Effectieve Constructie van Teamwork Games

Auteur: Ricardo Tangali
Datum:

Auteur: Ricardo Tangali
Datum:
Studienummer: 1275257
Universiteit: VU Amsterdam
Opleiding: Informatiekunde (Multimedia & Cultuur)
Begeleiding Getronics PinkRoccade: J. Gerbrandy, T. Thijssen
Begeleiding VU: A. Eliens, G. van der Veer.

Voorwoord

Dit afstudeerverslag vormt de afsluiting van mijn master studie Informatiekunde Multimedia & Cultuur aan de Vrije Universiteit van Amsterdam. Het afstudeeronderzoek is in opdracht van Getronics PinkRoccade uitgevoerd en in samenwerking met het Clima Futura project aan de VU.

Binnen de studie Informatiekunde Multimedia & Cultuur heb ik mij vooral gericht op multimedia technieken die waarde toevoegen aan computergames, film en webapplicaties. Computergames zijn van grote invloed geweest bij mijn studiekeuze. De computergame industrie is altijd één van de eersten die gebruik maakt van nieuwe computer technologie, waardoor deze ruimte geeft voor innovatie en creativiteit. Hierdoor valt vooral computergames binnen één van mijn specifieke interesses.

Dit onderzoek heeft mij meer inzicht gegeven op het gebied van game ontwikkeling. Vooral over herbruikbaarheid binnen game ontwikkeling heb ik veel kennis opgestoken. Daarnaast heb ik op technisch gebied veel bijgeleerd en dan heb ik het voornamelijk over het “Adobe Flex” framework, dat ik gebruikt heb voor een proof of concept.

Dit onderzoek biedt inzicht op het gebruik en ontwikkeling van high-level componenten, die helpen bij de bouw van (serious) games. Hierbij wordt voor het gekozen onderwerp, teamwork, gezocht naar mogelijkheden voor hergebruik. Het bevat adviespunten over herbruikbaarheid binnen een (serious) game project, die huidige en toekomstige projecten kunnen doen versnellen en stabiliteit en robuustheid kunnen bieden, en het omzetten van high-level concepten naar herbruikbare componenten.

Het afstudeeronderzoek is vooral bedoeld voor de opdrachtgever Getronics PinkRoccade, die overweegt om gebruik te maken van serious games als toegevoegde waarde voor haar producten. Tevens zal het verslag voor iedereen, die in (serious) game ontwikkeling geïnteresseerd is, inzicht en aandachtspunten bieden op het gebruik en ontwikkeling van herbruikbare componenten en tools binnen (serious) game development.

In eerste instantie ben ik veel dank verschuldigd aan dhr. J. Gerbrandy en dhr. T.Thijssen voor hun goede begeleiding, betrokkenheid en adviezen vanuit Getronics PinkRoccade. Ook dhr. A. Eliens en dhr. G. van der Veer wil ik bedanken voor hun begeleiding, betrokkenheid en adviezen vanuit de Vrije Universiteit van Amsterdam. De leden van het Clima Futura project bedank ik voor hun betrokkenheid en adviezen. Tenslotte wil ik mijn ouders en mijn broers bedanken voor hun hulp en steun.

Ricardo Tangali

Samenvatting

Inhoudsopgave

1 Inleiding

De computer game industrie vandaag de dag is enorm. Het aantal huishoudens dat over een game console (PC, Playstation, Xbox, Nintendo) beschikt, is in de afgelopen jaren sterk toegenomen. De steeds betere computertechnologie geeft computerspellen een blijvende aantrekkingskracht en heeft mede daardoor een groot aandeel in de entertainment industrie. Een publicatie van PriceWaterhouseCoopers [PWC] geeft weer dat de videogames markt in Nederland de komende jaren het snelst groeiende marktsegment binnen de entertainment- en mediasector is. Deze heeft in 2006 de verkoop van muziekdragers voorbij gestreefd. In 2008 blijft de videogames markt groot. Dit blijkt onder andere uit het wereldwijde succes van het spel GTA IV [1], dat op de dag van uitgave wereldwijd 3,6 miljoen keer is verkocht en in de eerste week een omzet van 500 miljoen dollar heeft opgeleverd. Dit is een officieel genoteerd wereldrecord [2].

Uit onderzoek naar de aantrekkingskracht en veelzijdigheid van computergames is gebleken dat computergames ook toegepast kunnen worden voor meer serieuze doeleinden. Hierbij kan gedacht worden aan educatie, reclame, training, werving en selectie.

Getronics PinkRoccade, de opdrachtgever van mijn afstudeeronderzoek, heeft interesse in serious games en doet daar onderzoek naar. Deze afstudeeropdracht levert een bijdrage aan dat onderzoek.

1.1 Serious Gaming

Purdy [Pur07] geeft de volgende definitie van een serious game:

"*Een serious game is een software applicatie die ontwikkeld is met game technologie en game design principes en heeft een primair doel dat anders is dan puur entertainment.*"

Het enige verschil tussen een serious game en een digitaal spel is het doel. Van elke serious game kan gezegd worden dat het een digitaal spel is. Het omgekeerde statement is niet geldig. Om deze reden zal voor het aanduiden van een serious game de termen serious game, spel en game gebruikt worden.

De sceptische visie op de serieuze toepassing van computergames bestaat bijvoorbeeld in het traditionele bedrijfsleven, waarbij computergames met kinderspelletjes worden geassocieerd. Het feit dat videogames het snelst groeiende marktsegment binnen de entertainment- en mediasector is, heeft bedrijven overgehaald om dit gebied te exploreren. Een van de meest voorkomende toepassing van serious games is terug te vinden in reclame op het Internet. De serious game wordt toegepast om de aandacht van de speler vast te houden, zodat een bedrijf de mogelijkheid heeft om meer punten van haar product te belichten, in de hoop dat de speler voor het product kiest. Naast reclame

[1] *http://www.rockstargames.com/IV/*
[2] *http://gamers.guinnessworldrecords.com/news/130508_GTA_IV_break_record.aspx*

noemen Michael en Chen [MC05] andere toepassingen voor serious games. Zij kunnen gebruikt worden voor bepaalde trainingen, zoals vliegtuig simulaties en militaire oefeningen en in de gezondheidszorg (b.v. chirurgie).

De overheid heeft het belang van serious games gezien en stimuleert de ontwikkeling en evolutie daarvan met opdrachten en prijsvragen [1]. Ook op Europees niveau hebben serious games erkenning gevonden [2].

1.2 Getronics PinkRoccade

Getronics PinkRoccade is expert op het gebied van workspace management services, applicatiebeheer en consultancy.

Sinds 23 oktober 2007 maakt GPR deel uit van KPN, de grootste aanbieder van telecommunicatiediensten in Nederland. Dit geeft het bedrijf meer financiële slagkracht, stabiliteit en innovatievermogen [3].

De afdeling Business Innovation is opgehangen onder Business Development en is onderdeel van BAS (Business Applications Services). Zoals eerder beschreven heeft GPR haar interesse onder ander gevestigd op serious games. Één van de mogelijke toepassingen, is om bestaande GPR producten een toegevoegde waarde te geven met serious games. Of serious games een reële toekomst heeft in dit ICT-bedrijf staat nog niet vast. Wellicht geeft dit onderzoeksrapport het management voldoende handvatten een business case te definiëren rondom serious gaming.

[1] *http://www.m-ict.nl/ (5e prijsvraag)*
[2] *http://www.seriousgameseurope.com/*
[3] *http://www.getronicspinkroccade.nl/*

2 Onderzoek

In dit hoofdstuk wordt de inhoud van het afstudeeronderzoek beschreven. Hierin wordt het probleem dat aanleiding is voor het onderzoek uiteengezet. Hierna volgt de onderzoeksvraag met de bijbehorende subvragen. Vervolgens wordt de aanpak van mijn onderzoek uitgelegd. Tenslotte wordt de opbouw van het verslag gegeven.

2.1 Probleemstelling

Binnen Getronics is al onderzocht wat serious games precies zijn, wat er onder wordt verstaan en waarvoor zij dienen. Getronics wil mogelijk haar producten met serious games een toegevoegde waarde geven. Hierbij is het van belang dat de productie van serious games zo efficiënt mogelijk verloopt.

Er bestaan meerdere manieren om tijd en geld te besparen bij het ontwikkelen van een spel. Enkele manieren zijn: bedrijfsreorganisatie[1], fusie, outsourcing en hergebruik. In mijn onderzoek richt ik mij op het laatste. Namelijk hergebruik.

Het gaat hier voornamelijk om hergebruik van software onderdelen. Naast software onderdelen worden ook modellen, architecturen, processen en (design) patronen hergebruikt. Dit idee is niet nieuw en wordt al binnen software projecten toegepast. Een verschil tussen *game development* en traditionele software productie is, dat voorafgaande het gehele ontwikkelingsproces een spelconcept wordt bedacht. In dit concept komen ideeën voort die niet uit te drukken zijn in reeds bestaande onderdelen, maar waarvan wel gewenst wordt dat deze in verschillende spellen kunnen worden toegepast. Dit komt omdat bij het bedenken van spellen gezocht wordt naar vernieuwende elementen. Deze elementen vereisen geheel of gedeeltelijk nieuwe software onderdelen, modellen, etc.

Tot één van deze ideeën behoort het concept van teamwork. Teamwork is als onderwerp gekozen voor mijn onderzoek. De efficiënte verwerking van dit concept brengt door haar veelzijdigheid een zekere moeilijkheid met zich mee.

Het doel van het onderzoek is om aan te tonen hoe het *high-level* concept teamwork in verschillende games toegepast kan worden.

2.2 Onderzoeksvraag

Het aantonen van mogelijkheden om high-level concepten in verschillende games toe te passen, wordt gedaan aan de hand van het gekozen onderwerp teamwork. Het onderzoek biedt antwoord op de volgende onderzoeksvraag:

"*Welke mogelijkheden bestaan er om het high-level concept teamwork her te gebruiken in verschillende games?*"

[1] http://tweakers.net/nieuws/48069/electronic-arts-splitst-zich-op-in-vier-divisies.html

De hoofdvraag is op te delen in een aantal subvragen.

- Wat is *game development*?
- Wat is het verschil *tussen game development* en traditionele software productie
- Wat is een *high-level* concept?
- Welke mogelijkheden bestaan voor het hergebruik van software binnen *game development*?
- Wat is teamwork?
- Hoe wordt teamwork toegepast in games?

2.3 Aanpak

In deze paragraaf wordt de aanpak van het onderzoek uiteengezet. De aanpak bestaat uit twee onderdelen, namelijk een literatuuronderzoek en een praktijk waarin een serious game wordt ontwikkeld. Gebaseerd op onderzochte literatuur zijn prototypes ontwikkeld die enkele aspecten van teamwork in games belicht. De prototypes en literatuur vormen samen de basis voor de serious game.

Literatuuronderzoek
Informatie over *game development*, hergebruik en teamwork wordt uit literatuur gehaald. Zij vormen de basis voor de toepassing van herbruikbare teamwork elementen in games. Het resultaat is beschreven aan de hand van de gemaakte prototypes en serious game.

Teamwork Game: Proof Of Concept
Om de toepassing van herbruikbare teamwork elementen in games te tonen is er een serious game ontwikkeld. Dit is een game op kleine schaal en dient als een proof of concept (PoC). Dit betekent dat het spel niet compleet afgerond zal zijn. In het spel komen teamwork onderdelen voor die ook in andere spellen kunnen worden toegepast. Criteria, die aangeven of een onderdeel herbruikbaar is, worden in het hoofdstuk over hergebruik beschreven. De ontwikkeling van de PoC vindt plaats in samenwerking met het Clima Futura[1] project aan de VU.

[1] *http://www.climafutura.nl/*

2.4 Opbouw verslag

Het verslag is verdeeld over zes hoofdstukken.

Hoofdstuk 3 biedt globaal inzicht op de context van het onderzoek; namelijk *game development*. Hierin wordt het ontwikkelingsproces van een spel beschreven.

Hoofdstuk 4 somt verschillende mogelijkheden van hergebruik binnen software ontwikkeling op. Deze mogelijkheden zijn afkomstig van software engineering praktijken binnen en buiten game development.

Hoofdstuk 5 gaat in op het concept van teamwork. Zij geeft een definitie en beschrijft aspecten die bij teamwork betrokken zijn.

Hoofdstuk 6 biedt een overzicht van teamwork toepassingen in games.

Enkele mogelijkheden uit hoofdstukken 4, 5 en 6 worden geëxploreerd in een Proof of Concept, welke gedetailleerd beschreven wordt in hoofdstuk 7.

Tenslotte zijn de conclusies en aanbevelingen terug te vinden in hoofdstuk 8.

3 Game Development

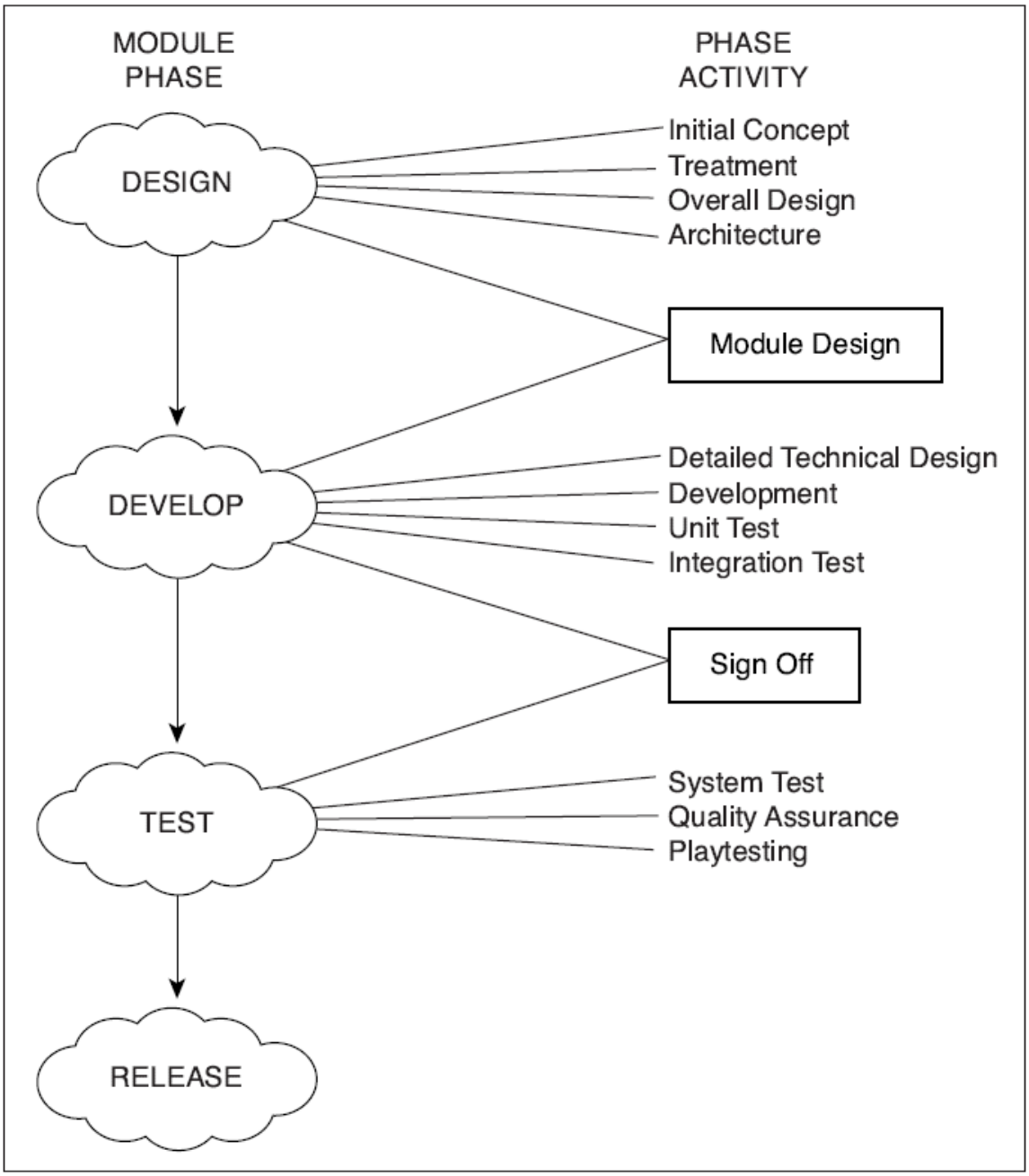
Game development is het proces waarmee een digitaal spel ontwikkeld wordt. Voor alle software geldt dat het ontwikkelingsproces en gebruikte methodes en technieken per bedrijf en per project kan verschillen. *Game development* valt onder software engineering.

Software engineering is betrokken bij problemen die te maken hebben met de constructie van ‘grote’ programma’s [Vli00]. De waterval methode (ontwikkelingsproces), object georiënteerd programmeren (gestructureerd programmeren) en C/C++ (programmeertaal) zijn enkele oplossingen voor het ontwikkelen van software. Samen vormen zij een verzameling van software engineering praktijken [FO05]. Per project wordt gekeken welke oplossingen het beste van toepassing zijn.

Game development liep enkele jaren achter op de heersende software ontwikkelingsmethodes, maar loopt nu ongeveer gelijk. Dit betekent dat business en game software in gelijkwaardige stijl ontwikkeld wordt [RM04].

3.1 Ontwikkelingsproces

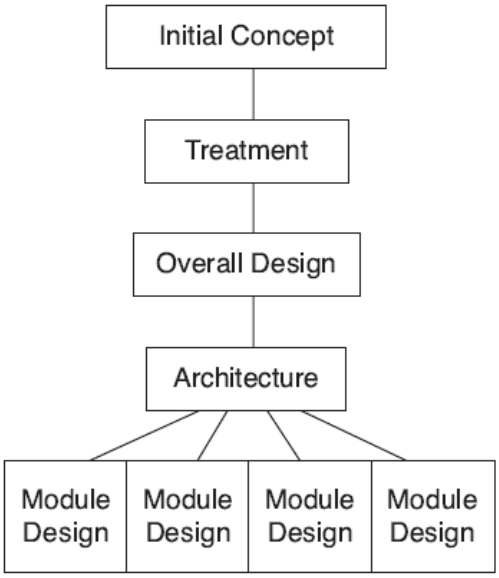
Het ontwikkelingsproces van een spel bestaat over het algemeen uit de volgende fasen [RM04]:



Figuur 1: project opgesplitst in fasen en activiteiten [RM04]

Design Fase

De design fase begint bij de formulering van een game concept en eindigt op het moment dat de algehele design voor een module geschreven is. Een game concept kan leiden tot meerdere modules, zoals het volgende plaatje laat zien.



Figuur 2: initiële concept tot module design [RM04]

Initial Concept

Het initiële concept voor een spel is puur creativiteit. Ideeën, notities, schetsen en diagrammen beschrijven het spel. Het initiële concept dient als startpunt van een project en wordt ter goedkeuring gepresenteerd aan de betreffende *stakeholders* (management, ontwikkelingsteam, uitgever, collega's, etc).

