deze deelvragen is er een concreet product uit het onderzoek gekomen die als antwoord op de hoofdvraag dient.

2.2.3 Verworpen deelvraag

Al snel werd het duidelijk dat niet alle problemen aangepakt konden worden binnen één semester daarom is er besloten om het onderzoek voornamelijk op het in-house development te richten. Dit betekende dat de volgende deelvraag verworpen werd:

- *Hoe zou een effectief testplan eruit zien voor de outsource development?*

2.3 Eindproducten

Het op te leveren eindresultaat voor Keesing Games zal uit de volgende producten bestaan:

- Een rapport over de verschillende methodieken en technieken die gebruikt worden voor software / game testing
- Een rapport over verhogen van user experience aan de hand van case studies naar andere concurrerende games
- Een testplan voor het testen van Yezzle
- Het testen van Yezzle

3 Analyse

In dit hoofdstuk wordt de probleemstelling verder uitgestippeld om een beter beeld te krijgen van wat precies het probleem is en hoe dit opgelost gaat worden. Er wordt een overzicht gegeven van wat er gaande is binnen KGG, er wordt dieper gekeken naar waar het precies misgaat in het development proces en tot slot wordt er besproken hoe dit probleem aangepakt gaat worden.

3.1 Overzicht

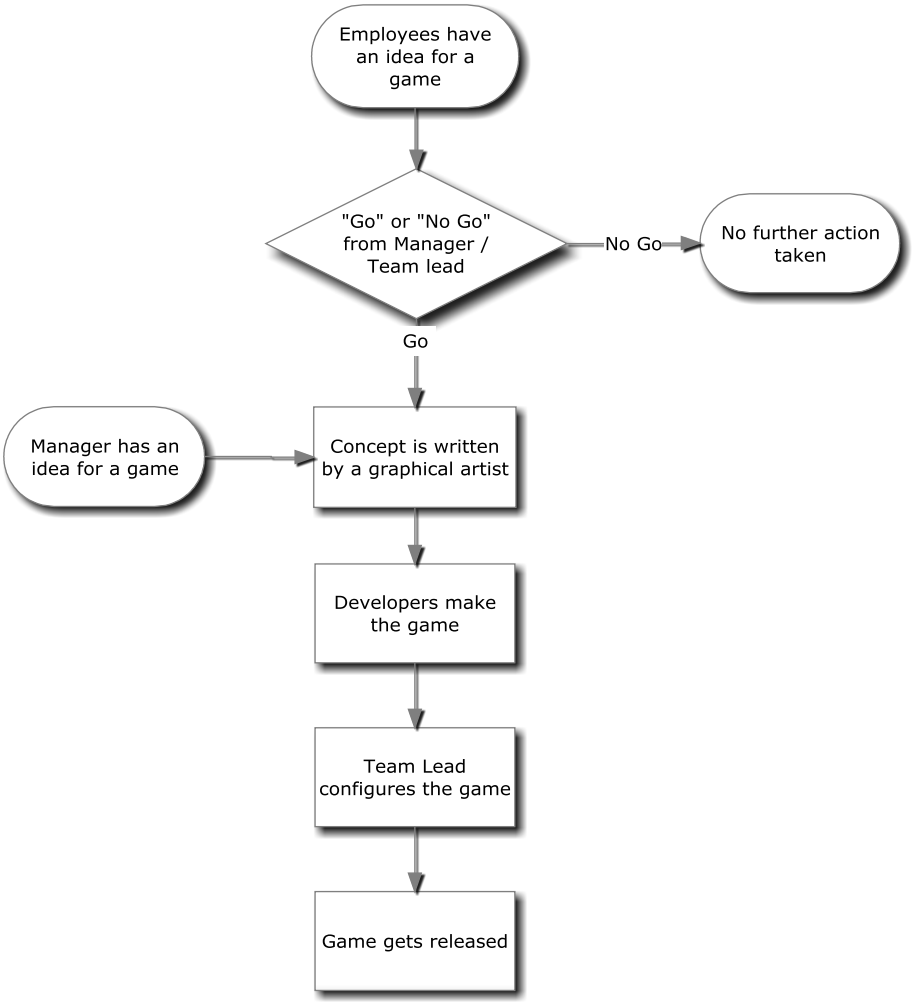
Keesing games had toen het project begon geen gespecialiseerde software testers in dienst. Het is zo dat er door de developers zelf getest wordt en wordt er eens in de zoveel tijd aan collega's gevraagd, die bezig zijn met andere projecten, of ze even de tijd zouden willen nemen om het product te testen en de gevonden problemen door te sturen via e-mail. Dit is niet noodzakelijk een slechte zaak, sterker nog het is juist slim om eens in de zoveel tijd een verse blik op een product te werpen. Maar het is wel zo dat door zulke test sessies alleen de globale bugs gevonden worden. Grote problemen zoals memory leaks, waar je pas na lang spelen achter kan komen, zullen niet gevonden worden. En hier komt dan ook bij dat zo een test sessie de werknemers van hun eigen werk afhoudt.

Wat wel gedaan wordt bij Keesing Games is het inhuren van het bedrijf "Jungle Minds" te Amsterdam om de zogenoemde User Acceptance Test (UAT) uit te voeren. Dit is een uitgebreid test sessie waarbij mensen van de doelgroep uitgenodigd worden om het product uit te proberen. Een UAT houdt zich niet zozeer bezig met de vraag hoe goed het werkt maar meer met de vraag: Hoe leuk vindt men het? Een UAT is ook voor Yezzle gehouden, de resultaten vertelde Keesing Games dat de doelgroep wel interesse Yezzle heeft, maar dat er nog veel aanpassingen gemaakt moeten worden. Dit is een grote reden geweest voor het uitstellen van de release van Yezzle (van februari naar mei).

De UAT heeft Keesing Games veel geleerd over hun doelgroep, en het is dan ook een erg effectieve instrument om de populariteit van een concept te meten voordat het uit is gebracht. Wat er echter niet gedaan is tijdens de UAT is het controleren of de games leuk zijn om te spelen en of de website geen fouten bevat, de test bleef namelijk alleen op concept niveau. Daarbij komt ook nog kijken dat een user acceptance test erg duur is, daardoor is het ook belangrijk dat het product pas door een UAT gaat wanneer deze al tamelijk compleet en bug vrij is. Om dit te bereiken is het dus nodig dat er onderzoek wordt gedaan naar het vak game / software testing, om hoger kwaliteit te kunnen leveren. Binnen Keesing Games is er namelijk maar gelimiteerde test kennis aanwezig, het doel van dit onderzoek is dan ook het overdragen van kennis en procedures betreffende testing en kwaliteit waarborging.

3.2 Analyse

Er is maar weinig zicht op de kwaliteit van het platform en de games van Yezzle. Hierdoor weet men niet of wat er gemaakt wordt ook daadwerkelijk goed werkt en daarbij ook nog eens leuk is om te spelen. Het gevolg hiervan is dat de (betalende) gebruiker als eerste geconfronteerd wordt met bugs of slechte gameplay in plaats van de developers. Daarom moet testen en playtesten meegenomen worden in de ontwikkeling van Yezzle. De huidige development proces van Yezzle ziet er als volgt uit:



Figuur 3-1 Huidige ontwikkelingsproces van Yezzle

In deze flow is te zien dat testen van de producten niet worden meegenomen in het ontwikkel proces. Er moet dus ruimte gemaakt worden binnen dit proces voor kwaliteitsbewaking. Om dit te bereiken moeten de werknemers het belang van het vak testen gaan inzien en daarbij kennis over testen zien te vergaren.

Er zullen meerdere oplossingen moeten komen om de problemen voor Yezzle op te lossen. De resultaten van dit afstudeerproject zal het volgende te bieden hebben:

- Kennis over testing overhandigen in de vorm van een samenvatting van verschillende aspecten van het vakgebied testen.
    - Hiermee elimineert men het gebrek aan kennis over testen.
- Informatie en kennis over fun in casual games overhandigen in de vorm van een rapport waarin aspecten als playtesting, achievement design, manieren van spelregels uitleggen, case studies naar succesvolle casual games te vinden zijn en worden de Braintrainers onder handen genomen met adviezen over hoe de games te verbeteren zijn.
    - Hiermee wordt er meer nadruk gelegd op het niet alleen werkend krijgen van de games maar ook het *fun* maken hiervan.
- Een testplan waarin het proces omschreven staat om kwaliteitsbewaking mee te nemen in de development cyclus van Yezzle.
    - Het volgen van dit document moet er voor zorgen dat er duidelijkheid is betreffende wie, wat, wanneer en hoe er getest moet worden.

3.3 Methodiek

De opdracht is op te delen in verschillende onderdelen, waarbij elk onderdeel anders is aangepakt. De onderdelen zijn als volgt:

1. Onderzoek naar QA / Game Testing
2. Onderzoek naar het verbeteren van de fun en playabilty van de games op Yezzle
3. Het opstellen van testprocedures in de vorm van een testplan

Elk onderdeel vereiste een ander aanpak om tot een goed eindresultaat te komen. Deze verschillende methodieken van realiseren worden in het vervolg van dit hoofdstuk besproken.

3.3.1 Onderzoek naar QA / Game Testing

Het onderzoek naar het vakgebied testing is uitgevoerd aan de hand van een literatuurstudie. De bestudeerde literatuur omvat drie boeken en verschillende artikelen geschreven door mensen uit het vakgebied software testing.

Er is begonnen met het doornemen van het boek *"Software Testing – Second Edition"* geschreven door Ron Patton. Dit boek geeft een goed globale beeld van software testing, maar gaat verder ook gedetailleerd in op bepaalde aspecten van het testen van software. Vervolgens is het boek *"Game Development Essentials: Game QA & Testing"* gelezen van de auteurs Luis Levy en Jeannie Novak. Dit boek geeft een erg globale kijk op de wereld van game testing. Tot slot is voor dit onderzoek het boek *"Game Testing All In One"* gelezen van de schrijvers Charles R. Schultz, Robert Bryant en Tim Langdell (Ph.D). Dit boek illustreert wat er allemaal moet gebeuren om een game te maken en wat de rol van een game tester hierin is, vervolgens gaat het de diepte in over alles wat je moet weten als game tester. Naast deze drie boeken zijn er verschillende artikelen bestudeerd van mensen uit het vakgebied en is er informatie vergaard van een online forum voor software testers.[1]

Na de boeken en artikelen bestudeerd hebben is er een samenvatting gemaakt van de informatie die toepasbaar was voor Keesing Games. Dit product is puur een samenvatting van het vakgebied testen en is verder niet in relatie gebracht met Yezzle. Het behandeld onderwerpen als:

- Test documentatie
- Black-box testing
- White-box testing
- Localisation testing
- Bug reporting
- Etc.

In dit document zijn ook resultaten te vinden van het onderzoek naar game testing in het specifiek. Per gaming genre is er onderzocht waar er extra opgelet moet worden voor het testen van een game binnen een specifiek genre. Ook zijn er veel verschillende games gespeeld om te kijken wat de meest voorkomende bugs zijn en onder welke categorie deze vallen. De resultaten zijn ook in de samenvatting te vinden.
Het doel van dit document is het bijbrengen van kennis over het vak testing aan de werknemers van Keesing Games.

[1] Zie http://forums.utest.com

3.3.2 Onderzoek naar fun en playability in casual games

Om op een feitelijke manier aanbevelingen te kunnen maken over de speelbaarbaarheid van de Yezzle games is het van belang om op een gestructureerd manier onderzoek te doen. Er is voor dit onderzoek besloten om het in tweeën te delen, namelijk als eerst een deel onderzoek naar casual gaming in het algemeen, en vervolgens een empirisch onderzoek waarvoor er case studies zijn uitgevoerd. Bij deze case studies is er gekeken naar twee succesvolle casual games en is er een case studie uitgevoerd naar de Braintrainers van Yezzle.nl waarbij er naar de leercurve, moeilijkheid en de progressie gekeken is.

Het eerste deel bestaat uit een literatuuronderzoek waarbij artikelen van game developers bestudeerd zijn en het boek "*Casual Game Design: Designing play for the gamer in all of us"* geschreven door Gregory Trefay bestudeerd is, om er zo achter te komen wat de belangrijke elementen zijn voor het succesvol designen van casual games. Dit is nodig om op een hoger niveau te kunnen gaan kijken naar de games van Yezzle. Voor dit onderdeel is er besloten om hoofdzakelijk naar 4 elementen van casual game design te kijken:

- Playtesting
- Casual level design
- Het uitleggen van de spelregels
- Achievements

Het empirisch deel van het onderzoek bestaat uit het onderzoeken van andere populaire casual games. Dit is besloten omdat er een hoop te leren valt van bestaande producten. Het kunnen begrijpen wat een goede casual game, een goede casual maakt, is waardevolle informatie. Dit kan vervolgens weer vertaald kan worden naar het verbeteren van de huidige en toekomstige games van Yezzle. Om achter deze "waardevol" informatie te komen is er besloten om naar de grootste casual games van het verleden en nu te kijken, Microsoft Solitaire en Bejeweled Blitz. Voor beide games wordt er de reverse engineering van de game mechanic gegeven. Tot slot wordt er gekeken wat nou de elementen zijn die ervoor gezorgd hebben dat deze games zo succesvol geworden zijn.
Aan de hand van deze handelingen moet het mogelijk worden om een zo objectief mogelijk antwoord te kunnen geven op de deelvraag: "*Hoe zou je de playability en replayability van Yezzle kunnen verbeteren?"*

3.3.3 Testplan

Het opstellen van het testplan wordt volledig geproduceerd aan de hand van het vooronderzoek over game / software testing en de gemaakte observaties van zwakte punten binnen het Yezzle projectteam.
Wanneer er fouten in het productie proces gevonden worden, kan er gekeken worden of er hier een oplossing voor te vinden is binnen het kader van testing. Daarbij zal het globale onderzoek naar software / game testing gebruikt worden, waarna er een concreet plan wordt opgesteld waarbij de te doorlopen procedures beschreven staan om zo de kwaliteit van toekomstige onderdelen van Yezzle.nl te kunnen waarborgen. Er is gekozen om niet een template te volgen van een standaard testplan, omdat het document dan niet alle informatie zou bevatten die nodig is om testing effectief in het proces te kunnen implementeren. De hoofdstukken van het testplan zijn in essentie oplossingen voor de gevonden tekortkomingen in het productieproces van Yezzle. Deze tekortkomingen worden in het volgende hoofdstuk "Ontwerp" besproken.

3.3.4 Scriptie

Het vervolg van de scriptie is als volgt opgedeeld: In het hoofdstuk "Ontwerp" worden de tekortkoming van het ontwikkelproces van Yezzle weergegeven. Één tekortkoming betreft de playability kan van zaken en zeven tekortkomingen betreffend het gebrek aan software testing. Deze scriptie dient niet alleen gezien te worden als een evaluatie van het proces en de producten. Het bevat ook veel technische en specifieke informatie die nodig is om de problemen op te lossen die in het hoofdstuk "Ontwerp" worden benoemd. Dit document is daardoor gedeeltelijk de test samenvatting, playability onderzoek en testplan in één, maar dan op een ander manier gepresenteerd.

3.4 Globale planning

Een afstudeeropdracht Mediatechnologie dient ongeveer 800 werkuren in beslag te nemen, plus 40 uur voor het afmaken van de scriptie. Om deze maatstaaf zo goed mogelijk aan te houden en om mijn eigen werktempo onder controle te houden, heb ik aan het begin van het traject een globale planning opgesteld voor deze stageperiode. Per week zullen deze taken opgesplitst worden in meer specifieke taken.

Week	Geplande activiteit
Week 1	Oriëntatie van Keesing Games en van Yezzle, opdracht formuleren
Week 2	Onderzoek naar software testing
Week 3	Onderzoek naar game testing
Week 4	Onderzoek naar game testing
Week 5	Resultaten van het onderzoek verzamelen en documenteren
Week 6	Bestaande persona analyseren en verder onderzoek doen naar de doelgroep
Week 7	Onderzoek naar fun en (re)playability in casual games
Week 8	Onderzoek en verwerking van goede en slechte voorbeelden van casual games
Week 9	Onderzoek en verwerking van goede en slechte voorbeelden van casual games
Week 10	Playability onderzoek in een rapport uitwerken
Week 11	In-house development testplan opstellen
Week 12	In-house development testplan opstellen
Week 13	In-house development testplan opstellen
Week 14	In-house development testplan opstellen
Week 15	Outsource development testplan opstellen
Week 16	Scriptie afmaken
Week 17	Outsource development testplan opstellen
Week 18	Uitvoeren van de testplannen
Week 19	Uitvoeren van de testplannen
Week 20	Presentatie voorbereiden
Week 21	Uitloop

Note: In het hoofdstuk "Proces en planning" worden strokenplanningen gegeven. Ook wordt er in dit hoofdstuk besproken hoe en waarom er niet aan deze globale planning gehouden is.

4 Ontwerp

In plaats van een ontwerp maken zoals dat gedaan wordt voor het maken van een software programma, wordt er in dit hoofdstuk de zwakte punten van het development proces van Yezzle besproken, met een focus op kwaliteitsbewaking. Aan de hand van deze problemen wordt het testplan en de case study opgesteld om hiermee advies te kunnen geven hoe de problemen eventueel op te lossen zijn. De benodigde kennis en oplossingen voor deze problemen zullen in het vervolg van dit document besproken worden en zullen in een uitgebreidere vorm aan het bedrijf overhandigd worden (QA samenvatting, case study/playability onderzoek en het testplan). De problemen worden hieronder besproken.

4.1 Wie test wat en wanneer?

Omdat er op het moment nog geen dedicated tester aanwezig is voor Yezzle, is er maar weinig duidelijkheid over wie er precies moet testen en wanneer dit moet gebeuren. Het gevolg hiervan is dat men de aanname maakt dat iemand anders het wel getest zou hebben. Doordat er meerdere werknemers zo denken komt het regelmatig voor dat er een ongetest feature of game op de website terecht komt. Doordat Yezzle.nl nu live is betekend dit dat de gebruiker als eerste de fouten ontdekken, dit laat een slechte indruk achter; wat kan betekenen dat deze gebruiker wellicht niet meer terug komt naar Yezzle.nl. Maar zelfs als er wel getest is kan het voorkomen dat een bug onopgemerkt door meerdere test rondes weet te komen, maar er moet alles aan gedaan worden om dit te minimaliseren. Daarom moet er duidelijkheid komen over wie, wat en wanneer er getest wordt zodat er geen aannames meer gemaakt kunnen worden over de kwaliteit van een product.

4.2 Waar moet er getest worden? (OTAP)

Onlangs is er een nieuwe development procedure geadopteerd voor het Yezzle projectteam, namelijk OTAP. OTAP staat voor Ontwikkel, Test, Acceptatie en Productie (in het Engels DTAP: Develop, Test, Acceptance and Production). Deze vier punten omschrijven het pad waardoor de ontwikkeling van software heen gaat. In het geval van Yezzle is er voor elk stap binnen OTAP een eigen omgeving, namelijk een website. Omdat niet iedereen te maken heeft met alle stappen is het niet voor iedereen duidelijk hoe de OTAP procedure precies in elkaar zit. Hierdoor komt het voor dat men op een verkeerde locatie aan het testen is of er wordt gevraagd om op de ontwikkel server te testen terwijl dit juist *niet* de bedoeling is.

Helderheid over de OTAP procedures zal ten goede komen van de snelheid en efficiëntie van het live krijgen van een nieuwe feature, waarbij de kwaliteit hiervan zo goed mogelijk gewaarborgd wordt.

4.3 Slechte bug rapportages

Er zijn verschillende mensen die niet door hebben hoe belangrijk het is om een bug goed te rapporteren in de bug tracking software. Het gebeurd daardoor dan ook regelmatig dat er een ticket in Mantis (bug tracking database gebruikt binnen Keesing Games) wordt ingeschoten die gewoonweg niet goed omschreven is. Zo worden er vage titels toegekend aan issues, slechte omschrijvingen gegeven en het gebeurd vaak dat er geen stappen worden gegeven om het probleem te reproduceren. Het slecht rapporteren van bugs zorgt vaak voor veel miscommunicaties, deze miscommunicaties worden vele malen verergerd doordat een groot deel van de werkzaamheden in India plaats vindt.

Voor een bug die intern opgelost kan worden zou je eventueel een bug rapport verbaal kunnen toelichten, maar deze luxe is niet beschikbaar wanneer de ontwikkeling buiten het bedrijf plaatst vind. Desondanks je in-house op te lossen bugs verbaal kan toelichten is het nog wel erg belangrijk dat men hier geen gewoonte van maakt en de bug helder en duidelijk rapporteert. Er kan namelijk veel tijd tussen het rapporteren van een bug en het fixen van een bug zitten, ook kan het voorkomen dat iemand onverwacht de bug moet fixen. Dit betekent dat het verbaal toelichten van het probleem niet meer effectief is. Men moet dus leren om heldere bug rapportages te schrijven.

4.4 Mantis als project management

Mantis is een gratis bug tracking pakket, het heeft alle basis functionaliteiten die de betaalde tegenhangers ook hebben. Deze pakketten zijn niet alleen voor bugs, het kan ook gebruikt worden als een project management tool. Het is wel zo dat andere professionele pakketten, zoals Jira, meer opties voor project management toelaten, dit wilt niet zeggen dat Mantis niet effectief te gebruiken is als hulpmiddel voor het bijhouden van alle openstaande projecten. Het gebruiken van de bug tracking database als project management tool zal het overzicht van wie waar precies mee bezig is verbeteren, waardoor er minder voortgangsmeetings gehouden hoeven te worden, dit betekent dat er meer tijd aan het ontwikkelen te besteden is.

4.5 Hoe test ik?

Het komt voor dat er voor de hand liggende bugs voorkomen terwijl deze eigenlijk door de developer zelf al gevonden moest worden. Er zijn bepaalde globale checks die door iedereen uitgevoerd kunnen worden en relatief weinig tijd kosten. Het is namelijk zo dat hoe vroeger de bug gevonden wordt, hoe makkelijker en hoe goedkoper het is om te fixen. Dus als de developers door een paar simpele testcases uit te voeren de grootste bugs er al uit weten te vissen dan komt dit ten goede van de stroomlijning van het development proces. Daarom moet men de kennis over het testen van software vergroten.

4.6 Overslaan van het verificatieproces

Een belangrijk onderdeel van het testen, wat vaak over het hoofd wordt gezien, is het verifiëren of bugs daadwerkelijk gefixt zijn en of deze fix geen omliggende elementen kapot heeft gemaakt. Het gebeurd vaak dat een developer een bug op zijn locale machine goed oplost en dit vervolgens integreert met de rest van het platform zonder te controleren of de fix bij de integratie wel correct werkt. Het toevoegen van nieuwe code kan in combinatie met oude code wellicht niet werken, of de nieuwe code kan omliggende code kapot maken. Daarom is de controle proces erg belangrijk. Een bug is pas opgelost wanneer deze geïntegreerd is op het juiste platform en gecontroleerd is of de bug daadwerkelijk opgelost is en geen andere elementen kapot heeft gemaakt. Het niet verifiëren van bug fixes kan tot grote complicaties leiden en moet daarom altijd gedaan worden. Daarom moet men op de hoogte zijn van de flow van de verificatie proces.

4.7 Te weinig documentatie

Binnen het Yezzle projectteam wordt er maar weinig gedocumenteerd, omdat men vindt dat het documenteren meer tijd kost dan het op kan leveren. Maar door niet of weinig te documenteren worden er een hoop onduidelijkheden gecreëerd, ook krijg je te maken met developers die game design keuzes gaan maken die hier niet altijd de “knowhow” of tijd voor hebben. Door weinig te documenteren wordt het ook erg lastig om mensen aansprakelijk te houden voor hun werkzaamheden.

Zodra het vak testen geïntegreerd wordt in de development cyclus van Yezzle dan is het verstandig om ook de bij behorende test documentatie hierin mee te nemen. Dit zorgt voor helderheid over wat er getest gaat worden, door wie het getest wordt, wanneer dit gedaan wordt en wat de test resultaten zijn. Zodoende kan testen effectief in de algehele planning meegenomen worden en creëert de documentatie weer aansprakelijkheid.

4.8 Games zijn niet fun genoeg

De braintraining games van Yezzle hebben de potentie om erg leuk te zijn en sommige zijn dit al, maar er zitten ook een paar Braintrainers tussen die niet goed zijn afgesteld. In sommige games zitten hele grote progressie sprongen, andere games zijn niet snel genoeg te begrijpen en dan zijn er ook games die gewoonweg te moeilijk gekalibreerd zijn. Nadat een game gemaakt is zou er meer nadruk moeten liggen op het goed afstellen van de game. Developers maken de game zoals zij het leuk vinden, dit wil niet betekenen dat het dan ook leuk is voor de doelgroep. Daarom is het noodzakelijk dat het platform en de games speciaal ontworpen en afgesteld worden voor de casual gamer, alleen is er een gebrek aan kennis van het vak game design.

5 Playability Onderzoek - De games zijn niet fun genoeg

In het hoofdstuk “Ontwerp” staat aangegeven dat er niet genoeg nadruk ligt op het juist kalibreren en *fun* krijgen van de games. Men moet zich meer gaan richten op het verhogen van het speelplezier van de gebruiker. Om dit te kunnen doen zijn er bepaalde elementen waar men van op de hoogte van moet zijn met betrekking tot casual gaming. Dit hoofdstuk bespreekt een paar van de belangrijke elementen van casual game design. Maar nu eerst een kort overzicht wat casual gaming precies is.

Casual gaming is een term dat in de laatste jaren een waar fenomeen geworden is. Het is begonnen als een deel van de gaming industrie dat gezien werd als minderwaardig. Maar in de afgelopen jaren heeft de casual gaming industrie zich bewezen als een waardig onderdeel hiervan. Een groot deel van het hervormde imago van casual gaming kwam door de komst van de Nintendo Wii. Nintendo heeft met hun Wii bewezen dat niet alleen de hard-core gamer computer games wilt spelen, maar dat ook de “normale” burger zin heeft om spelletjes te spelen. De Wii console is jaren lang uitverkocht geweest en is 86.1 miljoen keer verkocht, dit is ongeveer 33 miljoen meer in hardware sales dan de “hardcore” Microsoft X-Box 360. Naast de Wii zijn de casual social games ook steeds populairder geworden, dit is te zien aan de Facebook game “Farmville”. Zo had “Farmville” in februari 2010 meer dan 80 miljoen actieve maandelijkse gebruikers en meer dan 31 miljoen dagelijkse gebruikers. Dit is een user base waar de grootste hardcore games, zoals de “Call of Duty” franchise, niet aan kan tippen. Hierbij is het ook verrassend dat het grootste percentage casual gamers vrouwen zijn (74%).

Figuur 5-1 Meest succesvolle social game Farmville

De vraag wordt dan gesteld “Wat is het verschil tussen een casual game en een hardcore game?”. Casual games zijn videogames die gericht zijn op de massa. Casual gamers zien zichzelf niet als een gamer. Of je jezelf nou een gamer noemt of niet, iedereen zijn leven wordt voor een groot deel geleid door spelletjes. Gregory Trefry zegt in zijn boek “Casual Game Design” dat er een paar design elementen die bijna altijd aanwezig zijn in casual games:

- De regels en doelen moeten duidelijk zijn
- De speler moet een game snel kunnen beheersen
- Casual gameplay moet zich kunnen aanpassen aan de spelers´ leven en schema
- De spelconcepten “lenen” van herkenbaar inhoud en thema’s van het leven

Hieruit wordt dus duidelijk dat een casual game simpele gameplay moet hebben die in korte speelsessies gespeeld kan worden. In het vervolg van dit hoofdstuk wordt er dieper ingegaan op de design keuzes die ervoor kunnen zorgen dat een casual game een succes wordt. Daarbij worden de volgende onderwerpen besproken:

- Play testing
- Player archetypes
- Casual game design
    - Learning the rules
    - Casual level design
    - 10 casual design errors
- Achievements

Omdat er geen één juist antwoord is op de vraag “hoe maak je een game *fun*?” dienen de inzichten en informatie in dit hoofdstuk en bijlage A en B als een deel van het antwoord op de vraag.

5.1 Playtesting

5.1.1 Playtesting

Playtesting (ook wel usability testing genoemd) omschrijft het testen van een game op subjectieve kwaliteiten als balance, moeilijkheid en de "fun-factor". Het verschilt zich dan ook van het vak software testing omdat het zich niet bezig houdt met de vraag of de game werkt, maar het richt zich op hoe *goed* de game werkt. Software testing is van nature heel binair, de software werkt wel of het werkt niet. Play testing is in geen enkele manier binair, het richt zich volkomen op subjectieve zaken zoals:

- Is de game te makkelijk?
- Is de game te moeilijk?
- Is de game makkelijk om te leren?
- Is de game goed gebalanceerd?
- Is de besturing intuïtief?
- Is de interface schoon en makkelijk te navigeren?
- Is de game fun?

Het is de taak van de game designer om voor ervoor te zorgen dat deze zaken goed geregeld zijn om tot een *fun* game te komen. De game designer verzint een gameconcept en werkt dit dan uit in het zo genoemde game design document (oftwel, product specificaties), hierin staat uitvoerig beschreven wat de game precies moet doen onder alle omstandigheden en wat de "look and feel" van de game moet worden. Vervolgens gaan de developers en vormgevers hard aan de slag om de game te maken. Wanneer er een speelbaar prototype gemaakt is, dan komt de game weer terug bij de game designer om ge-playtest te worden.

> *Note: Omdat Keesing Games geen dedicated game designers heeft, valt de taak van play testing in de handen van de persoon die de game concept uitgeschreven heeft. Het play testen van games zou ook opgevangen kunnen worden door een tester die gevoel heeft voor usability en balancing.*

De game designer test elke iteratie van een game, en past de regels en/of waardes aan om er zo voor te zorgen dat de doelgroep de game leuk vind om te spelen, dit is een onderdeel van playtesting. Naast het zelf playtesten van een game is het belangrijk dat de game designer ook andere mensen de game laat playtesten.

Laat collega's de game spelen, vraag aan vrienden en familie of zij bereid zijn te testen, kijk buiten het bedrijf en vriendenkring voor playtesters. Zorg voor genoeg testers die zichzelf niet als een gamer beschouwen. Vraag hun of zij bereid zijn langs te komen om te testen in ruil voor een pizza of iets dergelijks. Of, als er tijd is, ga langs bij deze mensen en observeer ze terwijl ze aan het spelen zijn. Zorg ervoor dat je alleen laat weten wat ze moeten weten over de game, beïnvloed de testers op geen enkel manier, geef ze geen hint als zij vast komen te zitten. Wanneer zij vragen hebben probeer dit op een beleefde manier niet te beantwoorden. Pas ook op met de meningen van collega's, familie en vrienden. Zij zullen sneller dan een vreemde geneigd zijn om te zeggen dat de game leuk is en goed werkt. Familie en vrienden worden beïnvloed door een persoonlijke relatie en dus geen negatieve feedback willen geven. Collega's weten hoe de game hoort te werken en zien daardoor te snel fun in de game, daarnaast willen zij dat de game een succes wordt. Zorg er daarom voor dat de playtesters er bewust van zijn dat zij hun eerlijke mening moeten laten weten, ongeacht hoe positief of negatief. Hoe verder iemand van jou verwijderd is hoe eerlijker en hoe beter zijn feedback wordt.

Het is een misconceptie dat playtesten alleen op het einde van de ontwikkeling van een product plaats kan vinden. Wanneer een game nog niet alle gameplay elementen bevat doordat het in een vroeg stadium van ontwikkeling is, dan moet de playtest uitvoerig gestructureerd worden. Schrijf een script uit voor de test. Zorg ervoor dat de speler precies weet wat hij of zij moet testen, het kan zijn dat jij hun mening alleen wilt weten over hoe intuïtief de user interface is, of je wilt dat de tester een specifiek onderdeel van de game gaat spelen om te kijken hoeveel moeite hij hiermee heeft. Zorg er daarom voor dat het duidelijk is wat wel en wat niet hoort te werken.

Elk type game en elk iteratie hiervan vereist zijn eigen set vragen. Probeer de vragen specifiek te houden, waarbij je gericht vraagt naar specifieke features. Vragen naar de mening van het spel, of vragen wat zij zouden veranderen, dat zijn vragen waarbij opgepast moet worden doordat elke tester waarschijnlijk iets anders vind en/of iets anders wilt. Hierdoor is het belangrijk om niet alles aan te nemen van de testers, gebruik je eigen verstand, jij als developer weet uiteindelijk waar de game heen moet gaan. Om daarbij te helpen kunnen er kwalitatieve vragen gesteld worden zoals:

- Wat vindt men leuk en wat vind men saai?
- Wat vindt men te makkelijk en wat te moeilijk?
- Welke delen vindt men frustrerend?
- Heeft men alle regels geleerd?
- Was er een drastische stijging in de moeilijkheidscurve?
- Vinden hardcore gamers het saai of te makkelijk?
- Vinden casual gamers het te moeilijk?

Naast deze kwalitatieve vragen kunnen er ook kwantitatieve vragen gesteld worden:

- Hoe vaak is de speler dood gegaan? In welk level gebeurde dit?
- Waar klikken de gebruikers?
- Hoe vaak worden power-up gebruikt?
- Benodigde tijd om een level uit te spelen?
- Wanneer stopt de speler met spelen?
- Vraag de speler om een cijfer toe te kennen aan elementen als moeilijkheid, fun en begrip van regels.
- Vraag de speler om de level te rangschikken van makkelijk naar moeilijk, van minst leuk naar meest leuk en van makkelijk te begrijpen naar verwarrend.

De antwoord op deze vragen geeft een game designer een goede inschatting van wat er goed en fout is aan de game, waarna hij het design aan kan passen.

Om effectief te om te gaan met usability tests is het belangrijk dat de regression test ook meegenomen wordt in de ontwikkeling van de game. Het is dus niet voldoende om een test uit te voeren en aan de hand van de resultaten de game aan te passen zonder dit weer terug te koppelen naar de test gebruiker. Wanneer iets veranderd wordt aan de hand van gebruikersfeedback wil dit nog niet betekenen dat deze verandering juist is. Koppel de aanpassing weer terug naar de gebruiker en controleer of dit het probleem oplost.

Game designer Chris Viggers van Blitz Games Studios over usability testing:

> *"Does factoring usability testing into your pipeline mean more work?* Yes*. Does it add cost to your project? Definitely. Are the end results worth it to your company and your teams? Without a shadow of a doubt*."

Ter ondersteuning van het playtest process is het verstandig om te weten te komen welke verschillende type spelers er allemaal zijn. Daarom word er nu een overzicht gegeven van de verschillende type spelers en hoe dit gebruikt kan worden om de spellen beter te maken.

5.1.2 Player archetypes

Elke gamer speelt games op zijn eigen manier, dit zorgt ervoor dat je bij het designen van een game rekening moet houden met zoveel mogelijk verschillende type gamers. Richard A. Bartle omschrijft in het artikel "*Hearts, Clubs, Diamonds, Spades*"[2] vier verschillende categorieën gamer, de "Achiever", de "Explorer", de "Socializer" en de "Killer", daarnaast zijn in de laatste jaren de categorie casual gamer, hard-core gamer, button basher, customizer en exploiter bijgekomen.

Het is niet zo dat elke gamer maar in één van deze categorieën valt, vaak is er een combinatie van eigenschappen die het type gamer definieert. Bij het designen en testen van games is het belangrijk om de games vanuit deze verschillende categorieën te bekijken om een beter beeld te krijgen van hoe de game uiteindelijk door alle gamers gespeeld gaat worden. Vraag jezelf af hoe bijvoorbeeld een explorer een game zal spelen en ga vervolgens nadenken over de problemen die dit type gamer kan tegenkomen en pas hier het design voor aan of test de game zoals een explorer het zal spelen.
Hier volgt een korte omschrijving van de verschillende type gamers zoals die gedefinieerd zijn door Richard A. Bartle. Voor de vier hoofd types worden de implicaties gegeven van wat dit betekent voor het testen van een game.

Achiever

De achiever is er op uit om de game zo goed mogelijk uit te spelen. Hij krijgt voldoening uit het spelen van de game op een efficiënt manier en het halen van quests, missies, levels en achievements die andere spelers nog niet gehaald hebben. Hij zal de game meerdere malen spelen op hoger moeilijkheidsgraden of onder andere omstandigheden, zoals alleen een mes gebruiken in een first person shooter. Dit type gamer is ook geïnteresseerd in het halen van bonus doelen en het halen van bonus levels.

Om een achiever tevreden te houden zou je bijvoorbeeld extra rekening kunnen houden in de design met het juist definiëren van de achievements, quests en de moeilijkheid. En voor het testen ga je juist deze elementen testen of alles wel goed verloopt. Wat zou er bijvoorbeeld gebeuren als men meerdere achievements tegelijkertijd zou behalen?

Explorer

Explorers, oftewel verkenners, zijn geïnteresseerd in het uitvinden van wat de game te bieden heeft. Ze zullen rond reizen om obscure plekken en de randen van de map te vinden. "Unmapped" territorium zal hun aandacht trekken. De explorers zijn het type gamer die rond zullen kijken en van de art in de game genieten. Zij zijn ook diegene die interessante dingen zullen proberen, zoals het komen op plekken waar je eigenlijk niet hoort te komen, het openen van alle deuren en het bekijken van alle items in de game. De explorer is degene die zich het volgende afvraagt: "Ik ben benieuwd wat er gebeurd als ik ……. doe?"

Houd rekening in de design met deze mensen. Definieer goed de randen van de map en denk na over wat er zou moeten gebeuren als de speler ergens komt waar hij niet hoort te zijn. Geef deze mensen juist de ruimte om te verkennen als de game type dit toelaat. Voor het testen is het belangrijk dat je juist op plekken probeert te komen waar je niet hoort te komen.

[2] http://www.mud.co.uk/richard/hcds.htm

Socializer
Het doel van de socializer is het gebruiken van een game als rollenspel en daarbij zijn medespelers te leren kennen. Het kunnen aansluiten bij clans en guilds en het gebruik maken van de chat en forum functionaliteit zijn voor hem erg belangrijk. De socializer organiseert evenementen in de game zoals meetings, toernooien en trading events. Hij maakt gebruik van de speciale features in de game wanneer hij hiervan te horen krijgt van andere spelers.

Sociale aspecten in games wordt steeds belangrijker. Definieer goed hoe de chat en party functionaliteit in elkaar zit. Kan je bijvoorbeeld met iedereen praten? Ook met de vijand? Geef de spelers ruimte om samen te spelen. Test goed of deze elementen werken, test ook of de UI ergonomisch is.

Killer
Killers houden ervan om beter te zijn dan andere gamers, ze spelen om te winnen. Dit type gamer speelt vaak de gametypes waarbij hij tegen mensen moet spelen in plaats van samen spelen, zoals de free for all deathmatch gametype. Hij gebruikt zijn headset om zijn tegenstanders te lokken en te vernederen.
Voor de killer moet je goed rekening houden dat er veel statistieken over jezelf en andere spelers beschikbaar zijn. De killers zijn degene die de leaderboards nauw in de gaten houden en streven om hier op nummer 1 te komen staan, zij willen kunnen zien dat zij beter zijn dan anderen. Voor de killer archetype moet je dus elementen als voice communicatie en leaderboards / scoreboards goed testen.

Casual
De casual gamer beperkt zich meestal tot de functionaliteit die in de instructies of tutorial worden uitgestippeld (deze moeten daarom ook erg goed in elkaar zitten). Ze zitten vaak niet te wachten op erg complexe gameplay of lange speelsessies. Casual gamers zien zichzelf niet als gamers.

Hard-core
De hardcore gamer zal vaak een maxed-out hardware setup hebben om alles op max detail met een hoge framerate te kunnen spelen. Hij neemt de games serieus en maakt gebruik van macro's om hem beter te laten spelen. De hard-core gamer noemt zichzelf een gamer, en steekt veel tijd is het spelen van games en het volgen van alle game nieuws.

Button basher
De button basher houdt van snelheid en repetitie, in plaats van langzaam en defensief spelen. Hij houdt constant de "A" knop vast om sneller te rennen, om hoger te springen of om sneller te slaan.

Customizer
De customizer houdt ervan om alle features van de game zijn eigen te maken. Hij zal modificaties (mods) instaleren om de user interface te veranderen naar zijn wens.

Exploiter
De exploiter is iemand die altijd op zoek is naar shortcuts in de gameplay. Hij zal cheats gebruiken en hij zal zoeken naar fouten op een map waardoor hij een voordeel heeft over andere. Het is een type gamer die verder ook bots zal maken om items automatisch te creëren, punten te verdienen en om levels omhoog te gaan. In andere worden de exploiter zal altijd op zoek zijn naar manieren om een oneerlijk voordeel over anderen of de computer te krijgen. In een online omgeving is het erg belangrijk dat je het exploiten zo lastig mogelijk maakt voor dit type gamer.

Ga van te voren na voor welke speler archetype(s) je een spel aan het designen of testen bent. Verplaats je in hun schoenen vraag je af wat type "x" graag zou willen zien of doen in de game en pas hier je design of test voor aan.

5.2 Casual game design

De fun-factor is op verschillende manieren te verhogen of verlagen. Het maken van een leuk spel is dan ook een erg subjectief vakgebied waarbij bijna oneindig veel factoren meespelen. Er zijn wel enige design beslissingen waar de meeste game designers het over eens zijn. Daarom wordt er in het vervolg van dit hoofdstuk suggesties gedaan over het volgende:

- Casual level design
- Elementen die vermeden moeten worden in casual games

5.2.1 Casual level design

Het boek "Casual Game Design" van Gregory Trefry geeft een lijstje van elementen waar op gelet moet worden voor het designen van levels voor een casual game.

- Wees meelevend
- Als jij het niet kan halen, dan is het te moeilijk
- Design voor de casual, niet de hardcore
- Geleidelijk de moeilijkheid opvoeren
- Daag de speler uit
- Gebruik een centraal thema
- Level moet uitleggen hoe er gespeeld moet worden
- Breek je eigen regels
- Maak een plan
- Varieer de levels
- Verfijn, speel, verfijn
- Playtest

Deze punten worden nu kort toegelicht.