Treatment

De *treatment* beschrijft de gameplay, definieert de belangrijke features van het spel en geeft een beeld van de sfeer van het spel. Het is een formeel document dat het spel beschrijft (verhaal, *look and feel* en basis mechanisme).

Overall design

Het *overall design* is de eerste versie van een gedetailleerd ontwerp. Dit evolueert gedurende de loop van het project. Dit document is vereist voordat het project specifiek technisch werk kan beginnen. Het *overall design* document is semi-technisch en specificeert hoe het spel werkt, eruitziet, speelt en hoe de spelregels en onderdelen gedetailleerd werken. Alle onderdelen, karakters, verhaal elementen, fysieke weergave, sfeer en alle andere details van het spel worden beschreven. Dit document kan gebruikt worden als basis voor de handleiding.

Architecture

Het architectuur document is het initiële technische document. Dit document beschrijft hoe het project vanuit het module niveau gebouwd wordt. Dit omvat hoe het project binnen de richtlijnen van de totale bedrijfsarchitectuur past, inclusief een plan voor

hergebruik om optimaal gebruik te maken van componenten die al ontwikkeld zijn. Bij het kiezen van componenten kan er gekozen worden om gebruik te maken van componenten die door het bedrijf zelf ontwikkeld zijn of van 3de partij componenten. Een software component waarvan in de meeste gevallen licenties gekocht worden bij een 3de partij, is de 3D engine. Dit is een complex software component en zorgt voor de 3D weergave van een spel. De "Unreal Engine"[1] is een voorbeeld van een 3D engine.
De oplevering is een document dat de modules specificeert die het spel samenstellen en de connecties tussen deze modules.

Module design

Voor elke module wordt een module design document geschreven. Het eerste ontwerp wordt geschreven door een software planner of een game designer. Verdere herzieningen worden gewoonlijk door de ontwikkelaars behandeld.

Dit document is een high-level technische specificatie over hoe een module werkt en wordt gebruikt als handleiding bij het gebruik van een bepaalde module. Het module design document wordt hoofdzakelijk gebruikt om informatie te geven voor een module bibliotheek en speelt een belangrijke rol bij plannen voor hergebruik.

Een module kan van veel verschillende types zijn die elk van elkaar kunnen verschillen in belang en herbruikbaarheid. Zo is een code module (module die alleen computer code uitvoert) eenvoudig her te gebruiken en een artwork module (module dat gebruikt maakt van grafische elementen die uniek zijn voor een spel) niet.

Module design levert een document welke gedetailleerde instructie bevat over hoe een bepaalde module werkt en waarvoor de module gebruikt wordt. Ook bevat het voorbeelden van hoe de module gebruikt kan worden.

Development Fase

Dit is de fase waarin het spel wordt gebouwd. Programmeurs schrijven nieuwe code, design artiesten zorgen voor het visuele aspect van het spel, geluidtechnici maken de geluideffecten en componisten schrijven de muziek voor het spel.

Detailed technical design

Elke module wordt samen met het gedetailleerde technische design document geschreven.
Dit document kan gezien worden als een extensie van het module design document, maar met veel meer detail. Het doel van dit document is om andere ontwikkelaars exact uit te leggen hoe een module werkt, inclusief de redenering voor bepaalde ontwerp keuzes.

Dit document is het logboek bij het ontwikkelen van een module en is net zo belangrijk als de module zelf.

Er wordt een document opgeleverd dat gedetailleerde technische specificaties van een bepaalde module bevat, zoals interfaces, algoritmes en ontwerp keuzes.

Development

Een module wordt normaalgesproken ontwikkeld aan de hand van een design document. *Art* (achtergrondplaatjes, 3D modellen, etc) wordt ontwikkeld aan de hand van het spelontwerp in combinatie met een *style guide* (een document dat aangeeft welke stijl

[1] *http://www.unrealtechnology.com/*

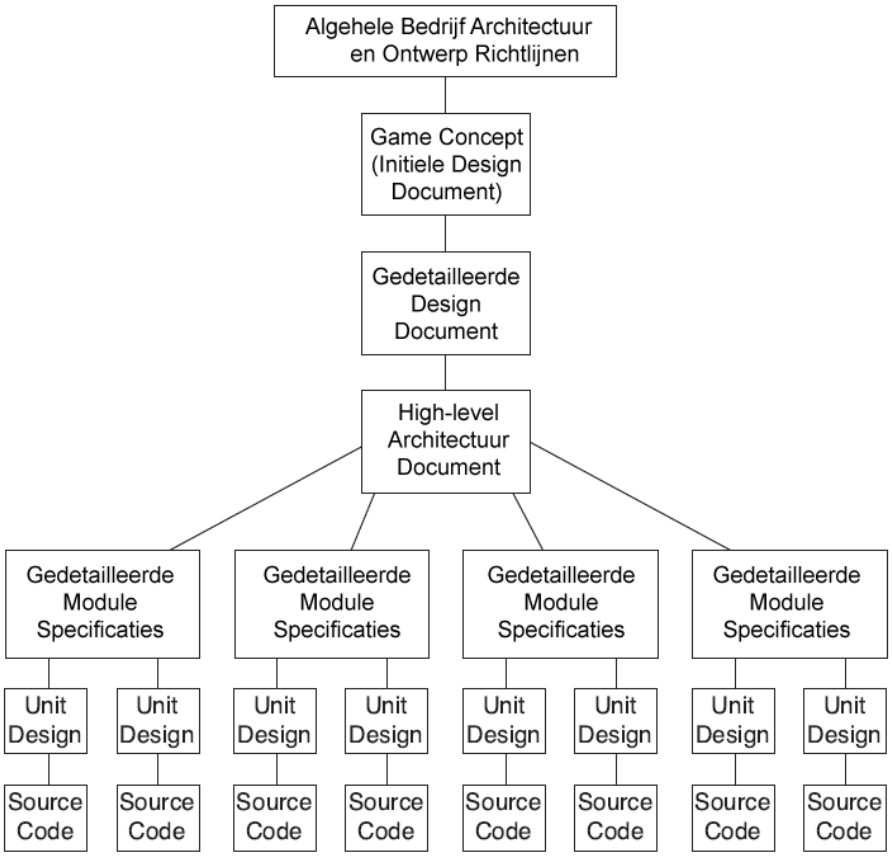
gehandhaafd moet worden). Code wordt geschreven, gebruikmakend van een gedetailleerd technische ontwerp.

Unit test
Unit testing is het testen van de door de programmeur gebouwde component. De *unit test* volgt een script dat deel uitmaakt van het technische design document. Test resultaten worden gerapporteerd aan de projectmanager.

Integration test
Integratie test is gewoonlijk de laatste test ronde voor een module. Er wordt getest of de voltooide module in het ontworpen systeem past. Integratie testen worden gezien als een extensie van de *unit test* en de gebruikte testen zijn deel van de *unit test* script.

Sign off
De *sign off* is de laatste fase waarin de ontwikkelaars aan een module werken.

De beschreven documenten zijn belangrijk voor een project. Het plaatje hieronder geeft de hiërarchische structuur weer van de documenten. Naar beneden toe worden de onderdelen complexer.



Figuur 3: hiërarchische structuur van documenten

Testing Fase

De testing fase betrekt drie kritieke niveaus van testen: *system test*, *quality assurance test* en *playtesting*.

System test

Een systeem test wordt zo vaak mogelijk uitgevoerd. De test is op een iets hoger abstractie niveau dan de *unit test* en levert een systeem test log op met veel algemenere beschrijving. Dit komt omdat bij een systeem test niet wordt uitgegaan van de kennis van hoe het systeem er van binnen uit ziet (black box test).

Quality assurance

Quality assurance is een hoger niveau van testen [RM04 blz. 354]. Dit verzekert dat het spel artistiek aangenaam is. Het is niet bedoeld om defecten in het programma te vinden, omdat deze er in dit stadium er al uitgehaald hadden moeten zijn. Het doel is om er voor de zorgen dat de spelsfeer, menu, handleiding, etc, gebruiksvriendelijk en consistent zijn volgens de aangegeven spelstijl uit het game design document.

Playtesting

Dit is het laatste test stadium. Hierin wordt de totale spelervaring getest. Hoe speelt het spel? Is de handleiding geschikt? Hoe is de leercurve? Zijn er problemen met de gameplay? De oplevering is een gameplay rapport.

Release Fase

In deze fase wordt het spel uitgebracht en geleverd aan de eindgebruiker. Na het uitgeven van het spel vindt onderhoud plaats.

Bij het maken van een spel wordt door de ontwikkelaars rekening gehouden met het platform waarop het spel wordt gespeeld. Zo kan een spel ontwikkeld zijn om gespeeld te worden op een Windows of Linux omgeving. Ondanks dat de omgeving gelijk blijft, kunnen hardware configuraties (moederbord, grafische kaart, geluidskaart, etc) van consumenten onder elkaar verschillen. Om deze reden bestaan er soms onvoorziende compatibiliteitsproblemen. Deze problemen worden opgelost met zogenaamde patches, die op de website van de ontwikkelaar of via een automatische update te verkrijgen zijn.

Naast het oplossen van compatibiliteitsproblemen vindt bij de game, zeldzaam onderhoud van het spel plaats. Via updates worden nieuwe versies uitgegeven die de inhoud en gameplay van het spel aanpassen.

3.2 Rollen binnen Game Development

Game development maakt gebruik van verschillende disciplines. In grote lijnen kan het personeel over vijf groepen worden verdeeld. Tabel 1 geeft een overzicht van vijf groepen en de rollen die daar bij horen [RM04].

Tabel 1

Groep	**Rol**
Management en Design	Software planner Hoofdarchitect Project manager Game designer
Programmeren	Hoofdprogrammeur Programmeur
Art	Lead artist Artist
Muziek en Verscheidene	Muzikant Geluid effect technici Verscheidene technici (zoals motion capture)
Support en Quality Assurance	QA leider QA team Playtester Support technici

Software planner

De taak van de software planner is om het spelontwerp op te splitsen in een set van gedetailleerde technische requirements en een inschatting te maken van tijd en inspanning dat nodig is om de features te implementeren.

De software planner werk normaal gesproken samen met de hoofdarchitect en de game designer om de gedetailleerde specificaties voor te bereiden en om deze in een technische architectuur document, welke de benodigdheden van het project omschrijft, te verwerken.

Hoofdarchitect

De taak van de hoofdarchitect is om in samenwerking met de software planner een set van module specificaties te produceren uit de technische requirements. De hoofdarchitect is verantwoordelijk voor de algehele architectuur van het project. Gedetailleerd ontwerp van de individuele modules, wordt meestal overgelaten aan de hoofdprogrammeur.

Project manager

De taak van de projectmanager is om de werklast, dat opgeleverd is door de software planner en de hoofdarchitect, te ordenen in een efficiënt en georganiseerd schema. Hij is verantwoordelijk om het project binnen de gestelde kaders; in tijd, in mancapaciteit, in uren en in geld tot een succesvol project te leiden. De projectmanager handelt interactie

tussen teamleden af, en gedraagt zich als de tussenpersoon tussen het project team en de marketing afdeling.

Game designer

De taak van de game designer is om de games te ontwerpen. De game designer werk samen met de software planner om de haalbaarheid van het gameontwerp te exploreren.

Hoofdprogrammeur

De taak van de hoofdprogrammeur is om de inspanning van het programmeer team te coördineren en te controleren. Hij zorgt er voor dat het team op schema ligt en rapporteert de vooruitgang van zijn team aan de projectmanager.

Programmeur

Een programmeur schrijft computer code aan de hand van getailleerde technische specificaties. Normaal gesproken is een programmeur verantwoordelijk voor een bepaalde subsectie van het hoofdprogramma. Dit blijft zijn verantwoordelijkheid gedurende het hele project.

Lead artist

Net als de hoofdprogrammeur coördineert en controleert de *lead artist* zijn team. De *lead artist* werkt samen met de hoofdprogrammeur en game designer om er voor te zorgen dat de gemaakte kunstwerken bruikbaar zijn voor het spel.

Artist

Een *artist* produceert kunstwerken die gebruikt worden voor één of meer projecten. Hierbij kan gedacht worden aan achtergrond decoraties voor het spel, ontwerp van de handleiding en ontwerp van de doos waarin het spel verkocht wordt.

Muzikant

Muziekanten werken over het algemeen gescheiden van het principale gedeelte binnen game development. Afhankelijk van het spel zitten hier naast instrument bespelers componisten en/of een dirigent tussen. Zij ontvangen een animatie, een demo of een beschrijving van een sfeer. Dit is voldoende om de muziek voor het spel te produceren. In het geval dat er interactieve muziek (muziek waarin tempo en sfeer verandert volgens gebeurtenissen in het spel) vereist is, dan wordt er wel meer samengewerkt met de rest van het *game development* team.

Geluid effect technici

Elk spel heeft geluidseffecten. Geluid effect technici produceren deze effecten. Dit omvat geluiden die te horen zijn bij het klikken op een knop of wanneer “Super Mario” de lucht in springt.

Verscheidene technici

Afhankelijk van het project zijn verscheidene technici nodig om taken te verrichten voor de ontwikkeling van het spel. Zo zijn er *motion-capture* technici, die betrokken zijn bij de animatie van een spel.

QA leider

De rol van de QA leider is het sturen van het QA team en samenwerken met de project manager en de game designer om er zeker van te zijn dat het spel grondig getest wordt op gebied van gameplay en functies [RM04 blz. 253]. De QA leider maakt de test plannen die door de QA technici worden uitgevoerd. Het resultaat wordt aan de project manager gerapporteerd.

QA technici

Een QA technici test de code, geschreven door het programeer team, uit. Alle functies en mogelijkheden worden uitgetest. Elk stukje geschreven code, hoe simpel het ook is, heeft een test plan die alle functies uittest. Er wordt samengewerkt met de programmeurs om er voor te zorgen dat de software modules grondig getest worden.

Playtester

Een *playtester* test hoe het spel gespeeld wordt. Een *playtester* kan een programmeur zijn of deel uit maken van het *game development* team. Omdat beiden betrokken zijn bij de ontwikkeling van het spel zijn zij niet objectief genoeg om het spel te kunnen testen. Een alternatief is om studenten te betalen om het spel enkele uren te laten spelen en daarover feedback te ontvangen. Het uitbrengen van een beperkte betaversie onder geregistreerde gaming deskundigen is een andere optie. Zij nemen op vrijwillige basis de *playtesting* voor hun rekening. Een ander alternatief is om een permanent *playtesting* team in het bedrijf op te nemen.

Support technici

Support technici onderhouden de computer omgeving waar de rest van het bedrijf gebruik van maakt. Verantwoordelijkheden omvatten het onderhouden van het netwerk, verzekeren dat alle machines de correcte software hebben en het uitvoeren van hardware upgrades (vervangen van defecte onderdelen en het uitwisselen van “oude” hardware onderdelen voor “nieuwe” krachtigere hardware onderdelen).

3.3 Game concept

Het bedenken van een spel concept is een activiteit die je binnen traditionele software engineering projecten niet terugvindt. Dit is een taak van de game designer. Ideeën voor spellen zijn er genoeg. Zo bestaan er ideeën voor een racespel en een avonturenspel dat zich afspeelt in een fantasie wereld. De moeilijkheid ligt in het uitwerken van het idee tot een aangenaam spel. Dit omvat onder andere de opbouw van het spel, de manier waarop het spel gespeeld wordt en het verhaal.

Volgens Rouse [Rou01] wordt het proces van het bedenken van een spel gelimiteerd door factoren afkomstig uit drie onafhankelijke gebieden: gameplay, technologie en verhaal. Hier hoeven schrijvers van fictie geen rekening mee te houden. Wanneer een game designer een spel bedenkt, neemt hij in gedachte wat voor impact zijn idee heeft voor alle aspecten van het spel.

Een start kan gemaakt worden uitgaande van gameplay, technologie of verhaal. Uitgaande van gameplay, ontwerpt een game designer een spel gebaseerd op de

gameplay van een ander spel dat hem goed is bevallen. Gameplay mag je vertalen als ‘scenario’ of ‘naar analogie van’. Hieraan zal hij aan het spel zijn eigen draai geven. Zo kan hij zijn idee bijvoorbeeld op de volgende manieren uitdrukken: “Het wordt een race spel”, “Het wordt een vliegsimulatie” en “Het wordt een avonturenspel zoals *Super Mario 64*”. Hiermee wordt het spelgenre bepaald. Enkele spelgenres zijn: RPG (Role Playing Game), Platform (springen en landen op platformen) en First Person Shooter. Daarnaast wordt een globale impressie gegeven van de uitwerking. “Zoals Super Mario 64” is een voorbeeld van zo een impressie.

Uitgaande van technologie kan er gekozen worden om een spel te bouwen op specifiek geselecteerd technologie. Zo kan het zijn dat er voor een nieuwe 3D engine wordt gekozen, waarbij het doel is grenzen te verleggen in 3D weergave.

Het spel kan ook beginnen met een verhaal. De gameplay en technologie hangt in dat geval af van de inhoud van het verhaal. Zo leent een verhaal, waarin een groep vrienden in een fantasie wereld moet strijden tegen vijandige wezens, zich voor een first-person shooter. Een verhaal waarin de speler met een groot aantal verschillende karakters moet spreken en op zogenaamde “quests” (voltooien van missies) moet gaan, kan het beste uitgedrukt worden in een RPG (Role Playing Game).

Een game concept wordt met gemiddeld 5-6 bladzijdes beschreven en dient ter communicatie van het idee en het overbrengen van de speelwijze (gameplay). Dit document kan anders verkocht worden aan spelontwikkelaars die het spel willen creëren.

Het is de taak van de game designer om iets nieuws te bedenken. Volgens Rolling en Morris [RM04] biedt de evolutie in computer technologie nieuwe spel mogelijkheden, en mogelijkheden om oude spellen op een nieuwe wijze over te brengen.

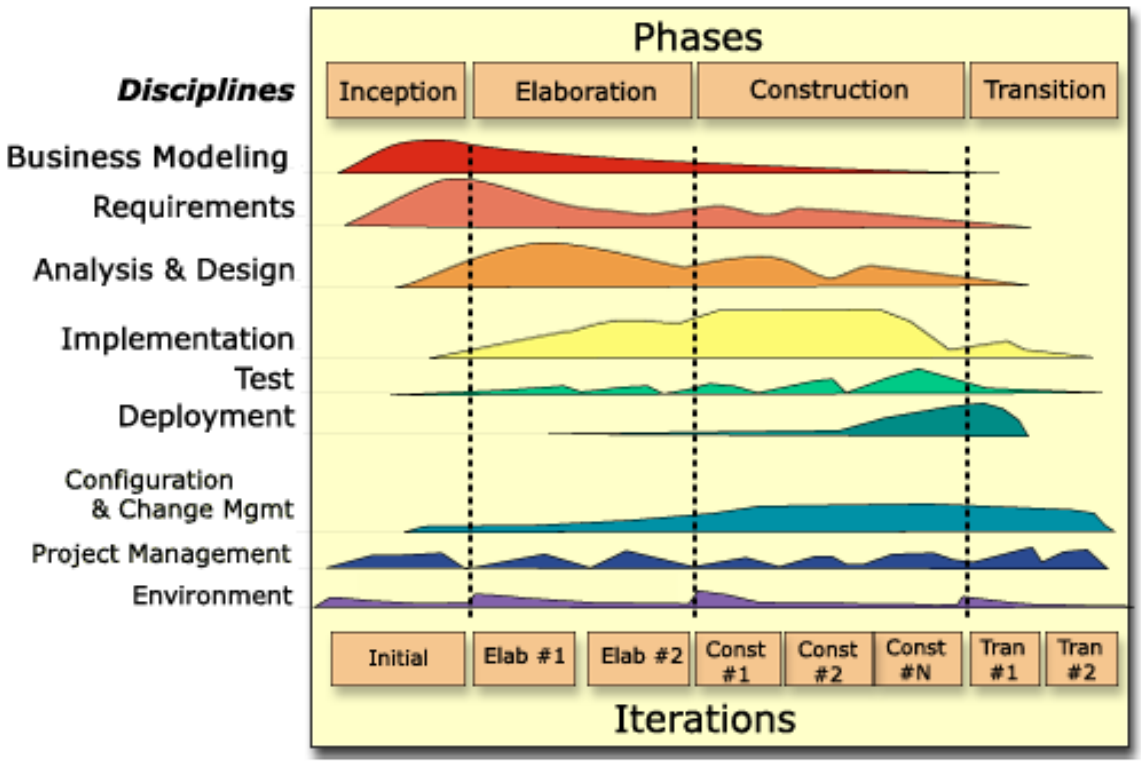
Figuur 4: nieuwe computer technologie geeft Super Mario (1985) links nieuw leven met Super Mario Galaxy (2007) rechts

3.4 High-level concept

Een high-level concept is een ruw concept in hoofdlijnen beschreven zonder enige detail. Het beschrijft een idee met abstractie van gedetailleerde specificaties om het idee te realiseren. Een game concept is zo een high-level concept. Onderdelen van een game concept, zoals gameplay en grafische weergave, zijn ook high-level concepten.

3.5 Vergelijking game development en traditionele software ontwikkeling

Het ontwikkelen van een game verschilt niet veel met het ontwikkelen van software. Een bekende ontwikkelingsmethode is het RUP (Rational Unified Process) [SK08]. RUP is een framework dat gemaakt is door “Rational Software”[1]. Met RUP wordt de functionaliteit van het systeem stapsgewijs opgeleverd in iteraties van vier tot zes weken doorlooptijd. Wijzigende inzichten en of tekortkomingen komen snel boven water. RUP beschrijft niet een concreet proces, maar eerder een aanpasbaar proces framework. Het vormt een set ‘*best practices*’, die de organisatie naar eigen inzicht kan oppakken en finetunen. Het plaatje hieronder geeft de fases, disciplines en iteraties weer over een tijdlijn. Voor elke discipline wordt de verwachte inspanning op een gegeven moment binnen het proces weergegeven.



Figuur 5: Rational Unified Process

Fasen

Inception

Het doel van de *inception* fase is om overeenstemming onder alle belanghouders te bereiken over de levenscyclus doelen van het project. De *inception* is hoofdzakelijk van belang voor nieuwe ontwikkelingsinspanningen waarin significante business en requirements risico's bestaan die verwerkt moeten worden voor het project door kan gaan.

[1] *http://www-306.ibm.com/software/rational/?S_TACT=105AGY59&S_CMP=13&ca=dtl-13*

Voor projecten, die zich concentreren op verbeteringen van een bestaand systeem, is de *inception* fase korter, maar er wordt nog steeds geconcentreerd op het verzekeren dat het project de moeite waard is en mogelijk is om uit te voeren.

Elaboration

Het doel van de *elaboration* fase is de grondlegging van de architectuur van het systeem om een stabiele basis voor het merendeel van het ontwerp en implementatie inspanning in de *construction* fase te verzorgen. De architectuur wordt ontwikkeld uit een overweging van de meest significante requirements (die grote impact op de architectuur van het systeem hebben) en een beoordeling van risico. De stabiliteit van de architectuur wordt door een of meer architectuur prototypes geëvalueerd.

Construction

Het doel van de *construction* fase is het ophelderen van de resterende requirements en het voltooien van de ontwikkeling van het systeem gebaseerde op de ontwikkelde architectuur. De *construction* fase is in enige zin een productieproces, waar de klemtoon wordt geplaatst op het beheren van resources en het controleren van operaties om kosten, schema's en kwaliteit te optimaliseren. In deze zin ondergaat het managen een overgang van de ontwikkeling van intellectueel eigendom tijdens *inception* en *elaboration*, naar de ontwikkeling van inzetbare producten tijdens *construction* en *transition*.

Transition

De focus van de *transition* fase is het verzekeren dat de software beschikbaar is voor zijn gebruikers. De *transition* fase kan enkele iteraties overspannen en omvat het testen van het product ter voorbereiding van de release en het maken van geringe aanpassingen gebaseerde op de feedback van de gebruiker. Op dit moment in de levenscyclus, zou de feedback van de gebruiker hoofdzakelijk gericht moeten zijn op het finetunen, configuratie, installatie en gebruiksproblemen van het product. Belangrijke structurele kwesties hadden veel eerder in de levenscyclus van het project moeten worden opgelost.

De tabel hieronder zet de RUP fasen en de globale game development fasen naast elkaar.

Tabel 2

RUP Fasen	**Game Development Fasen**
Aanvang Uitweiding	Design
Constructie	Development Test
Transitie	Release

Disciplines
In de onderstaande paragraaf worden de RUP disciplines van de traditionele software engineering omgeving beschreven om deze vervolgens te mappen op de disciplines van de gaming omgeving.

Business modeling
Het doel van *business modeling* is om het begrip en de communicatie tussen business en IT te verbeteren. Het begrijpen van de business betekent dat software ingenieurs de structuur en dynamiek van de organisatie (van de klant), de problemen binnen de organisatie en mogelijke verbeteringen, begrijpt. *Business modeling* beschrijft de visie van de organisatie waarvoor het systeem gemaakt wordt en gebruikt deze visie voor het aangeven van processen, rollen en verantwoordelijkheden.

Requirements
Deze discipline zet het verzoek van een stakeholder om in een set gedetailleerde requirements die aangeven wat het systeem moet doen.

Analysis & Design
De doelen van *analysis & design* zijn: het omzetten van de requirements naar een ontwerp van het systeem, het ontwikkelen van een robuuste architectuur voor het systeem en het aanpassen van het ontwerp aan de implementatie omgeving zodat het ontwerp praktisch inzetbaar is. *Analysis & design* laat zien hoe het systeem gerealiseerd wordt.

Implementation
De *implementation* discipline definieert de organisatie van code (subsystemen georganiseerd in lagen), implementeert design elementen (*source files*, *binaries*, uitvoerbare programma's en andere), test de ontwikkelde componenten als eenheden en integreert de geproduceerde resultaten in een uitvoerbaar systeem.

Test
Met testen wordt onder andere gecontroleerd op interactie tussen verschillende software objecten, op de integratie van verschillende software componenten en of alle requirements correct geïmplementeerd zijn.

Deployment
De *deployment* discipline beschrijft de activiteiten die geassocieerd zijn bij het verzekeren dat het software product beschikbaar is voor zijn gebruikers.

Configuration and Change Management discipline
Deze discipline legt uit hoe de ontwikkeling van een set *Work Products*, die samen een software systeem vormen, gecontroleerd en gesynchroniseerd kan worden.

Project Management
Deze discipline focust op project planning, *risk management*, bijhouden van vooruitgang. Het doel van project management is het aanbieden van een framework voor het managen van software-intensieve projecten, het aanbieden van praktische richtlijnen

voor planning, bemanning (aangeven hoeveel werknemers nodig zijn op een bepaald moment), uitvoering en controleren van projecten en het aanbieden van een framework voor risk management.

Environment discipline

De environment discipline beschrijft welke acties er nodig zijn om het project te ondersteunen. Het doel van deze activiteiten is om de software ontwikkelingsorganisaties te voorzien van een software ontwikkelingsomgeving om het ontwikkelteam te ondersteunen.

Activiteiten binnen de RUP disciplines zijn terug te vinden in activiteiten binnen game development. In een voorgaande sectie zijn enkele rollen binnen game development beschreven. De volgende tabel zet de RUP disciplines en de rollen binnen game development met overeenkomende activiteiten naast elkaar.

RUP Disciplines	**Game Development Rollen**
Business modeling	Game designer Hoofdarchitect
Requirements	Game designer Software Planner
Analyse & Design	Software Planner Hoofdarchitect Hoofdprogrammeur
Implementation	Hoofdarchitect Hoofdprogrammeur Programmeur
Test	Programmeur Quality Assurance leider Quality Assurance technici Playtester
Deployment	Hoofdprogrammeur Programmeur
Configuration and Change Management	Hoofdarchitect Hoofdprogrammeur
Project Management	Project manager
Environment	Support technici

Verschil tussen Game Development en RUP

Anders dan in RUP ligt de werkdruk voor architectuur en design binnen game development hoger. De architectuur en game design groepen zijn de enige twee groepen die van begin tot eind van het project actief blijven, met een relatief hoge werkdruk. Dit komt omdat de architectuur groep dient als middelpunt van een project en als contactpersoon tussen de verschillende groepen. De game design groep is verantwoordelijk voor het initiële game concept en werkt gedurende het hele project mee aan de inhoud van het spel. Voor het concept architectuur en game concept geldt dat deze

gedurende het hele project gewijzigd en aangepast kunnen worden. Dit is anders dan in RUP waarin aanpassingen en wijzigingen voornamelijk gebeuren in de eerste twee fasen van een project. Dit komt omdat er voor een spel na de release geen inhoudelijke wijzigingen meer plaatsvinden. Het resultaat is definitief. Voor een game bestaat het bèta testen en het maken van aanpassingen, na levering van het product aan de eindgebruiker, niet. Om er voor te zorgen dat een spel voldoet aan de verwachtingen van de speler vinden alle inhoudelijke wijzigingen voor de release plaats.

Oorzaak van dit verschil is dat in traditionele software productie zoals RUP een product ontwikkeld wordt voor een specifieke klant die zijn verwachtingen en eisen van te voren heeft opgesteld. Bij de ontwikkeling van een spel wordt er uitgegaan van een idee (game concept). Dit concept is niet expliciet afgestemd met de verwachtingen en eisen van de eindgebruiker. Om deze informatie in te winnen worden demo's vrijgegeven en bèta testen uitgevoerd voordat het spel uitgegeven wordt. Afhankelijk van deze informatie worden wijzigingen gemaakt om het spel af te stemmen met de wensen van de eindgebruiker.

3.6 Conclusie

Game development is het proces waarmee een digitaal spel wordt ontwikkeld. Dit proces verschilt per bedrijf en per project. Globaal is het proces in vier fases in te delen: *design*, *develpment*, *test* en *release*.

De ontwikkeling van een spel begint met een game concept. Dit is een high-level concept dat het spel in hoofdlijnen beschrijft, geabstraheerd van gedetailleerde specificaties om het spel te realiseren.

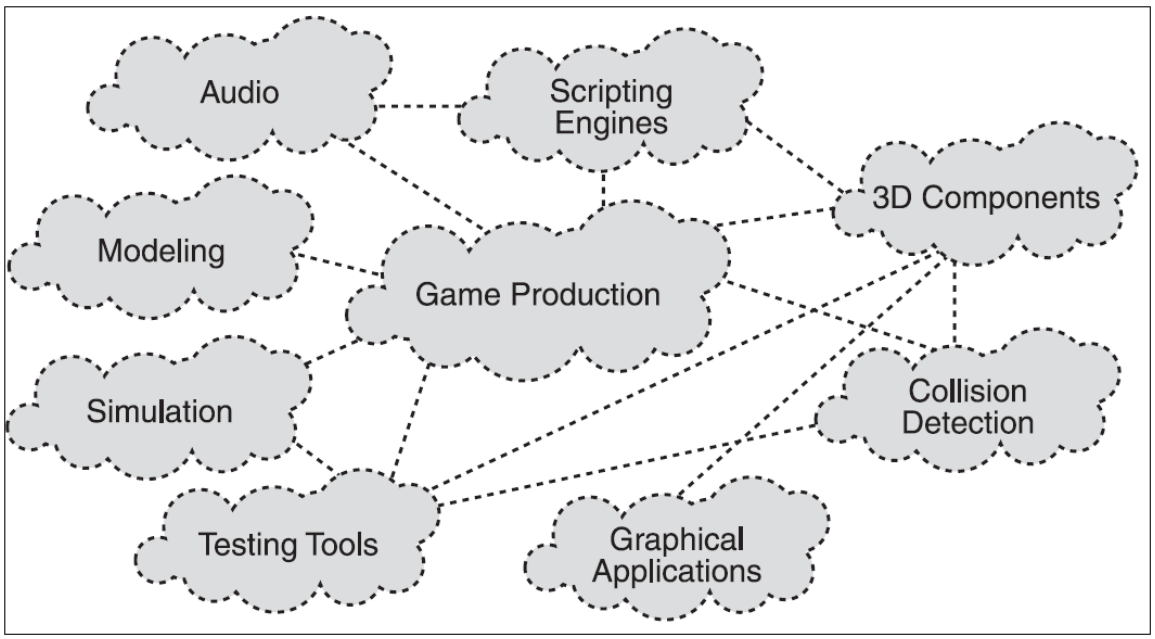
Het evolutiepad van *game development* heeft een andere route genomen dan traditionele software ontwikkeling. Traditionele software ontwikkeling begon bij grote mainframes en heeft zijn weg naar de PC gevonden. *Game development* begon met kleine computers waarbij ontwikkelaars zich aanpasten aan de systeemlimieten [RM04]. De twee verschillende aanpakken zijn inmiddels grotendeels samengekomen. De software engineering trends die over de decennia hebben geprevaleerd, zijn nu van sterke invloed op de *game development* industrie [FS05]. Dit betekend dat business en game software in gelijkwaardige stijl geschreven worden. Zij zijn beide in C of C++ gecodeerd, gebruiken de API (*Application Programming Interface*) van de bestemde *operating system* en maken gebruik van derde partij libraries.

Desondanks bestaan er permanente verschillen tussen business en game software die van invloed zijn op het ontwikkelingsproces. Een game is namelijk een creatief product. Net als bij een schilderij kan alleen de schilder bepalen wanneer zijn kunstwerk af is. Binnen game development heeft een game designer minder invloed op het spel (invloed van verschillende stakeholders spelen mee zoals de marketing afdeling en de rest van het ontwikkelingsteam) dan dat een kunstenaar als Rembrandt heeft op zijn schilderij. Maar zijn invloed is wel significant. Binnen traditionele software ontwikkeling wordt er uitgegaan van een specifieke klant die bepaalde eisen (requirements) heeft. Het product is af wanneer de software aan de eisen voldoet.

4 Hergebruik

Volgens Rollings en Morris [RM04] is herbruikbaarheid het concept van het produceren van software componenten die nuttig zijn bij meer dan één project.

De volwassen wording van de game industrie heeft takken van specialisatie doen ontstaan. Programmeurs zijn gewoonlijk niet de principale makers van spellen, maar worden gezien als “resources” die ingehuurd worden door producers om software te implementeren die spelontwerpen ondersteunen. Game designers maken spellen. Programmeurs ontwikkelen hulpmiddelen waarmee spellen gemaakt worden. Sommige programmeurs verlaten de game productie kant van de industrie en starten bedrijven die software (componenten) ontwikkelen om spellen te maken. Hier bevinden zich onder ander bedrijven die gespecialiseerd zijn in engines, audio productie en testing [FS05]. Het volgende plaatje schetst de situatie.



Figuur 6: gespecialiseerde producten binnen de game industrie

De gespecialiseerde software onderdelen zijn toepasbaar in verschillende projecten.

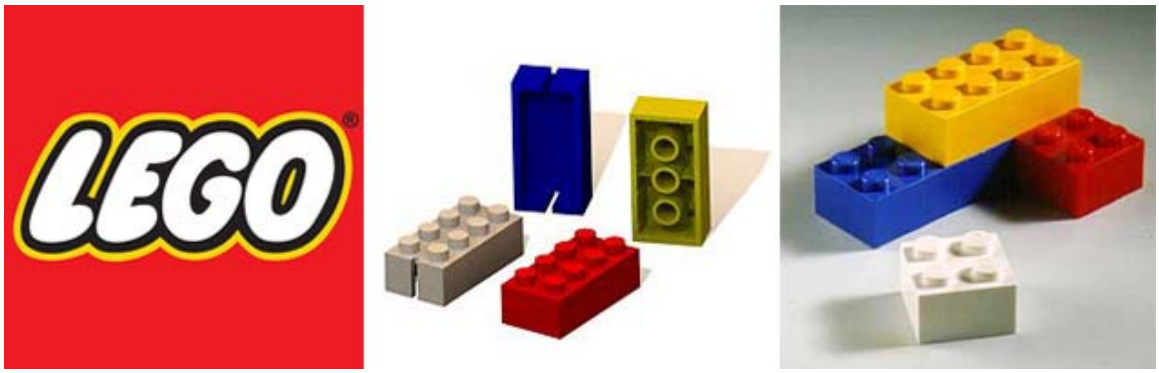
Software component

Een herbruikbaar software onderdeel wordt software component genoemd. Een software component is een onderdeel van een systeem dat een voorgeschreven service aanbied en kan communiceren met ander software componenten [Szy02]. Szyperski [Szy02] noemt vijf criteria voor een software component om aan deze definitie te voldoen:

- Het wordt meerdere malen gebruikt.
- Niet context specifiek.
- Samen te stellen met andere componenten.
- Ingekapseld. i.e. inhoud is niet te onderzoeken via de interface.

- Onafhankelijke eenheid. Heeft eigen release en versienummering.

Deze vijf criteria worden uitgelegd aan de hand van een abstract en concreet voorbeeld, namelijk “Lego” [1]. Daarbij wordt een link gelegd met software componenten.



Figuur 7: lego blokjes

De lego blokjes zijn de bouwstenen voor een object. Zij bestaan in verschillende vormen, kleuren en maten. Zo zijn software componenten de bouwstenen voor een software systeem. Enkele verschillende software componenten zijn: een 3D engine, een AI component en een collisie detectie (detecteert bijvoorbeeld botsingen tussen 3D objecten) component.

De lego stenen worden meerdere malen gebruikt om een object te construeren. In het plaatje hieronder is gebruik gemaakt van dezelfde type lego blokje.