Wees meelevend

Bij het designen van een level voor een casual game is het niet de bedoeling dat het een wedstrijd wordt tussen de designer en de speler, dus geen motto als "versla dit level maar!". Leid de speler door de level heen en leid de speler naar plezier. Soms betekend dit dat de speler uitgedaagd moet worden, maar soms betekend dit dat de speler een level zonder problemen moet kunnen oplossen. De spelers willen uitgedaagd worden, maar ze willen niet gestraft worden. Dit is erg lastig, je probeert de speler te laten winnen zonder dat hij door heeft dat jij hem laat winnen.
Om te weten te komen of de spelers het naar hun zin hebben moet je je in hun schoenen gaan verplaatsen. Doe alsof je de spelregels niet kent en probeer de game uit. Verder is het verstandig om de game te laten spelen door iemand uit de doelgroep (playtesting).

Als jij het niet kan halen, dan is het te moeilijk

De designer weet alles van de game, hij weet precies gebruik te maken van alle elementen en hij weet precies wat er gaat komen. Dit is de reden waarom de designer bij voorbaat al beter is dan degene voor wie de game bedoeld is. Als de designer een level niet kan uitspelen dan heeft de casual speler helemaal geen kans om het uit te spelen. Voor hardcore games geld deze regel ook maar in mindere maten, hardcore gamers zijn bereid veel meer tijd te investeren in het behalen van een uitdaging dan een casual gamer. Tweak de level niet voor jouw plezier maar voor de plezier van de doelgroep. Om te weten te komen wat de doelgroep wel en niet leuk vind is het belangrijk om te playtesten.

Design voor de casual, niet de hardcore
Zoals gezegd is, zijn hardcore gamers bereidt meer tijd in een game te steken. De hardcore is ook nog eens de groep gamers die hun mening laat horen via forums e.d. Het is dus makkelijk om te luisteren naar de feedback van deze gamers, maar pas hiermee op. Hardcore gamers kunnen goede suggesties hebben maar zij zijn daarbij ook weer de minderheid. Design dus altijd voor de meerderheid. Wanneer een level ontworpen wordt voor een casual game, dan moet deze met de casual gamer in gedachte gemaakt worden.

Geleidelijk de moeilijkheid opvoeren
Wanneer een game voor het eerst gespeeld wordt dan moet de speler veel dingen in één keer leren. Dit kan soms de speler erg afschrikken. Daarom is het verstandig om ingewikkelde gameplay geleidelijk te introduceren om zo geleidelijk de spelregels uit te leggen, paragraaf 5.3 gaat hier verder op in. Geef de spelers in het begin de ruimte om de regels te leren en voer vervolgens trapsgewijs de moeilijkheid op.

Daag de speler uit
Het is essentieel dat de speler het spel geleidelijk kan leren kennen, maar het is ook weer belangrijk om de speler uit te dagen. Als een game niet uitdagend is dan raakt het alle elementen kwijt van wat gamen nou precies zo aantrekkelijk maakt voor mensen. Daarom is het niet erg als er af en toe een level tussen zit waar de speler een paar pogingen moet doen om het uit te kunnen spelen. Let er alleen wel op dat elke level wel te halen moet zijn!

Bouw levels rond een centraal concept/thema
Bij het designen van een level is het verstandig om dit vanuit het centraal concept of thema te doen. Begin met het hoofdconcept en vandaar uit kunnen er variaties gemaakt worden op het concept. Zorg alleen dat het doel van het spel helder en duidelijk blijft.

Level moet uitleggen hoe er gespeeld moet worden
Een level moet signalen geven over hoe de level gespeeld moet worden. Wanneer er een nieuwe move/wapen/feature etc. geïntroduceerd word dan moet de level de ruimte geven aan de speler om deze eerst uit te proberen voordat de speler het echt moet gebruiken. Wanneer de speler bijvoorbeeld een bepaalde sprong moet uitvoeren om over een dodelijk gat te komen dan moet de level eerst een obstakel introduceren waar de speler overheen moet springen zonder dat hij hier dood van kan gaan. Hetzelfde met items en power-ups, dwing de speler deze te gebruiken zodat ze de basics leren kennen. En dan in de volgende level kan je de speler de item / power-up op een creatieve manier laten gebruiken.

Breek je eigen regels
Wanneer er patronen zijn gevestigd in de game, dan zou de designer zijn eigen regels kunnen breken. Doe dit wel voorzichtig, de game moet altijd aanvoelen alsof er logica in zit. Het veranderen van de gevestigde patronen of regels kan er voor zorgen dat de game fris blijft aanvoelen, dit kan er voor zorgen dat de speler langer geïnteresseerd blijft.

Maak een plan
Het is belangrijk om van te voren de levels uitgestippeld te hebben. Maak een plan voor wanneer nieuwe concepten, power-ups, vijanden, doelen, content etc. toegevoegd gaat worden. Maak een spreadsheet waar precies beschreven staat in welk level en waar in de level een element toegevoegd gaat worden. Dit is essentieel om op een overzichtelijke manier de game progressie te kunnen zien. Wanneer alles gedocuemnteerd wordt dan is het mogelijk om precies te kunnen zien waar er eventuele fouten in de game kunnen zitten. Een voorbeeld zou kunnen zijn dat er aan de hand van een plan gezien kan worden of er bepaalde elementen te veel of te weinig gebruikt worden.

Varieer de levels
Het variëren van de levels is makkelijker gezegd dan gedaan. Het is makkelijk om een game te herhaaldelijk te maken zonder dat ment dit door heeft, daarom moet er bewust moeite gestoken worden in variatie toevoegen in de levels. Er zou bijvoorbeeld een ander game designer gevraagd kunnen worden om een level te designen, ieder designer heeft een ander insteek voor het designen van levels, dit zorgt dus voor meer variatie. Zorg wel voor basis regels wanneer er meerdere game designers aan één product werken.

Verfijn, speel, verfijn
Het is praktisch onmogelijk om een level helemaal goed te hebben na de eerste design poging. Leg daarom de basis principes uit, speel de game, tweak de levels, speel de game opnieuw. Herhaal dit tot het compleet en "polished" is. Dit is één van de belangrijkste taken van level design, er gaat hier vaak dan ook veel tijd overheen. Soms is het verstandig om een dag even met iets anders bezig te gaan zodat er vervolgens weer met een frisse blik tegenaan gekeken kan worden.

Playtest
Het is mogelijk om een level constant te verfijnen totdat men hier gek van wordt, verstandiger is om iemand erbij te halen die de game nooit gespeeld heeft. Vraag buitenstaanders de game te spelen en noteer hoe ze de level spelen en wat hun feedback hierop is. Verleng de stukken die leuk gebleken zijn en kort de niet zo leuke stukken in.

Introduceren en uitdagen
Wanneer er goed gekeken wordt dan is er te zien dat er drie elementen zijn die telkens herhaald worden bij dezen "regels" voor casual level design: Het geleidelijk introduceren van de regels, het playtesten en het uitdagen van de speler.

Bij het testen op speelbaarheid is het dus belangrijk om de introductie van de game en de introductie van nieuwe elementen goed te testen of hier alles naar behoren werkt en of de uitdagende delen uitdagend genoeg zijn. Verplaats je in de schoenen van de eindgebruiker en speel de game door zoals de gebruiker dit zou doen. Voelt dit goed aan? Snap je alles? Is alles te halen? Nadat de game als een "domme gebruiker" doorlopen is, dan is goed aan te voelen waar de risicovolle plekken zitten. Vervolgens moeten deze plekken grondig doorgespeeld worden om er precies achter te komen wat er niet klopt. Er moet concreet gezocht worden naar waarom een level niet te begrijpen is, of waardoor het komt dat een level niet te halen is of juist te makkelijk is. Het zou bijvoorbeeld zo kunnen zijn dat er geen instructies worden gegeven over hoe je "wapen x" moet gebruiken. Of het kan zo zijn dat een puzzel niet op te lossen is omdat er niet genoeg tijd wordt gegeven om een erg complexe som uit te rekenen.

Het succes van een casual game wordt voor het grootste gedeelte in de eerste 10 minuten van spelen bepaald. Daarom is het dus zo belangrijk dat de uitleg van de game op een effectief maar "fun" manier wordt weergeven. Wanneer de speler eenmaal besloten heeft om na de eerste 10 minuten te blijven spelen, dan is het doel om ervoor te zorgen dat de speler blijft spelen. Dit gebeurd dus door het uitdagend houden van de game, zonder uitdaging is het eigenlijk geen spel meer te noemen. Dus controleer of de game uitdagend genoeg is voor de doelgroep, maar let er goed op dat het niet *te* uitdagend wordt.

5.2.2 10 dingen die je niet moet doen in Casual Games

De laatste jaren zijn er steeds meer voorbeelden van goede en slechte casual games erbij gekomen, daardoor zijn er nu bepaalde elementen waar casual game designers het redelijk met elkaar eens zijn over wat je het best niet kan doen.

Jason Kapalka van Popcap Games[3] (maker van Bejeweled) geeft een mooi lijstje van tien punten hoe je ervoor zorgt dat je een slechte casual game maakt.

> *Note: Popcap Games is lange tijd de markleider van de casual gaming industrie geweest, zij hebben veel van de standaarden neergezet voor het designen van casual games. Deze tien punten zijn uit jaren lang ervaring voortgekomen.*

1. Maak de game heel moeilijk
2. Geef veel middelmatige game modi in plaats van één goede modus
3. Distribueer de game als een grote download waarbij je een next-gen computer voor nodig hebt om het te kunnen draaien
4. Verkoop de game voor 35euro
5. Maak gebruik van de rechter muisknop en het toetsenbord
6. Geef het een afschuwelijke naam of thema
7. Deel lage scores uit
8. Verwacht dat de gebruikers gaan lezen
9. Maak het heel erg uitdagend
10. Negeer wat mensen te zeggen hebben over de game

Vermijd deze punten dus bij het designen van een casual game. Soms moet er juist het compleet tegenovergestelde gedaan worden zoals:

- Luister wel naar de meningen van de gebruikers
- Streef bij het designen van de games dat deze met alleen de linkermuisknop te spelen zijn
- Deel juist hoge scores uit (maar niet te hoog, te hoge scores werken ook averechts)
- Maak gebruik van plugins die de meeste mensen al op hun pc hebben zoals “Adobe Flash”
- Enz.

Omdat PopCap games zo een succesvolle casual game developer is is er besloten een analyse van één van hun meest populaire games, Bejeweled Blitz, te maken. In bijlage A-2 is deze analyse te vinden.

Het volgende onderdeel in dit hoofdstuk bespreekt de verschillende manieren waarop de spelregels uitgelegd kunnen worden.

[3] Zie http://www.casualgamedesign.com/?p=30

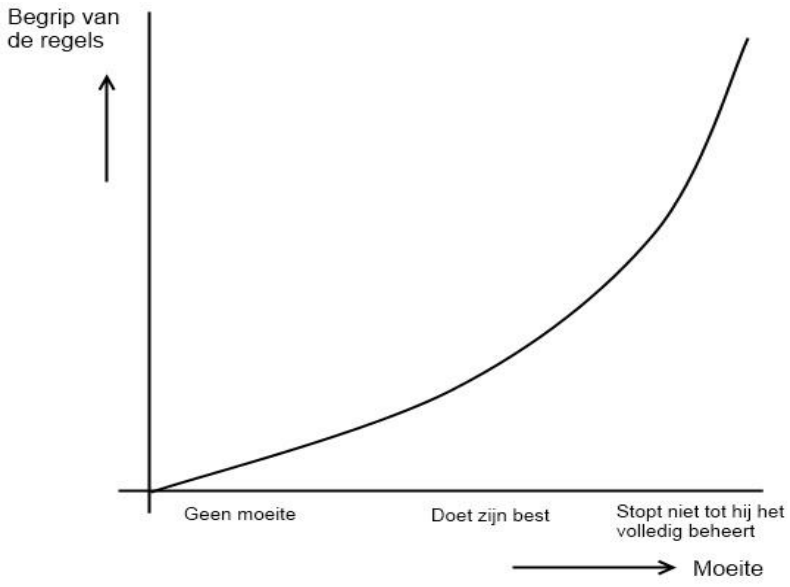
5.3 Leren van de regels

Om met de deur in huis te vallen: De speler wil niet zo snel mogelijk de regels leren, maar hij wil zo snel mogelijk spelen. Hiermee wordt bedoeld dat gamers gewoonlijk geen tijd willen investeren in het leren van de regels van een spel. Hier staat tegenover dat als een gamer de regels niet snel genoeg begrijpt dan wordt de game als slecht beschouwd. Dit betekent dus dat de manier waarop de regels worden uitlegt aan de speler een erg belangrijk aspect is tijdens het designen van een game.

> *Note: Als een tester betekend dit dat het erg belangrijk is om ervoor te zorgen dat de manier waarop de regels worden uitgelegd goed en volledig getest worden, hier mogen geen fouten in zitten.*

Het is onmogelijk om een game te maken zonder regels, dit betekent dat er altijd een overdracht zal moeten plaatsvinden van de regels naar de speler toe. Dit kan op verschillende manieren gedaan worden, met verschillende leercurves. Er is geen leercurve die altijd juist is, het ligt geheel aan het type spel en het type doelgroep. Wel is er een streven om ervoor te zorgen dat het de speler zo min mogelijk moeite kost om de regels te leren.

De onderstaande grafieken laten voorbeelden zien van leercurves waarbij het begrip van de regels uit gezet wordt tegen de hoeveelheid moeite die de speler in de game moet steken om de regels te leren. Waar developers naar moeten streven, is de speler zo snel mogelijk de regels laten leren zonder dat dit de speler veel moeite kost.



Figuur 5-2 Hoge instapsdrempel

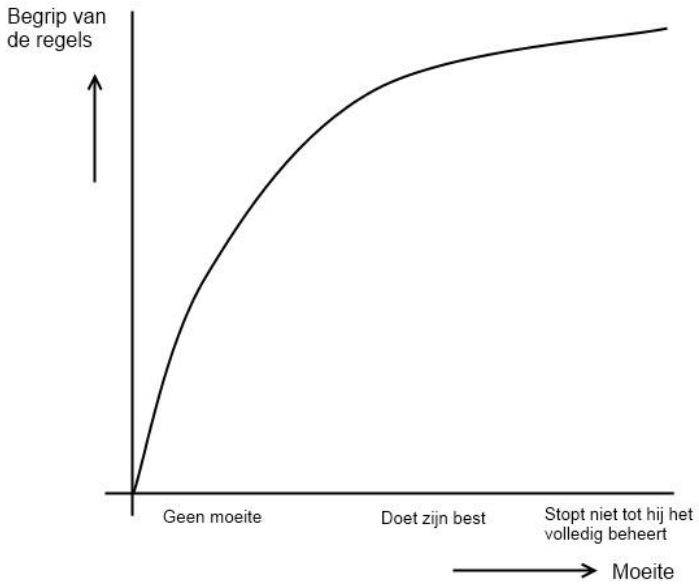
Hoge instapdrempel
Het bovenstaand voorbeeld hoort bij een game met een hoge instapdrempel, het zal de gamer hier veel moeite en waarschijnlijk ook veel tijd kosten voordat de game volledig beheerst wordt.

Games met een hoge instapdrempel zoals deze, horen gewoonlijk niet bij casual games. Maar er zijn natuurlijk ook games die gemaakt worden voor de hardcore gamer, dit type gamer is bereid veel meer tijd in een game te steken en zal niet snel afhaken als de regels niet direct duidelijk zijn. Er zullen ook hardcore gamers zijn die het juist leuk vinden om games te spelen die erg lastig zijn om te leren. Dit wil niet zeggen dat het een streven moet zijn om de introductie van de regels erg uitdagend te maken. Ontwerp de introductie van de regels altijd voor de meerderheid van de doelgroep.

Lage instapdrempel

Tegenover een leercurve van een game met een hoge instapdrempel, staat een leercurve met een lage instapdrempel, zie de grafiek hieronder. Games met deze leercurve zijn makkelijker te begrijpen, het kost de speler relatief weinig tijd moeite, wat vaak ook betekend dat het weinig tijd kost om de regels volledig te beheersen. Zo een leercurve kan me vaak in casual games tegen komen.

Het streven van elke game is om het spelen van de game de uitdaging te laten zijn en niet het leren van de regels. Voor bepaalde hardcore games is dit niet altijd mogelijk, maar voor casual games is dit zo goed als altijd een vereiste. Een goed voorbeeld is de succesvolle casual game Bejeweled, waarbij het zo is dat de game praktisch direct te begrijpen is door alle spelers. Je hoeft de game maar één keer te spelen om er achter te komen wat de core game mechanic is. (Zie bijlage A-2 voor een analyse van Bejeweled Blitz)

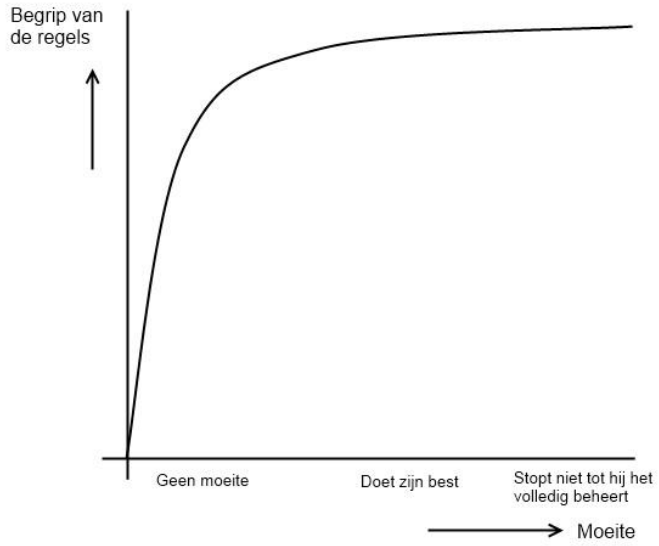


Figuur 5-3 Lage instapsdrempel

Zoals al gezegd is, alle games hebben regels en de speler zal deze altijd moeten leren. Één manier om deze regels over te brengen is om het gewoon, heel direct, te vertellen. Dus een scherm waarop staat: “Dit zijn de regels, leer ze en als je klaar bent met het leren dan kan je de game gaan spelen”. Dit is een aanpak die gebruikt wordt bij bordspellen, de speler moet de handleiding doornemen voordat hij kan gaan spelen. Table-top gamers zijn ook vaak bereid deze handleiding voor een groot deel door te nemen. Computer gamers daarentegen willen helemaal niks te maken hebben met handleidingen. Ook al komt een computer game met een uitgebreid handleiding, dan nog is het zo dat praktisch niemand dit doorleest. Om dit probleem te verhelpen is het tegenwoordig gebruikelijk om de handleiding in de game te verwerken, dit staat bekend als de tutorial.

Tutorial

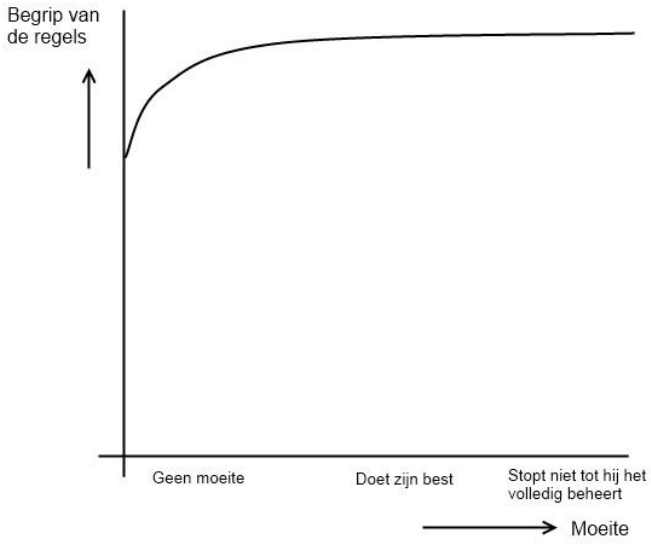
Hieronder zie je de leercurve van games die gebruik maken van een tutorial. Let er wel op dat de x-as niet de hoeveelheid tijd is maar de hoeveelheid moeite. Een tutorial leert de regels zonder veel moeite aan de speler, maar dit wilt niet perse zeggen dat het weinig tijd kost. Het streven bij het designen van een tutorial is het uitlegen van de regels op zo een manier dat het de speler weinig moeite kost om de game volledig te begrijpen.



Figuur 5-4 Lage instapsdrempel d.m.v een tutorial

Bestaande kennis

Het is slim om bij het designen van een game gebruik te maken van bestaande kennis en gewoontes van gamers, het wiel hoeft niet altijd opnieuw uitgevonden te worden. Gamers zijn van nature ongeduldig, dus informatie die de gamers al hebben opgedaan in andere games en situaties in het echte leven hoeven niet nog eens uitgelegd worden. Hieronder zie je de leercurve voor een game waarbij men al bestaande kennis van de game mechanic in huis heeft. Je ziet dat de game zonder veel moeite, en dus hoogstwaarschijnlijk ook snel, te begrijpen valt.



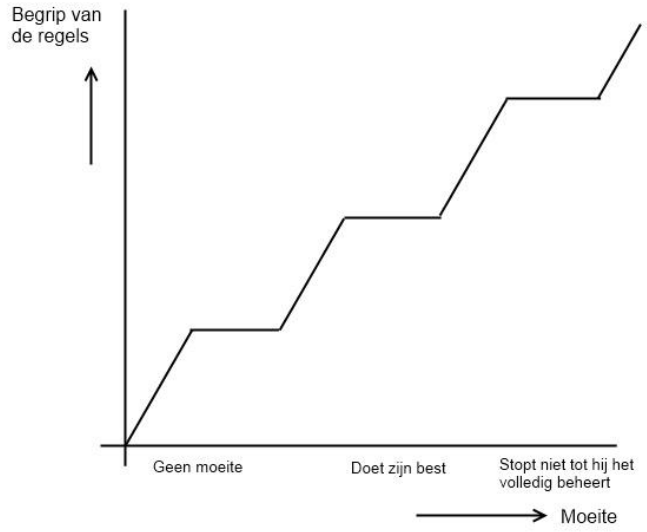
Figuur 5-5 Leercurve waar gebruik wordt gemaakt van bestaande kennis

Denk aan de game die geleverd wordt met elke windows pc “Microsoft Solitaire”. Één van de redenen waarom dit de grootste casual game ooit is omdat het een computer versie is van een game die al jaren lang bekend is in zijn “hardcopy” speelkaart variant. De regels van deze game hoeft niet uitgelegd worden omdat iedereen deze al kent. Dit zorgde voor een extreem lage drempel om de game te gaan spelen (zie bijlage A-1 voor een analyse van MS Solitaire). Verder kan je ook denken aan first person shooters, zoals Call of Duty. Spelers van dit genre weten dat bijvoorbeeld een rode ton zal exploderen wanneer je hierop schiet, terwijl een houten krat niet explodeert. Dit zijn willekeurige regels, maar het is wel verstandig om deze regels mee te nemen in de design zodat geen uitzonderingen op de gewoontes uitgelegd hoeven worden.

Er hoeft niet alleen gebruik te maken worden van bestaande kennis, ook is het mogelijk om gebruik te maken van de Intuïtie of de gevoel van gamers. Wanneer er een item in een game opgeraapt wordt en deze actie veroorzaakt een blij klinkend geluidje dan is dit een teken dat deze item een positief effect heeft en dus weet men voortaan deze items op te rapen wanneer ze gezien worden. Hetzelfde geld andersom, wanneer iets er eng uitziet en / of een eng geluid met zich meebrengt, dan weet de speler dat dit waarschijnlijk een vijand is en dus vermeden of vernietigd moet worden. Denk aan de game Pac Man, hier is geen tutorial aanwezig, maar het is wel na één keer spelen duidelijk wat wel en niet kan doordat er goed gebruik gemaakt is van geluid en beeld.

Trapsgewijs

Als laatst is er nog een leercurve die, nu dat de games steeds groter en complexer worden, steeds meer gebruikt wordt. Namelijk een trapsgewijs leercurve. Dit is een geschikte strategie wanneer een game niet uitgelegd kan worden aan de hand van de andere strategieën omdat er gewoonweg te veel regels aan gebonden zijn.



Figuur 5-6 Leercurve waarbij trapsgewijs de game wordt uitgelegd

Deze strategie gaat er vanuit dat niet alle informatie in één keer uitgelegd wordt omdat het door een grote hoeveelheid aan regels *te* veel moeite zou kosten om de regels in één keer te leren. Maar zoals al gezegd is, spelers willen de game zo snel mogelijk spelen, niet zo snel mogelijk snappen. Daarom is deze leercurve voor complexe games een goede oplossing, zodoende kan de speler direct gaan spelen zonder dat hij teveel informatie in één keer tot zich hoeft te nemen. De regels worden gedoseerd gegeven. De speler krijgt pas de regels uitgelegd wanneer hij met nieuwe gameplay elementen geconfronteerd wordt.

Een MMORPG (massively multiplayer online role-playing game) zoals World of Warcraft heeft immens veel verschillende regels. Als alle regels in één keer gegeven zouden worden, dan zou de speler dagen lang bezig zijn met het leren van de regels. Dit betekent dat de speler de game de eerste dagen niet eens zou kunnen spelen omdat hij eerst alle regels zou moeten leren. Daarom is er besloten dat je in World of Warcraft met veel beperkingen begint. Zo heb je maar de keuze uit een paar aanvallen, kun je geen specialisatie kiezen, kun je niet aangevallen worden door de vijandelijke factie en hoef je je geen zorgen te maken over de andere classes die beschikbaar zijn. Deze elementen worden pas geïntroduceerd wanneer de speler een bepaalde level bereikt heeft. Zodoende krijg men geleidelijk steeds kleine delen van de regels voorgeschoteld, in World of Warvraft kan dit ongeveer een maand duren. Het kost de speler weinig moeite om de kleine hoeveelheden aan regels telkens te leren, maar het kost de speler wel veel tijd eer hij de game volledig beheerst. Achievements kunnen helpen bij het trapsgewijs uitleggen van de regels. Zie hoofdstuk 6.4 – “Incremental en Meta-achievements”.

Wat betekent dit?

Wat opgevallen is van het analyseren van de twee meest populaire casual games, Microsoft Solitaire en Bejeweled Blitz[4], is dat beide games een leercurve hebben waarbij er gebruik wordt gemaakt van bestaande kennis. Dit betekent dat het de speler nauwelijks moeite kost om de spelregels te leren omdat hij deze regels, bewust of onbewust, al weet. Daarom is het verstandig dat er bij het bedenken van nieuwe concepten gebruik gemaakt wordt van alledaagse activiteiten en hier een op gegeven wordt in de vorm van een game. Bejeweled Blitz geeft bijvoorbeeld een draai aan de menselijke gewoonte om dingen te willen sorteren en Microsoft Solitaire maakt gebruik van het feit dat men de regels van de ‘analoge’ variant van Solitaire al kent.

Omdat Yezzle.nl gericht is op de casual gamer is het erg belangrijk om ervoor te zorgen dat, wanneer de spelers voor het eerst met een game geconfronteerd worden, deze met zo min mogelijk moeite te begrijpen is. Casual gamers hebben namelijk de neiging om snel af te haken, vooral wanneer een game niet snel duidelijk is. Daarom moet er veel tijd gestoken worden in de manier waarop de regels uitgelegd worden. Voor erg simpele games kan het voldoende zijn om alleen een instructie pagina te geven, voor andere games is een tutorial misschien handiger. Er moet per game gekeken worden naar wat de beste manier is om regels over te dragen.

[4] Zie bijlage A-1 en A-2

5.4 Achievements

In video gaming zijn achievements meta-doelen die buiten de parameters van de game worden gedefinieerd. De in-game doelen hebben direct invloed op verdere gameplay, het beheer van achievements vind gewoonlijk buiten de grenzen van de in-game omgeving plaats en heeft geen effect op het vervolg van de gameplay. De rol van achievements, in Yezzle genaamd “Spaarzegels”, is het helpen van de speler om plezier te hebben en het kunnen laten zien aan zichzelf en aan andere wat de speler bereikt heeft. Als het goed gedaan wordt dan kan het een echte toevoeging zijn, maar het is heel makkelijk om het fout te implementeren wat de ervaring van de gebruiker ernstig kan belemmeren. Sinds Yezzle in mei gelanceerd is komen er regelmatig klachten binnen over de spaarzegel systeem. Spaarzegels worden niet opgeslagen en / of de spelers klagen over het feit dat zij de meerwaarde van de spaarzegels niet begrijpen. Daarom is er besloten om achievements te bespreken om er achter te kunnen komen waar de zwakte punten liggen.

PhD onderzoeker aan de universiteit Central Florida en game designer Lucas Blair definieert, onder anderen, verschillende type achievements die elk een ander effect hebben op de speler. Om effectieve achievements te kunnen designen, ongeacht deze voor casual of hardcore gamers zijn, is het essentieel om te weten hoe de verschillende achievements de spelers beïnvloeden.

Er moeten een paar termen uitgelegd worden die relevant zijn voor het begrijpen van de reacties van de spelers:

Intrinsieke motivatie - De wil of de drang in iemand, dus intern (intrinsiek), om bepaalde handelingen te verrichten waarmee tot een zeker doel wordt gekomen.
Extrinsieke motivatie - De wil of de drang gecreëerd door externe factoren (beloningen), dus extern (extrinsiek), om bepaalde handelingen te verrichten waarmee tot een zeker doel wordt gekomen.
Self-efficacy - Het vermogen en de overtuiging om adequaat en efficiënt te handelen in een gegeven situatie.

Self-efficacy

Het verhogen van de self-efficacy, de overtuiging van de spelers skills, is belangrijk voor de volgende redenen:

- Het verhoogd de vastberadenheid van het behalen van doelen
- Het verhoogd de kans dat spelers eigen strategieën gaan bedenken en gebruiken
- Het zorgt ervoor dat de spelers beter omgaan met negatieve feedback

Het verhogen van de self-efficacy van de spelers kan dus gedaan worden met behulp van onder anderen achievements. Er moet daarbij op vier factoren gelet worden.

- Elke speler zit op een ander expertise niveau. Probeer dus voor alle skill niveaus achievements te maken
- Het zien van andere spelers die slagen verhoogd de self-efficacy van de “kijker”. Denk bijvoorbeeld aan leaderboards.
- De self-efficacy is te beïnvloeden door middel van sociale overtuiging. Geef positieve feedback wanneer een speler iets goeds gedaan heeft.
- Hoe een persoon zich voelt is erg bepalend voor hun self-efficacy niveau. Houd dus rekening met hun stress niveau, emotionele conditie en zijn fysieke staat

Meetbare achievements

Dit type achievement wordt gegeven voor het behalen van een taak op een manier waarbij de performance te meten is tegen die van jezelf, die van andere spelers of een standaard gezet door de designer. Een voorbeeld hiervan zijn de sterren die je ontvangt in de game “Angry Birds” afhangend van hoe goed je de level gehaald hebt. Dit type achievement geeft de speler feedback van zijn skill. De reflectie van deze feedback verhoogt het gevoel van competentie in de speler, wat weer de intrinsieke motivatie verhoogt.

Figure 5-7 - Angry Birds sterren

Completion achievements

Tegenover de te meten achievements staan achievements die gehaald worden door een taak succesvol te volbrengen. Dit type achievement kan in tweeën gedeeld worden, namelijk: performance contingent achievements en non-performance contingent achievements.

Een *Performance contigent* achievement vereist skill om te halen, denk bijvoorbeeld aan een achievement voor het eerste keer uitspelen van een dungeon in “World of Warcraft”. Deze beloning in de vorm van een achievement kan leiden tot een verhoging van de prestatie van de speler. Maar het heeft ook als negatieve effect dat het gevoel van autonomie in de speler verlaagd, vooral als deze achievements veel worden uitgedeeld. Het gevolg van het verlagen van het gevoel van autonomie is een verlaging van de intrinsieke motivatie van de speler. Dit betekent dat zodra de speler de beloning gekregen heeft het onwaarschijnlijk is dat hij verder gaat met die specifieke taak. In de praktijk kan dit een verlaging in de replay waarde betekenen.

Non-performance contigent achievements worden uitgedeeld voor alleen het deelnemen aan een actie. Bijvoorbeeld het krijgen van een in-game pet voor het aanwezig zijn tijdens een event. Dit type achievement heeft geen negatieve effect op de intrinsieke motivatie, maar omdat het niet afhangt van skill betekend dit ook dat de speler weinig motivatie zal hebben voor het willen behalen van dit soort achievements.

Het is dus verstandig om meetbare achievements te gebruiken in plaats van completion achievements om zo de intrinsieke motivatie van de speler te verhogen door middel van feedback.

Saaie taken

Saaie taken in games kunnen leuker gemaakt worden door er extrinsieke motivaters aan te koppelen, zoals achievements. Omdat de speler niet snel geneigd is om een saaie taak uit te voeren wordt de intrinsieke motivatie niet aangetast door het gebruik van beloningen. Één van de manieren om spelers toch te motiveren om een saaie taak uit te voeren is door de waarde van deze taak te laten zien door middel van de benaming van de achievement. Een ander manier om mensen te motiveren is door extra regels en/of fantasie aan de taak zelf te koppelen. Dit is eigenlijk de essentie van wat een achievement doet.

Interessante taken

Spelers zullen interessante taken voor het merendeel wel uitvoeren, daarom is het niet verstandig om hier beloningen aan toe te kennen. Als er toch achievements geïmplementeerd worden voor interessante taken dan kunnen deze het beste achievements zijn voor het hinten naar de meest effectieve strategieën voor het behalen van een taak.

Probeer dus beloningen te geven voor saaie taken uitvoeren en geef feedback voor het spelen van interessante taken.

Moeilijkheid van achievements
Er zijn twee verschillende elementen waar opgelet moet worden bij het vast stellen van de moeilijkheid, de achievement moet uitdagend maar haalbaar zijn en de spelers' self-efficacy moet hoog genoeg zijn om de taak voor de achievement te willen volbrengen.

Achievements moeten de speler op zo een manier uitdagend zijn zodat de speler na het behalen van de achievement beter presteert. Het is wel belangrijk om te weten dat de casual doelgroep in bijna alle gevallen achievements niet eens gaan uitproberen wanneer deze te moeilijk zijn ingesteld.

Doel van de speler
Naast het bepalen hoe moeilijk de achievements moeten worden, is het ook erg belangrijk dat er op het doel van de speler gelet wordt, dus op welke manier pakt hij een taak aan. Er zijn over het algemeen twee verschillende type doel oriëntaties, namelijk, "performance oriëntation" en "mastery oriëntation". Spelers die performance oriëntation prefereren houden zich gewoonlijk bezig met de beoordeling van hun competenties in de ogen van anderen. De spelers die neigen naar mastery oriëntation zijn gewoonlijk gefocust op het verbeteren van hun eigen skill niveau.

Tegenwoordig richten veel games zich op de performance gedreven spelers, doordat ze een zware focus hebben op tijddoelen en scores. Dit heeft wel als nadeel dat spelers bij performance gedreven achievements weinig risico's gaan nemen omdat ze bang zijn dat nieuwe dingen uit proberen hun "score" negatief zal beïnvloeden. In een shooter betekend dit bijvoorbeeld dat spelers altijd dezelfde wapen zullen kiezen en dezelfde routes zullen nemen omdat het bewezen is dat deze route tamelijk effectief is. Dit heeft als gevolg dat performance gedreven spelers vaak alleen goed zijn in kleine niet-complexe taken.

Je wilt niet dat de speler telkens maar één klein onderdeel van de game te zien krijgt, daarom is het verstandig om door middel van achievements de spelers meer te laten neigen naar een mastery mindset zodat de spelers alle gameplay elementen te ervaren krijgen. Mensen die eropuit zijn om een bepaalde taak volledig zien te beheersen, zoals een game spelen, zien vaak dat zij tegen fouten kunnen en actief opzoek gaan naar manieren om hun competentie te verbeteren. Dit zorgt ervoor dat mensen die streven naar het meesteren van een taak vaak goed zijn in complexe taken.

Probeer door achievements de moeite die de speler in de game aan het steken is te erkennen. Games moeten fouten die de speler maakt als een moment gebruiken om feedback te geven en de speler aan te moedigen.

Verwachte vs. onverwachte achievements
Het is gebruikelijk in games dat de speler de nog niet behaalde achievements ergens kan bekijken zodat hij een plan kan opstellen om een achievements te halen. Maar er zouden ook onverwachte achievements geïmplementeerd kunnen worden. Dit zijn achievements waarbij de speler niet van tevoren op kan zoeken hoe deze te halen zijn. Dit kan een beetje verwarrend werken, maar wanneer de speler weet dat er achievements te halen vallen die niet gedocumenteerd zijn dan bevorderd dit een experimentele speelstijl, dit is iets waar game designers soms naar streven.

Een experimentele speelstijl bevorderen is erg goed, alleen moet je wel op de hoogte zijn van de voordelen van verwachte achievements. Het grote voordeel van een verwachte achievement is dat het de speler de mogelijkheid geeft om een doel vast te stellen voordat hij begint. Er zijn veel voordelen aan spelers met een doel:

- Doelen zorgen ervoor dat de speler zijn middelen gaat organiseren. Dit kan het opfrissen van bepaalde skills zijn, extra tijd uitrekken voor het behalen van het doel en / of hulp van vrienden vragen.
- Een doel voor ogen hebben verhoogd de hoeveelheid moeite die iemand ergens bereid is in te steken, dit betekent meer speel tijd.
- Als een speler een doel voor ogen heeft, dan zal hij niet snel opgeven wanneer een taak erg uitdagend wordt.
- Mensen met een doel zijn bereidt nieuwe kennis en skills te leren om een doel te bereiken.

Verwachte achievements geven ook aan hoe de game is opgebouwd en welke acties genomen moeten worden om de game succesvol af te ronden.

Probeer hoofdzakelijk verwachte achievements te gebruiken zodat de spelers een doel voor henzelf kunnen stellen. Onverwachte achievements kunnen spaarzaam gebruikt worden om een creatieve speelstijl te bevorderen.

Gehaalde achievements weergeven

Er zijn verschillende momenten om behaalde achievements weer te geven. Het meest voor de hand liggende is om direct feedback te geven wanneer een speler de achievement unlocked. Direct feedback geven kan het leerproces en de efficiëntie verbeteren. Hierdoor werkt het direct feedback geven van het behalen van een achievement het beste voor beginnende spelers.

Direct feedback geven heeft als nadeel dat het de speler uit “the zone” haalt, in andere worden het verplaatst de focus van de speler waardoor de immersie gebroken wordt. Gamen is een vorm van escapisme, en het breken van de immersie zorgt ervoor dat de speler weer bewust is van de echte wereld, dit verminderd de motivatie om verder te willen spelen. Daarom kan er ook gekozen worden om pas feedback te geven van een achievement wanneer er een natuurlijke pauze in de game zit. De ervaren spelers kunnen dit waarderen doordat zij dan de tijd krijgen om hun eigen prestaties te beoordelen.

Permanente vs. tijdelijk achievements

Er is een onderscheid te maken tussen permanente achievements en tijdelijk achievements. Tijdelijke achievements zijn elementen van feedback naar de speler toe die na het vertonen direct verdwijnen en niet opgeslagen worden. Denk hierbij aan het visuele en verbale commentaar als “Multikill” en “GODLIKE” wanneer je meerdere vijanden achtereenvolgens dood in de Unreal Tournament games. Dit type achievement verhoogd de intrinsieke motivatie van de speler.

Hier tegenover staan de permanente achievements. Deze komen in twee verschillende variaties voor, namelijk de *digitale tastbare* variant en de *opgeslagen lijsten* variant. Het verschil tussen de twee variaties zit in de beloning. Digitale tastbare achievements zijn achievements waarvoor je een “tastbaar” beloning krijgt (voor zo ver je dit tastbaar kan noemen in een digitale omgeving), denk hierbij aan een speciaal embleem of een in-game huisdier als beloning krijgen. Deze beloningen zijn gewoonlijk te zien door andere spelers.
Tastbare beloningen uitdelen is een goede motivatie voor de speler om een doel te halen, maar let wel op het feit dat dit soort type achievements niet te vaak gebruikt moeten worden. Het verlaagd namelijk de natuurlijke drang om handeling uit te willen voeren (intrinsieke motivatie) nadat de speler de beloning ontvangen heeft. Dit geld voor digitale als “echte” tastbare beloningen.

De achievements waarbij de beloning enkel een opgeslagen melding is voor het behalen van een doel, zoals de achievements van Xbox Live en Yezzle op het moment, dienen als reflectie voor de speler voor het succesvol behalen van een doel. Reflectie op behaalde doelen zorgt ervoor dat de spelers beter begrip krijgen van de game ervaring. Zorg er dus voor dat er een lijst aanwezig is waar men de behaalde achievements terug kunnen vinden.

Achievements weergeven aan anderen

Nu dat een stabiele en snelle internetconnectie voor de meeste mensen een feit is geworden, is het zo dat multiplayer elementen voor veel games een "must" zijn geworden en zien we steeds vaker dat achievements gedeeld worden met andere spelers. Sociale goedkeuring is een grote reden waarom men computer spelletjes speelt. Hierdoor wordt het kunnen zien van de behaalde achievements van andere spelers een grote motivater om zelf ook bepaalde achievements te gaan halen. Het is gebleken dat dit een positief effect heeft op de prestatie van de spelers. Het streven naar en uiteindelijk halen van deze achievements heeft een positief effect op de self-efficacy van de speler (het geeft zelfvertrouwen om andere taken te behalen).