Figuur 8: lego constructie

In *game development* is het niet mogelijk om dezelfde type software component binnen één project meerdere malen te gebruiken, maar wel over verschillende projecten. De 3D

[1] *http://www.lego.com*

engine van “Unreal” [1] en de 3D engine van “id Software” [2] zijn software componenten van hetzelfde type. Zij zorgen voor de visuele weergave van een spel. Het samenvoegen van beide engines in één spel veroorzaakt conflicten omdat zij gelijkwaardige service bieden. Het is wel mogelijk om de “Unreal” engine voor de ontwikkeling van verschillende spellen te gebruiken.

Voor het bouwen van lego objecten is de context niet van belang. Lego blokjes kunnen op elkaar gestapeld worden. Deze eigenschap verandert niet. Met dezelfde bouwstenen is het mogelijk om verschillende objecten te construeren zoals het volgende plaatje laat zien.

Figuur 9: lego constructies

De eigenschappen van een software component veranderen niet. Een sound engine zorgt voor de weergave van geluid. Deze functie is constant en is niet afhankelijk van de soort muziek (rock, klassiek, jazz, etc.) dat weergegeven moet worden.

Lego blokjes kunnen op elkaar gestapeld worden. De interface van het lego blokje maakt dit mogelijk. De onderkant sluit precies aan op de cirkelvormige punten aan de bovenkant. De interface van een software component bepaald welke connecties met andere software componenten mogelijk zijn. De cirkelvormige punten van een lego blokje zijn te vergelijken met de methodes die te vinden zijn in de interface van een software component. Deze methodes bepalen wat er met de software component gedaan kan worden.

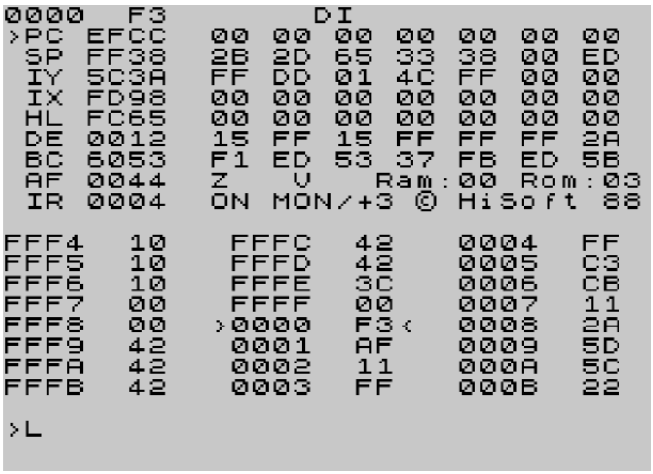
De lego stenen zijn simpele bouwstenen. Door het bekijken van het blokje aan de buitenkant is te zien wat de vorm is, hoe het blokje gemaakt is, wat er mee gedaan kan worden. In dit opzicht is een lego steentje vrij doorzichtig. Via de interface van een software component is het niet mogelijk om te achterhalen hoe een methode wordt uitgevoerd. Dit is een vorm van abstractie. De onderliggende werking van een software component wordt geabstraheerd.

Een lego steentje is een losse eenheid. Onafhankelijk van andere lego steentjes behoudt het zijn functionaliteit. Ook een software component is niet afhankelijk van andere software componenten om goed te werken. Het kan onafhankelijk getest worden en draagt een eigen versienummer. Dit betekent dat wijzigingen in een component geen wijzigingen in andere componenten tot gevolg hebben.

[1] *http://www.unrealtechnology.com*
[2] *http://www.idsoftware.com*

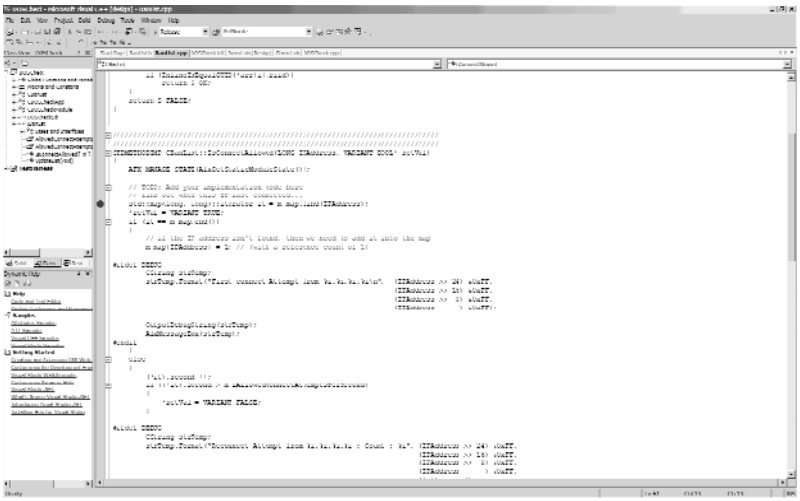
4.1 Game Architectuur

De ontwikkeling van games was vroeger een één-persoons-project. Deze spelontwikkelaars legden bij het programmeren de nadruk op systeem performance. Computers waren voorheen niet zo krachtig als de computers van tegenwoordig. Het was noodzakelijk om “slimme code” te schrijven die soepel op de computers gedraaid konden worden. De gemiddelde grootte van zo een spel is minder dan een zestiende deel van een megabyte, waardoor het nog mogelijk was dat een spel door één persoon gebouwd kon worden. Het scherm van een game ontwikkelaar zag er toen ongeveer uit als het volgende plaatje.



Figuur 10: scherm van een game ontwikkelaar circa 1983

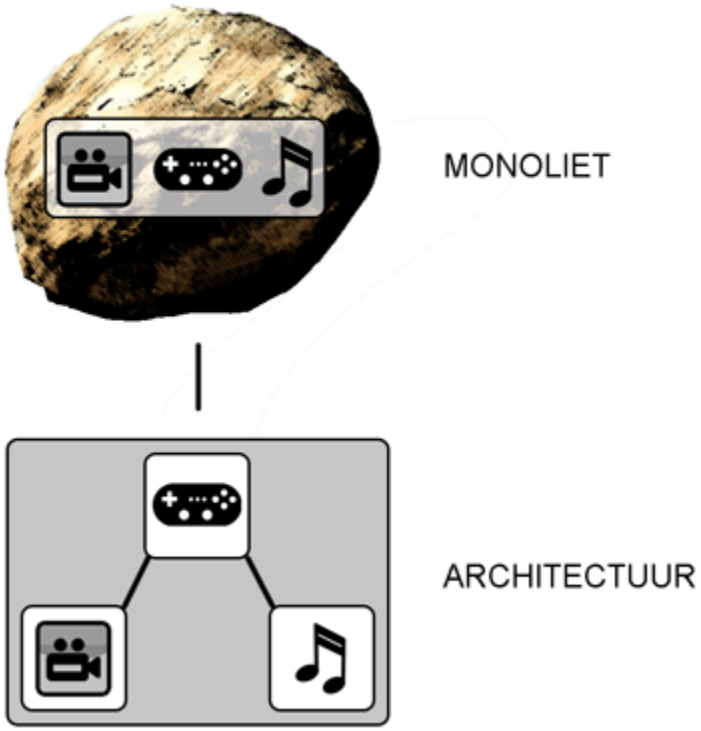
Tegenwoordig, door de explosieve groei van de game markt en computer technologie, zijn dit soort computerspellen verleden tijd. Om een goed product te leveren wordt er met meerdere mensen samengewerkt. Dit betekent dat programmeurs, elkaars code moeten kunnen lezen en begrijpen. De moeilijk te interpreteren code zoals in figuur 10 is nog wel aanwezig, maar wordt verstopt met menu opties en hogere programmeer talen (relatief eenvoudig te interpreten code).



Figuur 11: scherm van een game ontwikkelaar circa 2003

Een game architectuur zorgt voor ordening en beter begrip van het systeem. Vroeger werd architectuur alleen geassocieerd met gebouwen. Huidige commerciële computerspellen kunnen vergeleken worden met wolkenkrabbers waarvoor een architectuur absoluut noodzakelijk is.

Rollings en Morris [RM04] beschrijven game architectuur als de structuur van een programma, die de datastroom en interacties tussen de componenten van een systeem definieert. Het definiëren van een architectuur is de reductie van een monolithisch systeem naar een set van discrete logische subsystemen. De systeemarchitect onderzoekt het probleem domein vanuit een aantal verschillende perspectieven en produceert een model van het systeem, gebaseerde op criteria zoals gegevensstroom, natuurlijke verdelingen (bijvoorbeeld het scheiden van grafische en geluidsonderdelen) en beschikbare software componenten (een bedrijf heeft bijvoorbeeld licenties voor een bepaalde 3D engine).

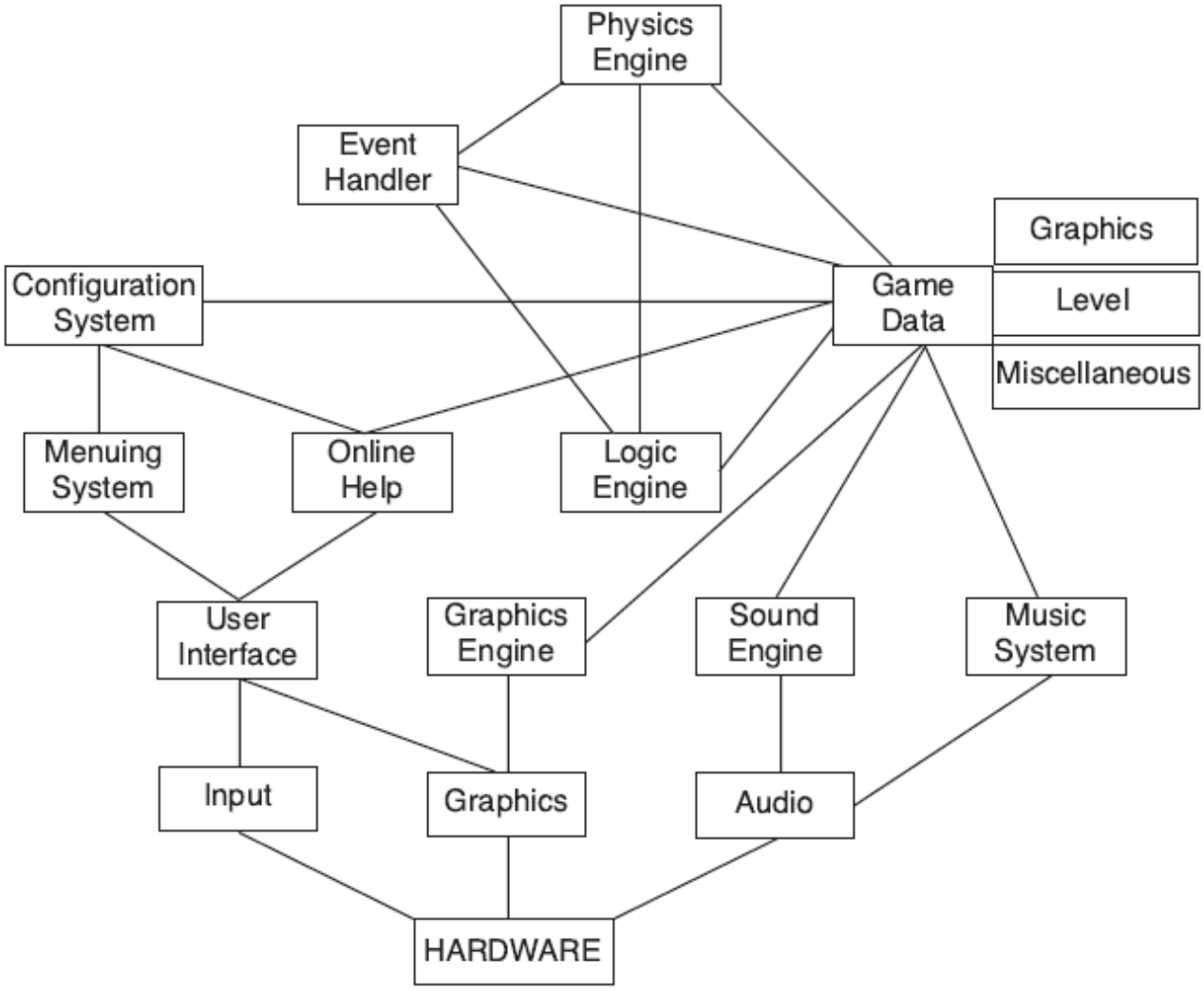


Figuur 12: reductie van een monolithisch systeem naar een set van discrete logische subsystemen

Monoliet ontwikkelen van een spel is het bouwen van een spel in één grote brok. De grote brok bestaat uit meerdere delen, maar zijn zo in elkaar verweven dat zij niet van elkaar te scheiden zijn. Elk gedeelte is afhankelijk van de andere delen, en steunen op interne kennis van hoe de delen functioneren, in plaats van gebruik te maken van de publieke interface. Dit maakt de delen onafscheidelijk van elkaar, alsof elke gedeelte aders heeft die reiken tot het diepste van de harten van de andere gedeeltes.

In contrast met het monoliet systeem staat de component gebaseerde systeem. Hierin wordt wel gebruik gemaakt van publieke interfaces en kunnen de componenten losgekoppeld worden, zonder dat daardoor de rest van het systeem instabiel wordt.

Het volgende plaatje is een voorbeeld van een game architectuur. Zoals de definitie aangeeft, wordt de indeling van het spel in verschillende componenten (structuur) weergegeven en geeft de architectuur weer hoe de componenten met elkaar in verhouding staan (datastroom en interactie).



Figuur 13: voorbeeld van een game architectuur

Een architectuur is geen statisch wezen. Een architectuur groeit en verandert. Ontwikkelaars stellen zich vragen bij het nut van een architectuur, omdat deze gedurende de ontwikkeling van een game steeds verandert. In het klassieke model van architectuur design in dit een valide argument, omdat een architectuur slechts voor één spel werd gebruik en daarna werd “weggegooid”. Dit argument geldt niet voor een architectuur dat flexibel en uitbreidbaar is. Een goed ontworpen architectuur laat flexibiliteit in code en algoritme toe.

Subsystemen

De architectuur deelt het spel op in systemen en subsystemen, die in stadia kunnen worden gebouwd, waarbij een werkend model het uiteindelijke doel is in elke stadium. Deze stadia worden bepaald door het ontwikkelingsproces van een spel. Elk stadium biedt meer functionaliteit dan de vorige zonder daarbij af te wijken van de interface.

Door het ontwerp te verkleinen tot afzonderlijke componenten, worden gebieden van verantwoordelijkheid gescheiden. Het is makkelijker om een systeem te begrijpen dat gemaakt is in vele kleine modules, dan een monolithisch systeem. Een monolithisch systeem is alleen te begrijpen wanneer het gehele systeem begrepen wordt. Grote complexe systemen maken zo een visie praktisch onmogelijk. Dit beeld is wel te creëren wanneer het complexe systeem gemaakt is uit kleine afzonderlijke modules, die relatief eenvoudig te begrijpen zijn. Het maken van kleine aanpassingen in een monolithisch systeem is een complexe taak, omdat het moeilijk is te overzien wat de bijeffecten zijn van de aanpassingen.

Door gebruik van modulaire componenten kan de ontwikkelaar zich concentreren op het juiste niveau van granulariteit. Hij hoeft zich bijvoorbeeld geen zorgen te maken of zijn werk andere delen van het systeem zal beschadigen. Volgens Szyperski [Szy02] is elke module onafhankelijk en vereist geen kennis van de omringende modules om correct te functioneren. Deze subsystemen, modules zijn de game componenten die samen de game architectuur vormen.

Computerspellen zijn essentieel hetzelfde. Er bestaan locale verschillen, maar het interne ontwerp ziet er ongeveer hetzelfde uit.

Op een basisch niveau bestaat een spel uit de volgende primaire subsystemen:

- User interface
- Event handler
- Data engine (graphics, levels, etc)
- Dynamics system (botsingen en algemene fysica)
- Logic engine (het hart van het spel)
- Graphics engine
- Sound engine
- Hardware abstraction layer (spreekt met de graphics, sound en controller hardware).

Verder bevat een game architectuur één of meer van de volgende secundaire subsystemen:

- Game configuratie systeem
- Menu systeem
- Online instructies en help systeem
- Muziek systeem

De lijst van secundaire subsystemen is geen definitieve lijst. Hier passen nog vele andere denkbare subsystemen tussen.

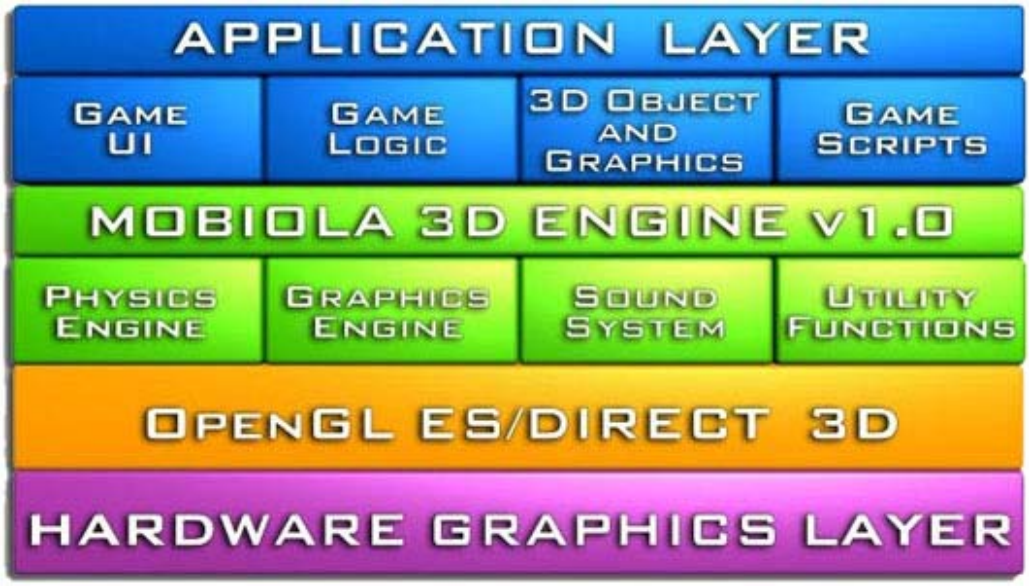
Component gebaseerde architectuur

Wanneer een architectuur, stukje bij stukje wordt opgebouwd, door elk stukje te kiezen uit een virtuele catalogus, dan resulteert dat tot een schonere en meer verdeelde architectuur. Deze virtuele catalogus is een verzameling van bestaande software componenten. Dit kunnen zelf ontwikkelde of derde partij componenten zijn. Zo bieden grote bedrijven als Sony (Playstation [1]) de mogelijkheid om derde partij *physics engines* (software component dat natuurkundige modellen simuleert zoals zwaartekracht en botsing) en ander componenten met hun SDK (Software Development Kit) te combineren.