Het delen van achievements met andere kan dienen als een soort gaming Curriculum Vitae. Het laat zien waar de speler zijn sterkte punten liggen en wat voor type speler hij is. Dit kan ervoor zorgen dat er aan de hand van andermans achievements gezien kan worden wie een goede teamgenoot, uitdager of vriend zou kunnen zijn binnen een spel. Deze gaming CV heeft wel als nare eigenschap dat het er ook voor zorgt dat beginnende spelers moeite kunnen hebben om medespelers te vinden, het sluit namelijk beginnende spelers uit. Dit is een fenomeen wat veel gezien wordt binnen MMORPG's zoals "World of Warcraft", waarbij ervaren spelers andere spelers niet toelaten in hun groep tenzij zij de juiste achievements hebben waarmee er te bewijzen is dat zij weten hoe zij effectief moeten spelen. Je moet dus ervaring hebben om ervaring op te kunnen doen (catch-22). Dit wil je tegengaan, dat kan door middel van slimme achievement design. Zo is er bijvoorbeeld de mogelijkheid om achievement te kunnen implementeren voor het helpen van andere (beginnende)spelers.

Negatieve achievements

Het gebruik van negatieve achievements moeten vermeden worden. Hiermee worden achievements bedoeld die gekregen worden nadat de speler heel slecht gespeeld heeft. Het geven van negatieve feedback kan ervoor zorgen dat de speler het gevoel van competentie en zelfstandigheid kwijt raakt, dit komt niet ten goede van de spelervaring. Er wordt heel anders gespeeld wanneer men de kennis heeft dat er negatieve achievements te behalen zijn, de gedachte hiervan kan erg vermoeiend zijn. In de plaats van dit soort achievements kan is het beter om feedback te geven die de speler kan helpen om een bepaald onderdeel uit te spelen.

Achievements als valuta

In sommige gevallen wordt er virtuele geld gekoppeld aan achievements. Dit kan in de vorm van punten, coins, goud, sterren enz. zijn. Dit virtuele geld kan dan vervolgens weer ingeruild worden voor in-game of real world items.

Het verdienen van in-game geld heeft dezelfde voordelen als een digitale tastbare beloning verdienen voor het behalen vaan een achievement (zie paragraaf "Permanente vs. tijdelijk achievements"). Maar het voordeel van het verdienen van virtueel geld in plaats van een tastbare beloning is dat de speler de keuze krijgt over wat hij met zijn geld gaat kopen.

Ook de nadelen van het belonen met virtueel geld zijn hetzelfde als het belonen met een digitaal tastbaar voorwerp. Nadat de speler een item gekocht heeft met het in-game geld, dan is er een kans dat hij niet erg gemotiveerd meer zal zijn om een specifiek onderdeel weer te doorlopen. Een ander nadeel hiervan is dat de speler meer gefocust zal zijn op de beloning systeem dan het spel zelf. Bij het designen van games moet dit fenomeen vermeden worden, de game zelf moet de main focus van de speler zijn.

Incremental en Meta-achievements
Incremental achievements zijn uitdagingen die steeds moeilijker worden wanneer er een bepaald doel bereikt wordt. Veel shooters dagen de spelers uit om b.v. 100 vijanden in-game dood te maken, nadat dit doel bereikt is veranderd deze uitdaging in het doodschieten van 250, 500 en 1000 vijanden. Meta-achievements worden behaald na het succesvol unlocken van een serie van andere gerelateerde achievements. Deze achievements splitsen dus eigenlijk een bepaald uitdaging op in hanteerbare onderdelen. Het opsplitsen van achievements vormt een goede manier om de speler te introduceren aan nieuwe game mechanics. Het leidt de spelers door middel van sub-achievements naar het volledig beheersen van complexe elementen die zij normaal gesproken niet zouden durven proberen. Dit soort achievement zijn een goede ondersteuning voor een game met een trapsgewijze leercurve (zie paragraaf 5.3 – “Trapsgewijs”).

Het nadeel hiervan is dat het kan voorkomen dat de speler het gevoel krijgt dat hij niks zelf meer kan beslissen omdat de game je precies leidt naar wat de game designer voor ogen heeft. Het is dus verstandig om wel ruimte en tijd tussen de achievements te laten zodat de speler nog wel het gevoel van autonomie heeft.

Competitieve achievements
Competitieve achievements vragen de spelers om een uitdaging direct aan te gaan met of tegen elkaar, of indirect door middel van het vergelijken van scores van individuele taken. Competitie kan de ervaring van een taak verbeteren, ook kan het ervoor zorgen dat de speler beter gaat presteren. Dit geld eigenlijk alleen voor de spelers met een relatief hoog niveau van skill, dit zijn gewoonlijk de hardcore gamers. Een hardcore gamer haalt veel plezier uit competitie en wordt in het algemeen niet sterk beïnvloed wanneer zij een wedstrijd verliezen, zij zullen dit juist als een uitdaging zien om beter te worden. Maar voor iemand die een relatief lage skill niveau heeft kan zo een achievement juist negatieve gevolgen hebben. Wanneer een achievement niet te halen is omdat de meeste andere spelers meer skill hebben, dan zal dit je het gevoel van minderwaardigheid geven waardoor de intrinsieke motivatie en de self-efficacy van de speler verlaagd wordt.

Het gebruik maken van competitieve achievements ligt dus erg aan de doelgroep. Voor de hardcore spelers werkt het erg positief, voor casual gamers werkt het vaak juist negatief.

Coöperatieve achievements
Coöperatieve achievements kunnen verdiend worden door samen met andere echte spelers een doel te halen. Dit kan bijvoorbeeld samen een monster in “World of Warcraft” dood maken kunnen zijn, of in een shooter als “Call of Duty” kan dit een x aantal assisted kills maken zijn. Het samenwerken om een gezamenlijk doel te halen is een goede manier om de prestaties van de speler te verbeteren. Zoals in het echte leven leert men meer wanneer iemand een taak voor doet dan wanneer het volledig zelf uitgezocht moet worden.

Een nadeel van het samenwerken is dat in grote groepen dit kan leiden tot vele miscommunicaties, zoals door gebrekkige chat functies of het missen van een Voice over IP (voip) systeem. Dit kan tegen gegaan worden door er voor te zorgen dat de achievements gericht zijn op kleinere groepen. Dit elimineert ook de belemmerende factor om af te hangen van heel veel andere spelers om een achievement te halen.
Een ander probleem is dat spelers makkelijk kunnen meeliften doordat hun individuele prestaties niet opvallen in grote groepen. Daarom is het verstandig om ervoor te zorgen dat de achievements ook op individuele inzet checken.

5.4.1 Wat betekent dit voor Yezzle?

De huidige achievements op Yezzle.nl zijn slecht geconfigureerd. Voor veel games worden dezelfde achievements gebruikt met in sommige gevallen een aangepaste omschrijving. Zo hebben alle Braintrainers de achievements “Het voltooien van level 5, 10 en 15” terwijl elke game anders in elkaar zit. Zo zijn er games waarbij het, met de huidige configuratie, zo goed als onmogelijk is om de level 15 spaarzegel te behalen, zelfs niet voor ervaren spelers. Dit is de meest opvallende fout in het spaarzegel systeem, dit heeft als gevolg dat de spelers niet eens bereid zullen zijn om deze *te* moeilijke achievements uit te gaan proberen. Zoals hierboven gemeld is:

Figure 5-8 De spaarzegels van Yezzle

“Het is wel belangrijk om te weten dat de casual doelgroep in bijna alle gevallen achievements niet eens gaan uitproberen wanneer deze te moeilijk zijn ingesteld.”

Het is dus nodig dat de achievements specifiek voor elke game gekalibreerd moeten worden. De achievements moeten uitdagend, maar niet *te* moeilijk zijn. Ook moeten de achievements niet generiek gaan aanvoelen. Het is oké om een zelfde set achievements voor elke game te gebruiken (mits deze per game afgesteld zijn), zoals haal level x, y en z, maar naast deze generieke achievements moeten er ook achievements komen die specifiek gericht zijn op de gameplay van de game in kwestie. Zo zou er bijvoorbeeld voor de Braintrainer “Golven” een achievement kunnen geïmplementeerd worden waarbij het de bedoeling is om level x met alleen de muis te halen. Zo een achievement laat de speler zien dat het ook mogelijk is om met de muis te spelen, dit is een vorm van een mastery achievement.

Er zijn bij Yezzle plannen voor een nieuw spaarzegelsysteem, waarbij je voor elk behaalde spaarzegel punten verdient. Deze punten zij weer inruilen voor een verlengen van het abonnement of een donatie aan een goed doel. Dit is in principe achievements als een valuta gebruiken. Hierbij moet men de gevaren van zo een systeem nauwlettend in de gaten houden. Het is belangrijk dat de main focus nog altijd op de games blijft en niet op het spaarzegelsysteem. Zolang de spaarzegelpunten in te ruilen zijn voor dingen als abonnementen of donaties naar goede doelen moet er geen probleem zijn, maar wanneer er punten in te ruilen zijn voor items die maar één keer te verdienen zijn, dan is er een risico dat men na het behalen van zo een item geen motivatie meer heeft voor het opnieuw willen doorlopen van de games om meer punten te verdienen.

Naast het feit dat het spaarzegelsysteem opnieuw ontworpen gaat worden, is het verstandig om de spaarzegels zijn opnieuw te designen waarbij deze per game geconfigureerd worden.

5.5 Conclusie / Aanbeveling – Playability onderzoek

In het hoofdstuk ontwerp is er gezegd dat Yezzle op het moment niet op alle vlakken leuk genoeg is. Van te voren was het al duidelijk dat hier geen één juiste oplossing voor te geven is omdat het een erg subjectieve vakgebied betreft. Wat voor de ene persoon leuk is, is niet per se voor ander leuk. Daarom is er besloten om als antwoord op deze vraag informatie en/of inzichten te verschaffen die ter ondersteuning gebruikt kunnen worden voor het verhogen van de user-experience. De punten die erg belangrijk zijn gebleken zijn als volgt:

- Playtesten is een must.
- De meest succesvolle casual games, zoals Microsoft Solitaire en Bejeweled Blitz, hebben een leercurve die gebruik maakt van de reeds aanwezige kennis van de speler.
- Geef de speler ruimte om de regels te leren kennen.
- Daag de speler uit maar maak het niet te moeilijk
- Achievements kunnen, wanneer goed ontworpen, een hele grote toevoeging zijn voor een game, maar het kan ook de speler negatief beïnvloeden wanneer achievements slecht ontworpen zijn. Achievement design eist daarom veel aandacht.

Wat betekent dit voor Yezzle.nl?
Er komt een stroom van klachten binnen van mensen die niet tevreden zijn over wat zij terug krijgen voor hun abonnementsgeld. Dit zou kunnen komen door het feit dat alles wel werkt, maar er zijn maar weinig features die goed werken. Bijvoorbeeld

- de MindIndex is een component die op het moment alleen een getal laat zien zonder dat het duidelijk is wat dit getal betekend.
- De trainingsprogramma's geven geen feedback over hoe de speler gegroeid is, waardoor het de speler niet motiveert om te beginnen aan een trainingsprogramma.
- De spaarzegels (achievements) die gehaald kunnen worden betekenen niks, het kan namelijk niet gedeeld worden met andere spelers en de speler krijgt er niks voor terug.
- De huidige achievements zijn te generiek, dit verlaagd de intrinsieke motivatie van de speler om deze te willen unlocken.
- Verschillende games zijn of te moeilijk, of te makkelijk of de spelregels zijn niet snel genoeg te leren. De casual gamer heeft hier in alle gevallen geen geduld voor.

Op het moment is er in principe niemand in staat om deze problemen op te lossen. In de meeste gevallen hebben de werknemers hier de kennis niet voor. Er zijn echter wel een paar werknemers die een goed gevoel hebben voor het oplossen van dit soort zaken, maar deze werknemers hebben het te druk met hun eigen werkzaamheden. Ook kunnen deze problemen niet opgelost worden aan de hand van de kennis die dit hoofdstuk de lezer geeft. Er is een poging gewaagd om enkele games van Yezzle.nl te verbeteren aan de hand van het analyseren van de games en het uitvoeren van playtests, de resultaten van twee van deze analyses zijn in bijlage B te zien. Deze analyses zijn alleen niet voldoende om te kunnen zeggen dat de games na het aanpassen erg leuk zijn geworden. Daarom moet er een betere oplossing komen voor het *fun* maken van Yezzle.nl

Oplossing
Game design is het vakgebied binnen de game industrie die zich volkomen bezighoud met de bovenstaande kwesties. Een game designer is degene die de concepten verzint en deze vervolgens tot in het detail uitwerkt. Hij is gespecialiseerd in het *fun* maken van een game. Op het moment is er niemand bij Keesing Games aanwezig die zich volkomen bezig houdt met game design. Het zijn voornamelijk grafische vormgevers die een deel van de taken van game design op zich nemen en daarnaast worden er ook een hoop game design kwesties over gelaten aan de programmeurs die de games ontwikkelen. Het nadeel van game design keuzes overlaten aan programmeurs is dat zij gewoonlijk een hele andere kijk hebben op games. Programmeurs beredeneren zaken vaak erg technisch, waarbij zij ook te snel geneigd zijn ideeën te verwerpen omdat het realiseren

technisch lastig zou zijn. Het is in eerste instantie essentieel om bij het designen van games je helemaal open te stellen voor alle ideeën die tot je komen. Programmeurs en grafische vormgevers die veel games spelen kunnen wel instaat zijn om goede games te designen, maar het gevaar is dat men heel snel geneigd is een game te maken die zij zelf leuk vinden, wat niet betekend dat het doelgroep het ook leuk gaat vinden. Een deel van de taken van een game designer is zich specialiseren in de doelgroep. Hij gaat verder dan alleen weten wie de doelgroep is. Hij verdiept zich in de games die de doelgroep leuk vind om te spelen en past hier zijn denkwijze en de games op de juiste manier op aan.

Mijn advies om het plezier voor Yezzle.nl te verhogen is dan ook om een game designer aan te nemen. Het hebben van een game designer brengt veel voordelen met zich mee. Een paar van deze voordelen zijn:

- Het haalt het schrijven van concept documenten weg van de grafische vormgever, wat hem meer tijd geeft om met zijn eigen werkzaamheden gefocust bezig te gaan.
- Het haalt game design beslissingen uit de handen van programmeurs die niet altijd de tijd hebben om hier mee bezig te zijn en niet altijd de juiste kennis in huis hebben om de juiste game design beslissingen te maken.
- De team lead van het game development team is op het moment degene die de games configureert terwijl hij een erg hoge workload heeft. Een game designer in huis hebben haalt deze taak uit zijn handen, dit geeft hem meer tijd om op een effectief manier de development team aan te sturen en met zijn andere werkzaamheden bezig te zijn.
- Omdat een game designer zich bezig houd met playtesten kan men ervan uitgaan dat de games op een logisch manier, op de doelgroep gericht en op ervaring gebaseerd geconfigureerd worden. Dit komt ten goede van de speelervaring.
- Normaal gesproken maken game designers prototypes van hun designs. Aan de hand van dit prototype kan de game designer al bezig zijn met playtesten. Zo kan een designer, voordat een game ontwikkeld wordt, controleren of het concept ook daadwerkelijk leuk zal zijn om te spelen en, wanneer nodig, kan hij het design document aan de hand van de resultaten aanpassen om het leuker te maken.
- De game designer overlegd met management over wat voor type game gemaakt moet worden, wat de reden van de game is en er wordt eventueel een richting van de stijl overlegd. Na dit overleg moet er zoveel mogelijk over gelaten worden aan de game designer. De game design beslissingen worden, als het goed is, met hele specifieke redenen gemaakt en kunnen beargumenteerd worden. Management heeft vaak niet de juiste kennis in huis om onderbouwde kritiek te geven en moet dit dan ook zo min mogelijk doen. Uitzonderingen nagelaten.
- De documentatie voor games gaat op het moment alleen zo ver als het uitschrijven van het concept, dit betekent dat er nog veel game design beslissingen open staan. Het gevolg hiervan is dat de developers eigenlijk nooit verantwoordelijk gesteld kunnen worden voor hun werk. Wanneer een game niet leuk is om te spelen, dan hebben zij als excuus dat zij geen game designers zijn. Wanneer een game niet compleet is, dan kunnen zij zeggen dat het concept bepaalde elementen niet omschrijft. Dit betekent dat zij een goed excuus hebben voor het afleveren van een niet complete game. Een game designer aannemen heeft als voordeel dat hij verantwoordelijk gesteld kan worden voor het *leuk* maken van de games. En doordat een game designer uitgebreide documentatie met zich mee brengt, betekend dit dat de developers verantwoordelijk zijn voor het *goed* maken van de game, zoals omschreven is door de game designer. Uit een quality assurance perspectief is dit erg belangrijk.
- Het hebben van een game designer brengt game design documenten met zich mee. Dit is een groot voordeel voor het vak testen. Hoofdstukken 12.1 en 13.1 gaan dieper in over hoe deze design documenten gebruikt worden voor het testen van software.

Nadelen?

Het aannemen van een game designer zou enkele nadelen kunnen hebben, deze worden hieronder besproken:

1. Het meest voor de hand liggende probleem zijn de kosten van het fulltime aannemen van een game designer.
2. Veel van de werknemers vinden het leuk om bezig te zijn met het vak game design tijdens het maken van de game.
3. Mogelijk niet genoeg werk binnen het Yezzle projectteam.

De voordelen wegen zwaarder dan de nadelen. Hier volgen de redenen waarom:

1. Te duur: De vraag is, hoeveel geld raakt men kwijt door producten op te leveren waarvan de kwaliteit erg laag is? Een game designer in huis hebben stroomlijnt het development proces. De workload van praktisch alle developers word verminderd doordat zij niet meer bezig hoeven te zijn met game design. Dit betekent dat er meer tijd vrij komt om bezig te zijn met het ontwikkelen van games. Daarnaast is het programmeren van een game aan de hand van een compleet game design document een erg efficiënte werk methode. Bijna alle middelklein tot grote game developers maken gebruik van dedicated game designers, het is dan ook een volledig bewezen manier van te werk gaan. Een game designer is het geld waard, doordat hij een verhoging in kwaliteit en kwantiteit met zich mee brengt.

2. Ontwikkelaars vinden game design leuk: Vormgevers en software developers zijn gewoonweg niet de juiste personen voor het maken van game design beslissingen. Grafische vormgevers zijn te gefocust op het mooi maken van een game en developers zijn te gefocust op het werkend krijgen van een game. Er is juist iemand nodig die vanuit een *fun* perspectief nadenkt, dit is wat een game designer in essentie doet. De development team hoeft niet bang te zijn dat zij nooit meer bezig zijn met de *fun* kant van zaken, want de game designer en de developer werken nauw samen. Er hoort altijd ruimte voor discussie te zijn. Wanneer iemand een goed idee heeft voor een game die in development is, dan kan dit in overleg met de game designer in het design document opgenomen kunnen worden.

3. Niet genoeg werk: Binnen het projectteam van Yezzle zou het zo kunnen zijn dat er niet genoeg werkzaamheden zijn voor een fulltime game designer. Maar verspreid over alle lopende en toekomstige project is er *meer dan genoeg* werk. Beide Zigiz.com en Yezzle.nl zijn game portals waarbij er een constante stroom van nieuwe games in ontwikkeling zijn. Het schrijven van game design documenten, het prototypen, het playtesten en het configureren van de games zullen zeker 40 uur in de week vol kunnen maken. Vooral als de andere platformen die in ontwikkeling zijn meegenomen worden is er ruim genoeg werk om fulltime een game designer in te huren.

Onderzoeksvraag

In hoofdstuk 2.2 zijn er onderzoekvragen opgesteld waaronder de volgende vraag:

- *Deelvraag 2: Hoe zou je de playability en replayability van Yezzle kunnen verbeteren?*

Het antwoord op deze vraag zoals hierboven te lezen is: Neem een game designer aan.

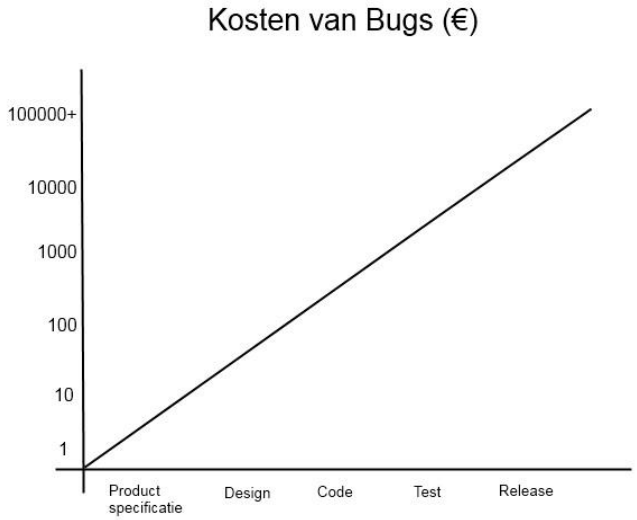
Tot die tijd kan al het gegeven informatie in hoofdstuk 5 ter ondersteuning gebruikt worden om het platform meer *fun* te maken. Ook kunnen de analyses van Microsoft Solitaire en Bejeweled Blitz in bijlage A-1 en A-2 gebruikt worden om te zien hoe deze twee succesvolle games in elkaar zitten en vervolgens deze goede elementen toepassen voor toekomstige games van Yezzle. Daarnaast zijn de geleerde lessen van deze twee games en het bovenstaande informatie toegepast in de vorm van analyses en playtests van de Braintrainers van Yezzle.nl. In bijlage B zijn de resultaten van twee van deze analyses / playtests te zien. Deze resultaten kunnen als een tijdelijke oplossing dienen. De overige analyses worden of zijn al overhandigd aan de desbetreffende personen binnen het Yezzle team.

6 Testing - Wie test wat en wanneer?

Het testplan is een document geworden waarin de verschillende aspecten, procedures en handelingen van het testen van worden beschreven. Aan de hand van dit document zouden de werknemers die aan Yezzle.nl meewerken de kwaliteit van hun producten kunnen waarborgen door middel de test technieken aan te nemen die in dit document besproken worden. De komende hoofdstukken geven dezelfde informatie als het testplan, maar dan op een ander manier gepresenteerd.

Als eerst wordt nu het probleem besproken dat men niet weet wie, er wat en wanneer moet testen.

In 2002 was er een onderzoek uitgevoerd door RTI International waar uit voort kwam dat software bugs de Amerikaanse economie jaarlijks 59.6 miljard dollar kost (onderzoek is onlangs offline gehaald, geen referentie). Het is essentieel geworden om zo snel mogelijk bugs te vinden en te repareren. De kosten van bugs kan zich met een veelvoud van 10 vermenigvuldigen, hoe later het in de productie cyclus gefixt wordt hoe meer het gaat kosten. Hieronder wordt een beeld geschetst van de kosten voor het fixen van bugs uitgezet tegen de productiefase.

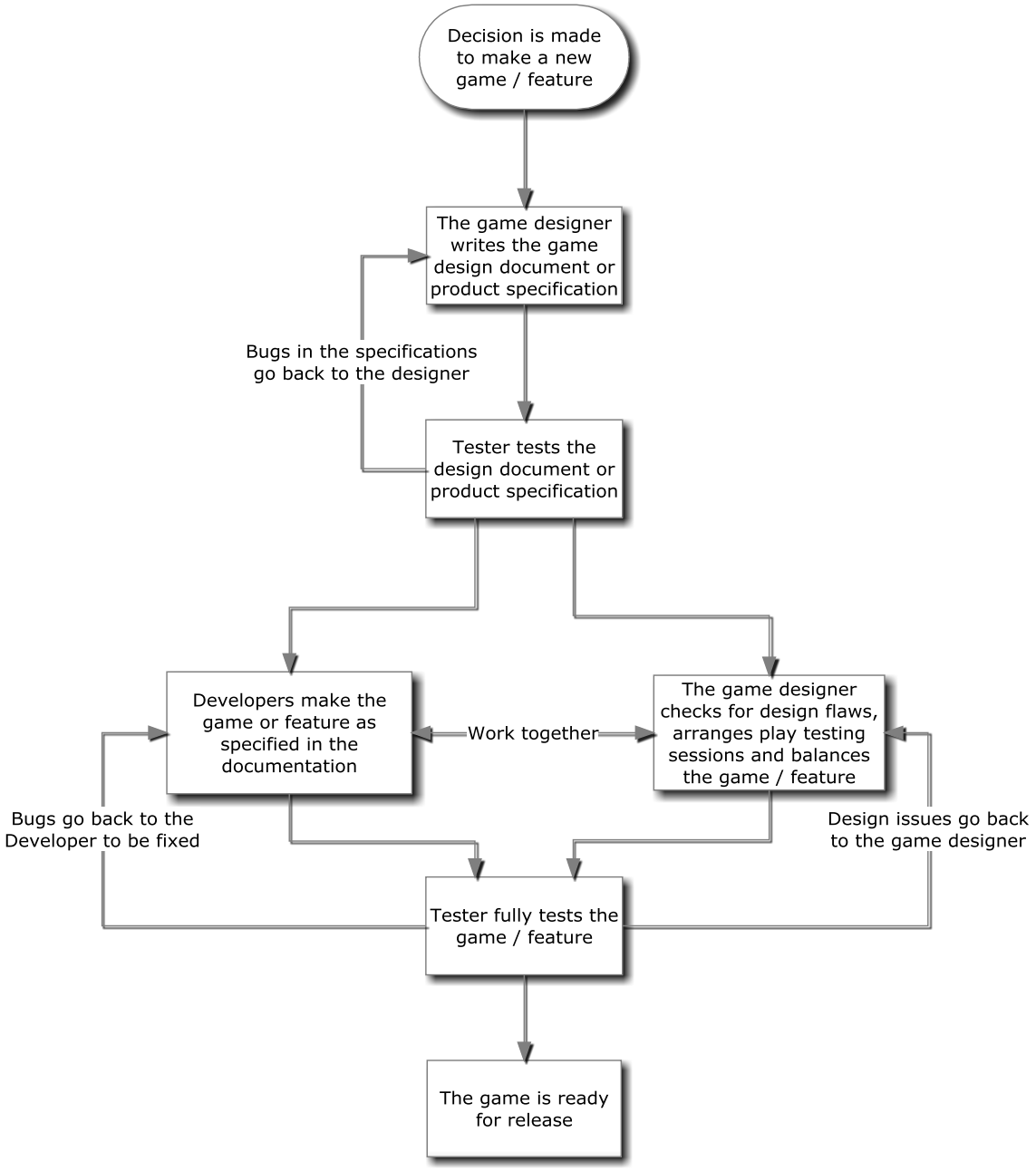


Figuur 6-6-1 Een globaal beeld van toenemende kosten van bugs

In extreme gevallen kosten software bugs meer dan alleen geld. Een voorbeeld hiervan is de Patriot Missile Defense System. Het systeem vormt een schild tegen inkomende raketten, zoals de Scud raketten gebruikt door Sadam Hussein tijdens de eerste golf oorlog. Een kleine timing error in het systeem klok zorgde ervoor dat het tracking systeem na 14 uur niet meer nauwkeurig was. Dit resulteerde in de dood van 28 U.S soldaten doordat er een raket niet uit de lucht was geschoten vanwege de bug in de software.

Om er dus voor te zorgen dat het product zo vrij mogelijk van bugs is en om de kosten laag te houden is het verstandig om al bij het opstellen van het product specificatie te beginnen met testen. Hoe je dit aanpakt wordt in het hoofdstuk “Te weinig documentatie” en “Static black-box testing” besproken.

Wanneer je ervan uit gaat dat er geen tester aanwezig is, dan moet deze werkzaamheden door de rest van het team opgevangen worden. In hoofdstuk 3 is de huidige flow van de ontwikkeling voor Yezzle weergegeven (figuur 3-1), deze flow is niet optimaal omdat het testen van de producten niet meegenomen wordt in de ontwikkeling van de games en features. De verbeterde ontwikkel cyclus, waarin testen en meer documentatie meegenomen wordt, is te zien in figuur 6-2.



Figuur 6-6-2 Effectieve ontwikkelproces

In dit ontwikkelproces zijn het schrijven van product specificaties en game design documenten en het testen van de producten meegenomen als onderdeel van het development proces. Hoe vroeger een bug gevonden wordt hoe minder het kost om deze op te lossen (zie figuur 7-1).

Er is te zien dat er vergeleken met figuur 3-1 veel meer terugkoppeling plaats vindt, dit komt ten goede van de kwaliteit en de kosten van de producten.

Er zijn twee test momenten te zien in de flowchart. Eén na het schrijven van de productspecificaties en één test moment nadat de game/feature volgens een developer af is. Maar wanneer er tussen de regels gelezen wordt dan zien is er te zien dat er meerdere test momenten zijn. Namelijk de developers moeten hun producten zo

uitvoerig mogelijk testen voordat zij deze aangeven als *klaar.* Het is namelijk niet de bedoeling dat de tester gebruikt wordt om te controleren of de feature wel echt af is. Een ander test momenten is wanneer er bugs gevonden worden in de documentatie of software en deze gefixt zijn. Na het fixen vind er weer een test moment plaats, namelijk het verifiëren of de bug daadwerkelijk opgelost is en of de oplossing geen nieuwe bugs veroorzaakt heeft. Voor een nader toelichting op het verificatieproces zie het hoofdstuk "Verificatieproces".

Normaal is de tester degene die verantwoordelijk is voor het testen van de productspecificaties en producten. Wanneer er geen tester aanwezig is binnen het bedrijf, dan moet iemand anders deze taken oppakken. Het testen van de documentatie kan het beste gebeuren door de product owner en door de website project manager, als het om een website feature gaat, en door de team lead games development, wanneer het een game betreft.
De product owner controleert of de specs voldoen aan de visie van het platform. De website manager of de lead games development controleert op een meer technisch niveau of de specificaties helder en compleet zijn. In het hoofdstuk "Hoe test ik? – Static black-box testing" wordt er dieper in gegaan op het testen van documentatie.

Na het testen van documentatie is de volgende test moment het testen van het gemaakte product. Degene die de productspecificaties of het design document geschreven heeft is het beste instaat om de game of feature te testen. Hij of zij weet het beste wat het hoort te doen en is daardoor het beste instaat om goed te kunnen controleren of de software daadwerkelijk doet wat het hoort te doen. Het voorgaand hoofdstuk "Playtesting" geeft informatie over het controleren van de playability van een game en het hoofdstuk "Hoe test ik? – Dynamic black-box testing" geeft manieren om software te testen.

7 Testing - Waar en wat moet er getest worden?

De nieuwe development methode die gebruikt word bij Yezzle, OTAP, zorgt voor strakke regels in het proces van een nieuwe game of feature online te krijgen. Elke stap binnen de procedure heeft zijn eigen omgeving, in dit geval zijn dat websites. Hieronder wordt weergegeven waar elke stap toe dient.

> Ontwikkel
  - De software developer creëert en test zijn software op een ontwikkel systeem. De ontwikkelingsserver dient voor het testen van nieuwe features en games binnen de Yezzle framework. Hier kunnen en mogen fouten op staan.

> Test
  - Wanneer de developer vindt dat het programma of deel van een programma klaar is om getest te worden, dan wordt dit overgezet naar een testomgeving waar het door een tester op bugs gecontroleerd wordt. Deze testomgeving moet goed gedefinieerd zijn en moet zo veel mogelijk lijken op de uiteindelijke productie omgeving.

> Acceptatie
  - Wanneer alles succesvol getest is dan wordt de testomgeving gekopieerd naar de acceptatie test omgeving. Tijdens de acceptatie test wordt het product getest door de klant of gebruiker om er zo achter te komen of het voldoet aan de verwachtingen van de klant / opdrachtgever. In het geval van Yezzle is het zo dat Keesing Games zelf de opdrachtgever is. Dit houdt dus in dat acceptance testing intern moet gebeuren of door een panel van gebruikers.

> Productie
  - Wanneer de klant het product accepteert dan word het uitgebracht op de productie omgeving (de live server). Deze omgeving is toegankelijk voor alle gebruikers van de software.

Figuur 7-3 laat de flow van zien van het OTAP methode.

Op het moment zien de verschillende servers er zo uit voor Yezzle:

- Ontwikkel server website development: http://test.nintec.yezzle.keesinggames.net/
- Ontwikkel server flash development: http://test.flashteam.yezzle.keesinggames.net/
- Test server: http://test.yezzle.keesinggames.net/
- Acceptatie server: http://acceptatie.yezzle.keesinggames.net/
- Live server: http://www.yezzle.nl/

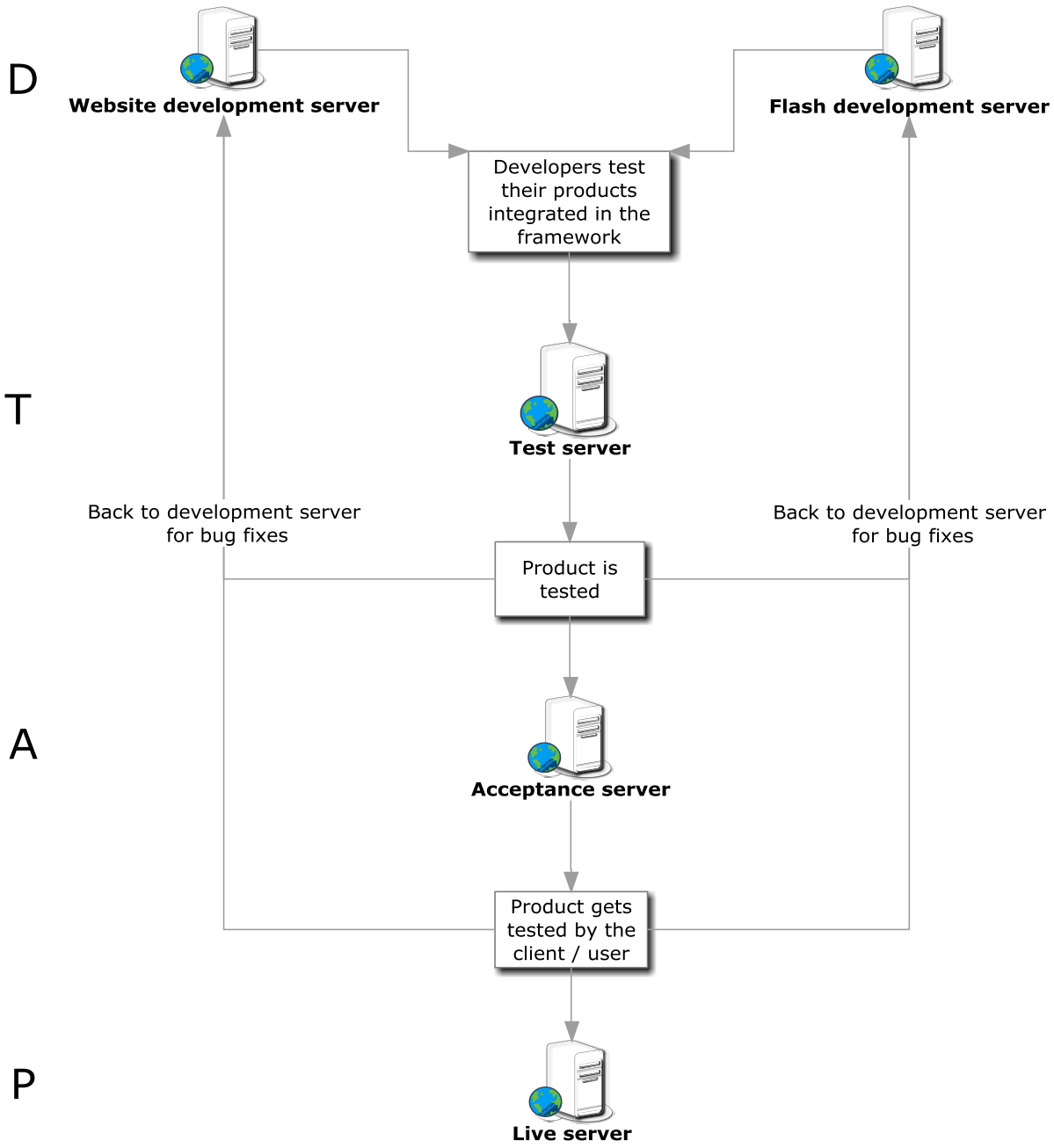
Hiervan zijn alle servers, op de live server na, alleen op het interne netwerk te bereiken.

Note: *De ontwikkel, test en acceptatie servers zijn zitten op dezelfde fysieke server*

Binnen de OTAP ontwikkelmethode is het de bedoeling dat het grootste deel van de testwerkzaamheden op de test server gebeurd. Het is belangrijk dat men zich hier aan houd zodat er overzicht is wat betreft welke onderdelen wel getest zijn en welke nog getest moeten worden. Zodra iets op de acceptatie server gezet is dan wordt dit als getest beschouwd en moet het alleen nog door de klant bekeken worden of de nieuwe elementen gemaakt zijn zoals gespecificeerd.

7.1.1 Hotfix

Het kan voorkomen dat er een ernstige bug gevonden wordt die direct gefixt moet worden. In deze uitzonderlijke gevallen is het mogelijk om onderdelen van de OTAP procedure over te slaan om het zo snel mogelijk live te krijgen. Dit heet een hotfix en moet alleen toegepast worden voor bugs die de gebruikerservaring ernstig aantast, zoals een game die constant crashed.



Figuur 7-7-1 Ontwikkel methode OTAP

8 Testing - Slechte bug rapportages

Het helder en compleet weten te omschrijven van bugs is één van de meest belangrijke elementen van game / software testing. Het uitgebreid rapporteren van een bug kan eventueel veel tijd kosten, maar dit is het altijd waard ten opzichte van alle tijd die verloren gaat door constante miscommunicaties. Hoe gedetailleerder een bug report is hoe makkelijker het zal zijn voor een developer om een bug te fixen.

Bug reports zijn niet alleen voor de developer bedoeld die het element gemaakt heeft. Het gebeurd vaak dat developers problemen moeten oplossen van features die zij zelf niet ontwikkeld hebben, daarnaast moet het probleem ook begrepen kunnen worden door de producers en ander management zodat zij de planning hierop kunnen aanpassen. In dit hoofdstuk worden goede en slechte voorbeelden gegeven van de title, description en steps to reproduce en worden de randvoorwaarden ook besproken zoals versie, repeatability en extra documentatie.

8.1 Mantis

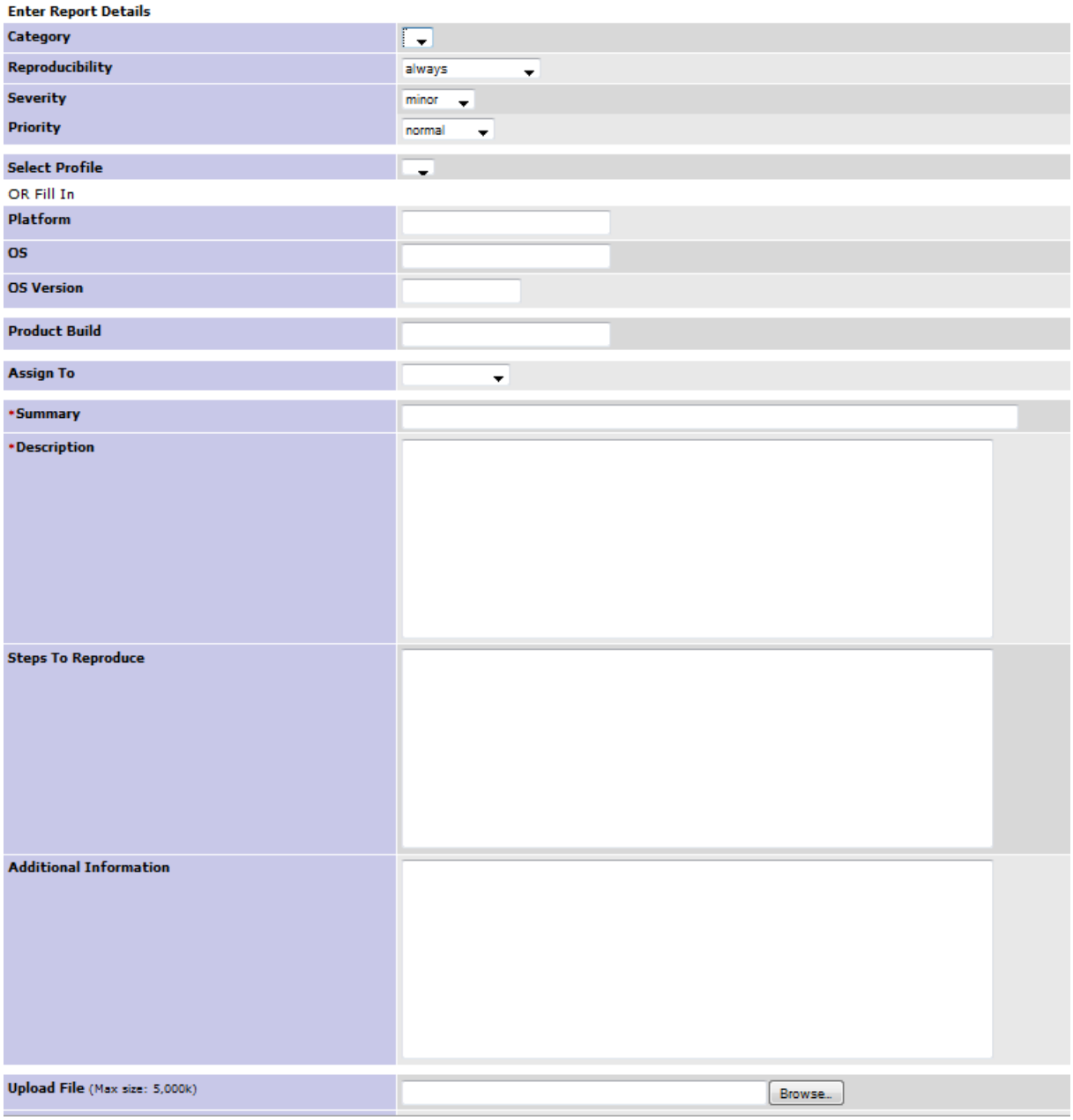
Wanneer er een bug gevonden wordt, dan is het aan de tester om hier een helder en compleet rapport van te maken in de bug tracking software die door het bedrijf gebruikt wordt. Mantis, Jira, DevTrack, TestTrack en Bugzilla zijn een paar voorbeelden van dergelijke bug tracking software. Elk bug tracking software packet werkt op zijn eigen manier, maar voor het rapporteren van een bug zullen de meeste pakketten informatie over de volgende elementen vragen:

- Category
- Reproducability
- Severity / Priority
- Test environement
- Assign to
- Summary
- Description
- Upload screenshot

De stappen die genomen moeten worden om effectief een bug te reporten zijn als volgt:

- Vind een bug
- Neem notities van wat er fout gaat
- Ga op zoek naar de stappen om het te reproduceren
- In de bug tracking software, selecteer het juiste project
- Zoek in de database naar soortgelijke bugs, zodat er geen dubbele entries ontstaan
- Voeg alle hierboven genoemde informatie toe in de bug report
- Voeg, als dit nodig is, screenshots toe
- Na het versturen, check of de bug is opgeslagen in het systeem

Op de volgende pagina is een screenshot te zien van de bug report pagina van de bug tracking software “Mantis”. Vervolgens wordt er dieper in gegaan op hoe bugs effectief gerapporteerd moeten worden.
Per onderdeel wordt dit geillustreerd aan de hand van 4 voorbeelden. De eerste drie voorbeelden komen uit het boek *Game Testing: All in One* geschreven door C.P. Schultz, R Bryant en T. Langdell en het laatste voorbeeld is een daadwerkelijk slecht gerapporteerde bug uit de bug tracking software van Yezzle.



Figuur 8-1 Bug tracking software Mantis

Voor een overzicht van verschillende bug tracking software pakketen zie:
http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems

8.2 Title

Een goede titel van een bug report zorgt ervoor dat het, zoals bij een krantenkop, dat het direct duidelijk wordt voor de lezer wat er aan de hand is. Het is dan ook slim om de vijf W's te gebruiken die in de journalistiek gebruikt worden: wie, wat, wanneer, waar en waarom.[5]

Slechte bug titels

a) Fell through the ground
b) Shotgun sucks
c) Can't load level
d) Schokkerige graphics

Waarom het niet voldoet

a) Er staat niet waar het gebeurde.
b) Dit is een mening, die ook nog eens incompleet is.
c) Er wordt niet gezegd over welk level het gaat
d) Het is niet in het Engels geschreven en er wordt niet gezegd waar en / of wanneer er schokkerige graphics zijn.

Betere, gedetailleerdere titel

a) Fell through the ground when crossing the hangar towards the exit
b) Allied shotgun is underpowered when compared to Axis shotgun
c) Can't load Poisson map in Capture the Flag game mode.
d) Golven - There is a noticeable frame drop after playing about 15 levels

Als we de verbeterde versie van voorbeeld 'd' bekijken dan zien we dat vier uit de vijf w's beantwoord worden.

- Wie / Waar? De game Golven
- Wat? Te merken frame drop
- Wanneer? Na 15 levels gespeeld te hebben

De vraag 'waarom?' wordt is meestal door de developer beantwoord, de vier andere w's helpen hem bij het vinden naar de 'waarom' vraag. Wanneer de tester met zekerheid de 'waarom' vraag toch kan beantwoorden dan is het belangrijk dat dit vermeld wordt. Dit wordt gewoonlijk in de description vermeld.

Om het navigeren in Mantis overzichtelijker te maken kan de titel van de bug het beste met de naam van de betreffende game of feature beginnen. Dit maakt het opzoeken van issues in Mantis veel makkelijker.

[5] http://en.wikipedia.org/wiki/Five_Ws

8.3 Description

Zoals te zien is in de vorige paragraaf dient de titel een één regel samenvatting te zijn. In de description ga je verder in op de bug en leg je het in detail uit.

Slechte omschrijvingen

a) I was walking through the level when suddenly my char fell through a humongous hole in the ground. He kept falling for ages until he died.
b) I can't stand the shotgun. I shoot from really close and the other guy doesn't die.
c) Every time I try to load the Poisson level, something happens and the screen goes dark.
d) Waarschijnlijk afrondingsfouten op pixels.

Waarom het niet voldoet

a) Deze omschrijving mist details, het maakt gebruik van onnodige afkortingen ("char" in plaats van character) en er staan onnodige woorden("humongous" en "for ages").
b) Deze omschrijving is een mening.
c) In deze omschrijving staat wel de naam van de level waarin het voorkomt, maar het is gewoonweg niet genoeg informatie voor een developer om het op te gaan lossen.
d) De omschrijving is niet in het Engels en de omschrijving is compleet onduidelijk, waarschijnlijk is het mondeling gecommuniceerd naar degene die het moet fixen. Door vakanties, ziektes en prioriteiten die bij andere projecten liggen gebeurd het vaak dat bugs worden opgelost door een ander developer dan degene die de software heeft geschreven. Schrijf de bug altijd zodat iedereen het kan begrijpen.

Betere omschrijvingen

a) Crossing the Hangar, about to finish the Enemy Fields level, I suddenly fell through a map hole. Facing the exit, the hole is about two feet to the left of the alien jet engine. I kept falling until my character finally died and I was taken back to the Load screen.
b) The Allied shotgun seems to be underpowered in comparison to the Axis shotgun. If I stand four feet from an enemy soldier and fire, it takes me two or three hits to kill him. If he shoots me from the same distance, he can kill me with a single shot. Other testers have been reporting the same issue in the past week or so.
c) The Poisson level failed to load 3 out of 10 times. This seems to happen a lot in Capture the Flag mode. We had a team of 16 in the lobby, all set, when two other machines got stuck on a black screen. (we readied at roughly the same time)
d) When you play around 15 levels of Golven without going game-over you will notice that there is frame drop (depending on how good your pc is). The higher the level, the easier it is to notice this frame drop.

De nieuwe omschrijvingen hebben veel meer detail. Zo heeft "a" nu een exacte locatie voor de developer om de map hole te vinden. "b" heeft nu een feitelijke reden waarom de Allied shotgun overpowered is en dus een bug is. "c" geeft nu alle informatie die de tester kan hebben over de situatie, ook al kan de tester niet precies zeggen wat er fout is, dit geeft de developer genoeg informatie om te weten waar hij moet zoeken in zijn code op fouten.

8.4 Steps to reproduce

De steps to reproduce zijn de instructies die iemand zou moeten uitvoeren om de bug te kunnen reproduceren. Dit is een onderdeel van bug reporting waarvan het essentieel is dat het goed gebeurd. Er valt over te discussiëren, maar in het algemeen worden de “steps to reproduce” als belangrijkst element van een bug report gezien.

Slechte steps to reproduce

a) 1. Walk toward exit.
   2. Die.
b) 1. Load game.
   2. Equip shotgun.
   3. Shoot anyone.
   4. Nothing happens.
c) 1. Boot game.
   2. Select multiplayer.
   3. Attempt to load Poisson.
d) No steps to reproduce

Waarom het niet voldoet

a) Te simplistisch. Het klinkt eerder als een grap.
b) Er worden stappen gegeven maar dit zijn niet de stappen die men moet uitvoeren om het probleem te reproduceren.
c) Hier wordt geen nieuwe informatie toegevoegd.
d) Steps to reproduce zijn van essentieel belang voor het goed overdragen van problemen die gefixt moeten worden. Geen steps to reproduce is altijd een no-go.

Betere steps to reproduce

a) 1. Load Enemy Fields level
   2. Advance through level following the usual path.
   3. Close to the end of the level, you’ll see an alien jet engine.
   4. Position your character about two feet to the left of the jet engine.
   5. You’ll fall through a map hole.
b) 1. Load any multiplayer level.
   2. Have another tester join in.
   3. Equip the Allied shotgun.
   4. Have the other tester equip the Axis shotgun.
   5. Have the other tester shoot you in the chest. You should die with 1 hit.
   6. Once you respawn, shoot the tester from the same distance. It will take 2 or 3 hits to kill the other tester’s character.
c) 1. Boot game.
   2. Start a multiplayer match in Poisson.
   3. Change the game mode to Capture the flag.
   4. While in the lobby, have two testers join your game at the same time.
   5. You should get a black screen 3 out of 10 times.
d) 1. Log in on the live server (yezzle.nl).
   2. Start Braintrainer Golven.
   3. Choose a level on which you do not go game over.
   4. Play this level until you notice a drop in the frames per second.

Waarom “Steps to reproduce” zo belangrijk zijn

De “Steps to reproduce” moeten ervoor zorgen dat de developers de bugs perfect kunnen reproduceren vanaf hun eigen werkplek. Als productie tester heb je meestal de luxe dat jij en de developer in het zelfde gebouw zitten en zou je problemen mondeling kunnen aanvullen. Het kan echter voorkomen dat delen van opdrachten ge-outsourced worden naar andere bedrijven. Ook als je test voor de uitgever zit je zo goed als nooit in hetzelfde gebouw als het development team. Wanneer de steps to reproduce niet duidelijk genoeg zijn gaat er, zeker als men niet in hetzelfde gebouw zit, te veel tijd zitten in het opzoeken van de bug door de developer zelf. Naast dat bugs oplossen hierdoor veel tijd kost, betekent het ook minder tijd voor het creëren van nieuwe features, wat het bedrijf in beide gevallen onnodig geld kost.

8.5 Version

Het developen van games is een heel evolutionair proces waarbij het product constant aan het groeien en veranderen is. Dit betekent dat een groot deel van het testen niet in de final build plaats vindt. Daarom is het belangrijk om de versie waarin een bug voorkomt in het bug rapport te melden. Dit elimineert veel miscommunicaties doordat de developer zodoende in de juiste build van de game aan de slag kan gaan met het fixen van de bugs. In een ideale wereld zou de subversion revision (versiebeheer programma) nummer erbij gezet moeten worden, dit is alleen niet altijd mogelijk. Vermeld in elk geval de server waarop de bug gevonden is en vermeld hierbij ook de datum.

8.6 Repeatability

Het is belangrijk om te weten hoe vaak een bug gereproduceerd kan worden daarom is er is in de meeste bug tracking pakketten dan ook een gereserveerd invoerveld voor. Een bug die altijd te reproduceren is, heeft een hogere prioriteit dan een bug die maar 1 op de 10 keer voorkomt.

Het bijvoegen van de repeatability kan een developer erg helpen met het localiseren van de bug in de code. Controleer daarom altijd of de bug te reproduceren is en geef dit altijd aan. Mantis geeft de volgende opties:

- Always – Gebruik deze aanduiding wanneer een bug 100% van de tijd te reproduceren is.
- Sometimes – Gebruik deze aanduiding wanneer een bug niet altijd voorkomt. Wanneer deze aanduiding gebruikt wordt moet in de description sectie worden beschreven hoe vaak deze bug voorkomt per 10 keer testen.
- Random – Gebruik deze aanduiding wanneer een bug niet altijd voorkomt en wanneer er ook geen enige regelmaat te vinden is. Noteer goed de omstandigheden in de omschrijving waaronder het probleem wel gevonden is.
- Have not tried – Gebruik deze aanduiding alleen wanneer er geen tijd is om na te gaan of het te reproduceren is. Deze aanduiding moet, tot zo ver dit kan, vermeden worden.
- Unable to reproduce – Gebruik deze aanduiding nadat er meerdere malen op verschillende manieren geprobeerd is om de fout te kunnen reproduceren. Geef ook weer in de description aan hoe vaak het geprobeerd is en op wat voor manier(en). Dit soort informatie elimineert mogelijke oorzaken voor de developer waardoor het makkelijker wordt voor hem om een oorzaak te vinden.

8.7 Bug classification Mantis

Om voor efficiëntie binnen de bug cyclus te zorgen is het belangrijk dat bugs worden gerangschikt naar prioriteit. Dit wordt gedaan door middel van de zogenoemde “Severity” aan te geven in Mantis waarbij men ook de prioriteit binnen deze categorie kan aangeven met het “Priority” veld. Zo kan iedereen die toegang heeft tot de bug tracking database zien welke problemen er aanwezig zijn en welke volgorde de bugs gefixt moeten worden.

In een ideale situatie is het zo dat de lead tester in combinatie met de lead developer en producer de zwaarte en prioriteiten van de verschillende bugs bepalen. In het geval van Yezzle is dit niet zo. Daarom moeten nieuwe issues eerst naar de project manager doorgespeeld worden als het om website bugs gaat en moeten game gerelateerde issues eerst doorgespeeld worden naar de team lead games development. Deze twee personen bepalen de zwaarte en prioriteit van de bug en assignen vervolgens de issue door naar de relevante developer of artist die het moet gaan fixen.

De bug tracking software Mantis geeft acht verschillende classificaties en vijf verschillende prioriteiten van bugs waaruit men kan kiezen. Hieronder is een kort overzicht te zien van de verschillende classificaties en prioriteiten.

Bug classificatie	Te gebruiken bij
Feature	Project management, speciaal bedoeld voor feature requests.
Trivial	Kleine schoonheidsfouten die de gebruikerservaring niet beïnvloed, alleen fixen wanneer er geen hogere prioriteit bugs zijn.
Text	Te gebruiken voor tekst gerelateerde problemen. B.v. tekstfouten en vertalingfouten (localisation).
Tweak	Te gebruiken als het om verbeteringen gaat van werkende onderdelen.
Minor	Te gebruiken bij opvallende fouten die de flow van de gebruiker *niet* breekt.
Major	Te gebruiken bij opvallende fouten die de flow van de gebruiker van een specifiek onderdeel *wel* breekt.
Crash	Te gebruiken bij kapotte elementen die de gebruikerservaring ernstig belemmerd.
Block	*Alleen* te gebruiken bij *ernstige* problemen die direct opgelost moeten worden. (werknemers moeten doorwerken tot dit probleem opgelost is)

De verschillende prioriteiten zijn Low, Normal, High, Urgent en Immediate. Deze prioriteiten moeten per issue aangegeven worden om de volgorde van fixen / developen aan te geven. De prioriteiten zijn voor de hand liggend, zo geeft de prioriteit Low aan dat iets een lage prioriteit heeft en Immediate geeft aan dat een bug direct opgelost moet worden.

8.8 Additional documentation

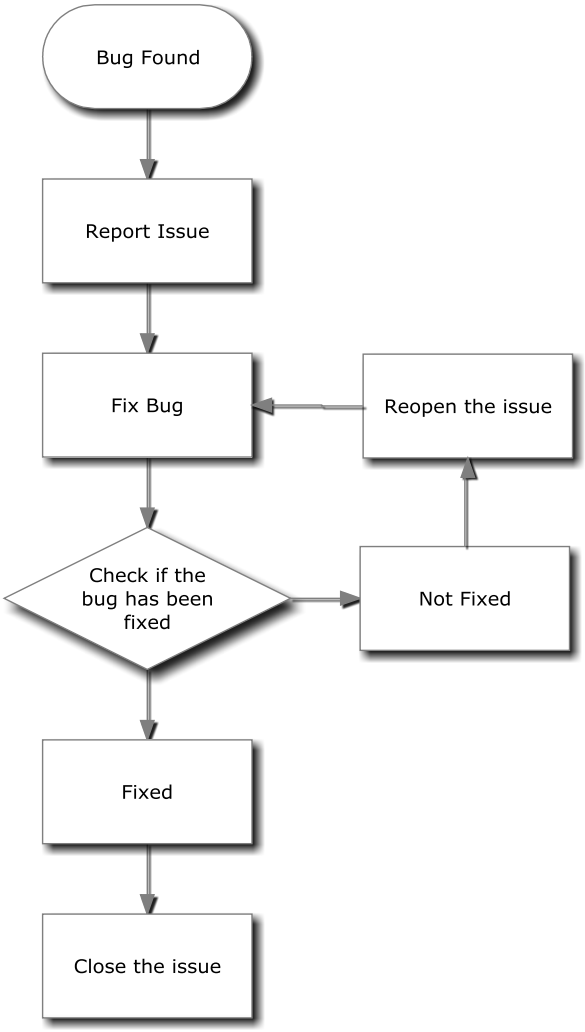
Bij het toevoegen van een bug in de tracking software is het belangrijk om zo veel mogelijk informatie te geven. Één van de belangrijkste onderdelen van bugs rapporteren zijn de stappen om het te reproduceren. Een probleem is niet altijd even makkelijk om met woorden te omschrijven. Vandaar dat er in de meeste bug tracking software een optie bestaat om bestanden aan een issue toe te voegen. Wanneer nodig en/of mogelijk; voeg screenshots toe, voeg een filmpje toe, voeg log bestanden toe, voeg de productspecificatie toe, etc. Hoe meer informatie gegeven kan worden aan de developers hoe makkelijker hun werk wordt.

9 Testing - Verificatieproces

Wanneer er een bug in de tracking database ingevoerd wordt dan wordt het doorgestuurd naar de nodige persoon die het vervolgens gaat fixen. In de tracking software krijgt de bug de status “New” wanneer het net ingevoerd is en wanneer de bug aan iemand toegekend is om te fixen dan krijgt het de status “Assigned”.

Wanneer de bug gerepareerd is, en de status van de bug op “Resolved” is gezet, dan komt de bug weer terug naar de tester om te controleren of de bug daadwerkelijk gefixed is, of dit op de juiste manier gebeurd is. Dit heet ook wel *regression testing*. Er moet, naast het controleren of de bug gefixt is, ook gecontroleerd worden of de fix geen andere bugs veroorzaakt heeft. Er moet dus met soortgelijke tests om de bug heeft getest worden, dit heet ook wel *halo-testing*.

Als de bug op de juiste manier gefixt is dan wordt de issue in de bug tracking software van “Resolved” op “Closed” gezet. Als een tester een bug closed dan link hij hierbij zijn naam aan de bug en draagt hij de eindverantwoordelijkheid hiervoor. Het closen van bug betekend dat de tester zegt dat het definitief gefixed is voor een bepaalde build. Het closen van een issue waarbij de bug niet opgelost is kan grote gevolgen hebben en dus is het verifiëren van bugs erg belangrijk.



Figuur 9-1 Bug verificatieproces

Echter, als het blijkt dat tijdens het verifiëren de bug toch niet goed opgelost is, dan re-opend de tester de issue. Dit gaat meestal gepaard met een notitie over wat er nog fout gaat. Bij het heropenen van een issue gaat het weer terug naar de developer die in eerste instantie de taak gekregen had om de bug te fixen. En moet de fix en verificatieproces doorlopen worden tot het probleem opgelost is.

10 Testing - Bug tracking software als project management

Bug tracking software hoeft niet alleen voor test doeleinden gebruikt te worden, ook is er de mogelijkheid om het als een project management tool te gebruiken. In dit hoofdstuk wordt kort beschreven hoe dit toegepast kan worden met een focus op de bug tracking software die op het moment voor Yezzle gebruikt wordt, Mantis.[6]

10.1 Huidige situatie

Voor het game development team wordt er op het moment niet of nauwelijks gebruikt gemaakt van Mantis. Er wordt aan een developer gevraagd om een game te maken en vervolgens wordt deze gemaakt. De voortgang van het ontwikkelproces wordt niet bij gehouden in Mantis. Verder is er ook geen centrale punt waar bij gehouden wordt wie waar mee bezig is.

Voor de taken die uitbesteed worden aan het bedrijf NinTec wordt er wel gebruik gemaakt van Mantis. Over het verloop van het afgelopen half jaar is men steeds meer Mantis gaan gebruiken wanneer het om taken gaan die door NinTec ontwikkeld moeten worden. Er wordt goed gebruik gemaakt van de feature request en sinds kort worden er ook product specificatie aan de rapporten toegevoegd. Alleen er wordt maar weinig commentaar geleverd in de feature requests. Het enige moment dat men gebruik maakt van de comment functie is wanneer er iets fout gegaan is. Meer documenteren kan er voor zorgen dat er minder problemen plaat kunnen vinden.
Het vervolg van dit hoofdstuk geeft weer hoe er effectiever omgegaan kan worden met Mantis voor de interne en externe team.

10.2 Feature requests

De eerste stap is het effectief gebruik maken van de feature classificatie. In het vorige hoofdstuk zijn de verschillende prioriteiten besproken, één van deze classificaties is de feature request.
Zodra er een nieuw element of game gemaakt moet worden dan moet deze in Mantis gezet worden onder de classificatie *feature*. Om een goed overzicht te krijgen in Mantis is het belangrijk dat de titel van een rapport in één opslag duidelijk maakt wat het precies betreft. Daarom moet de titel naast de omschrijving ook de volgende tags bevatten:

- Een projectfase tag
    - Ongoing - voor een rapport dat in de huidige projectfase verwerkt moet worden
    - Phase ‘x’ - om aan te geven in welke projectfase een rapport verwerkt moet worden
- Naam van de game wanneer het rapport om een game gaat
- Het type request
    - Functioneel – als het om functionele problemen gaat
    - Graphical – als het om grafische problemen gaat
    - Payments – als het om problemen met de betalingen gaat
    - Tweak – als het om een kleine aanpassing van een bestaand element gaat
    - General – als het meerde types problemen gaat
    - Feature – als het om een feature request gaat

Een goed voorbeeld van een titel uit de bug tracking software van Yezzle:

- *“Phase 2 - functional - former subscription users can still log in and have no play restrictions”*

Het betreft een functionele issue die in fase 2 opgelost moet zijn.

[6] Zie http://www.mantisbt.org/

Verder moet de feature rapport op zijn minst het volgende informatie bevatten:

- Een korte omschrijving van de feature aanvraag
- De deadline (sommige bug tracking software hebben deze optie als invoerveld)
- Product specificatie / game design document als bijlage

Het assignen van het rapport wordt voor game gerelateerde issues door de team lead games development bepaald. Voor website gerelateerde issues beslist de manager van de NinTec-Yezzle projectteam naar welke developer de issues assigned worden.

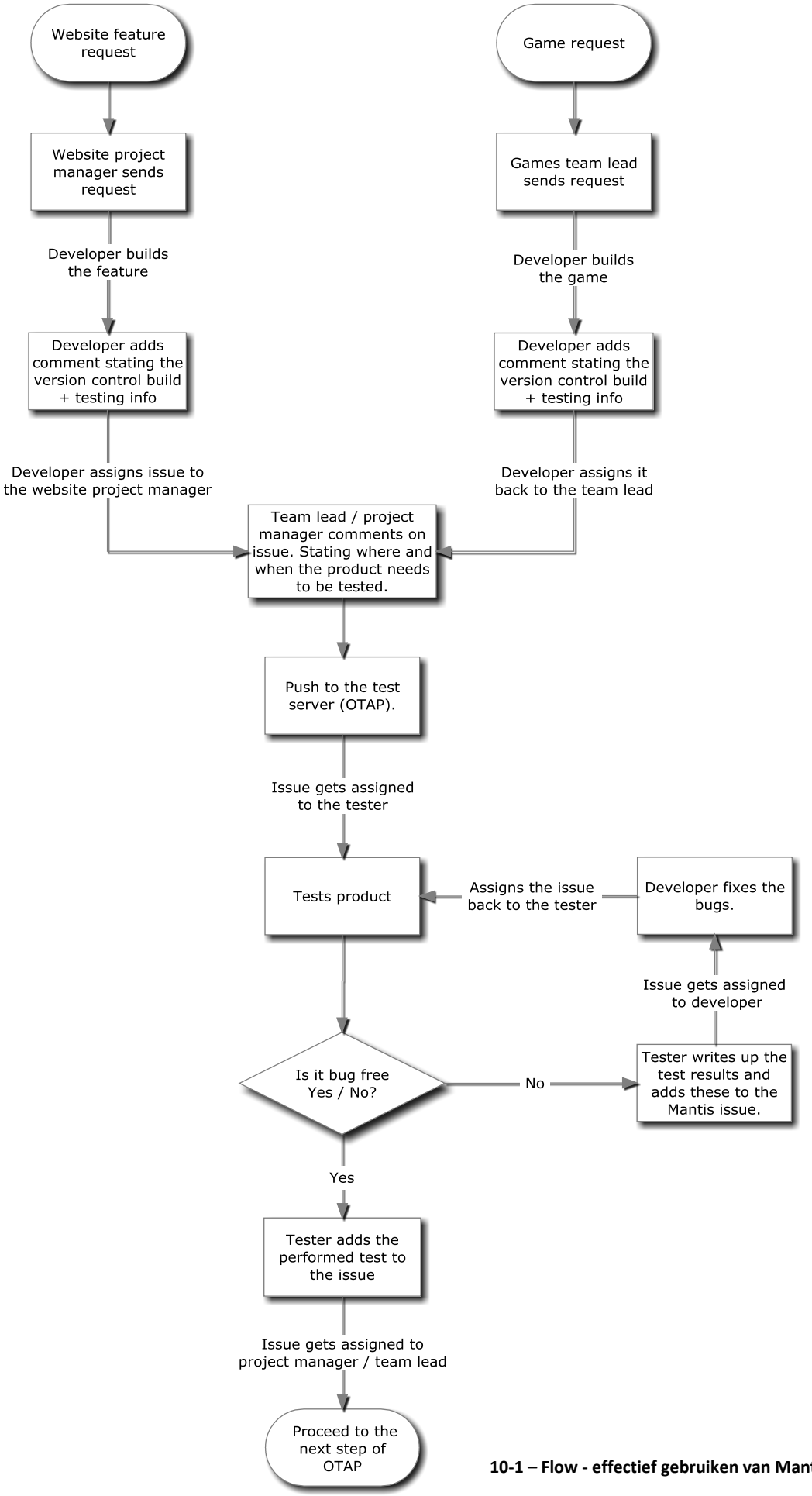
10.3 Prioriteiten

De volgorde waarin bugs gefixt of features gemaakt moeten worden, wordt bepaald door de prioriteit die het meekrijgt, zie hoofdstuk 8.7 “Bug classificaties Mantis”. Issues met de meest ernstige classificaties moeten eerst opgelost worden voordat men aan de lagere prioriteit issues begint. Wanneer bugs en feature requests dezelfde prioriteit en severity mee hebben gekregen, dan krijgt het fixen van de bug prioriteit boven het maken van de feature.

10.4 De flow

In figuur 10-1 is de flow te zien van het effectief gebruik maken van de bug tracking software Mantis. Er is hier één groot verschil te zien vergeleken met hoe men nu te werk gaat, namelijk deze nieuwe flow eist dat er na elke stap een comment geschreven wordt in Mantis. Hoe meer er gedocumenteerd wordt, hoe meer overzicht er gecreëerd wordt.
Het begint met een duidelijke omschreven feature request waar product specificaties aan toegevoegd zijn zoals in paragraaf 10.2 besproken is. Wanneer de developer eenmaal klaar is met het maken van het product, dan eist deze nieuwe methode dat er commentaar geleverd wordt in de bug rapport betreffende het build nummer van het versie beheer systeem en, wanneer van toepassing, de nodige test informatie. Met test informatie worden opmerkingen bedoeld zoals het aangeven van elementen die extra aandacht eisen, missende elementen die bekend zijn of placeholder graphics. Vervolgens kan de team lead of project manager aangeven waar het product te vinden is en wanneer deze producten getest moeten worden.

Het is dan de taak van de tester om test cases te ontwerpen die het product goed en volledig test. Ideaal gezien heeft de tester al test cases opgesteld na het testen van de documentatie, dit wordt in hoofdstuk 12.1 toegelicht. De resultaten van deze tests worden toegevoegd aan het rapport in de vorm van een bijlage. De gevonden bugs worden, zoals in hoofdstuk 8 beschreven is, gedocumenteerd en toegevoegd aan het rapport. Nu doorloopt het product een test-en-fix cyclus totdat alle bugs gevonden en gefixt zijn en het de volgende stap van het OTAP proces kan doorlopen.



10-1 – Flow - effectief gebruiken van Mantis

11 Testing - Te weinig documentatie

Dit hoofdstuk gaat verder in op het nut van goede documentatie bij het testen van software en hoe dit het proces vele male efficiënter kan maken.

11.1 Documentatie als test hulpmiddel

Om effectief te kunnen testen heeft de tester product specificaties of een game design document nodig. In deze documenten staat wat het product precies hoort te kunnen, wat de spelregels zijn en/of hoe het eruit hoort te zien. Wanneer dit soort documentatie aanwezig is dan levert dit meerdere voordelen op:

1. De tester kan heel vroeg beginnen met testen
2. De tester kan relevante testcases produceren
3. De tester kan makkelijk onderscheidt maken tussen een bug of een feature

1. Het hebben van de productspecificaties of het game design document maakt het mogelijk om aan *static black-box testing* te doen, dit kan al in de opstart fase van het project.

Static black box testing is het testen van iets wat niet draaiende is (dus statisch) en waarvan de tester niet onder de “motorkap” kan kijken. Denk aan het testen van een nieuwe auto. Het voelen van de banden, het controleren van de verf en het kijken naar de motor zijn allemaal voorbeelden van statische tests (je start de auto niet).

In het geval van software betekent dit het testen van de documentatie/specificaties. Static black-box testing maakt het mogelijk om fouten en gaten in de specificaties van een product te vinden. Het gevolg van fouten of gaten in de specificaties hebben is namelijk dat het product dan ook verkeerd ontwikkeld wordt. Dit maakt het controleren van de specificaties een essentiële stap om ervoor te zorgen dat het product in één keer goed ontwikkeld wordt. Want, wat al eerder genoemd wordt, hoe eerder bugs gevonden worden hoe goedkoper het is om deze op te lossen. Zo is een fout in de specificaties snel op te lossen, het is een kwestie van wat tekst aanpassen. Een fout in de software kan echter veel meer problemen opleveren, developers moeten meer tijd aan bug fixing spenderen in plaats van het daadwerkelijk ontwikkelen van producten. Deze verloren ontwikkeltijd kan het bedrijf veel geld kosten.

Om te demonstreren dat product specificaties erg handig kunnen zijn, volgt hieronder een onderdeel uit een design document waarin beschreven staat hoe een avatar in de online virtuele wereld “Club Galactic” gaat zitten (geschreven door mij, de wereld is onlangs offline gehaald, gemaakt voor het bedrijf Virtual Fairgorund).

"<u>Someone sat down on the seat you clicked on</u>
It is possible that another player has clicked on the same seat as you. If this happens then whoever gets to the seat first gets to sit on it regardless of who clicked first. When the other avatar has sat down then your avatar continues to walk to the seat but when he gets there he does nothing.

<u>No room</u>

- *For a seat to be a usable object it has to have enough room in front of it to walk or stand in front of it.*
    - *If there is not enough room in front of it then this seat is not a usable item.*

<u>The player clicks elsewhere</u>
If you click elsewhere before your avatar has sat down then the new command must be executed directly.

- *If this is an invalid command then continue the sitting down process*

- *If you click elsewhere whilst sitting down then the task must be performed of this click*
    - *If this is an invalid mouse click then do nothing (remain seated).*

- *If you click on another seat then the avatar must stand up and move to this location and sit down.*
    - *This also goes for seating which have more than 1 sit place (unless this seat is directly next to the seat you are sitting on, see chapter 2.4.1)."*

Het lijkt erg triviaal om een simpele feature zo uitvoerig uit te schrijven, maar er zijn meer keuzes die gemaakt moeten worden dan in eerste instantie gedacht wordt. Het opschrijven, en dus definiëren van al deze keuzes en problemen, zorgt ervoor dat de ontwikkelaar hier niet over na hoeft te denken. Een voorbeeld:

"<u>Someone sat down on the seat you clicked on</u>
It is possible that another player has clicked on the same seat as you. If this happens then whoever gets to the seat first gets to sit on it regardless of who clicked first. When the other avatar has sat down then your avatar continues to walk to the seat but when he gets there he does nothing."

Dit is een simpele keuze die gemaakt moet worden over wanneer een avatar op een zitplek mag zitten op het moment dat er twee spelers tegelijk op dezelfde zitplek hebben geklikt. Wanneer een dergelijke keuze nergens beschreven staat is het erg makkelijk om dit over het hoofd te zien tijdens het ontwikkelen van de zit functie, dit kan voor problemen zorgen. Hierbij komt ook kijken dat het een element is die niet snel door de developer zelf gezien kan worden doordat er meerdere spelers voor nodig zijn om de situatie te reproduceren. Als er geen rekening gehouden was met deze case in de documentatie dan betekende dit wellicht dat meerdere avatars op één zitplek konden plaatsnemen. Met als gevolg dat zich problemen met clipping voordoen doordat meerdere avatars elkaar overlappen.

Het niet meenemen van deze case had door het testen van de documentatie (static black-box testing) alsnog opgemerkt kunnen worden als een missend element of bug in de documentatie. Waarna de case alsnog meegenomen kon worden in de specificaties voordat het naar de developers was gegaan.

2. Aan de hand van product specificaties is het erg makkelijk om een testsuite (een verzameling test cases, zie paragraaf 11.2.3) op te stellen waar *alle* beschreven features, elementen en uitzonderingen in meegenomen worden. Wanneer er geen documentatie aanwezig is wordt het lastiger om test cases op te zetten wanneer een product nog niet af is. Er zou dan gezocht moeten worden naar iemand die wel op de hoogte is van wat er gebouwd gaat worden. Deze personen zijn dus als het ware een lopend en pratend product specificatie. Aan de hand van gesprekken met de "levende product specificatie" zouden er test cases verzonnen moeten worden. Dit is een erg inefficiënte proces, daarom zijn product specificaties erg belangrijk om te maken.

Hieronder zie je een testsuite voor het testen van het gaan zitten van avatars, uit het design document van Club Galactic.

Step	Pass	Fail	Comment
1.Have two players click on the same seat at the same time, but at different distances			
> Do both avatars walk to the seat?			
> Is the first avatar to arrive the only one to actually sit down?			
2.Place a seat in such a way that there is no room in front of the seat			
> When clicked on the seat does it show up as a not-usable item?			
3. Click on a valid seat, whilst the avatar is on-route make an invalid mouse click			
> Does the avatar continue to travel to and sit on the seat?			
4.Have an avatar take a seat and make an invalid mouse click			
> Does the avatar remain seated?			
5.Have an avatar take a seat and make a valid mouse click			
> Is the task of the valid mouse click performed?			
6.Have your avatar sit down and click on another seat which is not directly next to him			
> Does the avatar stand up and sit on the clicked on seat?			
7.Have your avatar sit down and click on another seat which is directly next to him			
> Is the avatar snapped to the seat directly next to him?			

Aan de hand van de design document is het erg makkelijk om specifieke en relevante test cases te produceren. Het opstellen van test cases die precies de productspecificatie testen heet ook wel test-to-pass. Het gaat namelijk om het controleren of alles werkt zoals omschreven staat in de product specificaties, dit kan daarom ook als een vorm van acceptance testing gezien worden. In het hoofdstuk "Hoe test ik?" worden meer technieken besproken voor test-to-pass testing.

3. Een veel voorkomend probleem bij het testen van software is dat iets als bug wordt opgegeven door een tester waarna de issue terug komt met als commentaar dat het geen bug is, maar een feature. Dit betekent dat de tester veel tijd heeft verspild aan het reproduceren van het probleem, het testen om de "kapotte" feature heen, het klaarmaken van screenshots (en een eventuele video opname) en het rapporteren hiervan. Bovendien volgt hierop de discussie met de developer waarom het een feature is in plaats van een bug. Dit is een hoop werk voor iets dat voorkomen had kunnen worden als er een productspecificatie was geweest waarin opgezocht kon worden of een element een feature of een bug is.

Product specificaties en game design documenten opstellen en testen kan de kwaliteit van een product erg verbeteren.

> *Further reading: Op de website http://www.eldergame.com/2009/02/whats-a-qa-team-without-a-spec/ is een voorbeeld te lezen van een game developer die het gebrek aan documentatie bespreekt van de development van de MMORPG Acheron's Call 2 en hoe dit de baan van QA'ers bijna onmogelijk heeft gemaakt om effectief te werk te gaan.*

In het vervolg van dit hoofdstuk worden de verschillende documenten besproken die gemaakt moeten worden voor een efficiëntere aanpak van het vak testen.

11.2 Test documentatie

In de meer gestructureerde ontwikkel modellen is het mogelijk om testen als onderdeel van de ontwikkeling mee te nemen. Dit betekent dat er vanuit een Quality Assurance perspectief geëist wordt dat er discipline, methodiek en regels toegepast worden om de productie cyclus snel en efficiënt te laten verlopen. Het gevolg hiervan is dat het testproces zorgvuldig en methodisch gepland moet worden. Dit betekent dat er voor het testen ook documenten geschreven moeten worden. Dit is belangrijk voor de volgende redenen:

- **Organisatie** – Zelfs bij een klein project is het mogelijk dat er duizenden test cases zijn. Deze test cases kunnen opgesteld zijn door meerdere testers over lange periodes. Zorgvuldig plannen zorgt ervoor dat elke tester en andere projectleden de testcases kunnen reviewen en effectief kunnen gebruiken.
- **Repeatability –** Het is belangrijk dat de gefixte bugs weer gecontroleerd worden of ze daadwerkelijk gefixt zijn en of de fix geen extra bugs gecreëerd heeft. Dit betekent dat je de test case opnieuw moet uitvoeren. Wanneer een test case nergens staat gedocumenteerd dan is het waarschijnlijk zo dat het veel tijd gaat kosten om de test case te reproduceren. Dit komt doordat er soms weken tussen het rapporteren van een bug en het fixen ervan kan zitten, wat betekent dat de tester het probleem waarschijnlijk al vergeten is. Vandaar dat goede documentatie hier van pas komt.
- **Tracking –** Hoeveel test cases ben je van plan uit te voeren? Hoeveel heb je op de vorige release uitgevoerd? Hoeveel cases zijn geslaagd? Hoeveel zijn er gefaald? Zijn er test cases overgeslagen? Deze vragen moeten beantwoord worden om een goed overzicht van het project te houden. Wordt er niet gepland en gedocumenteerd dan zijn deze vragen onmogelijk te beantwoorden.
- **Werkverantwoording** – In hoge risico industrieën, zoals in de medische wereld, is het soms wettelijk verplicht om te kunnen laten zien dat de test cases ook echt uitgevoerd zijn. Zelfs in lage risico industrieën, zoals de gaming industrie, is het verstandig om te kunnen bewijzen wie wat heeft gedaan, zodoende kan men zien wie er goed aan het presteren is en wie meer inzet moet tonen. Goede documentatie betekent dat er bijgehouden kan worden wie waar verantwoordelijk voor is.

Om dus methodisch te werk te kunnen gaan zal er altijd goed gedocumenteerd moeten worden. Er zijn verscheidene standaard test rapporten die nu besproken zullen worden.

11.2.1 Software Quality Assurance Plan

Elk game project zou een eigen plan moeten opstellen over hoe de kwaliteit bijgehouden gaat worden van de game. Dit wordt normaal gesproken bijgehouden in de zo genoemde Software Quality Assurance Plan of SQAP. In dit plan staat niets over wat er getest gaat worden, het houdt zich alleen bezig met het bijhouden en corrigeren van het product en het richt zich op kwalitatieve problemen in het proces. De SQAP is een onderdeel van de functie Quality Assurance en staat los van het daadwerkelijk testen van software. In een SQAP worden normaalgesproken de volgende punten besproken:

- **QA Personeel**
  Hier wordt de organisatorische structuur van het QA team beschreven. Omschrijf hier de hiërarchie van het QA team, wie werkt voor wie en naar wie rapporteert het hoofd van het QA team. Omschrijf de primaire rol van elke QA engineer. En omschrijf hier specifiek met welke activiteiten zij zich bezig houden.

- **Standaarden die gebruikt moeten worden in het product**
  Hier moeten twee type standaarden besproken worden, de productstandaarden en de processtandaarden. De productstandaarden gaan om de functie van elementen die geproduceerd worden als onderdeel van het project. Dit is inclusief de code, graphics, print, enz. De processtandaarden gaan om de manier waarop alle elementen geproduceerd worden. Denk hierbij aan bestand benaming, code formatting standaarden en het onderhouden van evoluerende project documentatie zoals het technische design document. Documenteer alle standaarden die relevant zijn en waar zij relevant voor zijn. En beschrijf dan ook hoe het QA team dit gaat bijhouden en hoe eventuele fouten aangepast kunnen worden in de standaarden.

- **Reviews en audits die uitgevoerd gaan worden**
  In dit hoofdstuk staat wanneer welke reviews en/of audits uitgevoerd gaan worden. Dit moet afgestemd worden met de algehele planning van de productie cyclus zodat men er zeker van is dat de producten of activiteiten beschikbaar zijn wanneer de geplande audit of review uitgevoerd wordt. Voor meer informatie over reviews en audits zie het hoofdstuk “Static white-box testing”.

- **QA verslagen en rapporten die gegenereerd worden**
  Hierin worden de type rapporten gedocumenteerd die door de software quality assurance activiteiten gegenereerd worden, hoe deze gecommuniceerd worden naar de rest van het team en hoe vaak dit gaat gebeuren.

- **QA probleem rapporteren en herstellende acties**
  Hier wordt gedefinieerd welke data en statistieken er gerapporteerd worden bij het vinden van fouten en wanneer dit gereviewed wordt met de rest van het project team.

- **QA tools, technieken en methodes**
  Definieer hier de tools, technieken en methodes die gebruikt worden binnen het QA team. Denk hierbij aan welke project management software, methodes van statistieken analyseren (b.v. fouten per 1000 regels code) en automated testing technieken met de bijbehorende tools.

- **QA metrics**
  Omschrijf hier welke statistieken er bij gehouden moeten worden en hoe dit weergegeven gaat worden.

- **Controle van extern geleverde producten**
  Geef hier aan welke onderdelen door welke bedrijven geleverd worden en wie dit moet controleren.

- **QA verslag opslag, onderhoud, retention**
  Geef hier aan hoe de verschillende documentatie opgeslagen wordt en hoe dit onderhouden wordt.

- **QA training**
  Om QA specifieke tools en technieken te leren gebruiken is bijbehorende training nodig. Documenteer hier deze technieken en tools om een budget voor de training te krijgen. Houd er ook rekening mee als er iets in-house ontwikkeld wordt dat het QA team een briefing krijgt van de ontwikkelaar.

- **Risico's**
  Documenteer hier de risico's die het QA team tegen kan komen. Definieer hier ook de mogelijke gevolgen van elk risico en hoe dit opgelost kan worden als het risico een realiteit wordt.