Hard en Soft Architecture

Binnen de game architectuur kan er een onderscheid gemaakt worden tussen twee type architecturen. Zo bestaat de "fysieke" architectuur, zoals subsystemen die zorgen voor de interactie tussen computer hardware en de speler (de input/output architectuur). Dit is de *hard architecture*. Binnen een project wordt naar de *hard architecture* ook wel gerefereerd met de term *horizontal solution*, omdat de *hard architecture* een vrij generiek framework is (tenminste binnen een bepaalde platform), dat niet onbedoeld het type van een game vaststelt. Een voorbeeld van een component uit de *hard architecture* is een module die kan "spreken" met een aspect van DirectX, zoals de grafische kaart en geluidskaart. De *hard architecture* zorgt voor de koppeling met hardware onderdelen en is per definitie afhankelijk van de interfaces die geleverd worden bij deze onderdelen.

Tegenover de horizontal solution staat de *vertical solution*. Dit is de *soft architecture* van een game. Het gedeelte dat het spel opmaakt. Hoewel de *horizontal solution* en de *vertical solution* op bepaald punten elkaar overlappen, staan dezen over het algemeen tegenover elkaar. De *hard architecture* kan gezien worden als de drager van het spel (*soft architecture*). Het volgende plaatje geeft het contrast tussen *hard* en *soft architecture* weer.



Figuur 14: abstracte representatie van low-level componenten naar high-level componenten

[1] *http://www.playstation.com*

Op het hoogste niveau bevindt zich de *application layer*. Dit is *soft architecture*. De componenten op dit niveau zijn domein specifiek en kunnen over het algemeen niet hergebruikt worden in andere projecten. Enkele componenten zijn de *user interface* (menu's en opties van een spel) en de *game logic* (spel regels).

De *soft architecture* wordt effectief geïsoleerd van de buitenwereld, en zit bovenop de *hard architecture*, gebruikmakend van de diensten die zij aanbiedt. In het plaatje wordt dit aangegeven met "Mobiola 3D Engine v1.0" [1]. Dit is een commercieel product en biedt de *hard architecture* voor een spel. Zij bevat onder andere de volgende componenten: *physics engine*, *graphics engine* (zorgt voor de visuele weergave van een spel) en *sound system* (zorgt voor de geluidsweergave van een spel). *3D engine* en *game engine* zijn termen die gebruikt worden voor het aangeven van de *hard architecture*. Enkele bekende commerciële *3D engines* zijn: Unreal engine [2], Quake III engine [3] en Gambryo engine [4].

Op het laagste niveau bevinden zich de hardware onderdelen. Enkele onderdelen zijn: grafische kaart, geluidskaart en moederbord. Onder de leveranciers van deze onderdelen bestaan onder ander Nvidia [5], Creative [6] en Asus [7].

Herbruikbare componenten

Bij het bouwen van de architectuur wordt er gekeken of er gebruik gemaakt kan worden van reeds bestaande componenten. Door gebruik te maken van bestaande componenten wordt er veel tijd bespaard bij het ontwikkelen van een game. De indeling van de architectuur in componenten maakt hergebruik van code mogelijk. Een van deze componenten kan de *physics engine* zijn, welke deel uitmaakt van de game framework.

Het is eenvoudig te bedenken dat een *graphics engine* en een *sound system* kan worden hergebruikt, omdat deze zich op een laag niveau in de architectuur bevinden. Maar het is ook mogelijk om componenten van uit de *soft architecture* her te gebruiken. Hierbij kun je denken aan GUI's (*Graphical User Interface*), buttons en menusystemen.

In het geval dat er een nieuwe component ontwikkeld moet worden is het belangrijk om potentieel hergebruik te overwegen. De belangrijkste gelegenheid voor hergebruik vindt plaats op het niveau van de hardware interface en algemene algoritme. Tegenwoordig heeft het game platform geen grote invloed op de keuze voor hergebruik, omdat de meeste game platformen van tegenwoordig krachtig genoeg zijn, zodat game specifieke code niet in de hard architectuur componenten geplaatst hoeven te worden.

Des te krachtiger computers worden des te meer mogelijkheden er gevonden worden voor hergebruik binnen de architectuur, en de mogelijkheden voor een complete *soft architecture*.

---

[1] *http://www.warelex.com/games/index.php*
[2] *http://www.unrealtechnology.com*
[3] *http://www.idsoftware.com*
[4] *http://www.emergent.net*
[5] *http://www.nvidia.com*
[6] *http://www.creative.com*
[7] *http://www.asus.com*

Game Engines

Zoals in de *hard* en *soft architecture* kopje is aangegeven, is een *game engine* de *hard architecture* voor een spel. *Game engines* spelen een grote rol in de ontwikkelingsduur van een spel. Een *game engine* is de softwarematige basis van een computerspel. Game engine is de verzamelnaam voor de modules die functionaliteiten aan het spel bieden. Elke module heeft een specifieke taak. Zo bestaat er bijvoorbeeld een *graphics engine* dat voor de grafische weergave van het spel zorgt. De modules kunnen over het algemeen hergebruikt worden in andere projecten. Sommige ontwikkelaars refereren met de term *engine* naar de complete *source code* van een spel.

Game engines kunnen onderling sterk van elkaar verschillen in complexiteit. Met de "Doom" engine is het bijvoorbeeld mogelijk om een nieuwe level binnen één of twee dagen af te ronden. Bij geavanceerde engines, zoals de "Quake III" engine, kan de ontwikkeling van een nieuwe level al snel meer dan twee weken in beslag nemen.

Figuur 15: links: level gemaakt met Doom engine – rechts: level gemaakt met Quake III engine

Hergebruik van engines

De kosten, om bij de ontwikkeling van een game een eigen *game engine* te bouwen, bedragen rond de 40 tot 70 procent van het totale ontwikkelingsbudget. Uiteraard betekent dit dat ook de duur van het project veel hoger ligt, dan wanneer een *game engine* van een derde partij gekocht wordt. Daarnaast zal het aantal testen die de eigen gebouwde engine ondergaat veel minder zijn dan een derde partij engine dat al voor verschillende projecten is gebruikt. Wanneer onderhoudt en ontwikkeling aan een derde partij wordt overgedragen, waarbij gewoonlijk meer dan één groep gebruik maakt van de engine, dan verzekerd dat de snelle ontwikkeling van een stabiele engine. Ook ondersteuning van nieuwste hardware wordt sneller toegevoegd. Door bij het begin van het project een volledige engine te nemen is het mogelijk om een gedeelte van de ontwikkeling-cycle in te korten, waardoor het team in een vroeg stadium aan de inhoud van het spel kan werken.

Het is van groot belang, voor een bedrijf met eigen engines, om deze uit te geven, om op de hoogte te blijven van de laatste standaarden.

4.2 Framework

Johnson [Joh97] en Riehle [Rie00] beschrijven een software framework als een herbruikbaar ontwerp voor een software systeem (of subsysteem). Een software framework kan support programma's, code libraries, scripting taal, of andere software bevatten, die de verschillende componenten van een software project bij elkaar brengt.

Frameworks voorkomen het besteden van tijd aan de ontwikkeling van componenten en onderdelen die al ontwikkeld zijn. Het voordeel van het gebruik van bestaande/commerciële frameworks, is dat deze door en door getest worden. De voordelen zijn gelijkwaardig aan de voordelen van commerciële engines. Het is niet zo dat zij allemaal perfect zijn. Net als elke software programma kunnen frameworks, welke als het ware een verzameling van software componenten zijn, bugs bevatten. Wanneer een bepaalde framework door veel groepen (over verschillende projecten) gebruikt wordt, dan worden onbekende bugs snel gevonden, en zullen updates voor nieuwe hardware support en bug fixes relatief snel uitgegeven worden. Dit verzekerd de stabiliteit van het framework.

Zo is DirectX [1] een veelgebruikt framework dat een verzameling van API's (Application Programming Interface) is, die het eenvoudig maakt voor programmeurs om computerspellen te maken op het Windows platform. Deze API's maakt het mogelijk om, uitgaande van een applicatie, te communiceren met een ander programma of onderdeel. Zo dient een onderdeel uit DirectX bijvoorbeeld als interface voor de videokaart in een computer. Hierdoor hoeft de programmeur niet de taal van de videokaart te kennen om iets op de monitor te doen verschijnen. DirectX abstraheert de programmeur van de onderliggende hardware communicatie. Naast DirectX Graphics bestaan ook andere onderdelen, waaronder: DirectSound, DirectShow en DirectPlay. Dit framework wordt regelmatig geüpdate met nieuwe functies, verbeteringen en hardware support.

Libraries

Libraries is een bibliotheek van software componenten, functies en routines. Net zo belangrijk als de code zelf is de documentatie. De documentatie legt uit hoe bepaalde softwarecomponenten en functies toegepast kunnen worden, en legt uit hoe zij werken. Zonder deze documentatie is het bijna onmogelijk om de code toe te passen bij verschillende projecten. Dit komt omdat de code door verschillende ontwikkelaars gebruikt moet kunnen worden. Dezen weten zonder documentatie niet wat de schrijver van een software component wil bereiken. Uiteraard is er voor het begrijpen van de documentatie enige voorkennis nodig. Deze voorkennis is voornamelijk te lenen uit objectgeoriënteerde programmeer technieken [Szy02].

Een bekent voorbeeld van een herbruikbaar software component is een array. Dit object is in de meeste programmeertalen en frameworks terug te vinden, maar kunnen onderling verschillend geïmplementeerd zijn. Zo kan de documentatie voor een array object uit het Adobe FLEX framework gevonden worden op de website van Adobe [2].

De bekende veelgebruikte Array object kent al veel functies die niet meer opnieuw geschreven hoeven te worden. Het is voor een programmeur/ontwikkelaar van belang om het framework waarmee gewerkt wordt goed te kennen, om op de hoogte te zijn van

[1] *http://www.microsoft.com/windows/directx/*
[2] *http://livedocs.adobe.com/flex/2/langref/Array.html*

bestaande componenten, en de mogelijkheden die zij met zich meebrengen. Een programmeur zonder goede kennis van een framework zal al snel overbodige code schrijven, simpelweg omdat hij niet weet dat dezelfde functies al in bestaande software componenten geïmplementeerd zijn.

Flex framework

Voor de proof of concept en prototypes maak ik gebruik van het Adobe Flex framework. Dit is een verzameling van technologieën, uitgegeven door Adobe Systems, voor de ontwikkeling van cross-platform RIA's (Rich Internet Application) die gebaseerd zijn op de eigenschappen van het Adobe Flash [1] platform. Flex biedt een workflow en programeer model waar de animatie georiënteerde Flash programmeur mee vertrouwd is. MXML, een XML mark-up taal, biedt een manier aan om GUI's (*Graphical User Interface*) te maken. Interactiviteit wordt bereikt door middel van Actionscript, welke de kern taal van de Flash Player is.

De Flex SDK komt met een set van interface componenten waaronder: buttons, list boxes, trees, data grids, verscheidene tekst controls, and meerdere layout containers. Enkele interactie features die het framework biedt zijn: drag & drop, animatie effecten, en formulier validatie.

Soft architectuur

Een programmeur dat gebruikt maakt van het Flex Framework kan zich compleet concentreren op de *soft architecture* van zijn applicatie, omdat de *hard architecture* in de Flash Player verwerkt zit. Met de Flash Player is het mogelijk om Flex applicaties af te spelen. Adobe biedt deze Flash Player voor verschillende platformen, waarin voor elk platform de *hard architecture* verwerkt zit. Een programmeur kan zich daarom compleet concentreren op de inhoud van zijn applicatie.

High-level & low-level componenten

De aanwezige software componenten binnen een framework bieden abstractie van onderliggende code. De mate van abstractie kan per component verschillen. Dit maakt het verschil tussen high-level en low-level componenten.

Low-level componenten zijn bijna doorzichtig. Hiermee wordt bedoeld dat de onderliggende code niet diep verstopt is. High-level componenten daarentegen, zijn makkelijker te interpreteren op een logische manier. Een button uit het Flex framework is redelijk high-level. Op een logische manier kan dit geïnterpreteerd worden als een knop met een label (tekst) en een actie, dat uitgevoerd wordt nadat er op de knop is geklikt.

Wat er geabstraheerd wordt is dat deze button uit meerdere componenten van lagere niveaus is samengesteld. Zo bestaat de button onder andere uit een grafische representatie van de knop, een grafische representatie van waneer de muis over button wordt geplaatst (dus niet geklikt), een grafische representatie wanneer de button ingedrukt is, een label, dat een object van het type String bevat, etc. Figuur 16 geeft een standaard Flex Button (ongewijzigd) in verschillende situaties (muis over button, button ingedrukt, etc.) weer. In Flex wordt de button op een hiërarchische manier samengesteld. Exacte details zijn terug te vinden op de website van Adobe [2].

[1] *http://www.adobe.com/products/flash/*
[2] *http://livedocs.adobe.com/flex/2/langref/mx/controls/Button.html*

Figuur 16: default Flex button

Complexiteit

Het complete plaatje van de Button component is tamelijk complex. De complexiteit brengt met zich mee dat er veel verschillende representaties en interacties van de button mogelijk zijn. Zo kan het uiterlijk en gedrag van de button compleet worden veranderd. In het plaatje hieronder zijn twee aangepaste Flex Buttons (“Yes” en “No, Thanks”) te zien. De grafische representatie van de button en het lettertype van de label zijn gewijzigd.

Figuur 17: custom flex button

Het niveau van complexiteit wordt vastgesteld door de ontwikkelaar van het software component. Deze kan voor de button vaststellen dat de grafische representatie en gedrag onveranderd blijft. Het enige wat ingesteld kan worden is een label en een actie. Het voordeel van vereenvoudiging van high-level componenten is dat applicaties sneller ontwikkeld kunnen worden, maar het nadeel is dat de applicaties, die met de relatief simpele framework gebouwd zijn, er allemaal hetzelfde uit zien en erg gelimiteerd zijn. Dit verschijnsel is terug te vinden in *3D engine* dat gebruikt is voor het spel Doom [1].

Figuur 18: van links naar rechts: Doom (origineel), ZDoom en CounterDoom

In het geval van het Flex framework geldt dat des te hoger het niveau van een component, des te hoger de complexiteit van de component. De complexiteit is te meten aan het aantal functies en eigenschappen dat de component heeft. Omdat een high-level component opgebouwd is uit meerdere low-level componenten, brengt het alle functies van de low-level componenten met zich mee.

Met hoge complexiteit wordt niet bedoeld dat het moeilijker is om high-level componenten toe te passen, in contrast met componenten van een lager niveau. Maar wel dat er meer mogelijkheden bestaan. Afhangend van de verlangens van de programmeur kan de toepassing van high-level componenten moeilijker gemaakt worden.

4.3 Tools

Naast het gebruik van herbruikbare software componenten wordt er gebruik gemaakt van specifieke tools die hulp bieden bij de creatie van de inhoud van een spel. Deze tools maken gebruik van de functies die een *game engine/hard architecture* biedt om spel specifieke inhoud te creëren. Hierbij kan gedacht worden aan de creatie van nieuwe levels (speelwereld), karakters en spelregels. De tools hebben alleen invloed op de *soft architecture* en laten de *hard architecture* ongewijzigd.

Mods

Mods, kort voor “modification”, zijn “add-ons” of aanpassingen van een bestaand spel. Mods zijn populair geworden door de open-architectuur van “id Software”, welke haar spelers de mogelijkheid gaf tot het creëren van eigen levels van hun spel “Doom”. Naast nieuwe levels geven bepaalde mods de mogelijk om nieuwe AI, art, verhaal, 3D modellen, etc, in het spel te introduceren. Spellen die de gebruiker een grote schaal van mogelijkheden bieden voor het maken van aanpassingen, kunnen zo aangepast worden

[1] *http://www.idsoftware.com/games/doom/doom-final/*

dat deze een compleet ander spel ervaring oplevert dan het origineel. Dit soort mods worden geleverd als een compleet losstaande game.

De ontwikkelingsduur van een mod kan verschillen. Wanneer de aanpassingen simpel zijn, kan de bouw van een mod binnen enkele dagen afgerond zijn. Bij complexe aanpassingen en ontwikkeling van nieuwe modellen en textures, kan het ontwikkelingsproces uitlopen in jaren. Een bekend voorbeeld is de “Star Wars Quake” mod, welke ondanks zes jaar ontwikkeling nooit is uitgebracht.

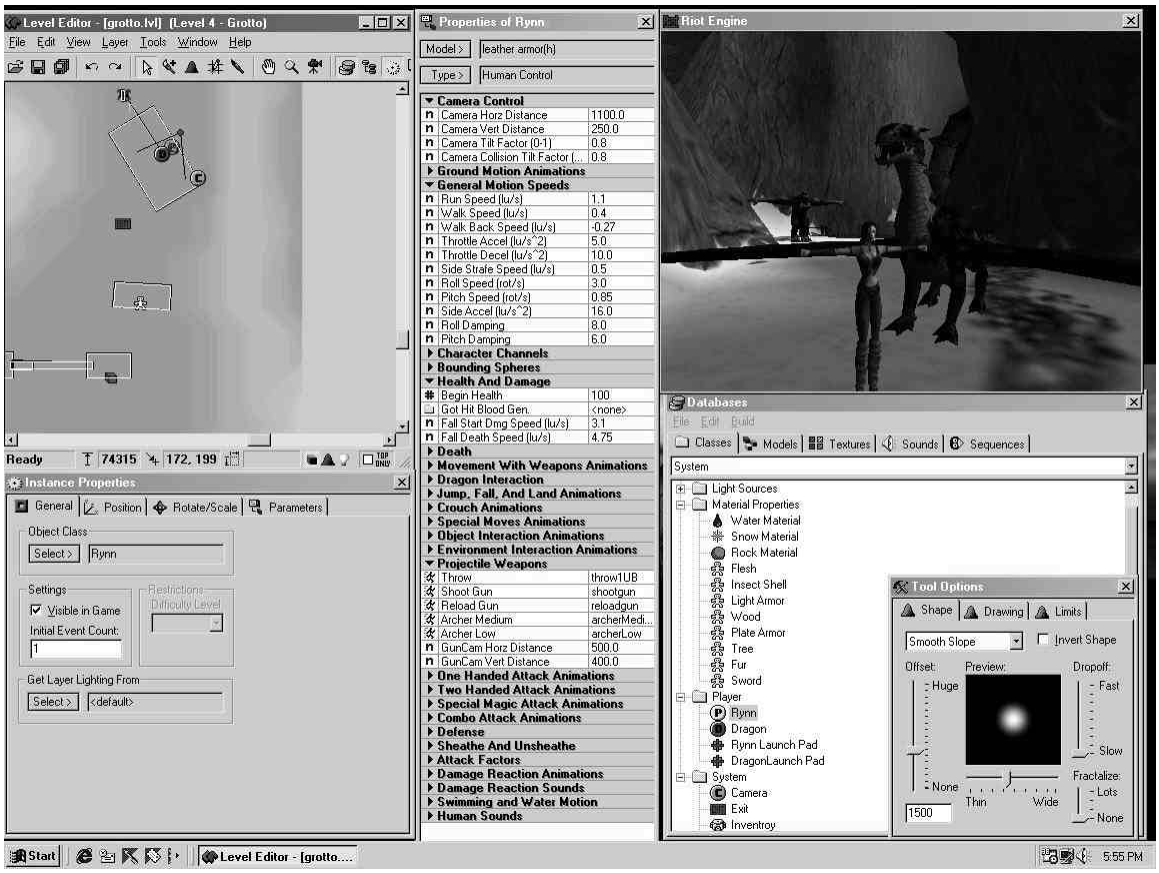
Script (engine) taal (level editor)

Het is de norm geworden om bij de ontwikkeling van games een systeem te gebruiken waar game designers, objecten en gedrag kunnen aanpassen, zonder raadplegen van een programmeur. Deze systemen worden ook wel *level editors* genoemd. Veel spellen bevatten script talen, die relatief simpel zijn en de mogelijkheid bieden om complexe entiteiten te creëren zonder dat daarvoor de game engine opnieuw gecompileerd hoeft te worden.