De SQAP geeft helderheid over wat er allemaal moet gebeuren voor Quality Assurance en hoe dit bijgehouden gaat worden. Een voordeel van het opstellen van een SQAP is dat er verschillende afspraken gemaakt worden over de rollen die men heeft, de standaarden die gemaakt worden waaraan men zich moet houden en hoe de test resultaten gerapporteerd gaan worden. Deze afspraken zorgen ervoor dat testers verantwoordelijk gesteld kunnen worden voor hun werkzaamheden. Ook zorgt het voor helderheid bij de developers over wat er met de test resultaten gebeurd, dit kan ervoor zorgen dat developers beter hun best gaan doen zodra men weet dat er objectief naar hun producten gekeken wordt en dat de resultaten ook zwart op wit gezet worden.

11.2.2 Test plan

Een test plan en een Software Quality Assurance plan hebben gelijkenissen, het verschil zit hem in het feit dat het test plan een fundamenteel document is voor het testen. Het gaat dieper in op de hoe/wie/wat/wanneer vragen. Het is het document dat de leidraad vormt voor een test team binnen een software project.

De "*IEEE Standard 829-1998 for Software Test Documentation"* zegt het volgende over wat een testplan precies is en waar het toe dient:

> *"To prescribe the scope, approach, resources, and schedule of the testing activities. To identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risk associated with the plan"*

Bij het opstellen van een test plan is het *document* niet wat het belangrijkst is, maar het *planningproces* dat ondernomen is waaruit het test plan voort gekomen is. Een test plan is dus een document dat voort komt uit het zorgvuldig plannen van alle uit te voeren test activiteiten van een project. Het wordt meestal geschreven door een lead test engineer die de planning goed afstemt met de andere afdelingen binnen een software productie cyclus.

Het is niet handig om een template te volgen van een test plan omdat je dan het risico loopt om het doel van het test plan voorbij te schieten. Om toch een beeld te schetsen hoe een test plan er uit kan zien, wordt hier de inhoud van een test plan uit het *"IEEE Standard 829-2008 for Software Test Documentation"* weergegeven:

- Test plan identifier
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Suspension criteria and resumption requirements
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

Zoals te zien is; houd het test plan zich voornamelijk bezig met het testproces en niet het reviewen van het testproces zoals dat in de SQAP besproken wordt. Het test plan is meestal door een tester, voor een tester gemaakt. Dit haalt niet weg dat het test plan ook essentieel is voor de complete planning van het game development proces, dus het test plan is ook voor het management belangrijk.

11.2.3 Test case / Test suite

Een test case is een test waar na wordt gegaan of een conditie of variabel binnen de software werkt of niet. Een collectie test cases noemt men een test suite. De test suite is in zijn simpelste vorm een serie van toenemende stappen die de tester achter elkaar kan uitvoeren.

Het is belangrijk dat de test cases zo geschreven zijn dat deze binair en duidelijk zijn. Het moet dus te beantwoorden zijn met ja of nee, waarbij "ja" de pass conditie is en "nee" de fail conditie. Dit is belangrijk voor het snelle overzicht van de problemen die verschillende tests wel of niet opgeleverd hebben, zonder dat alle test cases doorgelezen moeten worden om erachter te komen of de "ja" antwoord de pass of of fail conditie is. Een voorbeeld:

- *Does the text contain any typographical errors?*

Het probleem met deze test case is dat een pass gegeven zou worden met een nee antwoord. Wanneer dit voorkomt is het erg makkelijk om een foutje te maken en het te markeren als "fail", terwijl het een "pass" is. Een betere vraagstelling voor hetzelfde probleem zou het volgende zijn:

- *Is the text free of typographical errors?*

Wanneer hier "ja" op geantwoord wordt krijgt het de "pass" conditie met zich mee. Dit zorgt voor structuur in het testen en het creëert duidelijkheid in de documentatie.

Hieronder zie je een klein deel van de test suite die gebruikt zou kunnen worden bij het testen van Minesweeper, de game die standaard met Windows meegeleverd wordt.

Step	Pass	Fail	Comment
1. Launch Minesweeper			
->Menu options are “Game” and “Help”?			
->Left number (bombs left) displayed is 10?			
->Right number (time elapsed) displayed is 10?			
->Middle button shows a smiley face?			
2. Click the “Game” option on the menu and choose “Exit”.			
->Game closes?			
3. Relaunch Minesweeper			
4. Click the “Game” option on the menu			
->”Beginner” is checked?			
5. Click “Custom”.			
6. In the “Custom Field” submenu, enter 0 in the height box			
->0 is accepted as input?			
7. Click “Ok”			
->Playing grid is 9 rows high?			
->Playing grid is 9 rows wide?			
8. Click “Game”, “Custom…”			
9. In the “Custom Field” submenu, enter 999 in the height box.			
->999 accepted as input?			
10. Click “Ok”			
->Playing grid is 24 rows high?			
->Playing grid is 9 rows wide? (unchanged)			

Je ziet dat de stappen worden gegeven die nodig zijn voor het uitvoeren van de test cases en dat de test cases met “ja” te beantwoorden zijn wanneer deze cases slagen.

Een test suite opstellen en de cases uitvoeren is belangrijk voor het verificatieproces (zie hoofdstuk “Verificatieproces”). De cases die de tests niet doorstaan worden door een developer opgelost, waarna een tester moet nagaan of deze bugs daadwerkelijk gefixt zijn. Er kan soms weken tussen het rapporteren van alle bugs en het verifiëren van alle fixes zitten, daarom is de test suite zo handig. Aan de hand van de al ingevulde test suite kan met nagaan of deze juist opgelost zijn.

In bijlage C is er een globale test suite te zien voor de Braintrainers van Yezzle. Deze test suite is gebruikt om effectief de Braintrainers te kunnen controleren op de functionaliteit die zij gemeen hebben. Naast deze algemene tests cases zijn er per spel test cases opgesteld en uitgevoerd die uniek zijn voor een specifieke game.

12 Testing - Hoe test ik?

Er heerst bij sommige werknemers van Keesing Games de gedachte dat testing niet echt een vak is, dit stamt voornamelijk uit het feit dat men niet weet wat er allemaal bij komt kijken om een product goed te testen. Daarom wordt er dit hoofdstuk een paar van de verschillende technieken en methodes uitgelegd die de werknemers kunnen gebruiken om Yezzle.nl te testen.

12.1 Static Black-box testing (documentatie testen)

Hoe vroeger fouten gevonden worden hoe goedkoper het is om deze op te lossen. Daarom moet men bij het meest fundamentele element al beginnen met testen, namelijk het testen van de productspecificaties. Hieronder worden verschillende technieken uitgelegd over hoe dit aangepakt kan worden.

<u>High level review van de specificatie</u>
De eerste stap in het testen van de specificaties is een stap terug nemen en er van een afstand naar kijken. Dus niet direct gaan zoeken naar specifieke bugs. Lees de specs en zoek naar grote fundamentele problemen en dingen die over het hoofd gezien zijn. Deze stap is eigenlijk research doen, maar hoe beter je beeld is van wat er gemaakt moet gaan worden, hoe beter je er gedetailleerd naar kan kijken.

Doelgroep
Eén van de beste manieren om een goed beeld te krijgen van de specs is ernaar kijken door de ogen van de eindgebruiker. Doe wat research naar de doelgroep. Het is belangrijk om de verwachtingen van de doelgroep te begrijpen om zo de software te kunnen testen op de eisen van de gebruiker.

Aannames
Verder is het van belang bij de high level review van de documentatie dat je geen aannames maakt. Als een deel van de spec niet duidelijk is dan moet je er niet van uit gaan dat het klopt. Uiteindelijk is het de bedoeling dat je de spec gebruikt om je tests te designen, dus moet je het volledig begrijpen.

*“Assumptions are the mother of all f*ck ups!”*

Concurrentie
Om een beter beeld te krijgen van het geplande eindproduct is het altijd handig om naar de concurrentie te kijken. Waarschijnlijk is er al onderzoek gedaan naar de concurrentie door iemand in het team, maar als tester is het handig om zelf nog even naar vergelijkbare software / games te kijken. Dit helpt om missende elementen in de design te vinden en het helpt om over test situaties en test aanpakken te bedenken.

<u>Low level specificatie test technieken</u>
Na de high level review van de specificaties heb je als het goed is een compleet beeld van wat de product gaat worden en wat de externe factoren zijn die de product kunnen beïnvloeden. De volgende stap is het op een meer gedetailleerde niveau te bekijken. Waarbij de volgende vragen gebruikt kan worden als hulpmiddel:

Attributen checklist
Een goed doordachte product specificatie heeft acht belangrijke attributen. Hier volgen de acht attributen met de vragen die je per attribuut zou kunnen stellen om te controleren of het klopt.

- Het is Compleet – Mist het iets? Is er iets vergeten? Heeft het alles dat nodig is om als stand-alone document duidelijk te zijn?
- Het is Accuraat – Is het doel duidelijk gedefinieerd? Zijn er fouten?
- Het is Precies en Helder – Is de beschrijving helder en niet vaag? Is het maar op één manier op te vatten? Is het makkelijk te lezen en te begrijpen?

- Het is Consistent – Is de omschrijving zo geschreven dat er geen conflicten zijn met zichzelf of andere onderdelen in de specificatie?
- Het is Relevant – Is er extra informatie wat weggelaten kan worden? Sluit de feature aan op de originele gebruikers eisen?
- Het is Haalbaar – Kan de feature gemaakt worden met het beschikbare personeel, tools en middelen binnen het budget en planning?
- Het is Code-vrij – Beperkt de specificatie zich tot het definiëren van het product en niet het onderliggende software design, architectuur en code?
- Het is Testbaar – Kan de feature getest worden? Is er genoeg informatie beschikbaar zodat een tester tests kan designen?

Wanneer de productspecificatie getest wordt en een onderdeel niet voldoet aan al deze attributen dan is er een bug gevonden en moet dit aangevuld of verbeterd worden in het design document.

Terminologie
Als aanvulling op de attributen checklist zijn er een aantal woorden waar op gelet moet worden die gevoelig zijn voor fouten.

Note: De woorden staan in het Engels omdat software documentatie veelal in het Engels geschreven wordt.

Woorden	**Gevaar**
Always, Every, All, None, Never	Deze woorden impliceren dat iets absoluut is. Zorg ervoor dat dit ook daadwerkelijk absoluut is. Verzin test cases die dit op de proef stellen.
Certainly, Therefore, Clearly, Obviously, Evidently	Deze woorden halen je over om te geloven dat iets voor de hand liggend is. Val hier niet voor.
Some, Sometimes, Often, Usually, Ordinarily, Customarily, Most, Mostly	Deze woorden zijn te vaag. Het wordt onmogelijk om een operatie te testen die maar af en toe hoort te gebeuren.
Etc., And so forth, And so on, Such as	Lijsten die hiermee eindigen zijn niet testbaar. Lijsten moeten absoluut zijn of uitgelegd worden zodat er geen verwarring is over hoe de lijst opgebouwd is en wat er verder in de lijst voorkomt.
Good, Fast, Cheap, Efficient, Small, Stable	Deze termen zijn niet te definiëren. En dus niet testbaar. Als deze woorden voorkomen dan moet het verder gedefinieerd worden om exact uit te kunnen leggen wat er mee bedoeld wordt.
Handled, Processed, Rejected, Skipped, Eliminated	Deze termen kunnen grote hoeveelheden functionaliteiten verbergen die gespecificeerd moeten worden.
If…Then…(but missing Else)	Zoek voor “If…Then” statements die geen matchende “Else” hebben. Vraag jezelf af wat er gebeurd als de “If” niet gebeurd.

Wanneer er fouten of gaten in de documentatie gevonden word dan is het van belang dat dit op een heldere manier wordt gecommuniceerd met de schrijver van het document. Hij kan dit vervolgens aanpassen zodat het niet fout wordt aangeleverd bij de developers.

Zodra de productspecificaties uitvoerig getest zijn dan kan dit doorgestuurd worden naar de developers zodat het gemaakt kan worden. Zodra er een testbare versie van de software aanwezig is dan kan de tester bezig gaan met het doorlopen van de software.

12.2 Dynamic-Black-box testing (functional testing)

Dynamic black box testing is het testen van de software zonder dat de details bekend zijn van de onderliggende code waarmee de software geschreven is. Het is dynamisch omdat er draaiende software getest wordt, dit in tegenstelling tot static black box testing waar alleen de achterliggende documentatie getest wordt. Het is "black-box" testing omdat de tests worden uitgevoerd zonder dat men precies weet hoe de software tot zijn output komt. Dit hoeft ook niet, het is alleen wel belangrijk om te weten wat er hoort te gebeuren, dus wanneer "A" wordt ingevoerd en het is bekend dat vervolgens "B" de output hoort te zijn, dan kan er getest worden of de software dit ook daadwerkelijk doet.

Om weer terug te komen op het voorbeeld van het uitproberen van een nieuwe auto, dan is het starten van de motor, luisteren naar het geluid van de motor en er in rijden dynamische tests.

Er zijn veel verschillende technieken om software en/of games te testen. Er is niet altijd een duidelijk *juiste* techniek, verschillende situaties vragen om verschillende technieken. In deze paragraaf worden er verschillende test technieken uitgelegd die voor het meeste type software toepasbaar zijn, of belangrijk zijn om te weten.

12.2.1 Exploratory testing

Projecten die de big bang of code-and-fix model van ontwikkelen gebruiken zullen soms geen product specificaties hebben (zoals dat voor een deel van Yezzle zo is). Dit is niet ideaal voor een tester, maar je kunt een techniek genaamd "exploratory testing" toepassen. Dit is het simultaan leren van de software, het designen van tests en het uitvoeren van deze test cases.

Je moet de software als de productspecificatie behandelen. Ga methodisch door de software heen, feature per feature. Houd bij wat de software doet en geef weer welke features er allemaal zijn en analyseer het alsof het de specificatie is. Vervolgens voer je de dynamic black box technieken uit die hieronder besproken worden.

Het is een minder effectieve manier van testen dan wanneer je wel de specificaties hebt, maar door middel van exploratory testing is het wel mogelijk om methodisch te werk te gaan en alsnog zo veel mogelijk bugs te kunnen vinden.

12.2.2 Test to pass / test to fail

Het is de baan van een software tester om software te slopen (test-to-fail). Maar voordat er begonnen wordt met het uitvoeren van test cases die hiervoor bedoeld zijn, is het handig dat er eerst door de software heen gelopen wordt om te kijken of het doet wat het hoort te doen onder normale omstandigheden.

Om weer terug te komen naar het voorbeeld van een auto, wat als er een nieuw ontworpen auto getest moet worden? Het is niet verstandig om direct de motor te starten en op een race circuit op maximale snelheid te gaan racen. Het beste is om eerst rustig te beginnen met "de basics" checken. Werken de remmen? Maakt de motor rare geluiden? Kan er geschakeld worden? Zijn de banden het juiste formaat voor de auto? Dit zijn basic vragen die beantwoord moeten worden voordat een auto tot zijn extremen getest wordt. Dit geld ook voor

software. Check eerst of “de basics” werken (test-to-pass) voordat er verder gegaan wordt met het testen van extreme situaties (test-to-fail).

12.2.3 Data testing

Simplistisch gezien kan software in twee delen in gedeeld worden: data en het programma. Data zijn de mouse clicks, keyboard input, cd bestanden etc. Het programma is dan de uitvoerbare flow, overgangen, logica en berekeningen (het programma wordt in de paragraaf “*state testing”* in dit hoofdstuk besproken).

Wanneer er data getest wordt, dan worden de resultaten gecontroleerd van de input van de gebruiker. Een paar voorbeelden van data zijn:

- De woorden die je typt in een tekstverwerker
- Nummers ingevoerd in een spreadsheet
- De hoeveelheid kogels die je nog over hebt in een schietspel

Zelfs het meest simplistische programma kan een immense hoeveelheid aan data bevatten dat afgehandeld moet worden. Een rekenmachine bijvoorbeeld, hiermee kan bijna oneindig veel berekeningen uitgevoerd worden. Het is praktisch onmogelijk om elke berekening te testen. Daarom is het van belang om “*equivalence partitioning*” toe te passen. Dit is het zoeken van groepen die soortgelijke inputs, soortgelijke outputs en soortgelijke operaties hebben in de software. In plaats van alle data testen is het veel slimmer om bepaalde data sets te testen per equivalente partitie.

Boundary conditions

Eén van de technieken die gebruikt moet worden om goede test cases te ontwerpen voor een equivalente partitie, is het testen op boundary conditions (grens condities). Dit is het testen op data dat op een grens binnen de software valt. Programmeren is van nature heel binair, iets is waar of niet waar (0 of 1). Als een operatie wordt uitgevoerd op een reeks getallen, dan is de kans groot dat de programmeur het goed gedaan heeft voor de getallen in het midden, maar het komt wel vaker voor dat programmeurs fouten maken op de randen van data reeksen. Ron Patton geeft in zijn boek Software Testing: second edition een voorbeeld van een boundary condition error geschreven in BASIC:

```
Rem Create a 10 element integer array
Rem Initialize each element to -1
Dim data(10) As Integer
Dim i As Integer
For i = 1 To 10
        Data(i) = .1
        Next i
End
```

Het doel van dit stukje code is om een array van 10 elementen aan te maken en daarbij elke element op -1 te initialiseren. Een array(data) van 10 integers en een counter(i) worden aangemaakt. Een *For* loop loopt van 1 tot en met 10, en elk element van de array ,van 1 t/m 10, krijgt de waarde -1 toegekend.

Wanneer er in BASIC een array is met een aangegeven bereik (in dit geval *Dim data(10) As Integer)* dan is het eerste element 0, niet 1. Dit stukje code maakt dus een data array van 0 t/m 10, dit zijn 11 elementen. Dit betekent dat het programma de waardes 1 t/m 10 naar -1 initialiseerd, de waarde 0 word dus niet geinitialiseerd. Bij het uitvoeren van dit programma zie je de volgende resultaten:


```
data(0) = 0            data(6) = -1
data(1) = -1           data(7) = -1
data(2) = -1           data(8) = -1
data(3) = -1           data(9) = -1
data(4) = -1           data(10) = -1
data(5) = -1
```

Er is te zien dat data(0) de waarde 0 heeft en niet -1. Als dezelfde programmeur vergeet dat, of een ander programmeur er niet van bewust is hoe deze data array geinitialiseerd is dan kan het gebeuren dat het eerste element van de array, data(0), gebruikt wordt denkende dat deze op -1 is gezet. Dit soort foutjes kunnen tot vervelende bugs leiden. Hieronder wordt een overzicht gegeven van mogelijke grens condities die kunnen bestaan.

Type boundary conditions
Zoek of de software het volgende type data bevat:

- Numeriek
- Karakter
- Positie
- Kwantiteit
- Snelheid
- Locatie
- Grootte

En denk dan per type data na over de mogelijke grenzen die het kan hebben zoals:

1. Eerst / Laatst
2. Start / Finish
3. Minimaal / Maximaal
4. Over / Onder
5. Leeg / Vol
6. Langzaamst / Snelst
7. Grootst / Kleinst
8. Naast / Ver van
9. Kortst / Langst
10. Hoogst / Laagst

Note: Dit is niet een allesomvattende lijst maar dit zijn wel veel van de voorkomende grenzen.

De volgende stap is het testen van de grenzen. De meeste bugs zullen gevonden worden wanneer er twee equivalent partities gemaakt worden. De eerste partitie moet data bevatten waarvan de kans groot is dat het *wel* zou moeten werken, dus bijvoorbeeld twee *kloppende* waardes binnen de grenzen (test-to-pass). De tweede partitie moet data bevatten waarbij ervan uit gegaan wordt dat het een error genereert, dus bijvoorbeeld één of twee *niet* geldige waardes *buiten* de grenzen. In andere woorden test een waarde binnen de grenzen, test de laatst kloppende waarde en test niet geldige waardes buiten de boundaries. Het testen van waardes buiten de grenzen van de software is meestal zo simpel als ergens 1 bij optellen of aftrekken (test-to-fail). Bijvoorbeeld:

1. Eerst-1 / Laatst+1
2. Start-1 / Finish+1
3. Minimaal-1 / Maximaal+1
4. Net over / Net onder
5. Minder dan leeg / Meer dan vol
6. Nog langzamer / Nog sneller
7. Nog groter / Nog kleiner
8. Naast / Ver van
9. Nog korter / Nog langer
10. Hoogst+1 / Laagst-1

Hier volgen een paar toegepaste voorbeelden van het testen van de grenzen in software.

- Als een tekstveld 1 tot 255 karakters ondersteund, probeer dan 1 karakter in te vullen en probeer 255 karakters in te vullen (geldige partitie). Vul vervolgens 0 en 256 in als ongeldige partitie.
- Wanneer een postcode in een registratieformulier ingevuld moet worden, probeer dan 1000AA en 9999ZZ als geldige partitie. En probeer vervolgens 1 karakter meer of minder als ongeldig partitie(bv. 123AB of 1234ABC).
- Het testen van de grenzen voor een flight simulator zou betekenen dat er bijvoorbeeld precies op grond niveau en op de maximale hoogte van het vliegtuig gevlogen kan worden (geldige partitie). Probeer vervolgens onder grond en zee niveau te vliegen en probeer ook in de ruimte te vliegen(ongeldige partitie).

Het toepassen van equivalence partitioning rond de grenzen van de software is de meest effectieve manier om van praktisch oneindig veel combinaties naar een testbaar hoeveelheid effectieve tests te gaan.

Default, empty, black, null, zero and none

Een ander voor de hand liggend, maar veel voorkomende, bug is wanneer de software eist dat de gebruiker iets invult, bijvoorbeeld in een tekstveld, maar vervolgens niks invult. Hoe dit opgevangen moet worden wordt vaak over het hoofd gezien in de specificaties of vergeten door de programmeur. Goed functionerende software vangt dit af door de standaard waarde in te vullen of een error melding weer te geven.

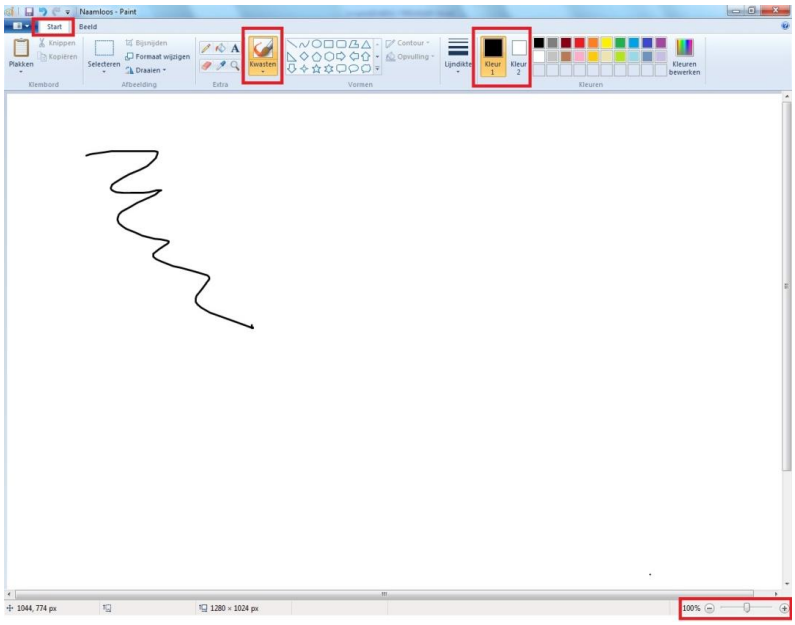
Overweeg altijd het creëren van een equivalence partition voor de default, empty, blank, null, zero of none condities.

Invalid, wrong, incorrect and garbage data

Het laatste type data testing is het testen van garbage data, dit is een methode die van “test-to-fail” uit gaat. Het is bedoeling dat er nu expres niet kloppende waardes ingevuld worden. Dus als er cijfers worden gevraagd, vul dan letters in. Als het alleen positieve waardes accepteert, vul dan negatieve waardes in. Als het verplicht is om een datum in te vullen, kijk dan of de software werkt wanneer het jaar 3500 ingevuld wordt. Typ met dikke worsten vingers, in andere woorden druk meerdere toetsen tegelijk in en observeer wat er gebeurd.

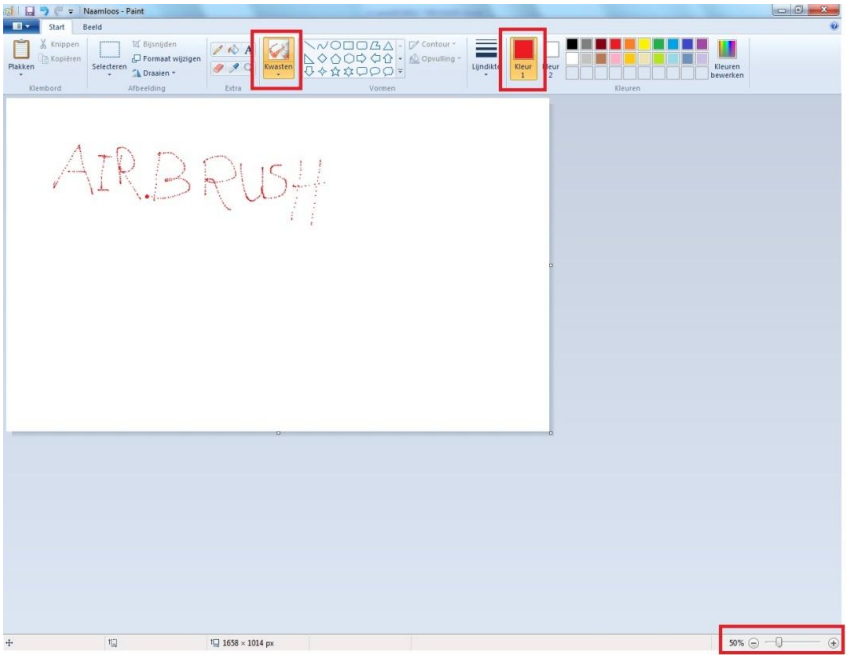
12.2.4 State testing

De hierboven genoemde technieken testen alleen data. Ook omvat black-box testing het controleren van de flow van de logica van staat naar staat (de transitie), dit wordt *state testing* genoemd. De *software state* is de conditie of modus waarin de software zich bevind.



Figuur 12-1 Windows 7 Paint in zijn begin staat

Figuur 12-1 laat het begin staat van Windows Paint (windows 7) zien. De rood gemarkeerde elementen worden altijd zo geselecteerd of ingesteld wanneer Windows Paint opgestart wordt. Hier is te zien dat de start tab, de selectie tool, standaard type brush, de kleur zwart en 100% grote als begin staat ingesteld zijn.



Figuur 12-2 Windows7 Paint in een aangepaste staat

Zodra er binnen Paint een optie veranderd wordt dan kan de software van uiterlijk veranderen, de menu's kunnen veranderen en de functionaliteit kan verander worden, in andere woorden de *state* van de software veranderd. In figuur 12-2 is te zien dat het brush type, de kleur en vergroting aangepast zijn. Dit zijn allemaal veranderingen van de software *state*, waarbij de software een pad door de code volgt, een paar bits aanpast, een paar variabelen set en nieuwe data inlaadt om tot de nieuwe *state* te komen.

State testing heeft dezelfde probleem als data testing, het is gewoonlijk niet realistisch om door elke *state* heen te gaan en deze uitvoerig te gaan testen. Ook hier moeten er equivalence partitions gemaakt worden om de hoeveelheid test cases terug te dringen naar een hanteerbaar hoeveelheid. Ron Patton geeft hier vijf mogelijke manieren om dit te doen in het boek *Software Testing: Second edition*

- Ga op zijn minst één keer door elke state.
- Test de state-to-state transities die eruit zien alsof deze het meest populair gaan worden of het meest voorkomen. De populariteit van te voren bepalen is een subjectieve factor, maar door de productspecificaties goed te analyseren is hier wel een goede schatting van te maken.
- Test de minst voorkomende paden tussen de *states*. Het is waarschijnlijk dat deze paden over het hoofd zijn gezien door de designer en de programmeurs.
- Test alle error *states*. Het komt regelmatig voor dat errors niet juist worden afgehandeld. Het errorbericht is niet juist of de software hersteld zich niet juist na het oplossen van de error.
- Test willekeurige *state* transities. Kies twee willekeurige punten en probeer van de ene staat naar de andere te komen.

Na het identificeren van de *states* die getest gaan worden, is het mogelijk om te beginnen met het definiëren van de test cases die de verschillende *states* op de proef gaan stellen.

Bij het testen van de verschillende *states* en de transities moet men naar alle variabelen van de staat gaan kijken; de statische condities, informatie, waardes en functionaliteit moeten per *state* gecontroleerd worden. Hier volgt een deel van de verschillende variabelen van de beginstaat van Paint (in Window 7, zie figuur 12-1):

- De grootte van het tekengebied wordt gezet naar dezelfde afmetingen waarmee de vorige sessie van Paint opgeslagen was.
- De grootte van het tekengebied begint altijd op 100%
- Het tekengebied is blanco.
- De standaard kleuren zijn geselecteerd (zwart voor de voorgrondkleur, wit voor de achtergrondkleur).
- Het document is genaamd Naamloos
- De normale kwast is standaard geselecteerd bij het opstarten.

Dit is maar een deel van de verschillende *state* variabelen waarmee rekening gehouden moet worden voor de beginstaat van Paint. Omdat er veel verschillende *states* kunnen zijn is het slim om de productspecificaties raad te plegen om een helder overzicht te krijgen van alle verschillende staten waarin de software zich kan bevinden. Veel productspecificaties bevatten diagrammen van de flow van het programma, deze diagrammen zijn een uitstekend hulpmiddel om te weten te komen wat elke staat doet. Wanneer deze diagrammen niet aanwezig zijn dan is het nog steeds verstandig om voor het testen deze zelf te maken.

Race condition

Het controleren of elke *state* en elke transitie klopt volgens de documentatie zijn voorbeelden van test-to-pass tests. Net zoals data testing is het ook belangrijk om test-to-fail test cases te designen. Een voorbeeld van een test-to-fail case is het checken van de zogenoemde "Race Condition". Hierbij wordt er gekeken naar hoe een programma omgaat met multitasking (het draaien van meerdere processen tegelijkertijd). Voorbeelden hiervan zijn:

- Het opslaan en laden van hetzelfde document op dezelfde tijd met twee verschillende programma's.
- Het delen van dezelfde printer of andere randapparatuur.
- Het drukken van toetsen of het klikken van de muis terwijl de software aan het laden is of van *state* aan het veranderen is.
- Het tegelijkertijd afsluiten en opstarten van twee of meer instanties van de software.
- Het gebruiken van verschillende programma's om tegelijkertijd een gemeenschappelijke database aan te roepen.

Race condition testing is belangrijk omdat gebruikers regelmatig dit soort acties per ongeluk uitvoeren. Hedendaagse software moet dit soort handelingen zonder problemen kunnen afvangen.

Repetition testing

Een ander voorbeeld van een test-to-fail *state* test is repetition testing. Dit is een techniek waarbij er één handeling herhaaldelijk uitgevoerd wordt. Dit kan zo simpel zijn als het herhaaldelijk opstarten en afsluiten van een programma, het herhaaldelijk opslaan en laden van data of het herhaaldelijk selecteren van dezelfde operatie. Het is mogelijk om een fout binnen een paar herhalingen te vinden, maar het kan ook wel duizenden herhalingen van een actie kosten om een probleem te ontdekken.

De voornaamste reden voor het uitvoeren van repetition testing is om memory leaks te vinden. Dit is een veel voorkomende software probleem waarbij de computer geheugen voor een operatie reserveert maar het geheugen niet volledig vrijmaakt wanneer de operatie uitgevoerd is. Het gevolg hiervan is dat het programma steeds langzamer gaat lopen waarbij het voor kan komen dat het programma onbruikbaar wordt of vastloopt.

Stress testing

Stress testing is het draaien van de software onder slechte omstandigheden zoals weinig geheugen ter beschikking stellen, lage schijf ruimte hebben, een trage CPU gebruiken, een langzame modem gebruiken etc. Kijk goed naar welke externe factoren de software gebruikt en minimaliseer deze. Observeer vervolgens hoe de software draait onder deze slechte omstandigheden. Vaak wordt er een zogenoemd *minimum spec machine* vast gesteld, dit is de minimale hardware waaronder de software moet werken. Tijdens een stress test kan direct gecontroleerd worden of de software acceptabel draait op minimale hardware. Stress testing is een vorm van een boundary condition test.

Load testing

Load testing is het tegenovergestelde van stress testing. Bij load testing is het de bedoeling om zoveel mogelijk handelingen tegelijk door de software uit te laten voeren. Voor een web applicatie betekend dit bijvoorbeeld dat er duizenden connecties met de server aangemaakt moeten worden om zo te weten te komen hoe de software draait wanneer de server op maximum capaciteit is.

Er is geen reden dat repetition, stress en load testing niet tegelijk uitgevoerd kunnen worden. Dit soort tests worden gewoonlijk geautomatiseerd met speciale tools die bijvoorbeeld duizenden connecties met een server kunnen aanmaken. Omdat automated testing een vak apart is wordt hier verder niet op ingegaan.

12.2.5 User interface testing

User interface (UI) testing kan een wat subjectieve test gebied zijn. Wat de één iemand een goed werkende UI vindt is niet noodzakelijk voor een ander ideaal. Voor het testen van UI's komt er gevoel bij kijken. Dit wil niet zeggen dat het compleet subjectief is. Er is genoeg onderzoek gedaan naar hoe een sterke grafische user interface opgesteld kan worden. Er zijn dan ook verschillende punten van een UI die getest kunnen worden. Hier volgen zeven punten waar naar gekeken kan worden. Houd er wel rekening mee dat verschillende onderzoeken, verschillende eigenschappen geven om getest te worden, deze zeven eigenschappen zijn dan ook niet een definitief lijstje.

Volg standaarden en richtlijnen

Één van de belangrijkste UI eigenschappen is dat het de huidige standaarden en richtlijnen volgt (of je moet een goede reden hebben om dit niet te doen). Bedrijven als Microsoft en Apple hebben voor hun besturingsystemen een standaard gedefinieerd. Zo hebben zij bijvoorbeeld standaarden neergezet voor de "Look and Feel" en wanneer een checkbox gebruikt moet worden in plaats van een optie knop. Voor deze richtlijnen zie de boeken "*Macintosh Human Interface Guidelines*" uitgegeven door Addison-Wesley en "*Microsoft Windows User Experience*" uitgegeven door Microsoft Press. Het is verstandig om software aan deze richtlijnen te laten voldoen. De software kan vervolgens getest worden of het voldoet aan de gedefinieerde richtlijnen.

Wanneer het platform voor de software geen standaarden ken of wanneer de software het platform zelf is, dan wordt het volgen van de komende eigenschappen belangrijk.

Intuïtief

Bij het controleren of de UI intuïtief aanvoelt zouden de volgende vragen gesteld kunnen worden.

- Is de user interface "clean", niet indringend en niet druk? De UI moet niet in de weg staan van wat er bereikt wilt worden. De functies en reacties van de software moeten te vinden zijn waar men verwacht deze te vinden.
- Is de UI georganiseerd? Is de UI makkelijk te navigeren? Is het logisch wat de volgende stap is? Is het mogelijk om op een willekeurig punt terug gaan of de applicatie stoppen? Wordt de input bevestigd? Gaan de menu's niet te diep?
- Is er te veel functionaliteit? Probeert de software te veel te doen? Wordt de software te complex doordat er te veel features zijn? Voelt het alsof er een informatie overload gegeven wordt?
- Als er problemen zijn is de help functie dan makkelijk te bereiken en is deze ook echt behulpzaam?

Consistent

Mensen raken gewend aan bepaalde dingen en creëren gewoontes, dit geld ook voor het gebruiken van computers. Mensen raken onbewust en soms bewust gefrustreerd door kleine veranderingen binnen de UI. Zoals als een zoekfunctie op één plek aangeduid wordt met "find" en op een ander plek met "search". Let dus op of de terminologie en benaming van elementen consistent zijn. Ook moet er gelet worden op of de shortcuts hetzelfde blijven, of de tekst op dezelfde doelgroep niveau blijft en of de plaatsing van buttons als "Ok" en "Cancel" consistent geplaatst worden.

Flexibel

Gebruikers houden van keuzes, maar niet te veel. Met veel keuzes komt ook veel complexiteit kijken. Let er dus op of de UI niet te ingewikkeld wordt voor de gemiddelde gebruiker. Als het wel te ingewikkeld is, denk dan na over een optie waar de gebruiker uit een simpele of uitgebreide view kan kiezen.

Comfortabel

De software moet comfortabel te gebruiken zijn voor de gebruiker. Dit is erg subjectief, maar er zijn wel een paar elementen waarop gelet kan worden.

- Is de “Look and Feel” van de software gepast voor de doelgroep?
- Geeft de software een waarschuwing voordat de gebruiker een kritieke operatie uitvoert? En is er een mogelijkheid om verloren data terug te halen na het maken van een fout (denk aan een undo / redo functie)?
- Is de software snel genoeg? Gebeurt er niets te snel, zoals een informatie scherm tijdens het laden waar geen tijd voor is om deze te lezen? Als een handeling langzaam uitgevoerd wordt, krijgt de gebruiker feedback over hoe lang het nog gaat duren?

Correct

Waar het controleren of het gebruik van de software wel comfortabel is subjectief is, is het controleren op correctheid weer erg objectief te uit te voeren. Correctheid houdt zich bezig met de vraag of de UI doet wat het hoort te doen. Het komt bijvoorbeeld vaak voor dat er een proces plaats vindt die de optie geeft op dit proces af te breken, maar bij het klikken van deze optie wordt het proces helemaal niet afgebroken. Waarom wordt de afbreek optie dan gegeven? Dit is niet correct, zo zijn er nog een paar andere elementen die op correctheid getest kunnen worden.

- Zijn er extra of missende functies?
- Is het taalgebruik en spelling correct? (Programmeurs zijn vaak snel geneigd te technisch te schrijven.)
- Controleer op slechte media gebruik. Zijn alle icoontjes hetzelfde formaat, en maken ze gebruik van dezelfde kleuren palette? Is al het geluid dezelfde format met dezelfde sampling rate? Etc.
- Controleer of wat de UI weergeeft, ook daadwerkelijk doet wat het hoort te doen. Als er bijvoorbeeld op “opslaan” gedrukt wordt, is de opgeslagen staan dan precies hetzelfde als wanneer het bestand weer ingeladen wordt? Wanneer er geprint wordt, is de output op papier precies gelijk aan de weergave op de computer?

Relevantie

Als laatst kan er gekeken worden of de gegeven features wel nuttig zijn. Hebben alle features meerwaarde voor de kwaliteit van de software? Helpt een feature de gebruiker te bereiken wat hij wil bereiken? Wanneer er een vermoede is dat een feature irrelevant is, doe dan wat onderzoek naar waarom het in de software zit. Het kan namelijk voorkomen dat een tester de achterliggende gedachte niet door heeft. Onnodige elementen in de UI moeten vermeden worden.

12.2.6 Localisation testing

Localisation is het proces van het veranderen of aanpassen van een applicatie voor een ander taal, land of cultuur. Dit omvat het aanpassen van de user interface, grafische designs en de initiële instellingen aan de hand van hun cultuur en vereiste.

Localisation testing komt neer op het testen van software die bedoeld is voor een buitenlandse markt. Het meest voor de hand liggende onderdeel van localisation testing is het controleren van de taal. Maar het is ook nodig dat de localised software op “normale” bugs getest wordt. Dit maakt het testen van buitenlandse software erg lastig omdat de tester waarschijnlijk de taal niet spreekt. Hoe beter de tester de taal begrijpt, hoe makkelijker het wordt om de localised versie te testen. Daarom is het gewoonlijk dat er speciaal iemand voor localisation testing wordt ingehuurd of dat er een externe partij voor gebruikt wordt.

Er zijn overigens wel onderdelen die getest kunnen worden zonder de taal goed te begrijpen. Eén van de meest logische problemen die je bij localisation testing tegen kan komen zijn elementen zoals buttons die niet meer goed passen. Dit komt doordat andere talen soms meer of minder karakters gebruiken om hetzelfde te zeggen. Daarom is het belangrijk om gebieden van de software te testen op elementen die problemen zullen ondervinden van het gebruiken van meer of minder karakters.

Verder zouden de hotkeys gecontroleerd kunnen worden. Bijvoorbeeld in Engelse talige software is “Ctrl + F” vaak de hotkey om op een pagina te zoeken. Dit zou in het Frans heel goed geen “Ctrl + F” zijn, maar “Ctrl + R” omdat zoeken “réchercher” is in het Frans.
Van een localisation standpunt is het belangrijk dat developers tekst nooit hard-coden (uitzonderingen nagelaten). Programmeurs zijn geen localisers, en dat hoeven zij ook niet te zijn. Maak een bestand aan met alle elementen die vertaald moeten worden zoals tekst strings en error messages, zorg er vervolgens voor dat de code verwijst naar deze tekst bronbestand. Dit maakt het mogelijk om localisers in te huren die geen kennis nodig hebben van programmeren.