Gebruik van scripting brengt een aantal voordelen met zich mee. Het meest belangrijke is dat het de aanbieding van unieke ervaringen in het spel stimuleert. Een ander groot voordeel is dat het compleet overdraagbaar is naar andere systemen. Dit betekent, wanneer een PC spel gepoort wordt naar de Playstation, dat alle gedragselementen, AI, etc die op de PC gescript zijn net zo functioneel zijn op de Playstation versie, uitgaande dat de script interpreter en haar functies goed gepoort zijn.

Dit betekent dat een robuust script taal een stabielere werkomgeving biedt dan bijvoorbeeld het programmeren in C (programmeertaal). De script taal geeft de scriptschrijver minder kans om het systeem te crashen. Een script taal is gewoonlijk minder complex dan een programmeertaal als C, waardoor designers met een beetje programmeerkennis de creatie van unieke spelervaringen over kunnen nemen. Een goed ontworpen en krachtig script systeem geeft gemotiveerde spelers de mogelijkheid om complexe aanpassingen te maken en hun eigen zogenaamde “mods” te creëren.

Naast voordelen kennen scriptsystemen ook een aantal nadelen. Het kost namelijk erg veel tijd om een stabiel en krachtig script systeem te bouwen. Meestal worden scripts gedurende “run-time” gecompileerd waardoor dezen significant langzamer zijn dan C/C++. Een van de voordelen van een script systeem is dat niet-programmeurs er mee kunnen werken, maar wanneer het script systeem krachtig genoeg is om AI te implementeren, komt het er vaak op neer dat een programmeur dit moet doen.



Figuur 19: Surreal Software Riot Engine Level Editor

In het plaatje hierboven is de “Surreal Software Riot Engine Level Editor” te zien. Naast het plaatsen van objecten in een ruimte, geeft de *editor* ook de mogelijkheid om AI, gedrag, etc aan te passen, waardoor dit niet alleen een *level editor* is, maar ook als het ware een *gameplay editor*.

Andere tools

Bepaalde bedrijven specialiseren zich op de ontwikkeling van tools, die de creatie van spel elementen uit de *soft architecture* faciliteren. Naast uitgebreide *level editors*, bestaan er onder andere 3D modeling tools (tools voor de ontwikkeling van complexe 3D objecten) zoals “3ds Max” [1] en “Maya” [2], animatie tools (tools die zorgen voor de animatie van 3D objecten) zoals “Poser 3D” [3] en g*ame user interface* tools (simplificeert de constructie van GUI’s) zoals “libRocket” [4].

[1] *http://usa.autodesk.com/adsk/servlet/index?id=5659302&siteID=123112*
[2] *http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7635018*
[3] *http://my.smithmicro.com/win/poser/index.html*
[4] *http://www.librocket.com*

4.4 Conclusie

Mogelijkheden voor hergebruik van software vinden op twee niveaus plaats. In de *hard architecture* en de *soft architecture*. Componenten uit de *hard architecture* hebben geen bepalende invloed op de inhoud van het spel en zijn om deze reden toepasbaar over verschillende projecten. De verschillende componenten zorgen onder ander voor de visuele weergave, de weergave van geluid en simulatie natuurlijke modellen (zwaartekracht, botsingen, etc.). Deze componenten kunnen zelf worden ontwikkeld of ingekocht worden bij een derde partij. Commerciële bedrijven bieden ook complete *hard architecture* oplossingen aan. Zij worden 3D engines of game engines genoemd en bevatten alle benodigde componenten uit de *hard architecture*.

Componenten uit de *soft architecture* bepalen de inhoud van het spel. Hergebruik binnen de *soft architecture* vindt voornamelijk plaats over een game van gelijke soort. Als voorbeeld kan er gekeken worden naar de "Zelda [1]" serie. Bij het vergelijken van drie titels uit de serie, namelijk "Ocarina of Time", "The Wind Waker" en "Twilight Princess", zijn grote overeenkomsten op te merken. De drie titels zijn ontwikkeld voor drie verschillende platformen. Overeenkomsten zijn te vinden in gameplay, doorlopen van de wereld, en opbouw van het spel (levels, missies, etc).

Figuur 20: beelden uit de "Zelda" serie
Van links naar rechts: "Ocarina of Time", "The Wind Waker" en "Twilight Princess"

In deze serie is duidelijk hergebruik op software niveau te zien. Dit is gelimiteerd tot de game serie. Het is mogelijk om componenten uit de *soft architecture* in een andere game serie toe te passen, maar dit is over het algemeen niet wenselijk. Het spel lijdt aan originaliteit wat teleurstelling bij de speler tot gevolg kan hebben.

Hergebruik op software niveau is vooral terug te vinden in tools waarmee componenten uit de *soft architecture* ontwikkeld kunnen worden. Deze kunnen net als componenten uit de *hard architecture* gebruikt worden in verschillende projecten. Zij faciliteren onder ander de ontwikkeling van nieuwe levels (level editor) en 3D modellen (3D modeling tools).

[1] *http://www.zelda.com/*

5 Teamwork

Om het concept van teamwork herbruikbaar op te stellen voor verschillende games is een brede kennis van teamwork noodzakelijk. Dit hoofdstuk beschrijft wat teamwork is en beschrijft een model/prototype dat een teamworkwaarde kan representeren.

Wat is teamwork?
Harris en Harris [HH96] geven de volgende definitie van teamwork:

"Teamwork is een werkgroep of eenheid met een gemeenschappelijk doel waarbij deelnemers wederzijdse relaties opbouwen om doelen te bereiken en taken te verrichten".

Voor teamwork is een werkgroep of eenheid nodig. Of anders, een team. Katzenbach en Smith [KS93] geven de volgende definitie van een team:

"Een team is een kleine groep mensen, met complementaire vaardigheden, die zich hebben toevertrouwd tot een gemeenschappelijk doel, prestatie doelen en benadering waar zij zelf wederzijdse verantwoordelijkheid voor dragen".

Samenwerken (teamwork) versus Werken in een groep
Er bestaan een aantal verschillen tussen samenwerken en werken in een werkgroep. Hieronder worden de karakteristieken van de twee opgesomd.

<u>Werken in een werkgroep</u>

- Lineair werken, individueel taakgericht.
- Werkers presteren binnen hun eigen verantwoordelijkheid.
- Interactie leidt de individuele prestatie van elk teamlid af.

Figuur 21: grote callcenter

Werk in een callcenter is hier een voorbeeld van. Het plaatje hierboven laat zien hoe een standaard callcenter er uit ziet. De werknemers zijn in hokjes van elkaar gescheiden. Het werk is individueel taakgericht. Afleiding zorgt voor prestatie vermindering (een werknemer kan bijvoorbeeld minder *calls* aannemen).

Samenwerken

- De focus ligt op gezamenlijke prestatie voor het behalen van doelen.
- Levert producten op die alleen gezamenlijk gerealiseerd kunnen worden.
- Gezamenlijke verantwoording
- Toegewijd aan een gedeeld doel en aanpak.
- Bij hoog presterende teams: toegewijd aan elkaars persoonlijke groei en succes.

Figuur 22: samenwerking

Samenwerking brengt verschillende disciplines samen om gezamenlijk één product te ontwikkelen. Dit is bijvoorbeeld te zien in *game development.*

De karakteristieken van beide vormen van werken in groepsverband kun je terug vinden als je naar het concept teamwork kijkt. Het doel van teamwork is om samen te werken, maar wanneer het team niet goed gemanaged wordt, beginnen zich al snel werkgroep karakteristieken te vertonen. Soms is een balans tussen beide vormen gewenst.

In hoeverre heeft goed teamwork invloed op prestatie?

Door goede samenwerking worden producten gecreëerd die alleen gezamenlijk gerealiseerd kunnen worden, waardoor zij meer waarde hebben dan producten als gevolg van individuele inspanning.

Producten worden gebouwd aan de hand van productbeschrijvingen (requirements). Bij het bouwen van een product worden de teamleden geplaatst bij verschillende ontwikkelingsprocessen. Wanneer de teamleden elkaar goed helpen en aanvullen, kunnen deze processen sneller lopen en kan er een product geleverd worden van hoge kwaliteit.

De prestatie van individuele inspanning is lineair, terwijl dit bij goed teamwork exponentieel groeit.

Wat zijn de karakter eigenschappen van teams met hoge prestatie?

De prestatie van een team wordt beoordeeld aan de hand van kwaliteit en snelheid van productie. Volgens Nash en Bolin [NB03] zijn bij een vergelijking tussen een hoog presterend team en een laag presterend team, duidelijke verschillen te vinden in de volgende aspecten.

- Samenhangende Strategie
- Duidelijke Rollen en Verantwoordelijkheden
- Open Communicatie
- Snelle Reactie
- Effectieve Leiderschap

Wat zijn de vereisten voor het vormen van een effectief (goed teamwork) team?

Voor het bereiken van effectief teamwork, zijn volgens Nash en Bolin [NB03] de volgende punten vereist.

- Klein aantal mensen: Het optimale aantal mensen in een team ligt algemeen tussen de vijf en negen. Meerdere teamleden brengen grotere verscheidenheid in perspectief en ideeën, maar de moeilijkheid om gemeenschappelijke beslissingen te nemen stijgt dramatisch.
- Complementaire vaardigheden: Bij het opstellen van een team is het belangrijk om een mix van verschillende maar complementaire vaardigheden te hebben.
- Toevertrouwing tot een gemeenschappelijk doel: Zonder een gemeenschappelijk doel heeft het team geen meetlat om haar prestaties te meten.
- Gemeenschappelijk prestatie doel: Teams delen prestatie doelen, of doelen. Als een doel niet behaald wordt, is het hele team verantwoordelijk. Toevertrouwing tot deze gemeenschappelijke prestatie doelen leidt tot hogere productiviteit en een verhoogde motivatie.
- Gemeenschappelijke benadering: Doelen representeren het taak element voor een succesvolle prestatie. Een gemeenschappelijke benadering representeert het groepsproces element van samenwerking. Zonder af te spreken hoe de interactie in het team verloopt, zullen de kansen om een taak af te ronden erg laag zijn.
- Gemeenschappelijke verantwoording: Gemeenschappelijke verantwoording verwijst naar gedeelde ownership en verantwoordelijkheid, welke fundamenteel zijn voor echt teamwork. Als iets verkeerds gaat zou er niet met vingers gewezen moeten worden, maar in plaats daar van spant de groep zich in om het probleem op te lossen.

Welke vaardigheden heeft elke deelnemer van een team nodig voor succesvol teamwork?

Er zijn zeven essentiële vaardigheden die iedereen moet leren om succesvol samen te kunnen werken. De vaardigheden zijn:

- Luisteren: het is belangrijk om te luisteren naar ideeën van andere mensen. Als mensen vrij zijn hun ideeën te uiten, dan zullen deze ideeën andere ideeën produceren.
- Vragen: het is belangrijk om vragen te stellen en te overleggen om de doelstellingen van het team te bereiken.
- Overtuig: individuen worden aangemoedigd om hun ideeën uit te leggen, te verdedigen en uiteindelijk hun ideeën aan te passen.
- Respecteer: Het is belangrijk om andere ideeën te respecteren en te ondersteunen.
- Help: het is essentieel om elkaar te helpen, dit is dan ook het algemene idee van teamwork.
- Delen: het is belangrijk om informatie te delen met het team om zo een omgeving van teamwork te creëren.
- Deelnemen: iedereen wordt aangemoedigd om deel te nemen in het team.

Zichtbare vaardigheden en emotionele intelligentie

Volgens Luca en Tarricone [LT01] zijn bij het kiezen van teamleden niet alleen zichtbare vaardigheden van belang, maar ook vaardigheden die niet zo zichtbaar (emotionele intelligentie) zijn. Bij zichtbare vaardigheden kun je denken aan fysieke kracht of programeer vaardigheden.

Er bestaat een sterke samenhang tussen emotionele intelligentie en harmonie van een team. Emotionele intelligentie bestaat uit vijf hoofdzakelijke elementen: zelfbewustzijn, zelfregulering, empathie, motivatie en sociale vaardigheden, welke moeilijk te testen zijn en niet zo zichtbaar zijn als bijvoorbeeld technische vaardigheden.

- Zelfbewustzijn is de bekwaamheid om doormiddel van zelfreflectie eigen gevoelens te begrijpen en te interpreteren
- Zelfregulering is de bekwaamheid om emoties te gebruiken om de vooruitgang van een taak of project te faciliteren (E. G. Lanser).
- De bekwaamheid om teamgenoten te **motiveren** hun best te doen is een krachtige vaardigheid. Werkers presteren, wanneer zij het gevoel hebben dat zij gesteund en geïnspireerd worden.
- Empathie is het inlevingsvermogen van gevoelens van collega's en zich kunnen identificeren met hun gevoelens en problemen door het begrijpen van het perspectief van mensen met een ander manier van leven als de eigen (D. Goleman).
- Sociale vaardigheden zijn vaardigheden die je nodig hebt om met andere mensen om te gaan, samen te leven en te werken.

Zij zouden te toetsen zijn met een vragenlijst, maar het resultaat is niet altijd correct. Zij kan niet naar waarheid ingevuld zijn, maar het is moeilijk om daar achter te komen. Toetsing van wiskunde weerspiegelt daarentegen de wiskunde kennis wel correct.

Relaties/Afhankelijkheid

Sociale afhankelijkheid verwijst naar hoe individuen met elkaar omgaan bij coöperatieve leer of werksituaties.

Volgens Johnson en Johnson [JJ95] [JJ99] heeft positieve afhankelijkheid goede interactie tot gevolg waarin individuen worden aangemoedigd om elkaars inspanning te faciliteren om doelen van het team, zoals het creëren van positieve relaties en gezamenlijke teamomgevingen, te bereiken. Negatieve afhankelijkheid of competitie resulteert over het algemeen in tegenstrijdige interactie.

Johnson en Johnson [JJ99] geven een lijst van essentiële attributen van positieve afhankelijkheid die nodig zijn voor succesvol teamwork:

- Het geven en ontvangen van hulp en assistentie voor taak gerelateerde en persoonlijke kwesties.
- Delen van hulpbronnen en informatie.
- Het geven en ontvangen van feedback over taken en teamwork gedrag.
- Elkaars redeneringen reviewen.
- Anderen aanmoedigen hun doelen te bereiken.
- Elkaars redenering en gedrag beïnvloeden.
- Gebruik maken van sociale vaardigheden om teamwork te versterken
- De effectiviteit van het team bewust weerspiegelen om verbetering te bevorderen en prestaties te erkennen.

Teams die om elkaar geven op een persoonlijk en professioneel niveau hebben meer kans succesvol te zijn dan teams die het belang van relatie tussen positieve inter-persoonlijke relaties, professionele relaties en doel prestatie, negeren.

Leiderschap

Elk team heeft een leider nodig die zorgt voor orde en organisatie. Nash en Bolin [NB03] sommen de taken en verantwoordelijkheden van een teamleider op.

- <u>Het doel bereiken</u>

  Leiden waar nodig is. Teamleiders leiden het team wanneer zij vast komen te zitten, maar zij laten ander leden toe om activiteiten te regisseren afhankelijk van het type project waar het team mee bezig is.

  Evalueren en tussenkomen waar nodig: Deel van de verantwoordelijkheid van een leider is om de totale verantwoording van het team te sturen, door het sturen van logistiek, lobbyen voor hulpmiddelen, communiceren met de organisatie over de activiteiten van het team, en het verwijderen van obstakels in het pad van het team.

- Mensen ontwikkelen
  Waarderen van teamleden. Het geven van feedback is een waardevolle tool om het team moraal en de verbondenheid van het team te verhogen. Deze waardering omvat ontwikkeling en positieve feedback.
  Individuele ontwikkelen door te trainen: Training omvat sturing van elk teamlid door het creëren van een dialoog over wat die persoon moet leren om zijn of haar huidige prestatie niveau te verhogen.
- Team bouwen
  Het team opstarten: Het gebruiken van teamopbouw technieken en oefeningen gedurende verschillende stadia van team ontwikkeling kan team prestatie motiveren en inspireren.
  Versterk resultaten: Een kritisch onderdeel in het bouwen van het team is het belonen en erkennen van voorbeeldige prestatie.

Team conflict

Conflicten binnen een team zijn onvermijdelijk. Deze conflicten resulteren in bepaalde reacties. Na een conflict kan een groep geclassificeerd worden als een winnaar of verliezer.

Sherif et al. [SHW+61] geven de processen aan die gebeuren in geval van winnen of verliezen van conflicten binnen een groep.

Winnaar

- Toename in groep samenhang.
- Verminderde serieusheid.
- Groep samenwerking (meer sociaal dan taakgericht).
- Versterkt zelf imago.

Verliezer

- Negeren, vervormen, ontkennen.
- Interne strijd en versplintering.
- Harder werken, toename in stress (meer taakgericht dan sociaal).
- Leerervaring -> reorganisatie -> meer samenhang en efficiëntie.

Het is ook van belang om te kijken naar de relaties die ontstaan met omringende groepen. De lijst hieronder geeft de reacties weer binnen een groep en tussen groepen die in conflict zijn.

Binnen een groep

- Toename in groepssamenhang.
- Toename in loyaliteit.
- Meer taakgericht.
- Autoritair in plaats van democratisch leiderschap.
- Structuur/organisatie
- Toename in interne/sociale druk

Tussen groepen

- Andere groep wordt gezien als de vijand.
- Toename in sociale complexiteit.
- Vervormd perspectief.
- Toename van conflict onderwerpen.
- Minder interactie.
- Stereotypering
- Pessimistische verwachting

Wanneer deze processen binnen groepen plaatsvinden is een reactie daarop onvermijdelijk. Jansen [Jan02] maakt een onderscheid tussen vijf type reacties of houdingen bij conflicten.

- Aanval/competeren.
- Samenwerken/oplossen.
- Overeenkomen/overeenkomst.
- Vermijden/verdedigen.
- Aanpassen/toegeven.