Een ander probleem dat voor kan komen bij het testen van een localised versie is het niet localisen van de verschillende eenheden die over de wereld gebruikt worden. Hier volgt een kort lijstje van verschillende data formats, waar onder andere op gelet moet worden:

Eenheid	Voorbeeld
Maten	Metric – Imperial, dus meters versus yards e.d.
Nummers	Komma’s - decimalen, 1.500,00 versus 1500.00
Valuta	De verschillende symbolen en waar deze geplaatst worden
Data	Volgorde van dag, maand, jaar. Engels dd/mm/yy versus Amerikaans mm/dd/yy
Tijden	12 uur of 24 uur systeem. 3:30pm versus 15:30
Adressen	Verschillende postcodes. 3511AA = Nederland, 03511 = Frankrijk

Figuur 12-3 Een bekende localisation fout onder gamers, afkomstig van de game Zero Wings

Bij localisation testing is het altijd lastig om te bepalen hoeveel tijd er aan besteed moet worden. Het beste is om een native speaker in te huren om de tekst te controleren. De vraag is alleen dan of de buitenlandse versie dan ook nog eens uitvoerig getest moet worden op normale bugs terwijl de "originele versie" al maanden lang getest is? Dit ligt aan het feit of de software ontwikkeld is om meerdere talen te ondersteunen, of is er pas in een late fase van het development proces besloten dat de software ook vertaald moet worden? In het eerste geval is een snelle check van de software waarschijnlijk voldoende, in het tweede geval is het verstandig om alles zorgvuldig na te gaan of er geen nieuwe bugs zijn bijgekomen door het localisen. Yezzle.nl gaat binnenkort vertaald worden in verschillende talen, maar omdat het platform problemen heeft met het cms en er binnenkort over gegaan wordt op een ander CMS, DotNetNuke, is het belangrijk dat er veel aandacht besteed gaat worden in het testen van alle verschillende localised versies.

<u>Overige black-box test technieken</u>

Na het volledig uitgevoerd hebben van de verschillende manieren om data te testen zijn er nog een paar methodes om de laatste bugs op te kunnen sporen.

Één manier is het testen als een domme gebruiker. Zet alle kennis van de software terzijde en ga gewoon dingen proberen zoals een nieuwe gebruiker dat ook zou doen. Het is hier handig om een collega of vriend er even naar te laten kijken die niet aan het project gewerkt heeft.

Een ander taak die verstandig is om uit te voeren is het zoeken van bugs waar er al bugs gevonden zijn. Hoe meer bugs er gevonden zijn, hoe meer bugs er in totaal zijn. Het komt vaak voor dat meerdere items worden beïnvloed door één bug.

Er zou ook nog als een hacker gedacht kunnen worden om zo onveilige onderdelen in de software op te kunnen sporen. In het geval van games is het belangrijk dat er ook gekeken wordt naar de game mechanics of dit niet exploit of cheat gevoelig is. Bijvoorbeeld wanneer een game met een hele simpele macro te kraken is dan zou hier iets aan gedaan moeten worden.

Als laatst is het belangrijk om ervaring en gevoel te gebruiken. Wanneer er een gevoel is dat er iets niet klopt neem dan niet zomaar aan dat het wel klopt. Ga op onderzoek uit. Leer van gemaakte fouten. Houd bij wat wel en niet werkt.

"Experience is the name everyone gives to their mistakes" – Oscar Wilde

13 Testing - White-box testing

White box testing is het testen van de software terwijl er *wel* toegang is tot de code. Dit betekent dat het mogelijk is om hints te krijgen over hoe de software het beste te testen is – de tester kan in de "box" kijken. Gebaseerd op het doornemen van de code, kan de tester zien welke onderdelen van het programma een verhoogde kans hebben op bugs te creëren, hier worden dan de tests op aangepast. Om aan white-box testing te kunnen doen is het wel nodig dat de tester kan programmeren, in tegenstelling tot black-box testing waar dit geen vereiste is (al is het wel handig om de basis van programmeren te kennen).

Net zoals black-box testing is white-box testing in static en dynamic white-box testing op te splitsen. Waarbij de de static methode om het reviewen van code gaat en de dynamic methode volledig draait om daadwerkelijk het testen van de software aan de hand van de code.

13.1 Static white-box testing

Zoals bij static black-box testing gaat static white-box testing om het testen van de software zonder het daadwerkelijk te draaien. De software design, architectuur en code worden op bugs gecontroleerd. Het wordt soms ook wel "structural analysis" genoemd. De grootste reden van het uitvoeren van static white box testing is het zo vroeg mogelijk vinden van structurele fouten in een programma. Hierbij komt als bonus dat het de black-box testers ideeën geeft voor specifieke test cases na het lezen van de commentaar van de static white-box tests. De verschillende manieren om aan static white-box testing te doen worden hieronder besproken.

13.1.1 Formal Reviews

Een formele review is het proces onder waar static white box plaats vind. Een formele review kan lopen van een simpele meeting tussen twee programmeurs tot een volledig gedetailleerde inspectie van de software design en/of de code. Er zijn vier elementen waar men zich aan moet houden bij een formele review.

1. **Identificeer problemen:** Het doel van een review is het vinden van problemen in de code. Problemen zijn niet alleen fouten in de code maar ook missende code.
2. **Volgen van de regels:** Hoeveel van de code wordt er doorgenomen? Hoeveel tijd mag het duren? Waar mag commentaar op gegeven worden?
3. **Wees voorbereid:** Een goede voorbereiding betekend dat de meeste fouten al gevonden worden en deze besproken kunnen worden tijdens de review. Dit helpt het efficiënt laten verlopen van de review.
4. **Schrijf een rapport:** De review groep moet een rapport produceren waarin samengevat staat wat de resultaten van de review zijn, en dit moet dan beschikbaar gemaakt worden aan de rest van de development team.

Het is belangrijk om je aan deze elementen te houden, wanneer een review op een ad-hoc manier gebeurd dan kan dit wel eens negatieve gevolgen hebben. Het is zonder structuur en regelmaat makkelijk om bugs te missen, dit kan men het gevoel geven dat de review een verspilling van hun tijd is geweest.

Een formele review is een effectieve manier om in een vroeg stadium van development bugs te vinden. Er kunnen altijd kleinere bugs door een review heen komen, maar dit kan in de volgende fase van testen opgevangen worden. En een review brengt ook nog een paar goede elementen met zich mee.

- **Communicatie:** Door de overdracht van informatie kunnen black-box testers inzicht krijgen in waar er eventuele problemen kunnen gaan liggen. Minder ervaren programmeurs kunnen door de review nieuwe technieken leren van de meer ervaren programmeur. En het management krijgt een beter zich van de voortgang van het project.
- **Kwaliteit:** Wanneer een programmeur weet dat elke regel code van hem bekeken gaat worden dan zal hij snel meer zorgvuldig te werk gaan tijdens het coderen.

- **Teamvorming:** Wanneer een review goed wordt uitgevoerd dan is het een mooie gelegenheid om voor testers en programmeurs wederzijdse respect op te bouwen. Men begrijpt elkaars baan beter en weet wat de ander nodig heeft.
- **Oplossingen:** Oplossingen voor grote problemen kunnen gevonden worden tijdens een review.

Je moet niet vertrouwen op deze indirecte voordelen, maar het kan wel voorkomen.

13.1.2 Peer reviews

Het meest informele en laagdrempelig vorm van een formele review is de “peer review”. Peer reviews vinden meestal plaats tussen degene die de code geprogrammeerd heeft en één of twee andere programmeurs of testers. Het is dan de bedoeling dat men de code samen doorneemt om zo fouten te ontdekken en elementen te vinden die over het hoofd zijn gezien. Peer reviews zijn goed doordat het vaak informeel uitgevoerd wordt waardoor men niet bang hoeft te zijn voor eventuele gevolgen vanuit het management. Het is nog wel verstandig om dat men zoveel mogelijk aan de vier regels van de formele review houd om er voor te zorgen dat het niet zomaar een kletspauze wordt.

13.1.3 Walkthroughs

Walkthroughs zijn de volgende stap in formaliteit van de reviews. Dit gebeurd meestal met de degene die de code geschreven heeft en ongeveer 5 andere programmeurs / white-box testers (waarvan minstens één een senior programmeur is). Belangrijk is dat de reviewers tijdig een kopie van de software ontvangen zodat zij zich kunnen voorbereiden op de review en alvast vragen kunnen verzinnen.

Degene die de code presenteert moet per regel door de code heen gaan of per functie de code bespreken (wat doet het en waarom?). De reviewers moeten goed opletten en stellen vragen wanneer ze iets tegenkomen waarvan ze denken dat het er verdacht uitziet. Na afloop van de review moet degene die zijn code gepresenteerd heeft een rapport schrijven met daarin wat er tijdens de review gevonden is en wat hier aan gedaan gaat worden.

13.1.4 Inspections

Een inspectie is het meest formele type review. Hier zit veel structuur in en vereist vaak een training voorafgaand van de inspectie. Het grootste verschil van de andere type reviews is dat degene die de code presenteert niet degene is die de code geschreven heeft. Dit dwingt dus de presentator om iemand anders code volledig te leren en te begrijpen. Dit zorgt voor een andere interpretatie tijdens de inspectie.

De andere participanten, genaamd inspectors, krijgen een de taak om de code te reviewen vanuit een ander perspectief. Zo kan er gevraagd worden om op de volgende manieren de code te bekijken: als tester, als gebruiker, als product support medewerker en wordt er zelfs iemand gevraagd om de code achterstevoren te reviewen om ervoor te zorgen dat de code evenredig gereviewed wordt. En dan heb je ook nog een deelnemer die de taak krijgt om de inspectie te modereren en te notuleren.

Na afloop van de inspectie volgt er gewoonlijk nog een meeting waarin de gevonden fouten per inspector gedeeld worden met degene die het rapport moet schrijven. Aan de hand van dit rapport gaat de programmeur wiens code het is de fouten herstellen en worden deze reparaties gecontroleerd door de moderator.

13.1.5 Checklist for Formal Reviews

Het reviewen van code gaat makkelijker wanneer er een lijstje van mogelijke problemen afgewerkt kan worden. Daarom zullen hier enige elementen besproken worden waarop gelet kan worden bij het reviewen van code. Deze punten zijn een toevoeging voor het controleren van de code tegen de design specificaties.

Data reference errors

Data reference errors zijn bugs die ontstaan doordat een variable, een constante, een array of een string niet goed gedeclareerd of geinitialiseerd zijn. Dit type probleem is de grootste boosdoener van een buffer overrun, dit is een groot probleem voor de beveiliging van de software.

Data declaration errors

Data declaratie bugs worden veroorzaakt door het niet juist declareren of niet juist gebruiken van variabelen of constanten.

Computation errors

Computation errors zijn fouten in de berekeningen en komt dus neer op het gebruik van slechte wiskunde. Het gevolg hiervan is dat de output niet is wat er verwacht wordt.

Comparison errors

Comparison errors zijn problemen waarbij waardes vergeleken worden maar daarbij de verkeerde vergelijking gedefinieerd is. Minder dan, groter dan, gelijk aan, niet gelijk aan. Vergelijkende errors zijn heel vatbaar voor boundary condition problemen(zie hoofdstuk Black-Box-Testing).

Control flow errors

Control flow errors zijn het gevolg van loops en andere controle constructies die zich niet gedragen zoals verwacht is. Ze worden gewoonlijk, direct of indirect, veroorzaakt door computational of comparison errors.

Subroutine parameter error

Subroutine parameter errors worden veroorzaakt door het incorrect doorgeven van data naar en van de software subroutines.

Input/output errors

I/O errors kunnen met veel issues te maken hebben zoals, het lezen van een bestand, het accepteren van input van een muis en keyboard of het schrijven naar een output device zoals een printer of een scherm.

13.1.6 Other checks

- Werkt de software met andere talen? Kan het extended ASCII afhandelen. Moet het Unicode gebruiken in plaats van ASCII?
- Is het de bedoeling dat de software omgezet kan worden naar andere compilers en CPU's?
- Is er rekening gehouden met de compatibility? Kan de software draaien op machines met ander hoeveelheden geheugen, ander interne hardware zoals videokaarten en geluidskaarten, en ander randapparatuur zoals printers en modems?
- Genereert het programma waarschuwingen bij het compilen? Meestal zijn informatie berichten en/of waarschuwings berichten een teken dat er iets niet helemaal goed gaat in de code.

13.2 Dynamic White-box testing

Note: Dit onderzoek is niet gericht op dynamic white-box testing, maar omdat het wel belangrijk onderdeel is van het testen wordt het hieronder kort besproken.

Dynamic white-box testing is het testen van draaiende software waarbij men toegang tot de code heeft. In essentie betekend dit dat je gebruik maakt van informatie die je krijgt van het zien van wat de code doet en hoe het werkt om er zo achter te komen wat er getest moet worden en hoe dit aangepakt moet worden. Dynamic white-box testing blijft niet alleen bij het kijken wat de code doet maar het gaat ook zo ver als het direct testen en besturen van de software. De vier gebieden die dynamic white-box testing volgens het boek van Ron Patton *Software Testing: Second edition* omvat zijn:

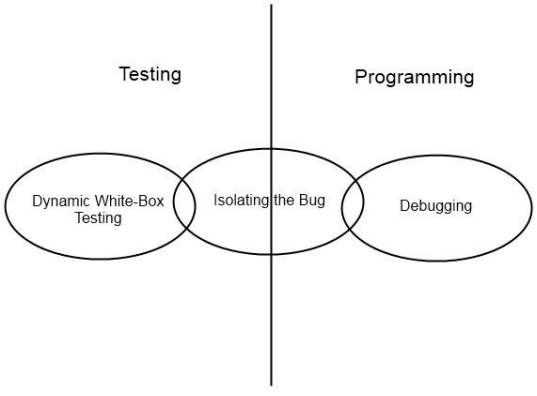
- Het direct testen van low-level functies, procedures, sub routines of libraries (API's).
- Het testen van de software als een compleet programma, maar het aanpassen van de test cases aan de hand van wat men weet van de operaties die de code uitvoert.
- Het toegang krijgen om variabelen en state informatie te lezen van de software zodat je kunt bepalen of jou tests doen wat je denk dat ze horen te doen. En, de mogelijkheid hebben om de software te dwingen om handelingen uit te voeren die niet mogelijk zijn wanneer er zonder code getest zou worden.
- Het meten hoeveel van de code en welk deel van de code 'geraakt' wordt wanneer de tests worden uitgevoerd en dan aan de hand van de resultaten de overbodige test cases te verwijderen en missende bij te voegen.

Naast deze punten geven C.P. Schultz, R Bryant en T. Langdell in hun boek "*Game Testing: All in One"* voorbeelden van situaties waar white-box testing effectiever is dan black-box testing wanneer we puur naar game testing kijken.

- Tests die uitgevoerd worden door de developer voordat nieuwe code geïntegreerd wordt met de rest van de game.
- Testen van code modules die onderdeel gaan worden van een hergebruikbare library voor meerdere games of platformen.
- Testen van code methodes of functies die essentiële onderdelen vormen van de game engine.
- Testen van low-level routines die de game gebruikt ter ondersteuning van specifieke functies in de nieuwste hardware apparatuur zoals een grafische kaart.

Dynamic white-box testing en Debugging

Het is belangrijk dat white-box testing niet verward wordt met debuggen. Het doel van white-box testing is om bugs te vinden, bij het debuggen ben je juist bezig met het fixen van bugs. Als een tester is het de taak om het probleem te demonstreren in de meest simpele case. Bij white-box testing kan dat betekenen dat er aangegeven wordt welke regels code er verdacht uitziet, waarna de programmeur gaat debuggen om er exact achter te komen wat de oorzaak van het probleem is.



Figuur 13-1 Verschil in white-box testing en debuggen

Voorbeeld van een white-box test

Om een beeld te krijgen hoe white-box testing ongeveer werkt is er hieronder een stuk C++ code te zien van de game *Wolfenstein: Enemy Territory:*

```
const char *TeamName(int team){
      if (team==TEAM_AXIS)
            return "RED"
      else if (team==TEAM_ALLIES)
            return "BLUE"
      else if (team==TEAM_SPECTATOR)
            return "SPECTATOR"
      return "FREE"
}
```

Er zijn 4 white-box tests nodig om het gedrag van elke regel code van deze module te kunnen testen. De eerste test zal het aanroepen van de `TeamName` functie zijn met de paramater `TEAM_AXIS` en vervolgens controleren of de string "`RED`" wordt terug gegeven. Ten tweede moet je de waarde van `TEAM_ALLIES` doorvoeren en controleren of "`BLUE`" wordt terug gegeven. Vervolgens test je de parameter `TEAM_SPECTATORS` om te controleren of "`SPECTATORS`" wordt terug gegeven. En als laatst voer je een niet bestaande waarde door zoals `TEAM_NONE` en check of "`FREE`" wordt terug gegeven. Samen hebben deze vier tests elke regel code doorlopen en wordt het gedrag van de "true" en "false" takken van de `if` statement gecontroleerd.

Het bovenstaand voorbeeld laat verschillen zien tussen black-box testing en white-box testing zoals:

- Black-box testing hoort alle verschillende manieren te testen waarop je een testwaarde in de software kan kiezen, zoals verschillende menu's en buttons. White-box testing eist dat de waarde daadwerkelijk doorgevoerd wordt in de code om te kijken wat de uitkomst hiervan is.
- Door het kijken naar een module wordt het mogelijk, door middel van white-box testing, alle mogelijke waardes te vinden die in de module ingevoerd en verwerkt kunnen worden. Niet alle mogelijkheden zijn altijd in de productspecificaties te vinden waar black-box testing op berust.
    - Toegang hebben tot de code kan een goed hulpmiddel zijn om black-box tests te designen.
- Black-box testing heeft als eis dat het zoveel mogelijk op een consistent platform en configuratie uitgevoerd moet worden. White-box testing heeft alleen de minimale resources nodig die een module vereist.

Er wordt verder niet meer ingegaan op dynamic white-box testing, het bovenstaande is vooral ter illustratie wat er naast het functioneel testen van software verder nog meer komt kijken bij het vak testen.

14 Testing - Conclusie en Aanbeveling

Dit onderzoek is aan de hand van de volgende hoofdvraag uitgevoerd:

- *Hoe kan Keesing Games de kwaliteitsbewaking verbeteren voor Yezzle?*

Hier zijn veel verschillende antwoorden op te geven, maar een groot gedeelte van dit onderzoek is hoofdzakelijk gericht op software testing. Zelf vanuit het perspectief van software testing is er geen eenduidig antwoord hier op te geven. Het doel van het onderzoek was om het vak testing te integreren in de development proces van Yezzle. Dit kan alleen maar lukken wanneer iedereen binnen het team het nut van testing gaat inzien. Alleen een rapport schrijven met informatie over testing is niet voldoende om dit te bereiken. De beste manier om testing door te laten dringen is als iemand het voordoet. Daarom is een deel van het afstudeerproject ook uitgevoerd als een tester, waarbij ik alle testwerkzaamheden van Yezzle tijdelijk op mij had genomen. Het gevolg hiervan was dat men ging inzien hoe makkelijk er fouten gemaakt kunnen worden en hoe belangrijk het dus is dat de software grondig gecontroleerd wordt. Naast deze inzicht moeten er voor het verbeteren van de ontwikkelingsproces in elk geval de volgende elementen aangepakt worden:

1. Er moet meer gedocumenteerd worden
2. Iedereen moet weten hoe het OTAP proces werkt.
3. Iedereen moet weten hoe er effectief omgegaan moet worden met het bug tracking software.

Deze punten worden toegelicht:

1. Ik kan niet sterk genoeg benadrukken hoe belangrijk documenteren is! Het is al ruim aan bod gekomen in de conclusie van hoofdstuk 5. Documentatie is belangrijk voor de speelbaarheid *en* voor het testen van producten. De documentatie moet helder en vooral specifiek zijn. Er moet niks over gelaten worden aan kans. Zodra er overal documentatie voor is dan wordt het mogelijk om al vanaf het begin van de ontwikkeling van een feature of game te gaan testen. Alleen dat is het mogelijk om de volledige potentie van het testen te benutten. Voor nu zou er al begonnen kunnen worden met het testen van de specificaties die uitgeschreven worden door de website project manager.

   Documenteren kost veel tijd, maar over een langere periode zal gezien worden dat het tijd en geld bespaard doordat de efficiëntie en kwaliteit van het hele developement proces en de producten verbeterd wordt.

2. Er heerst nog altijd enige onduidelijkheid binnen het teamen bij NiNtec over hoe OTAP in zijn werking gaat. Er kan helaas geen specifieke aanbeveling gegeven worden, doordat ik niet voldoende kennis heb van Subversion (versie beheer programma). Wat ik wel kan aanraden is dat er discussie gestart moet worden over hoe er verder gegaan wordt met OTAP. Denk na over meer regelmaat invoeren voor het overzetten van builds van de ontwikkel server tot de live server. Creëer vaste tijdstippen hiervoor, zodoende weet iedereen waar hij aan toe is en er komt meer regelmaat in het testen. Zodra er een vaste tijdstip is voor het doorvoeren van een build dan hoort iedereen te weten wanneer er getest moet worden. Dan is nog wel een probleem, men moet weten wat er getest moet worden. Er wordt door voornamelijk door maar één persoon binnen het projectteam op een wiki pagina gedocumenteerd wat de veranderingen zijn. Mijn advies is om meer gebruik te maken van deze wiki pagina. Maak intern en met NinTec concrete afspraken over wie er wat op zet en dan moet er ook voor gezorgd worden dat men deze wiki veel in de gaten gaat houden.

3. Er zijn te veel bugs ontstaan doordat NinTec, het Indiaans bedrijf dat alle website features ontwikkeld, issues in Mantis (bug tracking software) verkeerd interpreteert. Intern wordt NinTec beschuldigd van het leveren van slechte producten. Soms is dit waar, maar vaak ook niet. Het bedrijf NinTec heeft de (nare) gewoonte om precies te doen wat er gevraagd wordt en niet meer. Wanneer een bedrijf zo te werk gaat, dan wordt het des te belangrijker dat de feature requests en bug rapportages aan de kant van Keesing Games zo specifiek mogelijk opgesteld worden en geen gaten bevatten. Daarom moet iedereen die contact heeft met NinTec weten hoe belangrijk het is om dit goed te doen. Mijn advies is dan ook om mantis meer te gaan gebruiken als een project managment tool zoals in hoofdstuk 10 beschreven is en daarnaast voortaan alle bugs die aangemaakt worden in Mantis op te stellen zoals in hoofdstuk 8 omschreven is.

Alle drie de punten komen in essentie neer op het meer moeten documenteren. Vanuit een Quality Assurance (QA) perspectief is dit erg belangrijk. QA houd zich voornamelijk bezig met de kwaliteit van het ontwikkelingsproces waaronder auditing, tracking, en rapporteren vallen, het staat een stap boven het puur testen van software. Men kan een tester aannemen om alle producten te testen, maar dit haalt niet het onderliggend probleem weg. Veel beter zou zijn om de kwaliteit van het ontwikkelingsproces te verbeteren. Op het moment is er geen aansprakelijkheid, niemand kan volledig verantwoordelijk gehouden worden voor de producten. Vanuit een QA standpunt is dit een slechte zaak. QA houdt zich onder andere bezig met het bijhouden van de effectiviteit van de werknemers, alleen dit is op het moment niet mogelijk doordat er niet gedocumenteerd wordt. Men kan weg komen met slechte producten afleveren omdat er nergens zwart op wit staat wat er precies gemaakt moet worden. Maar zodra alle specificaties (in detail!) wel zwart op wit staan, dan kan er bewezen worden dat een developer goed of slecht bezig is. Zodoende kunnen er consequenties gesteld worden voor wanneer iemand onder de norm aan het presteren is, en zouden de werknemers die goed presteren beloond kunnen worden. Dit klinkt wellicht wat kinderachtig, maar het zit in de aard van de mens om goed te willen presteren zodra zij weten dat hun voortgang bijgehouden wordt. Een gevaar hiervan is wel dat het de sfeer negatief kan beïnvloeden, maar dit is niet per se waar. Men zal in het begin wellicht niet blij zijn, maar men zal vanzelf bijtrekken zodra zij merken dat de kwaliteit van de producten die geleverd worden aan het verbeteren is. Daarom wordt het nog één maal herhaald, documenteer meer en beter.

15 Evaluatie

De uiteindelijke wens van Keesing Games was het verkrijgen van een vorm van kwaliteitsbewaking voor de producten die gemaakt worden voor Yezzle. Mijn project heeft dit op verschillende manieren succesvol over gebracht wanneer het software testing betreft. De grootste en meest fundamentele problemen die ik ben tegen gekomen zijn opgelost doordat ik alles wat ik onderzocht had direct ook toepaste in de vorm van een dienst als tester. Het ging dan om het nut van testen niet inzien, het slecht rapporteren van bugs en het niet verifiëren van opgeloste bugs. Ongeacht of er een tester in dienst is of niet, deze drie problemen zijn opgelost doordat testen een norm is geworden. Wanneer we kijken naar de gestelde hoofdvraag van het project dan zien we dat hier geen eenduidig antwoord op te geven is:

- *Hoe kan Keesing Games de kwaliteitsbewaking verbeteren voor Yezzle?*

Wel is er een structureel verschil te zien in de ontwikkeling van de producten van Yezzle wanneer we het begin van het semester vergelijken met hoe men nu te werk gaat. Er is nu te zien dat testen een gewoonte is geworden, dit komt voor een groot deel doordat de drie, zojuist genoemde problemen, door mijn project aangepakt zijn. Keesing Games is zelfs een stap verder gegaan door de ontwikkel proces OTAP te adopteren voor Yezzle waar testing een groot onderdeel in speelt. Om alle test werkzaamheden effectief uit te kunnen voeren is het belangrijk dat alle werknemers de samenvatting over testing en het testplan uitvoerig bestuderen. Pas dan zijn de producten echt effectief. Er zal echter pas een verschil in de kwaliteit van de werknemers te zien zijn wanneer zij weten wat er allemaal bij komt kijken voor het testen van software. Wanneer een goede software developer weet waar tester naar zoeken dan gaat hij ervoor zorgen dat zijn software zo grondig ontwikkeld is dat alle standaard test die beschreven staan in dit document niet voor kunnen komen.

Het probleem is alleen dat software testing zoveel meer omvat dan wat er beschreven staat in dit document waardoor het onrealistisch is om te vragen aan de werknemers om zich verder te gaan verdiepen in de wereld van testing. Daarom komt het advies die ik maak van een meer Quality Assurance kant dan van een software testing kant, namelijk het advies dat er in alle opzichten meer of beter gedocumenteerd moet worden.

Naast de technieken van het testen willen overdragen was het ook nog mijn doel om binnen mijn tijd als afstudeerder bij Keesing Games de playability te verhogen van het Yezzle platform. Het kan worden gezien als een onderdeel van kwaliteitsbewaking omdat het plezier die de speler beleeft van het spelen van de games een onderdeel uit maakt van de kwaliteit van het platform. Echter dit is wel een volkomen ander vakgebied, het is een onderdeel van game design, niet testing. Het moeten opsplitsen van mijn werkzaamheden over twee verschillende vakgebieden is erg lastig en tijd consumerend gebleken. Al had ik wel als voordeel dat ik voorkennis had van game design doordat ik mijn vorige stage en minor uitgevoerd had als game designer. Daardoor zijn de analyse die ik gemaakt heb van MS Solitaire en Bejeweled Blitz degelijk (zie bijlage A). Hier zijn verschillende lessen uit geleerd die ik geprobeerd heb toe te passen op de Braintrainers van Yezzle.nl (zie bijlage B). Dit is op verschillende niveaus wel en niet gelukt. De grootste zwaktepunt van de playtests en analyses van de Braintrainers is dat ik niet ervaren genoeg ben in het vak game design om met zekerheid te kunnen zeggen wat er aan een game moet gebeuren om deze *fun* te maken. Bezig gaan met speelplezier is meer een kunst dan een wetenschap. Een kunst die puur gebaseerd is op ervaring. Daarom was mijn aanbeveling om een game designer aan te nemen. Dit heb ik niet lichtelijk besloten, het gaat om veel geld. Ik heb deze aanbeveling gemaakt omdat ik er *zeker* van ben dat dit ten goede komt van Keesing Games in zijn geheel.

Wat deze scriptie hoopt te bereiken is het overbrengen van de mindset die nodig is om te testen, en om meer nadruk te leggen op *fun* in games. Het kunnen switchen van de mindset van een developer naar de mindset van een tester is waardevolle skill om te hebben en zorgt voor het kunnen afleveren van kwalitatief hogere producten. Maar naast deze mindset hoop ik ook door te laten dringen dat het niet of nauwelijk documenteren hoogstwaarschijnlijk het onderliggend probleem is van het gebrek aan kwaliteit en dat dit dus opgelost moet worden.

De aanbeveling om een game designer aan te nemen is onlangs opgepakt door management. Er is op het moment een vacature open voor een grafische vormgever, maar aan de hand van mijn aanbeveling wordt er nu gekeken of er niet iemand gevonden kan worden die ervaring heeft in beide vormgeving en game design.

16 Proces en planning

In dit hoofdstuk wordt de manier van plannen toegelicht, de gemaakte strokenplanningen worden gegeven en besproken en wordt de procesevaluatie gegeven waarbij de vertraging in het uitvoeren van de planning wordt gereflecteerd.

16.1 Planning

Er is gekozen om een strokenplanning te maken in Microsoft Excel, deze planning is drie keer aangepast.

De reden voor het aanpassen van de strokenplanning is vanwege het niet accuraat kunnen inschatten hoe lang alles zou duren. Het vak Testing wordt niet gegeven op Mediatechnologie, in mijn vrije tijd had ik voordat ik aan het afstuderen begon mij al een beetje verdiept in Software Testing. Ik had alleen niet genoeg kennis in huis om aan het begin van het project een perfecte inschatting te kunnen maken over hoe lang bepaalde onderdelen zouden duren.

Er was van te voren afgesproken dat een deel van het afstuderen voor Keesing Games zou bestaan uit een dienst, namelijk het testen van Yezzle. De afstudeerperiode viel precies in een periode die vol met deadlines zat voor Yezzle. Dit betekende dat ik de beslissing had genomen om het testen van Yezzle.nl op mij te nemen. De testwerkzaamheden koste mij meer tijd dan ik gedacht had, en daarom zijn er bepaalde onderdelen verschoven.

Het onderzoek naar QA / testing is op de planning doorgetrokken naar het einde van het project. Dit is gedaan omdat ik elke keer weer nieuwe elementen van het testen tegen kwam die meegnomen moest worden in het onderzoek. Dit onderdeel verleende zich ook goed voor het overlappen van de andere onderdelen. De gemaakte planningen zijn in het vervolg van dit hoofdstuk te zien en worden daarbij ook toegelicht.

16.1.1 Initiële planning

Week / Activiteiten	01-02 / 04-02	07-02 / 11-02	14-02 / 18-02	21-02 / 25-02	28-02 / 04-03	07-03 / 11-03	14-03 / 18-03	21-03 / 25-03	28-03 / 01-04	04-04 / 08-04	
Orrientatie											
Onderzoek Testing											
Onderzoek testing verwerken											
Onderzoek fun / playability											
Onderzoek fun / playability uitwerken											
Testplan Yezzle opstellen											
Presentatie voorbereiden											
Scriptie											
Uitloop											

Week / Activiteiten	11-04 / 15-04	18-04 / 22-04	25-04 / 29-04	02-05 / 06-05	09-05 / 13-05	16-05 / 20-05	23-05 / 27-05	30-05 / 02-06	06-06 / 10- 06	13-06 / 17-06	20-06 / 24-06
Orrientatie											
Onderzoek Testing											
Onderzoek testing verwerken											
Onderzoek fun / playability											
Onderzoek fun / playability uitwerken											
Testplan Yezzle opstellen											
Presentatie voorbereiden											
Scriptie											
Uitloop											

Zo zag de planning er in eerste instantie uit. Dit was een ruwe schatting waarvan na ongeveer een maand duidelijk werd dat de planning aangepast moest worden doordat er tijdens de opstart van het project al vertraging was opgelopen. Het project had tijdens de opstart nog geen officiële goedkeuring gekregen waardoor het nog niet duidelijk was welke onderdelen van het testen onderzocht moest worden. Het was kon namelijk vier richtingen op: black-box testing, white-box testing, automated testing of play testing. Hierdoor werd het tijdens de eerste maand lastig een specifiek onderzoek uit te voeren. Dit is de reden waarom er opnieuw gepland werd waarbij de verloren tijd heringedeeld werd.

16.1.2 Tweede planning

Week / Activiteiten	21-02 / 25-02	28-02 / 04-03	07-03 / 11-03	14-03 / 18-03	21-03 / 25-03	28-03 / 01-04	04-04 / 08-04	11-04 / 15-04	18-04 / 22-04	25-04 / 29-04
Orrientatie										
Onderzoek Testing										
Onderzoek testing verwerken										
Onderzoek fun / playability										
Onderzoek fun / playability uitwerken										
Tesplan Yezzle opstellen										
Presentatie voorbereiden										
Scriptie										
Uitloop										

Week / Activiteiten	02-05 / 06-05	09-05 / 13-05	16-05 / 20-05	23-05 / 27-05	30-05 / 02-06	06-06 / 10- 06	13-06 / 17-06	20-06 / 24-06
Orrientatie								
Onderzoek Testing								
Onderzoek testing verwerken								
Onderzoek fun / playability								
Onderzoek fun / playability uitwerken								
Tesplan Yezzle opstellen								
Presentatie voorbereiden								
Scriptie								
Uitloop								

Nadat het project officiële goedkeuring had gekregen werd het vanuit school duidelijk dat er voornamelijk in de richting van black-box testing en play testing onderzocht moest worden, in andere woorden functioneel testen. De nadruk moest op het functionele liggen doordat er dan aan het mediatechnologische gehalte voldaan werd, er is dan een link van techniek naar de gebruiker. In deze tweede planning is er voor gekozen om het onderzoek te laten overlappen met het uitwerken van het onderzoek. Dit zorgde voor meer efficiëntie in het onderzoekende proces. Op deze manier werd het mogelijk om de verloren tijd in te halen waarbij de kwaliteit niet of nauwelijks omlaag zou gaan.

16.1.3 Derde planning

Week / Activiteiten	21-03 / 25-03	28-03 / 01-04	04-04 / 08-04	11-04 / 15-04	18-04 / 22-04	25-04 / 29-04	02-05 / 06-05	09-05 / 13-05	16-05 / 20-05	23-05 / 27-05
Orrientatie										
Onderzoek Testing										
Onderzoek testing verwerken										
Onderzoek fun / playability										
Onderzoek fun / playability uitwerken										
Testplan Yezzle opstellen										
Presentatie voorbereiden										
Scriptie										
Uitloop										

Week / Activiteiten	30-05 / 02-06	06-06 / 10- 06	13-06 / 17-06	20-06 / 24-06
Orrientatie				
Onderzoek Testing				
Onderzoek testing verwerken				
Onderzoek fun / playability				
Onderzoek fun / playability uitwerken				
Testplan Yezzle opstellen				
Presentatie voorbereiden				
Scriptie				
Uitloop				

De planning is vervolgens nog een derde keer aangepast doordat het onderzoek naar het vakgebied testing eigenlijk een doorlopend onderdeel werd. Dit was het gevolg van het testen van Yezzle. Het testen van Yezzle liet steeds meer slechte aspecten van het development proces zien waardoor er constant meer onderzoek gedaan moest worden naar oplossingen voor de problemen.

Een voorbeeld hiervan is het feit dat er punten werden aangemaakt in Mantis als bugs, waarna deze issues weer terug kwamen met als commentaar dat het een feature is en geen bug. Dit gebeurde zo vaak dat er gekeken moest worden waardoor dit precies kwam. In dit geval kwam het door het gebrek aan documentatie, zonder de productspecificaties is het niet mogelijk om tijdens het testen precies te weten wat het product hoort te doen, het gevolg hiervan is dat er bugs gevonden worden die features zijn, of bugs niet worden gevonden doordat bepaalde elementen als features worden aangezien, terwijl deze eigenlijk bugs waren. De oplossing hiervoor is was het

advies geven dat er documentatie geschreven moet worden voordat men begint met bouwen, maar dit betekende dat er verteld moet worden hoe je het beste documentatie kunt testen.

Een zodanig onverwacht onderdeel kost veel tijd om precies uit te zoeken waar het probleem zijn oorsprong vindt en hoe dit vervolgens het beste opgelost kan worden. Meerdere van deze “onverwachte” problemen hebben ervoor gezorgd dat de planning aangepast moest worden. Dit is op zo een manier gedaan dat er overlap van producten plaats moest gaan vinden (het testplan en playability onderzoek overlappen). Er is hiervoor gekozen om ervoor te zorgen dat er nooit een “dood” moment plaats kan vinden. Zodra er een moment was waar een activiteit even stagneerde werd er geswitcht van product zodat er later met een frisse blik naar het probleem gekeken kon worden, in plaats van veel tijd kwijt zijn doordat je eventjes vast zit.

16.1.4 Uitloop van het project

Bij de deadline van inleveren van de scriptie werd het duidelijk dat deze deadline niet gehaald ging worden. De reden hiervoor was dat veel producten maar half af waren. Hierna is er besloten om alle focus te leggen op het afmaken van het playability onderzoek. Het afmaken van dit product was alleen erg tijdconsumerend doordat er gewerkt moest worden aan een onderdeel dat van andere mensen afhing. Er moest namelijk geplay test worden door verschillende mensen en omdat niet alle familieleden, kennissen en collega’s vaste tijdstippen konden aangeven werd het plannen van dit onderdeel erg lastig. Het gevolg hiervan was dat er besloten is per dag de beschikbaarheid van de vrijwilligers te bekijken en het maken van het product hieraan aan te passen.

Op 28 juni werd door een e-mail van Petra Verbeek duidelijk dat ik verkeerd te werk ben gegaan met het maken van de scriptie. Dit zorgde ervoor dat alles “on hold” ging zodat er 100% op het schrijven van de scriptie gericht kon worden. De hele maand juli is gebruikt om de scriptie te herschrijven.

Week / Activiteiten	06-06 / 10-06	13-06 / 17-06	20-06 / 24-06	27-06 / 01-07	04-07 / 08-07	11-07 / 15-07	18-07 / 22-07	25-07 / 29-07	01-08 / 05-08	08-08 / 12-08
Playability onderzoek										
Testplan										
Scriptie										

Week / Activiteiten	15-08 / 19-08	22-08 / 26-08
Playability onderzoek		
Testplan		
Presentatie voorbereiden		

16.2 Proces Evaluatie

Er zijn op verschillende momenten planningen gemaakt en aangepast waarvan er een paar in het vorige hoofdstuk te zien zijn, dit is op een serieuze manier uitgevoerd. Maar om met de deur in huis te vallen, er zijn fouten gemaakt in het plannen van het project. De voornaamste reden van de gemaakt fouten is het feit dat het afstudeertraject om een project ging waar geen aandacht aan besteed is binnen de opleiding. Hierdoor werd het erg lastig in te schatten hoeveel tijd de verschillende onderdelen zouden kosten. Praktisch alle elementen waren meer tijdconsumerend dan ik in eerste instantie verwacht had.

Ook bij het houden aan de planning zijn er fouten gemaakt. Sommige fouten zijn bewust gemaakt, andere niet, maar één ding is wel zeker, ik heb enorm veel geleerd van het afstudeertraject. De redenen waarom er niet altijd aan de planning gehouden is luidt als volgt:

1. Erg trage start
2. Veel voor het bedrijf gewerkt
3. Te veel hooi op mijn vork genomen
4. De scope niet strak genoeg gedefinieerd
5. Producten niet afmaken
6. Scriptie verkeerd ingeschat
7. Evoluerende ontwikkel methodes
8. Problemen met taal

1. Erg trage start

Doordat ik het afstudeer traject begonnen ben met een voorlopige goedkeuring, vond ik het erg lastig om tijdens de eerste maand echt hard aan de slag te gaan omdat ik er nog niet zeker van was wat het uiteindelijke project precies ging worden. Hierdoor is de eerste maand van het project erg inefficiënt ingevuld. Er is tijdens de opstart van het project wel een hoop gedaan, maar dit ging voornamelijk om activiteiten als het juist opstellen van de afstudeer overeenkomst, het testen van Yezzle.nl en zonder een doel research doen naar testing. De planning eiste eigenlijk om tijdens de eerste maand van het project een samenvatting van alle relevante onderdelen voor het project klaar te hebben. Dit is uiteindelijk niet gelukt waardoor er een sneeuwbal effect ontstond en alle andere onderdelen van het project last hadden van deze “verloren” maand.

Verder kwam het ook voor dat er tijd kwijt geraakt is in maand 2 doordat er op bepaalde momenten niet full time aan het project gewerkt kon worden. Dit kwam door een bruiloft in Amerika waarvoor ik “best man” was en het overlijden van mijn oma waardoor ik een kleine week voor de begrafenis naar Engeland gegaan was.

Ondanks de bruiloft en begrafenis is het een bekend probleem onder studenten dat men de neiging heeft om pas aan het einde van een project te beginnen, waardoor er tijdsnood ontstaat. Ook al was het was het geen pretje dat het zo gelopen is, het is wel een goede les om nog één laatste keer te leren voordat ik de professionele wereld betreedt. Besteed alle tijd nuttig! Een goed begin is het halve werk!

2. Veel voor het bedrijf gewerkt

Ik heb relatief veel meegedraaid in het bedrijf als tester. Één van de op te leveren producten die ik gedefinieerd had was een dienst, namelijk het testen van Yezzle.nl. Het idee hierachter was alle opgedane kennis direct uit te proberen in de praktijk. Uiteindelijk vond ik het toepassen van de skills leuker dan het werken aan mijn project, waardoor ik snel geneigd was om alle gevraagde producten te testen. Dit is de keuze die ik soort van bewust gemaakt had doordat ik het erg relevant vond om zoveel mogelijk ervaring op te doen. Sinds mijn oriënterende stage ben ik mij ervan bewust dat in de professionele wereld, en dan met name de game industrie, ervaring bijna alles is! Men heeft veel liever iemand met veel ervaring dan iemand die alleen een opleiding achter de rug heeft. Daarom heb ik ervoor gekozen om mijn afstudeertraject ook te gebruiken om

professionele ervaring op te doen. Omdat ik mijn prioriteiten gedeeld had gebeurde het dus dat er moeilijkheden waren met het opvolgen van de planning, daardoor is mijn project ook uitgelopen.

Deze keuze heeft ervoor gezorgd dat het afstudeerbedrijf de afspraak met mij gemaakt heeft dat zodra ik klaar ben met afstuderen ik een baan krijg aangeboden als software tester binnen Keesing Games. Dit lijkt mij uiteindelijk de reden waarom het afstudeertraject voor Mediatechnologie binnen een bedrijf moet plaatsvinden.