5.1 Prototype: Teamwork score

In één van de vertalingen van teamwork naar games kan teamwork gezien worden als een waarde. Deze waarde geeft aan hoe goed een team samenwerkt. Uit literatuur is op te maken dat persoonlijke vaardigheden en onderlinge relaties invloed hebben op samenwerking. Met dit gegeven als basis heb ik een model ontworpen dat in staat is om een teamworkwaarde te berekenen.

Om beter te begrijpen hoe het prototype in elkaar zit en welke aannames gemaakt worden, is het van belang om uit te leggen wat de relatie is tussen een speler en een spelkarakter en hoe teamwork tot stand komt.

Game AI

In de breedste zin bevatten de meeste games enige vorm van Artificial Intelligence (AI). Spelontwikkelaars hebben over de jaren AI gebruikt om ogenschijnlijk intelligent leven te wekken in talloze game karakters. Van “Pac Man” [1] tot bots (AI agents) in first person shooters als “Halo” [2]. De grote variëteit in game genres en game karakters heeft een brede interpretatie nodig van wat onder game AI wordt verstaan.

In spellen is men niet altijd geïnteresseerd om een niet-speelbaar karakter intelligentie van menselijk niveau te geven. In sommige gevallen wordt er code geschreven om niet-menselijke wezens zoals draken en robots te controleren. Daarnaast hoeven niet-speelbare karakters niet per se slim te zijn. Game AI wordt vaak ingeschakeld voor het oplossen van redelijke complexe problemen. Zo kan AI gebruikt worden om niet-

[1] *http://nl.wikipedia.org/wiki/Pac-man*
[2] *http://www.xbox.com/en-US/games/h/halo3/hub/default.htm*

speelbare karakters verschillende persoonlijkheden te geven of om emoties weer te geven zoals angst en blijdschap [BS04].

AI wordt door “The American Heritage Dictionary of the English Language, Fourth Edition (Houghton Mifflin Company)” als volgt beschreven:

“*De bekwaamheid van een computer of andere machine om activiteiten uit te voeren waar intelligentie voor vereist is.*”

Andere bronnen definiëren AI als het proces of wetenschap om intelligente machines te creëren.

In essentie is de definitie van game AI nogal breed en flexibel. Volgens Bourg en Seeman [BS04] kan alles dat de illusie van intelligentie op een geschikt niveau opwekt, wat games meer diepgang, uitdaging en plezier geeft, als game AI beschreven worden.

Agent AI

Agent AI gaat over het produceren van autonome karakters die spel informatie gebruiken om te beslissen welke acties er genomen moeten worden en voeren deze acties vervolgens uit [Mil06].

Er bestaan twee verschillende vormen van AI gedrag, namelijk deterministisch en niet-deterministisch [BS04].

Deterministisch

Deterministisch gedrag of prestatie is gespecificeerd en voorspelbaar. Er bestaat geen onzekerheid. Een voorbeeld is het volg algoritme. In het spel “Final Fantasy VIII” [1] is hier een voorbeeld van te vinden. In dit spel bestuurt de speler één tot drie karakters. Wanneer de spelwereld doorlopen wordt bestuurd de speler één karakter en de andere karakters volgen.

Figuur 23: volg algoritme in Final Fantasy VIII

[1] *http://www.final-fantasyviii.com*

Niet-deterministisch

Niet-deterministisch gedrag is het tegenovergestelde van deterministisch gedrag. Gedrag heeft een graad van onzekerheid en is (gedeeltelijk) onvoorspelbaar (graad van onvoorspelbaarheid hangt af van de gebruikte AI methode en hoe goed de methode wordt begrepen). Een voorbeeld is te vinden in het *action adventure RPG* "Outcast" [1] waarin vijanden zich aanpassen aan de gegeven situatie. Zij gebruiken gehoor, zicht en acrobatiek om bepaalde taken op te lossen. Zij kunnen zelfs samenwerken bij het uitvoeren van complexe taken.

Figuur 24: niet-deterministisch gedrag van vijanden in Outcast

Relatie tussen AI agent en gamer

Een gamer (persoon die een spel speelt) bestuurt altijd één of meer karakters. De acties die uitgevoerd kunnen worden met de karakters zijn gelimiteerd aan haar capaciteiten, vaardigheden en karaktereigenschappen. Dit staat los van de capaciteiten, vaardigheden en karaktereigenschappen van de gamer. Een gamer beslist wel zelf wat hij met het karakter wil doen. Een voorbeeld is te vinden in het voetbalspel "FIFA 08" [2].

Figuur 25: schot op doel in FIFA 08

[1] *http://www.outcast-thegame.com*
[2] *http://fifa08.ea.com*

Een gamer kan zelf bepalen waar hij de voetbalspeler wil positioneren, wanneer hij een tackle wil uitvoeren en wanneer hij een schot op doel wil wagen.

Op niveau van het karakter (voetbalspeler, basketbalspeler, etc.), betreffen AI beslissingen persoonlijke tactische (of niet tactische) gedragspatronen die alleen het karakter betrekt [Sch04]. In voetbal zijn voorbeelden: het maken van een schijnbeweging om vrij te komen en in de verdediging aan het shirtje trekken.

Dit soort beslissingen en gedragsvertoningen houden rekening met persoonlijke attributen van het karakter, als reflectie van zijn *real-life* tegenhanger (als deze bestaat). Door het AI gedrag te modeleren aan de hand van reële statistieken, voelt de speler alsof hij met een karakter speelt met evenredige vaardigheden als die van de werkelijke persoon (voetballer, basketballer, etc.).

Op deze manier bevat de AI van sport spellen (en andere game genres) een groot simulatie element om te vermijden dat iedereen een superheld is. In plaats daarvan missen personen, die slecht zijn in het geven van een pass, vaker en slechte verdedigers zijn relatief makkelijk voorbij te lopen.

Hoe wordt de gamer betrokken bij teamwork in een spel?

In het vormen van teamwork zijn de beslissingen van de gamer van belang. De gamer wordt op verschillende manieren in een game betrokken bij de vorming van teamwork.

Teamwork bestaat in spellen waar de gamer samenspeelt met één of meer spelers. Hier zijn de persoonlijke vaardigheden (communicatie, strategie, etc.) van de gamer van belang. Dit soort teamwork is onder andere terug te vinden in *first person shooters* (FPS), *massive multiplayer online role playing games* (MMORPG) en teamsport games (bijvoorbeeld voetbal en basketbal). Het plaatje hieronder illustreert de samenwerking tussen gamers in de FPS “Counter-Strike” [1].

Figuur 26: teamwork tussen gamers in Counter-Strike

Een ander manier waarbij de gamer betrokken raakt bij teamwork in een spel is een spel waar een gamer een team van AI agents beïnvloed. De gamer bestuurt alle karakters, die elk verschillende eigenschappen en vaardigheden hebben, van een team. De gamer gebruikt de capaciteiten van de karakters om de beste teamprestatie neer te zetten. Dit is terug te vinden in RPG’s en manager spellen zoals “The Sims” [2], “The Movies” [1] en

[1] *http://www.counter-strike.net*
[2] *http://thesims.ea.com*

"FIFA Manager 08" [2]. Het volgende plaatje komt uit de "Final Fantasy X" [3] game en geeft een vecht scène weer waar de gamer alle karakters om de beurt bestuurd.

Figuur 27: gamer bestuurt verschillende karakters van een team in Final Fantasy X

In andere spellen werkt de gamer actief samen met AI agents. De gamer bestuurt één karakter en wordt gesteund door één of meer AI agent(s). Dit is vooral terug te vinden in de *single player* optie van een FPS zoals "Halo 3" [4].

Figuur 28: gamer werkt samen met AI agents in Halo 3

[1] *http://www.lionhead.com/themovies/*
[2] *http://en.wikipedia.org/wiki/FIFA_Manager_08*
[3] *http://www.ffx-europe.com*
[4] *http://www.xbox.com/en-US/games/h/halo3/*

Spellen die meerdere vormen adapteren bestaan ook. Dit is bijvoorbeeld te zien in sport spellen als voetbal en basketbal. Twee spelers kunnen samen één team besturen. Op een gegeven moment kunnen maximaal twee karakters worden bestuurd. De acties van de rest van het voetbal team (AI agents) is afhankelijk van de acties van de spelers en de spelsituatie (bijvoorbeeld aanval en verdediging).

Prototype aannames

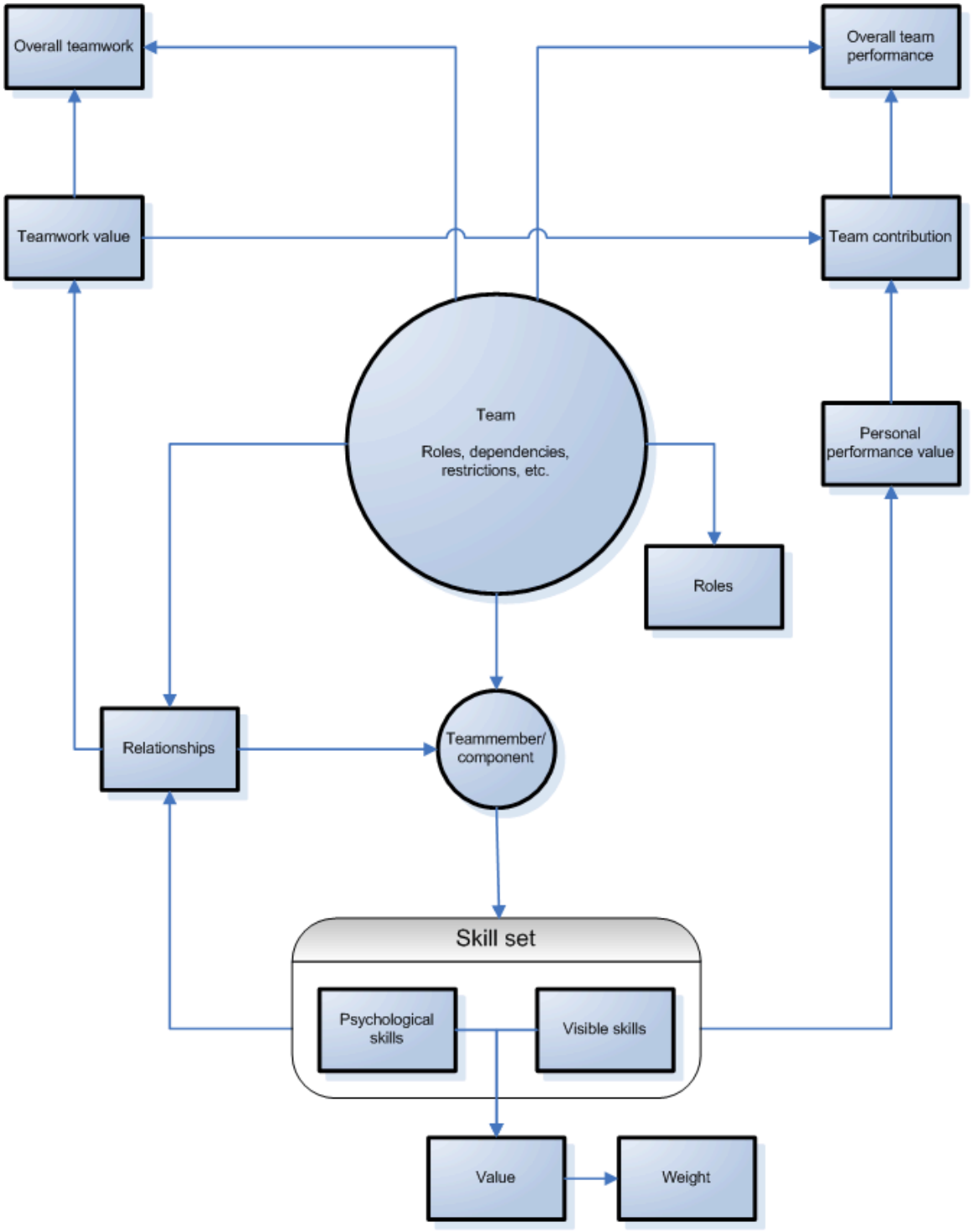
Het model, waar het prototype gebruik van maakt, is gebaseerd op het belang van persoonlijke vaardigheden en onderlinge relaties in een team. De vaardigheden en relaties van teamleden in het prototype staan los van de persoonlijke vaardigheden en relaties van de gamer zelf. Zij zijn de persoonlijke attributen van AI agents in het prototype en kunnen door de spelontwikkelaar worden ingesteld.

Het is de taak van de gamer om alle karakters te besturen/beïnvloeden om de beste teamprestatie neer te kunnen zetten, net zoals dat gebeurd in RPG's en manager spellen.

Gebruikte formules in dit model zijn zelf ontworpen met als doel de impact van teamwork op een simpele manier te illustreren.

Teamwork model

Hieronder is een model te zien dat aangeeft hoe er voor een team, een teamwork waarde wordt berekend.



Figuur 29: teamworkwaarde model

Team

Hierin staat de team definitie vast. Hierbij kun je denken aan het maximaal aantal leden dat een team kan hebben en welke rollen/afhankelijkheden er binnen het team bestaan.

Roles

Hierin staan de rollen gedefinieerd die teamleden binnen het team kunnen aannemen. Bij het definiëren van rollen kun je aangeven hoe sterk de impact van de rol binnen het team is. Een slechte leider heeft bijvoorbeeld een grotere impact op het team dan een slechte programmeur.

Teammember/component

Elk teamlid maakt deel uit van de teamcomponent en neemt een rol aan dat het teamcomponent aanbiedt.

Skill set

Elk teamlid heeft een set persoonlijke vaardigheden. Deze skill set hoeft niet voor elk teamlid gelijk te zijn, maar welke vaardigheden van belang zijn per teamlid, hangt wel af van de rol die het teamlid aanneemt. Zo zijn bij een programmeur andere vaardigheden van belang dan bij een designer. Iemand die een rol aanneemt dat niet bij zijn persoonlijke skill set past, zal slecht presteren, terwijl hij binnen een andere rol beter zou presteren. Met andere woorden: om een wezenlijke bijdrage te kunnen leveren aan het team moet de persoonlijke skill set zoveel mogelijk compatible zijn met de rol die het teamlid aanneemt.

De vaardigheden die een teamlid beschikt kunnen onderverdeeld worden over twee groepen.

Visible skills

Dit zijn vaardigheden die je kunt meten. Een voorbeeld hiervan is programmeer vaardigheid, welke te beoordelen is aan de hand van de ervaring van de programmeur met het bouwen van (complexe) systemen, of door toetsing.

Psychological skills

Dit zijn vaardigheden die je niet zo goed kunt meten. Een voorbeeld hiervan is empathie.

Elke vaardigheid die een teamlid beschikt kan de persoonlijke prestatie van het teamlid en/of de mate waarin het teamlid relaties met andere teamleden op kan bouwen, bepalen.

Value

De waardes die een vaardigheid kan aannemen wordt door de spelontwikkelaar ingesteld. Zo kan aan de programmeer vaardigheid van een teamlid (AI agent) een waarde tussen de -100 en 100 gegeven worden, waarbij een positieve waarde de prestatie en/of relatie positief beïnvloed, en een negatieve waarde de prestatie en/of relatie negatief beïnvloed. Het is ook mogelijk om een andere waardeschaal te hanteren. Bijvoorbeeld 0 tot 10 of -10 tot 10. De spelontwikkelaar bepaalt welke waarde schaal hij voor zijn spel wil hanteren.

Weight

Het zou handig zijn om bij elke vaardigheid dezelfde schaal te gebruiken (bijvoorbeeld -100 tot 100), maar bepaalde vaardigheden hebben meer invloed op de prestatie en/of relatie van bepaalde teamleden dan andere. Daarom is aan elke vaardigheidswaarde een gewicht verbonden, welke de mate van beïnvloeding, ten opzichte van andere vaardigheden, representeert.

Een zijn bijvoorbeeld twee teamleden genaamd Jan en Piet. Beide hebben dezelfde vaardigheden, maar hebben daar verschillende gewichten aan verbonden, zoals het volgende tabel laat zien.

	Programmeren		**Motivatie**	
	Value	Weight	Value	Weight
Jan	60	1	60	1
Piet	60	1	60	3

De motivatie van Jan en Piet kunnen door een bepaalde gebeurtenis voor beide spelers dalen naar een waarde 40. De impact hiervan is veel groter bij Piet. Met de gewichtwaarde is het mogelijk om de gevoeligheid van bepaalde vaardigheden voor elk teamlid ten opzichte van andere teamleden in te stellen. Dit is een manier om verschillende persoonlijkheden aan de teamleden (AI agents) te geven, zoals kalm, nerveus, onvriendelijk en vriendelijk.

Personal performance value

Door alle meetellende vaardigheden van een teamlid, die de prestatie bepalen, bij elkaar te nemen, kan een persoonlijke prestatie waarde gegenereerd worden.

Relationships

Aan de hand van de team definitie en rol van een teamlid, worden relaties opgebouwd met andere teamleden. De kwaliteit van de relaties hangt af van onderlinge vaardigheiden om goede relaties op te bouwen. Zo zullen twee personen met goede vaardigheden om relaties op te bouwen, sneller goed kunnen samenwerken dan anderen die daar minder goed in zijn.

Het opbouwen van relaties kost tijd, en groeit in momenten waar taken samen moeten worden gerealiseerd. Een voorbeeld kun je vinden in bestaande voetbal games. In een voetbalclub zie je voetbalspelers voortdurend komen en gaan. Binnen de club bestaan er meestal een aantal spelers die al heel lang in dezelfde club spelen. In die tijd hebben zij de mogelijkheid genoten om tijdens trainingen en wedstrijden goede relaties op te

bouwen met de rest van het team. Wanneer deze teamleden gedurende een wedstrijd in de basis staan, heeft dat een sterke positieve invloed op het teamwork van het team. Wanneer deze spelers ingevallen worden door nieuwelingen beïnvloedt dat de samenwerking negatief.

Teamwork value

Door alle relatie waardes van een teamlid bij elkaar te nemen kan een teamwork waarde gegenereerd worden.

Team contribution

Een goede teamwork waarde betekent dat het desbetreffende teamlid goede relaties heeft binnen het team. Met goed teamwork kan een teamlid met zijn persoonlijke prestatie een waardevolle contributie leveren aan het team. Door de teamwork waarde en de persoonlijke prestatie waarde samen te nemen, kan een team contributie waarde worden gegenereerd. Dit is in principe de prestatie dat een teamlid aan het team levert. Deze waarde zal in de buurt liggen van de persoonlijke prestatie waarde, welke positief of negatief beïnvloed is door de teamwork waarde.

Overall teamwork

Dit is de waarde die gegeneerd kan worden door de teamwork waardes van alle teamleden samen te nemen. Deze waarde representeert het teamwork van het team.

Overall team performance

Dit is de waarde die gegeneerd kan worden door de team contributie waardes van alle teamleden samen te nemen. Deze waarde representeert de prestatie van het team.

5.2 Voorbeeld: Game development team

Hieronder zal ik het model uit de vorige sectie met een simpel concreet voorbeeld illustreren.

Team: Game development team

In het team component wordt het team gedefinieerd. In mijn voorbeeld is het team een drie-persoons game development team.