3. Te veel hooi op mijn vork genomen

Mijn afstudeerproject zijn eigenlijk twee volwaardige projecten in één. Enerzijds is er het project betreffende software/game testing, anderzijds is er een project betreffende "fun" in games (hier houdt een game designer zich voornamelijk mee bezig). Het zijn twee verschillende vakgebieden waar wel enigszins overlap in zit maar voor beide onderwerpen is een volwaardig onderzoek aan te besteden. Deze twee onderwerpen zijn gecombineerd (aan de hand van het advies van een mediatechnologie leraar) om zo het mediatechnologisch gehalte van het project te verhogen. Door ook op speelbaarheid te testen is er een link van techniek naar gebruiker, hier staat Mediatechnologie voor, wat niet het geval was geweest als er puur gekeken was naar het testen van software.

4. De scope niet strak genoeg gedefinieerd

Het definiëren van het bereik van het project was in het afstudeervoorstel al aangegeven als een gevaar. Mijn vermoedens werden helaas gerealiseerd. Omdat het vak testen erg breed is en er ook nog eens een deel van het vak gebied game design meegenomen werd was het erg lastig om het project te beperken tot specifieke onderwerpen zonder dat het als incompleet overkwam. Elke keer dat er gekeken werd naar een specifiek onderwerp, werden er nog drie onderwerpen gevonden die ook onderzocht zouden moeten worden. Alle verschillende onderdelen stapelde zich zo hoog op dat het onmogelijk werd om alles uitvoerig aan te pakken. Hierdoor is er veel tijd verloren gegaan omdat ik toch een poging heb gedaan om zoveel mogelijk mee te nemen. Uiteindelijk was het slimmer geweest om het project te beperken tot maar één specifiek onderwerp en dit uitvoerig te bespreken.

Het te breed houden van het bereik van het project had wellicht voorkomen kunnen worden als mijn begeleiders vanuit school en van het bedrijf zelf specialisten waren in het vakgebied van testen, dit was alleen niet zo. Mijn bedoeling is niet om mensen de schuld te geven, ik probeer hiermee aan te geven dat dit een gevaar is wanneer men met een onderwerp bezig gaat waar geen voorkennis van aanwezig is.
Pas aan het einde van het traject kwamen er nieuwe werknemers bij KGG die ruime test ervaring hadden, alleen was het toen al te laat. Ik heb er expres voor gekozen om mijn project *niet* met deze mensen uitvoerig te bespreken doordat dit erg veel verwarring zou creëren in een stadium van het project waar er geen tijd meer was om het uitvoerig aan te passen.

5. Producten niet afmaken

Ik ben tot de realisatie gekomen dat ik een nare gewoonte heb om ergens aan te beginnen en dit tot ongeveer 75% compleetheid te realiseren en dit vervolgens niet direct af te maken. Dit is een bekend probleem voor mij, de oorsprong van dit probleem kan gevonden worden in faalangst hebben. Mijn faalangst stamt uit te perfectionistisch zijn. Hierdoor eis ik van mezelf dat een product perfect moet zijn. Maar wanneer ik van te voren weet dat ik iets niet perfect kan maken, dan durf ik er gewoonweg niet aan te beginnen. Dit betekend dat ik pas op het allerlaatste producten af maakt, omdat het dan niet anders meer kan.

Testen is hierdoor perfect voor mij omdat ik door deze vorm van faalangst juist mijn kritische (perfectionistische) drang kan gebruiken zonder dat ik tegen de barrière aanloop dat ik een product niet *durf* af te maken.

Het grote nadeel hiervan is dat ik tijdens het realiseren van mijn onderzoek maar om heel weinig feedback heb gevraagd doordat ik geen producten wilde laten zien die maar deels af waren. Ik heb mijn bedrijfsbegeleider wel constant op de hoogte gehouden waar ik mee bezig was, maar ik heb geen tussentijdse producten durven laten zien waardoor ik ook maar weinig aangestuurd kon worden door mijn begeleider. Dit is een erg slechte eigenschap geweest voor een project waarbij ik mezelf sowieso al in het diepe heb gegooid. Sterker nog ik ben van mening dat dit probleem de voornaamste reden is geweest voor het uitlopen van het project.

6. Scriptie verkeerd ingeschat

Nadat het project in de herkansingsfase terecht gekomen was, had ik een opgelucht gevoel omdat ik dan niet meer hoefde te haasten om het project af te ronden. Dit gevoel duurde niet erg lang doordat ik uit een e-mail van Petra Verbeek begreep dat ik verkeerd te werk ben gegaan met het realiseren van mijn scriptie. Hoe ik het uit een template (verstuurd door school) begrepen had, is dat de scriptie puur evaluerend hoort te zijn over de producten die gemaakt worden voor het bedrijf, waarbij deze producten de bijlagen vormen. Maar de scriptie, weet ik nu, hoort een alleenstaand product te zijn waar alle benodigde informatie over het gerealiseerde product in de scriptie gegeven moet worden (dus niet als bijlage).

Deze realisatie zorgde ervoor dat de laatste maand voor het inleveren van de scriptie deze herschreven moest worden. Dit zette andere elementen weer on hold.

7. Evoluerende ontwikkel methodes

Sinds ik bij Keesing Games ben begonnen is er een hoop veranderd waardoor bepaalde elementen van mijn onderzoek aangepast moesten worden of geen relevantie meer hadden. Zo is er vlak voordat ik begonnen ben een project manager (een oud Mediatechnologie student) bij gekomen waardoor het ontwikkel proces gestroomlijnd werd. En ik was voor een groot deel zelf de test werkzaamheden op mij aan het nemen, waardoor het ontwikkel proces veranderde waarin testen meegenomen werd. Daarbij is er begin juli ook een senior tester aangenomen en is er een vacature speciaal voor mij gemaakt om als tester te komen werken voor KGG waardoor het ineens vreemd werd om de test procedures op te zetten waarbij ik ervan uit ging dat er geen testers aanwezig zijn (dit was het idee van het project). Hierdoor probeerde ik het project nog aan te passen terwijl er eigenlijk geen tijd meer voor was, dit heeft voor mij veel onduidelijkheden gecreëerd.

Het is een gevaar geweest waar ik tijdens de opstart van het project geen rekening mee had gehouden. Het gevolg hiervan was dat er ook geen plan was hoe ik dit probleem het beste kon aanpakken. Dit heeft uiteindelijk voor een hoop problemen gezorgd en daarbij is ook veel tijd verloren gegaan.

8. Problemen met taal

Ik ben volkomen tweetalig opgevoed, thuis wordt Engels gesproken, op school Nederlands. Dit heeft ervoor gezorgd dat ik goed ben in beide talen, maar beide talen niet volledig beheers. Het schrijven van alle documenten is een constante strijd geweest om de spelling en grammatica goed te krijgen. Ruim de helft van de tijd die gebruikt is om dit document te schrijven is naar het spellchecken gegaan. Dit is uitermate frustrerend geweest, vooral omdat het niet een probleem is dat snel opgelost kan worden. Ik had aan het begin van het semester met mijn bedrijfsbegeleider afgesproken om alles in het Nederlands te schrijven, achteraf was het misschien toch makkelijker geweest om het in het Engels te doen.

Dit document is wel door verschillende mensen gecontroleerd op spelling, maar door de grote hoeveelheid aan woorden en het gebrek aan tijd is het waarschijnlijk zo dat er alsnog tamelijk wat spelling, grammaticale en interpunctie fouten gemaakt zijn. Mijn excuses hiervoor! Het is iets waar ik me bewust van ben en ik ben mijn best aan het doen om mijn kennis van taal weer op te frissen.

17 Reflectie

Dit hoofdstuk geeft de reflectie van het proces en de te behalen competenties weer.

17.1 Reflectie op de competenties

Het afstudeertraject is niet vlekkeloos verlopen maar dit heeft er wel voor gezorgd dat er bepaalde fouten niet meer gemaakt zullen worden.

Inzicht krijgen
In het afstudeervoorstel stond beschreven dat ik inzicht in het game development proces, outsource development en de doelgroep moest krijgen om het project succesvol te kunnen afronden. Dit streven is behaald, door mee te draaien in het bedrijf is er snel inzicht in de bedrijfscultuur gekregen. Dit inzicht heeft ervoor gezorgd dat bepaalde prioriteiten van het project verschoven zijn. Zo was er eerst het idee om voornamelijk adviezen te doen betreffende het vak testen, maar door goed te kijken naar waar het fout gaat bij de ontwikkeling van de games is er ook besloten om het belang van documenteren te laten zien. Het adviseren om product specificaties / game design documenten te schrijven en deze te testen is een onderdeel dat geen onderdeel uitmaakt van het vak testen maar thuis hoort bij het hoger liggende vak Quality Assurance. Naast het inzicht krijgen in de bedrijfscultuur is er ook erg veel kennis en inzicht in verschillende testtechnieken opgedaan. Van de volgende vaardigheden is ruime kennis opgedaan:

- Static Black-box testing
- Dynamic Black-box testing
- Instruction design
- Achievement design
- Casual level design
- Playtesting

Ontwerpen
Omdat ik veel bezig ben geweest met het vak testing is het zo dat het onderdeel “ontwerpen” niet erg veel naar voren is gekomen. Wat het belangrijkste is voor het testen van software (of adviezen maken over het ontwikkel proces) is het herkenen van zwakte punten. Het kunnen analyseren van processen en software is cruciaal. Bij het maken van een test suite ga je zo te werk dat je bezig bent met het reverse engineeren van software. Als er product specificaties of game design documenten zijn dan creëer je de test cases aan de hand van deze ontwerpen. Als dit niet het geval is dan is het maken van een testsuite een vorm van een ontwerp maken van de software, je stippelt namelijk precies uit wat de software moet doen en dit controleer je zodra het ontwikkeld is.

Plannen
Er zijn twee onderdelen van plannen voorgekomen tijdens het afstudeerproject, namelijk het inplannen van het project en het inplannen van testwerkzaamheden. Bij het plannen van het project zijn er hier en daar wat problemen ontstaan doordat het lastig was om iets in te plannen waar ik nog nooit mee bezig ben geweest (zie hoofdstuk proces en planning). Het inplannen van de test werkzaam ging wel goed, ik had geen moeite met het bijhouden van alle test werkzaamheden. De voornaamste reden hiervoor is dat je als tester vast zit aan de deadlines van de developers, iets kan namelijk pas getest worden wanneer de developers het product aanleveren om getest te worden. Dit betekent dat het inplannen van testwerkzaamheden aan de hand van de planning van de developers gebeurd. Daarnaast zijn er werkzaamheden die dagelijks uitgevoerd moeten worden, dit zorgt voor regelmaat. Dit geeft mij als tester veel flexibiliteit doordat het altijd duidelijk is wat er moet gebeuren.

Uitvoeren
De vaardigheden die ik bij “inzicht krijgen” benoemd heb blijkt uit het feit dat men blij is met mijn test resultaten. In mijn test resultaten geef ik niet alleen bugs aan, maar ik geef ook interactie/game design issues aan. Het aangeven van design issues is niet in eerste instantie de rol van een tester maar omdat er geen game/interactie designer aanwezig is bij KGG en ik hier wel gevoel voor heb (bij mijn oriënterende stage was ik full time bezig met game design), heb ik het op mij genomen om hier toch adviezen over te geven. De onlangs aangenomen senior software tester vraagt ook advies van mij over het testen van games omdat zij een specialisatie heeft in white-box testing en niet in functioneel testen.

Naast de vier generieke Bachelor of Engineering competenties zijn er ook een twee van de Dublin descriptoren die erg van toepassing geweest zijn:

Communicatie
Als software tester is communicatie een erg belangrijk element. Je baan is in principe tegen iemand zeggen dat hij slecht bezig is geweest, dit betekent dat je erg tactisch te werk moet gaan in het rapporteren van een bug. Tijdens het verloop van het afstudeertraject ben ik er achter gekomen dat één van de betere manieren om bugs te rapporteren is het puur feitelijk houden van het rapport. Probeer niet grappig of sarcastisch te doen in de bug rapportages, dit kan namelijk helemaal verkeerd opgevat worden. Wanneer je alleen feiten doorgeeft dan kan dit onpersoonlijk overkomen maar het is dan niet meer discutabel.

Om bugs puur feitelijk door te geven is het dus erg belangrijk hoe men een bug doorgeeft. Toen ik net begon was ik iemand die gewoonweg zei dat “element x niet werkt, fix het”, nu schrijf ik mijn bug rapportages erg uitgebreid, zelfs wanneer het om erg kleine of simpele bugs gaat. Ik geef de precieze stappen mee om het te reproduceren en ik omschrijf het probleem compleet en helder. Dit doe ik zodat er nooit een miscommunicatie kan ontstaan, er is niks meer vernederend voor een tester om een bug terug te krijgen met als commentaar dat het probleem niet begrepen wordt. Dit is niet alleen vernederend maar ook tijdconsumerend en daarbij kost het dus ook geld.

Leervaardigheden
Één van de taken van een tester is op de hoogte zijn van nieuwe technieken. Dit betekent dat zodra er een nieuwe techniek wordt toegepast om een product te maken dan is het essentieel dat een tester zich hierin ook verdiept. Dit betekent dat een tester officieel altijd wat te doen heeft. Zodra er een moment is dat er niks te testen is dan is het een onderdeel van de baan van een tester om deze tijd te gebruiken om nieuwe vaardigheden te leren. Dit is dan ook een groot element geweest van het afstudeertraject van mij. Ik heb het vak testen mijzelf moeten aanleren, elke dag vond ik wel een nieuw techniek of methode om te testen. Dit maakte het een erg interessante periode, maar daarnaast ook een erg frustrerende periode omdat het lastig was om mij te richten op één specifiek element. Het kunnen richten op een enkel onderwerp is dus nog een aandachtspunt voor mij.

17.2 Reflectie professionele competenties

Het is mij verteld dat Mediatechnologie staat voor de koppeling tussen mens en techniek. Als we hierbij puur kijken naar software testing dan zien we dat het de koppeling naar techniek heel erg heeft maar het mist enigszins de koppeling naar de mens toe. Doordat ik mij de afgelopen twee jaar gefocust heb op game design is het voor mij juist leuk geweest om mij binnen het vak testen te richten op functioneel testen. Wanneer men functioneel test is er een duidelijke koppeling naar de gebruiker toe, een deel van de testcases worden ook opgesteld vanuit de schoenen van de gebruiker.

Er is onlangs een senior software tester aangenomen, zij is volledig gericht op white-box testen, het testen van de code, ik vul dit goed aan door mij te richten op het functionele. Naast het vinden van technische bugs houd ik mij ook bezig met het checken voor vormgevingsfouten, tekst fouten, interactie fouten en game design fouten. Omdat ik bij KGG blijf werken krijg ik een unieke kans om kennis te delen met de white-box tester en nieuwe kennis op te doen door samen te werken met de white-box tester. Deze mogelijkheid maakt het mogelijk om mij bezig te gaan houden met automated testing en white-box testing, in andere woorden, er is de mogelijkheid om een volwaardig software tester te worden.

17.3 Profielschets

Na lang zoeken ben ik er nu zeker van dat het beroep *testen* is waar ik mij de komende jaren mee bezig ga houden. Zoals in het hoofdstuk proces evaluatie genoemd is ben ik iemand die problemen heeft met het creëren van producten doordat ik faalangst heb in de vorm van extreem perfectionisme. Deze tekortkoming is juist een voordeel als het om testen gaat, het stelt mij in staat om erg kritisch naar producten te kijken en ben ik niet tevreden met een “Go” geven voor een product waarvan de kwaliteit niet optimaal is. Daarnaast heb ik onlangs de switch gemaakt van game design naar testen, dit was een lastige keuze, maar mijn liefde voor game design stelt mij in staat om mee te denken met design kwesties.

In de toekomst ben ik van plan om mijn skillset door te laten groeien door middel van white-box testing en automated te leren. Dit is een grote onderneming, maar binnen Keesing Games is er de ruimte om hierin te groeien.

Het aannemen van mij en de senior tester betekend dat de organisatie enigszins moet gaan veranderen om effectief gebruik te kunnen maken van het vak testen. Eenmaal afgestudeerd is mijn eerste stap om testen volledig te integreren in de development cyclus van KGG. Wilt men echt vooruitgang zien in de kwaliteit van de producten dan zullen er bepaald structurele veranderingen moeten vinden. Dit zal in overleg met de senior tester en het management van de verschillende projectteams moeten gebeuren.

18 Afkortingen en begrippen

Afkorting / begrip	Uitleg
Acceptance testing	Een test techniek waarbij men nagaat of het product voldoet aan de specificaties of contract.
Black-Box-Testing	Een test methode waarbij er geen kennis is van de interne structuur of code van het programma of systeem. Wordt ook wel functioneel testen genoemd
Casual gamer	Een type gamer die zichzelf niet als gamer beschouwd. Speelt voornamelijk in korte speelsessies en speelt games met weinig complexiteiten / diepgang.
Extrinsieke motivatie	De wil of de drang gecreëerd door externe factoren (beloningen), dus extern (extrinsiek), om bepaalde handelingen te verrichten waarmee tot een zeker doel wordt gekomen.
Casual gamer	Een type gamer die zichzelf niet als gamer beschouwd. Speelt voornamelijk in korte speelsessies en speelt games met weinig complexiteiten / diepgang.
Hardcore gamer	Een fanatiek type gamer. Laat zijn mening graag horen over games en is altijd op de hoogte van de nieuwste games. Spendeert veel tijd aan gamen en speelt graag complexe en diepgaande games.
Intrinsieke motivatie	De wil of de drang in iemand, dus intern (intrinsiek), om bepaalde handelingen te verrichten waarmee tot een zeker doel wordt gekomen.
KGG	Keesing Games, stage bedrijf
Playability	Video game jargon. De term wordt gebruikt om het gemak waarmee het spel te spelen is te omschrijven. Wanneer men de playability wilt verbeteren dan betekend dit dat de game design verbeterd moet worden.
Product specifications	Een document waarin staat waaraan een product moet voldoen.
Self-efficacy	Het vermogen en de overtuiging om adequaat en efficiënt te handelen in een gegeven situatie.
User Acceptance Test (UAT)	Een test techniek waarbij men nagaat of het product voldoet aan de user requirement specifications
White-Box-Testing	Een test methode waarbij men gebruik maakt van kennis van de interne structuur of code van een programma of systeem

19 Bronnen

19.1 Boeken

- Ron Patton, 2006. *Software Testing: Second edition*. USA: Sams Publishing
- C.P. Schultz, R Bryant, T. Langdell, 2005. *Game Testing: All in One*. USA: Thomson Course Technology
- L. Levy, J. Novak, 2010. *Game development Essential: Game and QA & Testing.* USA: Cengage Learning
- Gregory Trefry, 2010. Casual Game Design: Designing play for the gamer in all of us. USA: Elsevier

19.2 Websites

- Richard Bartle, Hearts, Clubs, Diamonds, Spades: Players who suit MUDs
  http://www.mud.co.uk/richard/hcds.htm
- William, Learning the Rules
  http://www.casualgamedesign.com/?p=27
- William, 10 Ways not to design a videogame
  http://www.casualgamedesign.com/?p=30
- Josh Levin, Solitare-y Confinement: Why we can't stop playing a computerized card game
  http://www.slate.com/id/2191295/pagenum/all/
- Bilal Hameed, Farmville about to cruise past 80 million users
  http://www.allfacebook.com/farmville-about-to-cruise-past-80-million-users-2010-02
- Bejeweled Blitz application statistics
  http://statistics.allfacebook.com/applications/single/bejeweled-blitz/40343401983/
- Bejeweled Blitz
  http://apps.facebook.com/bejeweledblitz
- Greg McClanahan, Achievement design 101
  http://www.gamasutra.com/blogs/GregMcClanahan/20091202/3709/Achievement_Design_101.php
- Mary Jane Irwin, Unlocking Achievements: Rewarding skill with player incentives
  http://www.gamasutra.com/view/feature/3976/unlocking_achievements_rewarding_.php?print=1
- Lucas Blair, The cake is not a lie: How to design effective achievements
  http://www.gamasutra.com/view/feature/6360/the_cake_is_not_a_lie_how_to_.php
- Yezzle
  http://www.yezzle.nl/
- Lumosity
  http://www.lumosity.com/

- Wikipedia: Casual Game
  http://en.wikipedia.org/wiki/Casual_game

- Chris Viggers, Usability Testing: Face the Fear and Learn to Love It
  http://www.gamasutra.com/view/feature/6431/usability_testing_face_the_fear_.php

- PDYXS, How to playtest
  http://www.throwthelookingglass.com/category/how-to-playtest

- uTest, Forum door testers over tesing
  http://forums.utest.com

Bijlage A: Case studies

In dit onderdeel van de bijlage zijn de case studies te vinden van populaire casual games. Deze games zijn reversed engineerd om zo goed zicht te krijgen van de verschillende gameplay mechanics die de games zo goed maken.

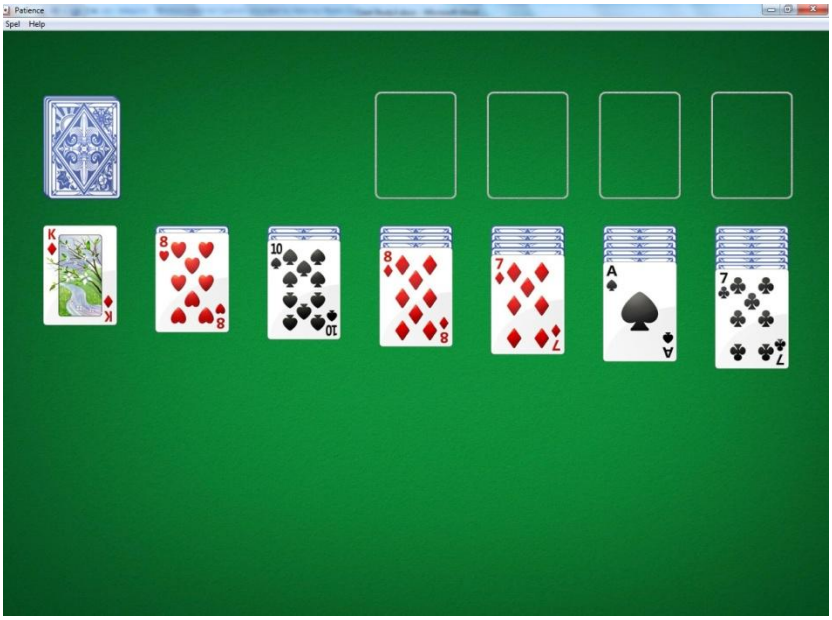
A-1 Microsoft Solitaire

Microsoft Solitaire is de computer versie van de kaart spel Klondike Solitaire, ook wel patience genoemd. Het wordt sinds 1990 geleverd met alle versies van de operating system Windows. De game was in 1989 door een stagiair gemaakt, waarvoor hij geen royalty's gekregen heeft.

Windows Solitaire was in eerste instantie bedoeld om mensen gerust te stellen die geïntimideerd werden door een operating system met een graphical user interface (gui). De game zorgde ervoor dat de gebruiker de basis handelingen leerde gebruiken van een operating system met een GUI. Zo leert het je omgaan met een muis, het principe van klikken en hoe je gebruik maakt van het click-and-drag functionaliteit.

Sinds de release van Windows 3.0 is Solitaire het meest gespeelde computer game in de wereld geworden, en is direct ook het begin van casual gaming. Daarom is dit een perfect voorbeeld om te gaan analyseren.

A-1.1 De game mechanic



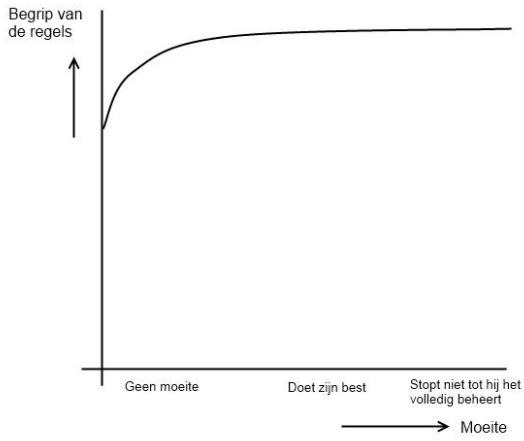
Microsoft Solitaire heeft dezelfde spelregels als Klondike Solitaire. Het is in principe een spel waarbij je op een heel omslachtige manier de kaarten aan het sorteren ben. Je neemt een standaard dek speelkaarten (52 kaarten zonder jokers), waarbij je het speelveld zoals hiernaast indeelt. Zes gesloten kaarten met één (willekeurig) open kaart op de rechter positie. Hier links van liggen er vijf gesloten kaarten met één open kaart erop. Hier weer link van 4 gesloten kaarten met één open kaart erop. De gesloten kaarten verminderen telkens met één kaart tot je alleen een open kaart neerlegt. De vier posities rechtsboven worden gebruikt om kaarten op vorm en grote te sorteren van laag naar hoog (waarbij de aas laag is). Het doel van de game is om alle kaarten in deze vier posities gesorteerd te hebben. De regels van het behalen hiervan zijn als volgt:

- Je mag elke bovenste open kaart op een ander open kaart plaatsen
    - Deze kaart moet wel 1 “punt” lager zijn dan de kaart waarop het geplaatst wordt
    - Deze kaart mag niet dezelfde kleur hebben als de kaart waarop het geplaatst wordt
- Je mag een groep geplaatste kaarten op een ander stapel plaatsten (mits het aan de vorige twee regels voldoet)
- Wanneer je van een stapel alle open kaarten weggespeeld of verplaatst hebt dan mag je de bovenste gesloten kaart open draaien
- Wanneer alle kaarten verplaatst of weggespeeld zijn van een stapel, dan mag op de open plek alleen een koning geplaatst worden
- De resterende kaarten niet gedeelde kaarten worden setjes van drie open getrokken
    - Wanneer alle drie de kaarten weggespeeld zijn, dan klik je op de dek resterende gesloten kaarten en krijg je drie nieuwe open kaarten
    - Wanneer er geen zet mogelijk is met één van deze kaarten, dan klik je op de dek resterende gesloten kaarten en krijg je drie nieuwe open kaarten
- De rechtsboven geplaatste kaarten kunnen terug gezet worden op de stapels hieronder

A-1.2 Goede en slechte punten

A-1.2.1 Instructies

Één van de grootste redenen waarom Solitaire een spel is die zo extreem veel gespeeld is, is omdat de spel voor de meeste mensen geen uitleg nodig heeft. Vooral toen Solitaire net uit was gekomen samen met Windows 3.0 was het nog een kaartspel die erg veel gespeeld werd om wat tijd te doden. Als we dus kijken naar het type learning curve die bij deze game hoort dan zien wat dat het eruit zou zien als de grafiek hiernaast. De spelers begrijpen de regels al voordat ze eraan beginnen, het enige wat ze moeten leren is de switch maken van de fysieke variant naar de computer variant van de game. De controls zijn erg simpel opgezet, hierdoor kan een speler die de regels van Klondike Solitaire al snapt de game in minder dan een minuut oppakken en spelen.



A-1.2.2 Computer vs. Analoog

Één van de grootste irritaties tijdens het spelen van Klondike Solitaire is het moeten schudden van de kaarten en het indelen van het speelveld. Je bent bijna net zoveel tijd kwijt aan het opzetten van het spel als het daadwerkelijk spelen van Solitaire. En zelf tijdens het spelen kon het voorkomen dat je geen zetten meer kan zetten, wat betekend dat je alles weer opnieuw moet gaan schudden en indelen.

Met de komst van de computer versie van Solitaire was deze irritatie ineens verdwenen doordat de computer de kaarten schud en het speelveld voor jou indeelt. De drempel om te gaan spelen is daardoor veel lager geworden. En als er geen zetten meer mogelijk zijn dan geeft de computer dit aan en geeft je direct een nieuw speelveld om mee verder te gaan.

Wat vroeger heel gewoon was, maar ook heel vervelend was, is wanneer men een spel wilde spelen of ander software wilde gebruiken dan moest met dit via een floppy / cd-rom geïnstalleerd worden. Dit maakt de drempel van kopen, instaleren en spelen heel hoog. Dit is niet een probleem waar Microsoft Solitaire last van had omdat het meegeleverd werd op alle versies van Windows (vanaf Windows 3.0).

Met de computer versie van Solitaire hoef je geen kaartdek meer te gaan zoeken, je hoeft de kaarten niet te schudden, je hoeft het speelveld niet in te delen en je hoeft niets te instaleren om het te spelen. Het enige wat je moet hebben is een computer met Windows en je kunt direct gaan spelen.

A-1.2.3 Tijd

Casual gameplay moet zich kunnen aanpassen aan de spelers zijn leven en schema. Wat dit in de praktijk betekend is dat casual games zo ontworpen moeten worden zodat zij in korte speelsessies te spelen zijn. Microsoft Solitaire is zo een game die in korte sessies gespeeld kan worden. Het is een game die tijdens een korte pauze op werk gespeeld kan worden. Begin jaren 90 was Solitair een hot topic in de media en op de bedrijfsvloer. Bazen vonden dat de werknemers te veel Solitaire aan het spelen. Het liep zelfs zo uit de hand dat bedrijven als Coca Cola en Boeing de game van Windows pc's verwijderden en daarbij een verbod op het spelen van game hebben gezet.

Het geniale aan Solitaire is niet alleen dat het geschikt is voor korte speelsessies, maar ook voor lange speelsessies. Er zijn genoeg mensen die meer dan een uur Solitair spelen zonder dat dit gaat vervelen.

A-1.2.4 Actief vs. Passief

Solitaire is perfect voor wanneer je even je verstand op nul wilt zetten, om bijvoorbeeld even ergens anders over na te denken dan je werk. Maar het kan ook actief gespeeld worden. De latere versies van Microsoft Solitaire houden de statistieken van de speler bij. Sommige mensen spelen dus actief om een bepaald doel te halen binnen de game. Een belangrijk element van een casual game is dat je feedback geeft aan de gebruiker. Deze feedback in de vorm van statistieken zorgen in dit geval voor een extra manier om de game te spelen.

A-1.2.5 Bereik

De reden waarom het de meest gespeelde game ooit is en tot op heden ook het meest geopende applicatie is op het Windows besturing systeem is omdat praktisch iedereen de game heeft. De game wordt bij de installatie van Windows geïnstalleerd en Windows is nog verre weg het meest gebruikte besturingssysteem, zo'n 81% van alle computers maakt gebruik van een versie van Windows.

A-1.2.6 Voor iedereen

Solitaire is één van de weinige games die niet gebonden is aan leeftijd, geslacht, intelligentie, game ervaring etc. Iedereen die de regels kent kan het spelen. De besturing is zo simpel dat het zichzelf uitlegt. Daarom kan een oma van 70 jaar oud het net zo goed spelen als een tiener die met computers is opgegroeid.

Het uiterlijk en de gameplay zijn in de loop van de afgelopen 20 jaar nauwelijks veranderd. Waardoor men een constante band heeft met de game. Iemand uit 1990 zou bijvoorbeeld de game in zijn huidige staat kunnen oppakken en gaan spelen.

A-1.2.6 Eenzaam

Één van de weinige slechte punten van MS Solitaire is dat het alleen in je eentje te spelen is. Highscores worden niet online bijgehouden, je kunt geen vrienden toevoegen. Het is dus echt zoals de naam al suggereert een singleplayer game. Dit heeft als grote nadeel dat men niet snel geneigd is de game regelmatig te spelen. Alleen de echte fanatieke spelers zullen genieten van de statistieken die bijgehouden, waar als je de statistieken van vrienden van jou zou kunnen inzien dan had dit ervoor gezorgd dat de game een competitieve element zou krijgen en dit creëert een reden om terug te blijven komen.

A-1.2.7 Samenvattend

De goede punten van Microsoft Solitaire zijn als volgt:

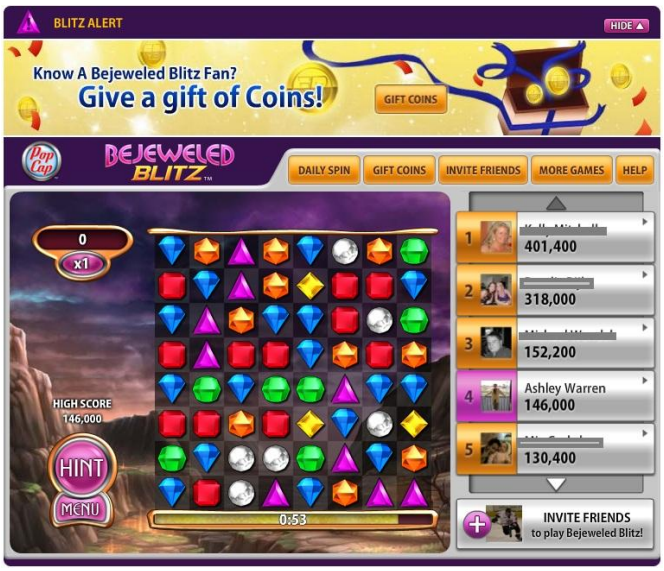
- De meeste mensen kennen de regels van Solitaire al, waardoor de game door geen uitleg nodig heeft.
- Het is een computer versie van een analoog, waarbij de computer versie alle vervelende elementen geautomatiseerd heeft.
- Er is geen installatie nodig
- Het is een game die korte speelsessies gespeeld kan worden wat goed past in het leven van de casual gamer.
- Het is een spel dat zonder na te denken gespeeld kan worden, maar je kan ook actief gaan spelen en je highscore proberen te verbeteren.
- Iedereen die een pc met Windows gebruikt heeft de game.
- De game is erg simplistisch, waardoor iedereen het kan spelen

De slechte punt is:

- Het is singleplayer waardoor er geen prestatie drang wordt geforceerd.

A-2 Bejeweled Blitz

Bejeweled Blitz is de variant van de casual game Bejeweled die te vinden is op Facebook. De originele Bejeweled kwam uit in 2001 en is gemaakt door PopCap Games en is sindsdien meer dan 150 miljoen keer gedownload en 25 miljoen keer verkocht. Daarom is er besloten om deze populaire casual game ook op de social network Facebook te plaatsen waar het op dit moment 3.4 miljoen actieve dagelijkse spelers heeft (zie http://statistics.allfacebook.com/applications/leaderboard/ voor de actuele statistieken). Bejeweled Blitz wordt ook ondersteund door bepaalde mobiele devices, zoals de iPhone.



PopCap heeft met Bejeweled de weg gebaand voor de casual gaming industrie. Zonder bejeweled was er waarschijnlijk geen Zinga geweest, wat betekend geen Farmville wat weer de weg gebaand had voor casual social gaming wat nu een immens groot industrie is geworden.

A-2.1 De game mechanic

De game mechanic van Bejeweled Blitz verschilt vrij weinig met de originele versie van Bejeweled. Waar het op neerkomt, is nog steeds het verplaatsten van gekleurde stenen om zo aaneenschakelingen van drie dezelfde gekleurde stenen te maken.

- Ruil aaneen liggende stenen met elkaar om sets van drie of meer te maken
- Een winnende set is drie of meer stenen van dezelfde kleur
- Na het maken van een set worden de stenen van de set verwijderd en score wordt toegevoegd
- Nieuwe stenen vallen naar beneden om de verwijderde stenen op te vullen
- Sets van meer dan drie stenen creëren bonus stenen
- Maak zo veel mogelijk sets om een zo hoog mogelijke score te bereiken

Dit zijn de basic game mechanics van de game Bejeweled, dit zijn de enige regels die je echt nodig hebt om de game te begrijpen. De game is wel iets complexer dan dit, maar het blijft erg simpel. De volgende regels maken ook onderdeel uit Bejeweled:

- Het speelveld bestaat uit 8x8 = 64 gekleurde stenen
- De stenen worden onderverdeel in 7 verschillende kleuren
- De speler mag alleen horizontaal of vertikaal een steen ruilen als deze een set van drie of meer vormt
- Wanneer nieuwe setjes gemaakt worden door de nieuwe vallende stenen dan verdwijnen deze direct en wordt er bonus score toegevoegd

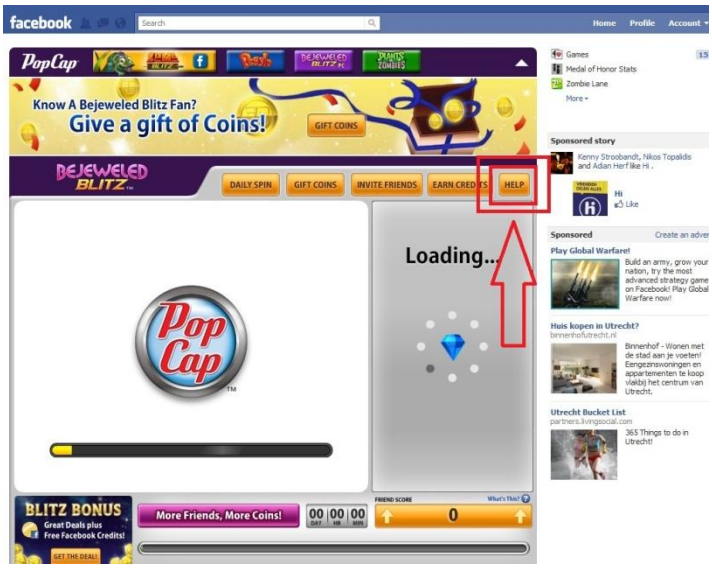
En dan zijn er nog een paar regels die de Facebook Blitz variant iets anders maakt.

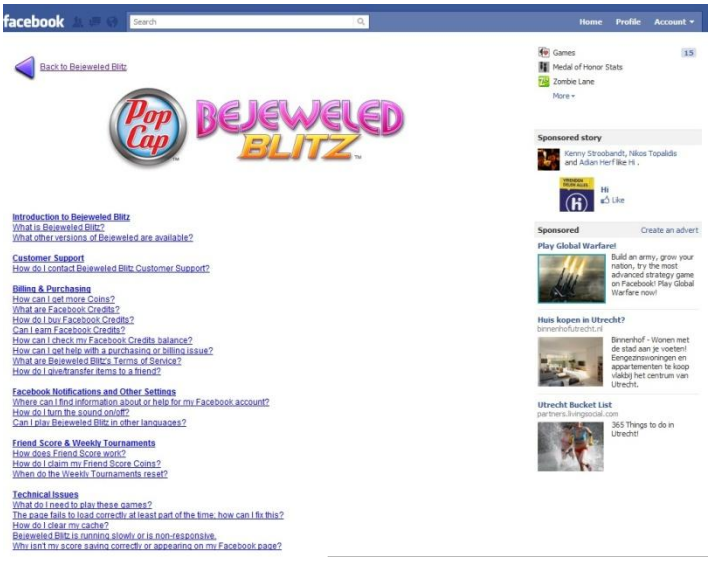
- Er is maar één speel modus, waarbij je binnen 60 seconde zoveel mogelijk setjes moet maken en dus een zo hoog mogelijke score moet zien te behalen
- Aan het einde van elke ronde krijgt de speler aan de hand van zijn score in-game geld
- Met de in-game currency kunnen er 5 verschillende power-ups gekocht worden
- Per ronde kunnen er maximaal drie power-ups gekocht en gebruikt worden
- In-game currency kan via facebook credits gekocht worden
- Één keer per dag kan de speler een draai aan een rad geven om zo gratis in-game geld te winnen
- Je hoogste score wordt opgeslagen en weergegeven samen met de scores van jouw Facebook vrienden die ook Bejeweled Blitz spelen
- Één keer per week worden de scores reset
- Er zit een leveling-up system in, hoe meer de speler speelt hoe hoger hun level wordt.
- In de Blitz variant is het niet mogelijk om geen zetten te kunnen maken

A-2.2 Goede en slechte punten

A-2.2.1 Instructies

Het grote succes van de Bejeweled franchise heeft niet te maken met een groots opgezette marketingcampagne, het succes is bijna 100% te danken aan de design van de game. Het geniale aan Bejeweled is zijn simpliciteit. Als we kijken naar hoe de regels van het spel in Bejeweled Blitz worden uitgelegd dan kunnen we zien dat de game voor een groot deel gebruikt maakt van een strategie dat berust op al bestaande kennis en de intuïtie van de speler. Als je de game namelijk opstart dan moet je moeite doen om een uitleg van de regels te vinden. Het proces is: Start Bejeweled Blitz - > Click Help -> Click “What is Bejeweled Blitz”. Dit lijkt in eerste oogopslag niet als veel moeite maar als je kijkt naar andere games dan is het proces: Start game -> Instructions. Waarbij de knop voor de instructies groot is en in het midden staat. Dit is in het geval van Bejeweled Blitz niet zo, de help knop is klein en rechtsboven en dan kom je in een soort “veelgestelde vragen” lijst waarin je nog moet zoeken welke link de regels zal uitleggen. En zelfs dan is de omschrijving van de game meer als een marketing omschrijving opgeschreven in plaats van een helder uitleg van de regels.





"From award-winning PopCap Games, this one-minute gem-matching challenge is the most intense version of Bejeweled ever! Rack up the highest score you can in one minute by swapping gems to make matches of three or more. Enjoy new score multipliers for huge combos and cascades, and new Star gems and Hypercubes for extra-explosive fun. Accumulate Coins while you play and use them to purchase boosts for even more gem-swapping, explosive action."

De uitleg van de Blitz variant van Bejeweled gaat er dus eigenlijk al vanuit dat de speler al bekend is met het concept van Bejeweled. Wanneer de speler niet bekend is met het concept van Bejeweled dan kan de speler de game nog steeds binnen één of twee minuten snappen omdat de mensheid van nature dingen willen uitproberen. Daarnaast doet de game een beroep op het feit dat mensen altijd dingen willen sorteren. Denk bijvoorbeeld als je een kaartspel speelt, bijna iedereen sorteert de kaarten in zijn handen op kleur en van hoog naar laag. En laat het nou zo zijn dat de core game mechanic van Bejeweled in zijn simplistische vorm gewoon het sorteren van steentjes is. Mensen weten onbewust dus al wat ze moeten doen zodra ze het speelveld zien. Een volledig werkende tutorial of instructie scherm is dan ook niet nodig voor Bejeweled. De eenvoud is één van de grootste succesfactoren van de game.

A-2.2.2 Limiet

Goed

Er is eigenlijk maar één core-regel waar heel Bejeweled op berust: De speler mag alleen een steen omwisselen als dit tot minstens één horizontale of verticale reeks van drie of meer stenen leidt. Alle gameplay stamt af van deze regel. Dit geeft de speler wel een direct gevoel van haast en claustrofobie. Je ziet vaak twee stenen bij elkaar maar je hebt dan twee zetten nodig om de combinatie van drie te maken, of je ziet gewoon geen mogelijkheden. Maar het mooie is dus dat dit gevoel van haast en claustrofobie opgeheven doordat je de kennis hebt dat er altijd een combinatie mogelijk is. De Blitz variant zorgt ervoor dat er altijd een combinatie van drie stenen te maken is en in de normale versie van Bejeweled is het zo dat de game stopt zodra er geen zetten meer mogelijk zijn. En als je het echt even niet meer ziet dan kan je altijd op de "Hint" knop drukken die simpelste reeks aanwijst. De speler wordt dus nooit afgestraft voor het slecht spelen. Dit maakt het voor de casual gamers onbewust zo aantrekkelijk.

Slecht

Waarom heeft Bejeweled zo een negatieve reputatie onder de gamers? Dit komt door dezelfde core game mechanic die het zo een goede casual game maakt. Hardcore gamers spelen games omdat deze hun uitdagen, waarbij je beloond wordt wanneer je goed speelt en afgestraft word wanneer er slecht gespeeld wordt. Dit proces is onmogelijk in Bejeweled omdat je namelijk niet "dood" of "af" kan gaan en je krijgt geen tijd straf wanneer je een zet probeert te zetten die niet tot een reeks van drie leidt. Je wordt zelfs geholpen wanneer je even geen zet meer ziet doordat de game een "Hint" knop heeft en de simpelste combinatie na een tijd automatisch aanwijst. Het feit dat de game je geen negatieve feedback geeft, wat het een briljant casual game maakt, is een reden waarom de game door velen hardcore gamers gehaat wordt.

A-2.2.3 Replayability

Easy to learn, Hard to master

Games die zo makkelijk te leren zijn, zodat ze eigenlijk geen instructies nodig hebben, lopen het risico om gewoonweg te simplistische gameplay aan te bieden waardoor de gamer niet uitgedaagd wordt, het gevolg hiervan is dat de speler niet terug zal komen. Bejeweled heeft dit probleem niet. De game is "easy to learn, hard to master", dus makkelijk te leren, maar moeilijk om er goed in te worden. Je kan de game gewoon spelen op zo een manier dat je alleen reeksen van drie aan een liggende stenen maakt, maar het is ook mogelijk om voor de grotere reeksen te gaan. Dit vereist wel dat de gamer meer bewust bezig is met de game, je moet vooruit gaan denken. Bejeweled krijgt dan een element zoals je dat bijvoorbeeld in schaken tegen kan komen, "als ik deze pion opoffer, dan kan ik zo met mijn paard zijn koningin slaan", in Bejeweled ziet dit er zo uit "als ik deze reeks van 3 groene stenen wegwerk, dan zal deze paarse steen drie plaatsen vallen zodat ik dan een reeks van 5 paarse stenen kan maken.". Hoe groter de kleuren reeks is die gemaakt wordt, hoe meer score je krijgt

en je krijgt dan ook nog eens speciale bonus steen waarmee ook weer meer punten te verdienen zijn. De paar verschillende bonus stenen geven de game een veel uitdagender karakter, dit maakt de game ook na de 20ste keer spelen nog steeds leuk.

Free spin
Naast de “easy to learn, hard to master” concept van Bejeweled, is er ook een element in de game waarbij je elke dag gratis munten kan winnen. Je krijgt één keer per dag de mogelijkheid om, gratis, een zwaai te geven aan een rad om zo tussen de 500 en 1 miljoen munten te winnen. Met deze in-game currency kun je bonus stenen kopen, deze stenen geven je een voordeel tijdens het spelen. En er valt dan ook nog bonus munten te verkrijgen als je meerdere dagen achter elkaar een draai aan de rad geeft. Dit zorgt ervoor dat men hun best gaan doen om elke dag in te loggen op facebook om zo deze gratis munten niet mis te lopen.

Social
Naast de al verslavende gameplay van Bejeweled introduceert de Blitz variant op Facebook nog een element van verslaving, namelijk de high-score lijst. In deze lijst kun je de scores zien van jouw Facebook vrienden die Bejeweled Blitz ook spelen. Deze lijst wordt elke week gewist zodat je elke week weer een nieuwe kans hebt om de beste score neer te zetten van jouw vrienden. Als vrienden waar je veel contact mee hebt, in het echt of op Facebook, dan werkt deze high-score lijst uitermate frustrerend. Het is namelijk ook mogelijk om je score op je publieke Facebook wall te posten, dit heeft als gevolg dat jij geconfronteerd wordt met de score van jouw vriend. Hierbij komt weer de menselijke psychologie bij kijken, mensen willen van nature beter zijn dan andere. Dus wat doet men bij het zien van de high-score van een vriend? Ze starten de game op en gaan bezig met het verbeteren van deze score. Het feit dat de high-score lijst elke week gewist wordt betekend dat er elk week weer een cyclus is van vrienden die elkaars scores proberen te verbeteren. Zo blijven mensen de game spelen ook al hebben ze er eigenlijk al genoeg van gehad. Deze “sociale druk” zien we steeds vaker terug nu dat social gaming groot is geworden door Facebook.

A-2.2.4 Tijdelijke achievements
Wanneer de speler enkele goede combinaties achter elkaar maakt dan krijgt hij feedback hiervan in de vorm van sound bites en onscreen teksten als “Good”, “Awsome” en “Excellent”. Dit worden ook wel tijdelijke achievements genoemd, nadat de notificatie weg is dan is dit nergens meer terug te vinden. Deze tijdelijke achievements verhoogt de intrinsieke motivatie van de speler en staat niet in de weg van de vastberadenheid van de speler.

A-2.2.5 Tijd
Bejeweled Blitz caterd voor het leven van de casual gamer. Je speelt het in korte sessies, elke ronde duurt maar 1 minuut, mensen spelen vaak zo’n 5 potjes achter elkaar. Casual gamers willen geen lange periodes achter elkaar aan het gamen zijn. Bejeweled is een perfect voorbeeld van een game die even tussen door gespeeld wordt op kantoor. Het is ook te spelen op smartphones, wat dus betekend dat men snel een paar potjes Bejeweled spelen wanneer ze ergens op moeten wachten, zoals het wachten op de bus. De “sweetspot” voor casual games is een game die in intervallen van 5 tot 15 minuten gespeeld kan worden, dit is precies wat Bejeweled aanbied.

A-2.2.6 Paying customers

Het is mogelijk om met Facebook Credits in Bejeweled Blitz in-game goud te kopen waarmee je power-ups kan kopen. Deze power-ups selecteer je voordat de game begint waar je er maximaal drie van kan gebruiken per ronde. Je hebt de keuze uit vijf verschillende power-ups.

- Mystery Gem – Zet een willekeurige speciale steen op het speelveld
- Detonator – Explodeert alle speciale stenen op het speelveld
- Scrambler – Wisselt alle stenen op het speelveld
- +5 Seconds – Voeg 5 seconde toe aan de duur van een spel
- Bonus Multiplier – Plaatst een multiplier op het speelveld

Deze power-ups geeft de speler een behoorlijk grote voordeel tijdens het spelen van de game, vooral als er drie tegelijkertijd geactiveerd zijn. Je hebt, zoals al gezegd, een in-game currency (coins) waarmee je deze power-ups moet kopen. Elke keer als je speelt krijg een kleine hoeveelheid coins als beloning. Je zou alleen ongeveer tien games moeten spelen voordat je genoeg coins hebt om een power-up te kopen. Je moet dus zorgvuldig omgaan met je in-game geld wil je op een goede manier gebruik maken van deze power-ups. Of, je zou ook coins kunnen kopen door middel van Facebook credits, wat echt geld kost. Voor 7 euro heb je een miljoen coins, een power-up kost gemiddeld 6000 coins. Voor 7 euro kan je er dus voor zorgen dat je voor een lange periode genoeg coins hebt om altijd drie power-ups geactiveerd hebben, in andere woorden, je hebt een lange periode waar je een voordeel hebt over de mensen die geen coins kopen. Het gevolg hiervan is dat niet betalende spelers zich benadeeld voelen.

A-2.2.7 Samenvattend

De goede elementen van Bejeweled Blitz zijn als volgt:

- Het heeft geen instructies nodig.
- De simplistische gameplay zorgt voor positieve feedback.
- Het is erg makkelijk te leren, maar erg lastig om goed in te worden.
- Er is elke dag gratis in-game geld winnen, waardoor mensen elke dag terug blijven komen.
- Het kunnen zien van de scores van de Facebook vrienden betekend dat er een competitief element inzit waardoor men meer gaat spelen.
- Bejeweled Blitz wordt in korte speelsessies gespeeld, dit sluit goed aan in het leven van de casual gamer.

De enigszins slechte elementen van Bejeweled Blitz zijn als volgt:

- De simplistische gameplay zorgt ervoor dat de speler nooit afgestraft wordt, waardoor de game geen aantrekkingskracht heeft onder de hardcore gamers
- Betalende klanten hebben een groot voordeel over niet betalende klanten

Bijlage B: Playtests

In dit onderdeel van de bijlage zijn enkelen resultaten van de playtest te vinden die aan de hand van het onderzoek naar playability in casual games zijn uitgevoerd. Deze resultaten worden voorgelegd aan de team-lead games development en de developer verantwoordelijk voor het maken van de game. De resultaten zijn puur bedoeld als advies en om eventueel een discussie te starten over de helderheid van de regels, de moeilijkheid en de progressie van de games, in andere worden ik hoop hiermee meer aandacht te creëren voor *fun* binnen de games.

B-1 Methode

Uit het vooronderzoek naar Casual Gaming is voort gekomen dat een paar elementen in het specifiek erg belangrijk zijn. De grootste kwesties zijn de introductie van de gameplay/regels, de moeilijkheid, en de progressie. Deze punten hebben veel met elkaar te maken, maar zijn wel verschillende elementen die op verschillende manieren getoetst kunnen worden.

Eerst is er naar de XML configuraties gekeken, daarnaast zijn de games door mijzelf gespeeld om zelf een idee te krijgen waar eventuele fouten kunnen liggen. Vervolgens is de game voorgelegd aan drie verschillende personen, twee hiervan vallen onder de doelgroep. De eerste stap was de testers de game vrij laten spelen, hierbij zijn er notities bijgehouden van hun voortgang. Wanneer de playtesters eenmaal paar keer game-over zijn gegaan en er geen vooruitgang meer geboekt werd zijn er verschillende vragen gesteld om hun mening te weten te komen over bepaalde elementen van de game. Een voorbeeld is level 3 en 4 opnieuw laten spelen en vragen wat zij vinden van het verschil in moeilijkheid tussen de twee level.

Aan de hand van de playtest resultaten en mijn eigen bevinden heeft elk Braintrainer per element een cijfer toegewezen gekregen op een schaal van 1 t/m 10, waarbij 1 heel slecht is en 10 heel goed. Hieronder wordt besproken waar er per element precies op gelet is.

> **Note:** *Alle configuraties van de games zijn gehaald uit de XML's van mei. Deze kunnen in de tussentijd veranderd zijn.*

B-1.1 Helderheid van de regels

Tijdens het leren van de gameplay/regels bepaald de gebruiker direct of hij de game wel of niet leuk gaat vinden. Hoe langer het duurt om de regels te leren, hoe groter de kans is dat een casual gamer de game weglegt en niet verder gaat spelen. Daarom wordt er gekeken of de uitleg die gegeven wordt in het instructiescherm duidelijk en helder is. Zijn de regels van de game meteen duidelijk na de eerste keer doornemen van de instructies? Wordt er overbodige informatie gegeven? Of zijn de instructies te kort door de bocht?

Er wordt niet alleen naar de instructie scherm gekeken, maar ook naar hoe makkelijk de regels/gameplay te begrijpen is zonder de instructies te lezen. Dit wordt bestudeerd omdat de meeste mensen de instructies overslaan en direct beginnen te spelen. Als een game goed is, dan zouden de regels voor een groot deel duidelijk moeten worden tijdens het spelen hiervan. Wanneer het niet duidelijk wordt wat het doel van het spel is dan zal je merken dat een groot aantal spelers niet bereid zijn de game een tweede kans te geven en stoppen zij met het spelen hiervan.

Op de schaal van 1 t/m 10 betekend dit dat een 1 wordt gegeven wanneer de regels helemaal niet duidelijk zijn. Niet door het spelen van de game en ook niet door het meerdere malen doorlezen van de instructies. Een 10 wordt gegeven wanneer de regels helemaal compleet zijn en de game de speler perfect de regels spelenderwijs weet uit te leggen.

B-1.2 Moeilijkheid

Één van de "regels" van casual game design luid als volgt: Als jij, de designer, de game niet kan halen / uitspelen dan is het veel te moeilijk. Je moet de speler uitdagen, maar je moet het niet zo moeilijk maken dat de speler gefrustreerd raakt omdat hij het niet kan halen. Het is een "balancing act". Een ideaal situatie is dat de moeilijkheid wordt aangepast aan de speelstijl van de speler, in andere worden een dynamisch moeilijkheidsgraad. Sommige games doen dit ook, de game wordt steeds lastiger zolang de speler succesvol is, zodra de speler af gaat dan wordt de moeilijkheid automatisch iets makkelijker. Zo zorg je ervoor dat de speler altijd uitgedaagd wordt zonder dat hij vast komt te zitten omdat hij niet voorbij een bepaald punt komt. Dit is helaas niet het geval voor Yezzle, daarom moet er nauwkeurig op de moeilijkheid van de games gelet worden.

De braintrainers zijn allemaal geheel te testen op absolute moeilijkheid. Hiermee wordt bedoeld dat alle braintrainers eigenschappen bevat die direct te vertalen zijn naar moeilijkheid. Neem bijvoorbeeld de maximale tijd die je krijgt om een level te halen. Wanneer je in level 1 60 seconde krijgt om de level te volbrengen en in level 2 maar 30 seconde voor hetzelfde krijgt dan is dit een absolute toename van twee in de moeilijkheidgraad van level 2. Veel games hebben geen absolute moeilijkheid maar een relatief moeilijkheid. Wanneer je voor level 1 30 seconde krijgt en in level 2 voor dezelfde handeling 30 seconde krijgt, *maar* je krijgt een "wapen" die hetzelfde level makkelijker maakt, dan spreken we over een relatieve moeilijkheid. De braintrainers geven de spelers geen extra "krachten" naarmate een game verloopt, dus spreken we over een absolute moeilijkheidsgraad wat elke braintrainers erg toetsbaar maakt.

Op de schaal van 1 t/m 10 betekend dit dat een 1 wordt gegeven de game echt veel te moeilijk is en gewoonweg niet haalbaar, een 10 wordt gegeven wanneer de game perfect gebalanceerd is en precies de juiste hoeveelheid uitdaging biedt.

B-1.3 Progressie

In games is het belangrijk dat de moeilijkheid geleidelijk opgevoerd wordt. Het zou makkelijk moeten beginnen zodat de spelers de game rustig kunnen uitproberen. Tijdens de eerste level (bij de complexere games de eerste twee levels) zou het eigenlijk zo moeten zijn dat de speler moeite zou moeten doen om "game over" te gaan. Ze moeten eerst de ruimte krijgen om de game mechanic uit te proberen en vervolgens moet de moeilijkheid steeds opgeschroefd worden.

De braintrainers van Yezzle hebben een vaste hoeveelheid selecteerbare levels waarbij de level opnieuw gespeeld kan worden wanneer de speler af gaat. Na de laatst selecteerbare level wordt het een uitputtingsslag. Dat betekent dat de game geleidelijk steeds lastiger wordt tot de speler "game over" gaat waarna de speler niet meer de optie heeft om de level opnieuw te proberen.

Er wordt dus gekeken op wat voor manier de games van level 1 tot en met de laatst selecteerbare level moeilijker worden, dit is mogelijk door de absolute moeilijkheidsgraden per level. Een 1 wordt gegeven wanneer een game totaal geen logische opbouw heeft. Bijvoorbeeld als level 4 makkelijker zou zijn dan level 3. Een 10 wordt gegeven wanneer de game op een goede en logische manier geleidelijk steeds iets lastiger wordt. Daarbij wordt ook gekeken hoe de progressie is bij de levels na het laatst te selecteren level.

B-2 Boodschappenband

Boodschappenband is een game waar het puur om het geheugen van de speler draait. Verschillende producten komen langs op een band en het is dan aan de speler om deze te onthouden en in de juiste volgorde uit te kiezen uit een lijst van andere producten. Daarbij komen er per level steeds meer producten bij en steeds meer variaties van deze producten, zo wordt het steeds lastiger om deze objecten te onthouden. Hieronder vind je de configuratie van alle levels zoals die waren op 30-05-2011:

Boodschappenband	Aantal rondes	Time	Time penalty	# Objecten mogelijk / # Objecten die langs komen	Hoeveel variaties per object	Snelheid van de objecten	Viewing tijd	In sequence ja / nee
Level 1	5	50 sec	5 sec	3 / 3	1	6	3	Ja
Level 2	10	80 sec	5 sec	3 / 3	1	7	3	Ja
Level 3	10	80 sec	7 sec	3 / 4	1	7	3	Ja
Level 4	10	80 sec	10 sec	7 / 4	2	8	3	Ja
Level 5	10	70 sec	12 sec	7 / 4	3	8	3	Ja
Level 6	10	70 sec	10 sec	13 / 3	3	8	3	Nee
Level 7	10	70 sec	12 sec	13 / 4	3	8	3	Nee
Level 8	10	70 sec	12 sec	13 / 5	3	8	2	Nee

Helderheid van de regels

- Boodschappenband geeft de gebruiker geen hints in de eerste levels over wat de bedoeling is van de game. Dit betekent dat het voor iemand die de game nog nooit gespeeld heeft *en* de regels niet gelezen heeft moeite met de game zou kunnen hebben.
    - Er zou in level 1 een in-game een tekst kunnen komen waarin staat wat de core bedoeling van de game is (producten onthouden en in de aangegeven volgorde reproduceren).

- De core game mechanic wordt direct duidelijk op pagina twee van de instructie scherm: “Op de boodschappenband zie je een rij producten die je moet onthouden in de volgorde waarin ze voorbij komen”. Met deze informatie is het mogelijk voor de gebruiker om met een beetje trial en error de game succesvol te spelen.

- Op pagina drie van de instructies staat dat je de volgorde moet aangeven van links naar recht waarin de voorwerpen voorbij zijn gekomen. Dit is niet consistent met de game mechanic, in de game moet je namelijk van rechts naar links de objecten onthouden.

- Op pagina 5 van de instructies staat dat de tijdlimiet steeds korter wordt wanneer je een product aanklikt dat niet voorbij is gekomen. Dit kan enigszins verwarrend overkomen omdat de tijdslimiet niet korter wordt, maar de resterende tijd wordt verminderd. En je krijgt niet alleen een tijdstraf voor het aanklikken van een product dat niet voorbij is gekomen maar ook wanneer je producten in de verkeerde volgorde aanklikt.

- In de level omschrijving van level 7 staat dat het identiek is aan level 6 maar dan met 4 voorwerpen i.p.v 3. Dit is niet correct, de time penalty is in level 6 namelijk 10 seconde en in level 7 is dit 12 seconde. Deze fout staat ook in de level omschrijving wanneer je level 7 selecteert en op spelen drukt.

Op een paar foutjes na zijn de instructies direct en helder. Het is een relatief makkelijk spel om te begrijpen, hierdoor krijgt boodschappenband een 7.5 voor de helderheid van de regels.

Moeilijkheid / progressie
Er wordt per level gekeken naar hoe moeilijk een level is waarbij er ook direct gekeken wordt of dit klopt in verhouding tot de voorgaande level, er wordt in andere woorden naar de moeilijkheid en progressie gekeken.

- Level 1: Erg makkelijk, totaal geen problemen met uitspelen hiervan.
- Level 2: Erg makkelijk, totaal geen problemen met uitspelen hiervan.
- Level 3: Makkelijk, het is wel complexer geworden door een vierde product.
- Level 4: Erg moeilijk, heeft 5 pogingen gekost om de level te halen. De level is van 3 verschillende producten naar 7 producten gegaan en daarbij ook per product twee variaties. Dat is dus een 466% toename in hoeveelheid mogelijkheden.
    - Suggestie: Laat de hoeveelheid mogelijke producten op drie met twee variaties per product.
- Level 5: Te moeilijk, bijna niet te halen. In level 4 heb je 14 mogelijkheden waarvan je er vier langs komen, wat al erg moeilijk was, level 5 heeft 21 mogelijke producten . Dit zorgt ervoor dat level 5 heel moeilijk te halen is.
    - Suggestie: Verlaag de time penalty naar 10 seconde en verlaag de hoeveelheid variaties naar 2 (i.p.v 3)
- Level 6: Moeilijk maar na een paar pogingen wel te halen.
    - Suggestie: Met een oog op level 7 is het verstandig de hoeveelheid producten te veranderen naar 10 in plaats van 13.
- Level 7: Zo goed als niet haalbaar. Het feit dat je er vier moet onthouden met 39 mogelijke variaties maakt het te moeilijk.
    - Suggestie: Laat maar 3 objecten voorbij komen, in plaats van 4.
- Level 8: Onmogelijk. Vijf objecten met 39 combinaties is gewoonweg te moeilijk.
    - Suggestie: Neem de exacte waarde over van level 7. Dat is dus: 13 objecten waarvan er 4 langs komen, met drie variaties.

Als al deze suggesties gevolgd worden dat ziet dit er als volgt uit:

Boodschap penband	Aantal rondes	Time	Time penalty	# Objecten mogelijk / # Objecten die langs komen	Hoeveel variaties per object	Snelheid van de objecten	Viewing tijd	In sequence ja / nee
Level 1	5	50 sec	5 sec	3 / 3	1	6	3	Ja
Level 2	10	80 sec	5 sec	3 / 3	1	7	3	Ja
Level 3	10	80 sec	7 sec	3 / 4	1	7	3	Ja
Level 4	10	80 sec	10 sec	3 / 4	2	8	3	Ja
Level 5	10	70 sec	10 sec	7 / 4	2	8	3	Ja
Level 6	10	70 sec	10 sec	10 / 3	3	8	3	Nee
Level 7	10	70 sec	12 sec	13 / 3	3	8	3	Nee
Level 8	10	70 sec	12 sec	13 / 4	3	8	3	Nee

(elementen in rood zijn veranderd)

Het gaat voornamelijk op één plek goed mis in Boodschappenband. De overgang van level 3 naar level 4 heeft een veelte groot verschil in moeilijkheid. De game progressie in de latere levels zijn zo te zien geconfigureerd in relatie tot level 4, waardoor de volgende levels te snel te moeilijk zijn. Om zeker te zijn van het feit dat de progressie niet klopt is de overgang van level 3 naar voor door meerdere mensen geplaytest.

Cijfer: Voor de progressie geef ik deze game een 4 en voor de moeilijkheid een 3.

Conclusie

Boodschappenband heeft de volgende cijfers gekregen:

Boodschappenband	Cijfer	Commentaar
Helderheid van de regels	7.5	Op een paar foutjes na zijn de regels duidelijk
Moeilijkheid	3	Sommige levels zijn voor de meeste mensen niet haalbaar
Progressie	4	Te groot verschil in moeilijkheid van level 3 naar 4
Gemiddeld	4.83	

De game krijgt een 4.83 als gemiddelde cijfer. Dit komt voornamelijk door het feit dat sommige levels echt te moeilijk zijn en doordat er een te grote verschil in moeilijkheid zit tussen level 3 en 4.

Één van de elementen die de game een extra dimensie van moeilijkheid geeft is het feit dat, op level 1 na, alle levels 10 rondes hebben. Elk ronde duurt gemiddeld 15 seconden, wat betekend dat, wanneer je van level 1 t/m level 8 zou spelen, je gemiddeld 18 minuten bezig bent met boodschappenband. 18 minuten pure concentratie is te lang om de game nog als leuk te ervaren. Het is opgevallen dat na zo'n 5 minuten spelen de meeste gebruikers al veel minder goed gaan presteren en de game zelf als frustrerend gaan ervaren. Hierdoor zou er over nagedacht kunnen worden om de hoeveelheid rondes per level ook aan te passen. Het gaat nu van level 1 waar er 5 rondes zijn naar level 2 t/m 8 waar er 10 rondes zijn. Er zou gespeeld kunnen worden met een progressie in de rondes, waarbij je in level 1 drie rondes zou hebben en dan per level één ronde toevoegen tot de 10 rondes bereikt zijn. Dit kan de game minder frustrerend maken en dus leuker, want op het moment is de "fun" factor van boodschappenband erg laag.

De elementen die veranderd zouden kunnen worden om de game leuker te maken zijn dus als volgt:

- Geef in level 1 in-game instructie aan de speler zodat hij de core game-mechanic kan begrijpen zonder de instructies te hoeven lezen.
- Pas de bovengenoemde fouten in de instructies aan.
- Configureer de game opnieuw om er een beter progressie in te krijgen.
- Configureer de game op zo een manier dat de latere levels iets makkelijk worden.
- De hoeveelheid rondes zou eventueel verlaagd kunnen worden.

B-3 Golven

Golven is een game die voornamelijk om behendigheid draait. Het is de bedoeling om de juiste knop op het juiste moment in te drukken zoals “Guitar Hero” en “Dance Dance Revolution”, waarbij de uitdaging zit in het feit dat het niet altijd voor de hand liggend is welke knop er gedrukt moet worden. In latere levels komen figuren naar beneden die gekoppeld zijn aan een pijltjes toets die op te zoeken is in een legenda.

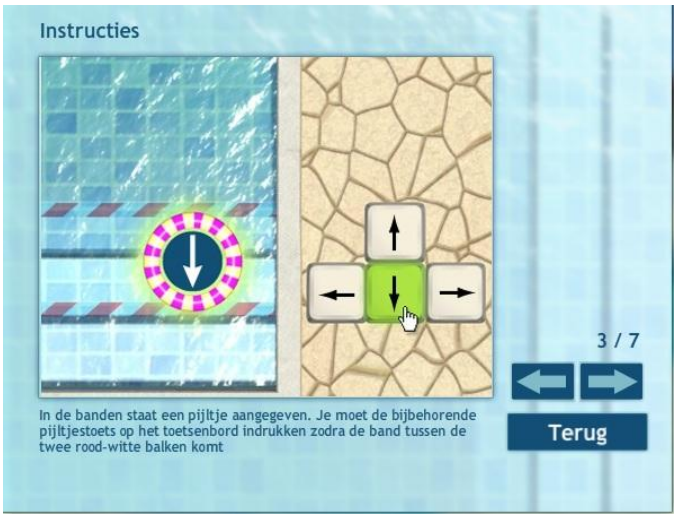
Golven is één van de weinige games die gebruik maakt van een meer “hardcore” game mechanic, het is dan ook een game waar gamers weinig problemen mee hebben, maar de doelgroep wel.

Helderheid van de regels

Een gamer die het concept van “Guitar Hero” of “Dance Dance Revolution” kent, zal weinig moeite hebben met het spelen van Golven. Uit de User acceptance test (UAT) is gebleken dat de doelgroep van Yezzle het concept niet kent en dus zullen zij moeite hebben met het spelen van Golven. Hierdoor staat er nu een in-game beschrijving van de core mechanic in level 1 en 2, dit maakt de core mechanic al iets duidelijker voor de gebruiker.

Er bevinden zich een paar minor issues in de instructies namelijk:

- Pagina 3 van de instructies maakt niet het niet duidelijk genoeg voor de gebruiker wat, waar en hoe de pijltjes toetsen gebruikt moeten worden. Dit is gebleken door meerdere playtests.
    - Suggestie: Laat in de screenshot op pagina 3 ook een plaatje zien van een toetsenbord met daarop de pijltjes toetsen gehighlight.



- Pagina 4 zou beter moeten aangeven dat de objecten met de bij behorende pijl in elk level weer anders geconfigureerd zijn. Zo weet de gebruiker dat er in elk level weer opnieuw gekeken moet worden naar de legenda.
- Het kan makkelijk voorkomen dat de gebruiker in een level af gaat omdat hij de regels van de game niet snapt, dit is het moment dat de meeste gebruikers de instructies willen inzien maar hier is geen knop voor in het game-over screen.
    - Suggestie: Implementeer een knop voor de instructies in de game-over scherm.

Verder zijn de instructies helder en duidelijk genoeg, om na het aandachtig lezen hiervan, de game te kunnen begrijpen. Het feit dat in level 1 en 2 er ook in-game een korte uitleg gegeven wordt is geen overbodige luxe. Een zorg voor Golven is wel, zoals al gezegd is, dat het een game mechanic is die niet snel te beheersen is door de doelgroep. Zoals is hoofdstuk 3.3.1 beschreven staat en te zien is in “Bejeweled” en “Solitaire”, is een streven dat je de games zo ontwerpt zodat je gebruik maakt van al bestaande kennis van de speler, dit is niet het geval in Golven. Dit kan een probleem zijn als meerdere games game mechanics hadden die niet aansluiten op de doelgroep, maar Golven is de enige game die gebruik maakt van een “hardcore” mechanic. Deze “core” game mechanic is dus in het geval van Golven niet een groot probleem, het zorgt juist voor variatie. Als we terug kijken naar de player archetypes dan zien we dat er casual gamers zijn die waarschijnlijk ook onder de archetype “achiever”zullen vallen, dit is het type speler die Golven wel leuk zal vinden. Het is dus goed dat er hier en daar games zijn die niet direct te begrijpen zijn voor de “echte” casual doelgroep, zolang deze games wel in de minderheid blijven.

Er is weinig mis met de helderheid van de regels maar omdat het berust op een mechanic die de gebruiker niet kent krijgt Golven een 7 en niet een 8 of 9.

Moeilijkheid en Progressie

De game is op het moment als volgt geconfigureerd:

Golven	Aantal "levens"	Aantal objecten	Aantal symbolische objecten	Snelheid van de objecten	Delay tussen objecten (min / max) (ms)
Level 1	10	5	0	70	4000 / 5000 ms
Level 2	8	20	0	100	2000 / 3000 ms
Level 3	8	20	1	110	1500 / 2500 ms
Level 4	6	25	2	120	1500 / 2000 ms
Level 5	6	25	3	130	1200 / 1800 ms
Level 6	5	25	3	140	1000 / 1800 ms
Level 7	4	25	3	150	800 / 1600 ms
Level 8	4	25	4	160	700 / 1400 ms
Level 9	3	25	4	170	500 / 1200 ms
Level 10	3	30	4	170	300 / 1000 ms

Golven is door vier mensen geplaytest, dit zijn de resultaten van deze playtests:

- Level 1: Na het lezen van de instructies is het voorgekomen dat 1 gebruiker de game mechanic niet in één keer door had. Zij drukt op het juist moment op een verkeerd pijltje. Na dit twee keer fout gedaan te hebben snapte zij de mechanic en ging het vervolgens zonder problemen.

- Level 2: Hier waren geen problemen mee, het gaat langzaam maar het geeft de gebruiker de mogelijkheid om te wennen aan de game mechanic.

- Level 3: Wanneer je de regels niet gelezen hebt dan zal level 3 een struikelblok zijn omdat je dan niet weet wat je moet doen met de nieuwe figuren. Je zou de instructies erg aandachtig moeten lezen om er achter te komen wat je precies moet doen met de figuren die geen richting aangeven. De meeste gebruikers zijn niet bereid dit te doen, zij zullen of direct beginnen, of de instructies even snel en niet aandachtig doorlezen.
    - Suggestie: Er zou gedacht kunnen worden aan een glow (of iets dergelijks) om de figuur in de legenda zodra er een figuur zonder pijl in beeld is.

- Level 4: Ervan uitgaand dat de gebruiker de mechanic snapt dan is level 4 geen probleem om te halen.

- Level 5: Geen problemen met het uitspelen van level 5

- Level 6: Levert weinig problemen op, een enkeling zal hier een fout of twee maken.

- Level 7: Een enkeling zal hier een fout of twee maken. Vanaf level 7 krijgen de meeste spelers het gevoel van paniek en haast.

- Level 8: Dit level zal voor sommige problemen leveren. Het gevoel van paniek en haast is hier hoog voor een beginnend gebruiker.
- Level 9: Hier zullen de meeste gebruikers voor het eerst echt problemen hebben met de moeilijkheid (in tegenstelling tot problemen hebben met het leren van de regels). Een nieuwe gebruiker zal hier wat tijd in moeten investeren voordat hij hier voorbij komt.
- Level 10: Hier is het gevoel van haast en paniek op zijn grootst. De objecten komen erg snel naar beneneden. Level 10 is alleen te halen voor mensen die de game leuk vinden en bereidt zijn er tijd in te steken om er beter in te worden.
- Level 11+: Na level 10 begint de uitputtingsslag, hierbij gaat de game steeds sneller maar verschilt verder niet van level 10. De meeste gebruikers zullen deze fase niet bereiken. De gebruiker die wel zo ver komen zullen merken dat zodra je level 10 kan halen zonder problemen dan zullen de levels daarna ook geen problemen opleveren.
    - Suggestie: Er zou over nagedacht kunnen worden om na level 10 steeds meer objecten naar beneden te kunnen laten vallen. Zo blijft de game uitdagend.

De moeilijkheid en progressie van Golven is uitstekend. De game begint erg langzaam en vergevinggezind zodat de speler tijd heeft om de mechanic te leren en wordt vervolgens met contante stappen moeilijker totdat het alleen nog maar te halen is door spelers die bereid zijn het vaker te spelen. Het begint ook zonder een gevoel van paniek of haast maar dit wordt steeds groter tot je het gevoel krijgt dat het bijna onmogelijk is in level 10. Het gevoel van paniek is niet altijd een goed element om in een casual game te hebben, maar zolang dit gevoel alleen in level 10 erg prominent aanwezig is is dit niet zo een groot probleem. Sommige type spelers zullen hier juist van genieten.

Alles bij elkaar krijgt Golven een 6 voor de moeilijkheid en een 9 voor de progressie. De moeilijkheid scoort wat aan de lage kant omdat het niet voor elk type gebruiker een spel is die snel door te krijgen is, gelukkig is de game zo geconfigureerd dat het een geleidelijk moeilijkheid curve heeft.

Conclusie

Golven krijgt de volgende cijfers:

Golven	Cijfer	Commentaar
Helderheid van de regels	7	Regels zijn alleen duidelijk na het lezen van de instructies
Moeilijkheid	6	De game berust op een mechanic die de doelgroep niet kent
Progressie	9	Er zit een goed opbouwende structuur in
Gemiddeld	7.33	

Golven is een goed en logisch functionerende spel. Het grootste nadeel dat de game heeft is dat het niet berust op een casual game mechanic. Dit betekent dat er bepaalde elementen heel overduidelijk naar voren moeten komen. Tijdens één van de playtests kwam het bijvoorbeeld voor dat een gebruiker niet wist welke toetsen precies de pijltjes toetsen waren. Dit bevestigd de regel zoals deze omschreven staat in het hoofdstuk *"10 dingen die je niet moet doen in Casual Games"*. Ook al zit de functionaliteit erin om met de linkermuisknop te spelen wil je dit alsnog vermijden, de game is namelijk bedoeld om met het keyboard te spelen. Daarom moet er een groter focus liggen op het duidelijk maken van de besturing.

Naast de besturing moet er ook meer aandacht besteedt worden aan het duidelijk maken van de figuren die geen pijltjes hebben. Dit zou je bijvoorbeeld kunnen doen om in level 3 (waar deze voor het eerst voorkomen) de legenda te highlighten wanneer er zo een figuur naar beneden valt.

Het aanpakken van deze twee elementen zal ervoor zorgen dat de game een groter bereik krijgt.

Bijlage C: Test Suite Braintrainers

Hier onder is een generieke test suite die te gebruiken is voor alle nieuwe braintrainers (mits het design van deze games hetzelfde blijft). Deze test suite controleert niet op hoe leuk de game is, maar is puur bedoeld om alle functionaliteit van een nieuwe braintrainer te controleren.

Steps for testing a Braintrainer	***Pass***	***Fail***	***Comment***
Game:			
Before you start, clear the cache of your browser			
1. Go to test.yezzle.keesinggames.net			
2. Make sure you are logged out			
3. Navigate to the page were the Braintrainer in question is located			
> Is the correct box art displayed?			
> Is the title spelt correctly?			
> Is the correct game index shown underneath the title?			
> Does the correct mouse-over get displayed when mousing over the game index?			
4. Open the game that needs testing			
> Is the game name spelt correctly in the breadcrumb navigation?			
> Are the in-game top menu items mute, help, Opties and close?			
> Have the top menu items got a contrasting color to the background color making them easy to see?			
> Is the correct background music running?			
5. Click the mute button			
> Is all sound muted? (background, effects, click sounds, in-game etc.)			
6. Click the question mark in the in-game top menu			
> Does the instructions screen open?			
7. Click “sluiten” to go back to the main menu			
8. Click “Opties” in the in-game top menu			
> Does the Options overlay appear?			
> Is the background blurred?			
> Is the text of the different difficulty setting correct?			
> Does changing the quality affect the graphics?			
9. Click on “Doorgaan”			
> Are you taken back to your previous page? (in this case that should be the main menu)			

10. Click on the cross (close button) in the in-game top navigation			
> Are you taken back to “alle-spellen” page?			
11. **Restart the game**			
> Has the side-panel only got the tab “info”?			
> Are the correct competencies displayed in the side-panel?			
> Do the correct mouse-overs get displayed when mousing over the gameindex			
> Is the correct game description displayed in the side-panel?			
> Is the game description in the side-panel free of spelling and grammar mistakes?			
12. **Click on “Instructies”**			
> Do the next / previous buttons respectively take you to the next and previous page of the instructions?			
> When you’re on the last instruction page and next page is clicked. Does it go to page 1?			
> When you’re on the first instruction page and previous page is clicked. Does it go to the last page?			
> Do the instructions match the gameplay?			
> Do the screenshots match the explained feature?			
> Are the instructions free of spelling / grammar mistakes?			
13. Click on “Sluiten”			
14. Click on “Spelen”			
> Due to being logged out, are the correct levels disabled?			
15. Start all available levels			
> Do all available levels start correctly?			
16. Play and complete the highest logged out available level			
> When you click on “Verder” (next level), do you get the pop-up asking you to register to play the rest of the levels?			
> When you click “Probeer Yezzle gratis” in this pop-up, do you get redirected to the registration page?			
> When you click “Sluit aanbieding en ga naar het hoofdmenu” in this pop-up, do you get redirected to the games main menu?			
17. Log in with an active account, open the game and click on “Spelen”			
> Are all levels available to be played?			
> Is the MindIndex tab added to the side-panel?			

18. Click on all levels in the level select screen			
> Do all level descriptions match the gameplay? (check against the game's config xml)			
> Are all level descriptions free of spelling and grammar mistakes?			
19. Select a level and click on "Spelen" (do this for all levels)			
> Do all level start descriptions match the gameplay? (check against the game's config xml)			
> Are all start level descriptions free of spelling and grammar mistakes?			
20. Select level 1, click "Spelen" and then click "Verder"			
> If the gameplay requires you to click with your mouse, then does the mouse cursor change on the clickable elements?			
> Is the clickable element fully clickable as to be expected?			
> If the gameplay requires you to press a keyboard button, then does the keyboard input trigger the correct response?			
> Does the game correctly handle when two or more keys are pressed at the same time?			
> With the sound on, do all the sound effects play at the right time?			
21. Click the "Options" button on the top menu			
> Is the game paused?			
> Is the difficulty setting disabled (grayed out)?			
22. (For time based games) Restart a level			
> Does it state "Resterende Tijd" above the remaining time graphic			
23. Let the time run out without completing a single round			
> Does the game over screen appear after the time runs out?			
> Do you receive 0 points?			
> Do you receive 0 MindIndex points?			
24. (For live based games) Start a level			
> Does it state "Levens" above the remaining lives graphic?			
25. (For all types of games) Restart the level and force the game to go game over by entering invalid input in quick succession			
> Does the game over screen correctly get displayed?			
26. Click on "Titelscherm"			
> Do you get directed to the game's main menu?			

27. Click "Spelen", select level 1 and start it and successfully complete the level			
> Did the side panel switch to the MindIndex tab?			
> Did you receive MindIndex points?			
> In the level recap screen do you get bonus points for the remaining lives or remaining time (depending on game typ)?			
> In the case of time based does it state "Je hebt x bonuspunten verdiend voor je overgebleven tijd."?			
> In the case of round based games does it state "Je hebt x bonuspunten verdiend voor je overgebleven levens"?			
28. Make sure the MindIndex tab is open in side-panel			
> Have all achievements ("spaazegels") got the correct graphics?			
> Have all the achievement got correct names and descriptions?			
> Are all of the achievement names and descriptions free of spelling and grammar mistakes?			
> Are all of the achievements possible to complete when you take the gameplay mechanics into consideration?			
29. Complete all achievements			
> When an achievement is triggered does an animation play of the achievement in question?			
> Does only one achievement animation get shown when 2 or more achievements are triggered at the same time? (not possible for all games)			
30. Reload the page and open the MindIndex tab in the side-panel			
> Are all previously unlocked achievements still marked as unlocked? (graphics are colored in, instead of grayed out)			
31. Take note of your MindIndex score and navigate to the page "Mijn Voortgang"			
> Is the correct MindIndex score displayed on this page?			
32. Click the button "Spaarzegels", open the correct category for the game in question			
> Are all the unlocked achievements marked as unlocked? (graphics are colored in, instead of grayed out)			

Bijlage D – Afstudeer overeenkomst