Roles

Hier worden de rollen gedefinieerd met de vaardigheden die noodzakelijk zijn om goed te kunnen presteren binnen de rol en vaardigheden om relaties op te kunnen bouwen. De vaardigheden die in de rollen gedefinieerd worden zijn de enige vaardigheden die mee zullen spelen bij het genereren van een prestatie en relatie waarde. Mocht een teamlid één van de vaardigheden niet beschikken, dan heeft deze vaardigheid een waarde 0 en het teamlid zal deze gedurende het spel moeten ontwikkelen. Naast de vaardigheden wordt er ook een impact waarde aan de rol toegekend.

Rol	Impact	Prestatie skills	Relatie skills
Leider	2	Leiderschap Motivatie	Communicatie Motivatie
Designer	1	Ontwerpen Motivatie	Communicatie Motivatie
Programmeur	1	Programmeren Motivatie	Communicatie Motivatie

Opvallend in deze tabel is dat de motivatie vaardigheid gebruikt kan worden voor het genereren van een prestatie waarde en een relatie waarde.

Teammember/component

Dit zijn de personen die een rol in het team vervullen. Elke persoon heeft een *skill set* en per vaardigheid een waarde die aangeeft hoe goed hij daarin is. Bij het toekennen van een waarde wordt een schaal van -100 tot 100 gehanteerd. Het gewicht geeft aan hoe gevoelig de vaardigheid is ten opzichte van de andere vaardigheden.

Teamlid	Skill set		
	Skill	Value	Weight
Jan	Leiderschap Motivatie Communicatie	50 20 80	1 3 2
Piet	Ontwerpen Motivatie Communicatie	70 90 80	3 2 1
Klaas	Programmeren Motivatie Communicatie	90 -30 30	2 1 2

Van de leden in de tabel neemt Jan de leider rol aan, Piet de designer rol en Klaas de programmeur rol.

Personal performance value

Met de gegeven waardes uit de *skill set* en rol definitie kan van elk lid een persoonlijke prestatie waarde gegenereerd worden. Om deze waarde te genereren is een formule nodig. Hoe deze formule er uit ziet hangt af van de spelontwikkelaar. In dit voorbeeld zal ik de volgende formule gebruiken.
Formule: van de prestatie vaardigheden uit de rol definitie: (som van waarde x gewicht) / (som van alle gewichten).

Teamlid	**Persoonlijke prestatie waarde**
Jan	27,5
Piet	78
Klaas	50

Relationships

Relatiewaardes worden opgebouwd over tijd en kunnen niet op dezelfde manier gegenereerd worden als de persoonlijke prestatie waarde. Na elke opdracht of event dat samen wordt uitgevoerd zal de relatie waarde veranderen.
Formule: van de relatie vaardigheden uit de rol definitie: (som van waarde x gewicht) / (som van alle gewichten). Deze waarde wordt per teamlid berekend. Vervolgens wordt het gemiddelde per koppel genomen. Dit gemiddelde per koppel wordt door 10 gedeeld. Na elke '*relationship event*' zal de relatie waarde met het berekende aantal punten toenemen of afnemen.

In het voorbeeld ga ik uit van 4 '*relationship events*' waarbij de waardes in vaardigheden onveranderd zijn gebleven.

Teamlid	**Relatie**	**Relatie waarde**
Jan (44)	Piet Klaas	34,94 10,8
Piet (86,7)	Jan Klaas	34,94 19,34
Klaas (10)	Jan Piet	10,8 19,34

In moment van conflict of ruzie, waarbij het probleem niet is opgelost, kan de opgebouwde relatie waarde met bijvoorbeeld 20 punten verminderen, waardoor die punten opnieuw moeten worden opgebouwd.

Teamwork value

Dit is een gemiddelde waarde van hoe goed een teamlid in het team kan samenwerken.
Formule: gemiddelde van alle relatie waardes

Teamlid	**Teamwork waarde**
Jan	22,87
Piet	27.14
Klaas	15.07

Teamwork contribution

Dit is de prestatie die een teamlid aan het team levert. Hierbij wordt de persoonlijke prestatie waarde door de persoonlijke teamwork waarde beïnvloed.
Formule: Teamwork waarde gedeeld door 10 + prestatiewaarde.

Teamlid	**Team contributie**
Jan	29,78
Piet	80,71
Klaas	51,50

Overall teamwork

Dit is de waarde die het teamwork van het team representeert.
Formule: Gemiddelde van alle teamwork waardes met de impact waarde van de rollen meegerekend.

Team	**Teamwork**
Game development team	21,99

Overall team performance

Dit is de waarde die de prestatie van het team representeert.
Formule: Gemiddelde van alle teamwork contributies met de impact waarde van de rollen meegerekend.

Team	**Prestatie**
Game development team	47.94

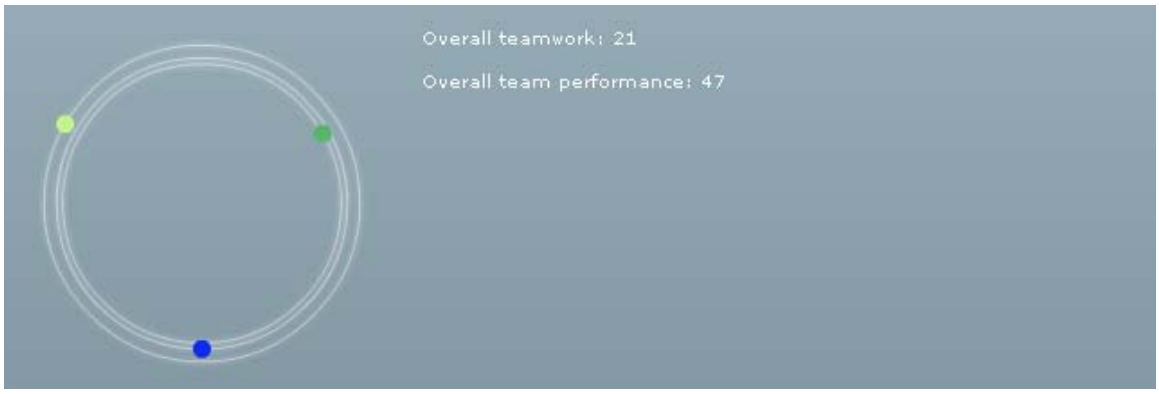
Toepassing

Dit model kan gebruikt worden om een teamworkwaarde te generen voor verschillende onderwerpen. Zo kan het model aangepast worden voor een voetbalspel of een managerspel.

In het voorgestelde model en voorbeeld worden de vaardigheden geplaatst onder relatie en/of prestatie. Om meer informatie te halen uit het team is verdere opsplitsing mogelijk. Dit geldt voornamelijk voor prestatie. Als we kijken naar voetbal games, dan zouden de prestatie vaardigheden verdeeld kunnen worden over verdediging, aanval, snelheid en strategie prestaties.

Het model geeft de mogelijkheid om individuele statistieken en relaties van de teamleden weer te geven. Dit is te vergelijken met bekende netwerk websites als hyves.nl en hi5.com.

In het prototype dat ik gemaakt heb, is het voorbeeld uit de vorige sectie verwerkt. Het prototype geeft een abstracte visualisatie van teamwork, welke in computer games gebruikt kan worden als een geavanceerd scorebord.



Figuur 30: screenshot van het prototype teamwork score.

De balletjes representeren de teamleden van het team. De afstand van het middelpunt van de grote cirkel tot aan een balletje, representeert de teamwork waarde van een teamlid. De balletjes draaien in een cirkel beweging. Beweegt het balletje met de klok mee, dan is de persoonlijk prestatie waarde van het teamlid positief. Beweegt het balletje tegen de klok in, dan is de persoonlijke prestatie waarde van het teamlid negatief (het teamlid werkt het team tegen). De snelheid van het balletje representeert de persoonlijke prestatie waarde van het teamlid.

De teamwork scorebord kan in het bedrijfsleven als monitor functie en management hulpmiddel gebruikt worden. Als dit model gevoed wordt met realtime informatie kan met een visuele representatie als deze, bottlenecks en problemen in één opslag gedetecteerd worden. In dit geval zijn de teamleden echte personen en geen AI agents waarvoor de waardes van persoonlijke vaardigheden door een spelontwikkelaar ingesteld kan worden. Het is de vraag hoe deze waardes van echte personen bemachtigd kunnen worden.

Ook interessant is dat oplossingen voor problemen voorspeld/gevonden kunnen worden door simulatie. De handleiding van het prototype is terug te vinden in appendix B.

5.3 Conclusie

Teamwork is een werkgroep of eenheid met een gemeenschappelijk doel waarbij deelnemers wederzijdse relaties opbouwen om doelen te bereiken en taken te verrichten [HH96]. Dit is een veelzijdig concept. Haar veelzijdigheid maakt de verwerking van teamwork in een spel een complexe taak. Dit betekent dat het concept van teamwork veel variabelen (team, teamleden, vaardigheden, relaties, etc.) met zich meebrengt die op een bepaalde manier met elkaar in verhouding staan.

Voordat teamwork in een spel toegepast kan worden is het noodzakelijk om de verhouding tussen de variabelen met een model in kaart te brengen. Dit model is toepasbaar over verschillende projecten, en biedt als het ware een framework voor het opzetten van samenwerking in een spel.

Verschillende teamwork spellen vereisen verschillende vormen van teamwork. Dit betekent dat er meerdere teamwork modellen bestaan. Zo heeft een voetbal game een ander teamwork model dan een basketbal game.

6 Teamwork in games

De grote interesse in computer games heeft voor een groot aanbod van verschillende type games gezorgd. Je kunt het zo gek niet bedenken en er bestaat een spel van. Door de groei van het Internet en het aantal mensen dat dagelijks online spellen spelen, worden steeds meer spellen uitgebracht die je samen kunt spelen.

De meeste van deze multiplayer games bestaan in de vorm van één tegen één, waarbij de beste speler wint. Een andere vorm van multiplayer games bestaat uit twee of meer spelers die samen hetzelfde doel willen bereiken. Hierbij is samenwerking van belang. Spellen die teamwork bieden kunnen nogal van elkaar verschillen. Meestal worden niet alle aspecten van teamwork geïmplementeerd. Of alleen een bepaalde vorm van teamwork wordt geïmplementeerd. Hierdoor verschilt per spel de ervaring van teamwork dat door de speler wordt beleefd.

Ook worden de verschillende aspecten van teamwork per spel verschillend geïmplementeerd. Communicatie is een aspect van teamwork, maar hoe dit in een spel geïmplementeerd wordt hangt af van de spel ontwikkelaar. De ene oplossing is beter dan de andere, of beter geschikt voor het spel dat de ontwikkelaar wil maken. Een voorbeeld van twee verschillende communicatie oplossingen waaruit spel ontwikkelaars kunnen kiezen is: spraak-communicatie en chat-communicatie. Dit betekent dat de effectiviteit van communicatie per spel verschilt.

In computerspellen wordt het concept van teamwork op verschillende manieren overgebracht, maar over het algemeen kun je twee vormen specificeren.

- Er bestaat een teamwork waarde die de spelervaring beïnvloed. Dit soort implementaties van teamwork zie je meestal terug in teamsport spellen als voetbal en management spellen.
- Vorming van teamwork wordt overgelaten aan de spelers, maar wordt gefaciliteerd door het bieden van hulpmiddelen. Dit soort teamwork in games vindt je meestal in first person shooters en strategie spellen waarin teamwork gefaciliteerd wordt door communicatiemiddelen als chat. Dit soort games kun je meestal niet alleen spelen.

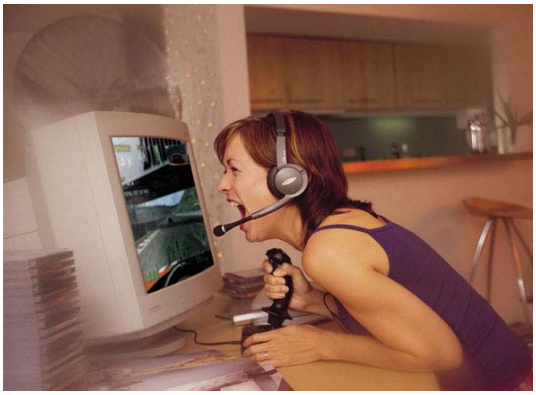
Het betekent niet dat elk teamwork spel tot één van de twee vormen behoort. Een balans tussen beide vormen is mogelijk. Zo kunnen twee gamers in een game samen een voetbal elftal besturen. Teamwork hangt af van de communicatie tussen de gamers onderling, maar ook van de harmonie (teamwork waarde) binnen het elftal (harmonie tussen de AI agents).

6.1 Bestaande vormen van teamwork in games

Baumann [Bau03] somt enkele mogelijkheden en frameworks op die in bestaande games worden gebruikt om de speler een teamwork ervaring mee te geven. Het is geen compleet beeld, omdat creativiteit nieuwe mogelijkheden toelaat.

In-game communicatie

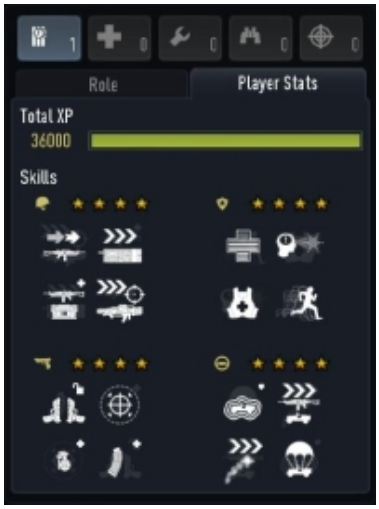
In het spel wordt de speler de mogelijkheid gegeven om met anderen te communiceren. Dit kan onder andere in de vorm van chat of spraak-communicatie. Door communicatie is het mogelijk om strategie te bepalen, rollen te verdelen, etc. In vele gevallen werkt spraak beter dan chat, omdat gedurende intense momenten in het spel veel minder gelet wordt op de chatbox.



Figuur 31: in-game communicatie via headset

Experience-system

Dit idee stimuleert de speler zijn teamleden te helpen. Elke keer wanneer een speler zijn teamgenoot helpt, ontvangt hij daarvoor experience-points. Des te meer experience-points de speler heeft, des te effectiever de acties van de speler worden.



Figuur 32: overzicht van vaardigheden van een speler met zijn ervaring uitgedrukt in “sterren” en “Total XP”

Gespecialiseerde classes/Rolverdeling
Door spelers in te delen in gespecialiseerde classes zijn de spelers beter geschikt voor bepaalde taken dan anderen. Bijvoorbeeld een dokter is niet goed uitgerust om deel te nemen in een vuurgevecht, maar biedt een grotere bijdrage door gewonde soldaten te genezen. Dit heeft tot gevolg dat spelers elkaar opzoeken om gebruik te maken van elkaars specialiteiten. Dit soort implementaties van teamwork gaat meestal samen met het Experience-system.

Het spel zelf
Het spel zelf kan teamwork forceren, door bijvoorbeeld doelen te stellen die alleen met twee of meer personen behaald kunnen worden.

Commander-system
Dit is het idee om één persoon al het werk te laten plannen. Dit werkt alleen goed als de leider weet wat hij moet doen.

Rank
Aan de hand van ranken worden de spelers speciale privileges gegeven. Personen met een lagere rank zullen in bepaalde situaties hulp van spelers met een hogere rank moeten vragen.

Clans
Een clan is een vorm van online community, en is een initiatief van de spelers zelf. Clans worden gevormd om samen als een groep online spellen te spelen. Communicatie tussen leden van een clan vindt plaats buiten het spel zelf, zoals op een forum of via chat programma's als MSN Messenger. De bedoeling hiervan is om organisatie, planning en strategie te bepalen.
Nieuwe teamwork games bieden zelf de spelers de faciliteit om clans te vormen. Dit gebeurt dan via een forum of webpagina van de spelontwikkelaar zelf.

6.2 Problemen in teamwork games

Hieronder worden een aantal punten opgesomd, die illustreren waar het mis kan gaan bij het spelen van teamwork spellen.

Organisatie
In de veel gevallen belemmert het gebrek aan organisatie het teamwork. De ene speler weet niet waar de andere mee bezig is, en zelfs de leider weet niet wat iedereen aan het doen is.

Communicatie
Bij slechte communicatie middelen is het vragen om hulp, gedurende intense momenten in het spel, tijdverspillend. Vaak gaat slechte organisatie samen met slechte communicatie.

Gedrag

Slecht gedrag van bepaalde individuele spelers kan het spel voor de rest van de spelers bederven. Dit is een andere reden waarom teamwork niet altijd werkt.

Kennis & Ervaring

Soms werkt het gebrek aan kennis en ervaring over hoe het spel werkt het teamwork erg tegen. Deze beginners verpesten onbedoeld de basis voor goed teamwork.

Orde

Een ander geval waarom teamwork niet goed werkt is het gebrek aan orde. Vaak hebben spelers een eigen wil en lopen waar ze zelf willen, al is het de verkeerde kant op. Het gebrek aan orde is ook gelinkt aan het gebrek aan organisatie.

6.3 Toepassing van teamwork elementen

In teamwork games worden lang niet alle mogelijke teamwork aspecten verwerkt. Alleen wat als requirements is opgesteld, en welke meedoen als spel elementen worden verwerkt in het teamwork concept. Dit betekent dat teamwork aspecten als onderlinge relaties en rollen als leiderschap niet noodzakelijkerwijs in een teamwork spel aanwezig hoeven te zijn.

In multiplayer games zoals first person shooters, wordt succesvol teamwork bepaald door de spelers zelf. Een model dat succesvol teamwork bepaald, wordt niet geïmplementeerd in het spel, alleen de tools zoals communicatie tussen spelers worden geleverd. Een team dat slecht presteert zal niet achterhalen waarom zij falen. Alleen individuele prestaties zoals (in geval van voetbal) aantal schoten op doel en aantal passes worden door het spel geleverd. De strategie, planning en organisatie om samen succesvol een doel te bereiken moet zelf worden uitgezocht door de spelers.

Figuur 33: wedstrijd statistieken uit het voetbalspel "FIFA 2008"

6.4 Communicatie

Communicatie speelt een grote rol binnen samenwerking in teamverband. Zonder communicatie is teamwork praktisch onmogelijk. Spellen kunnen communicatie faciliteren met componenten die herbruikbaar zijn.

Chat

Chat is een populair fenomeen op het internet, welke in teamwork games bijna onmisbaar wordt geacht. De zogenaamde chat-box was in het begin erg gelimiteerd. De groei van de "chat-communitie", en de daarbij horende chat specifieke taal (lol = laughing out loud), heeft chatboxes geavanceerder gemaakt. Een goed voorbeeld is de tegenwoordige "Windows Live Messenger" van Microsoft [1]. Deze begon jaren terug als MSN Messenger, en was geïmplementeerd in het besturingssysteem Windows XP. Dit was een relatief gelimiteerde chat tool, waarmee je met email contacten live kon "chatten". Het toenemende aantal messenger gebruikers heeft de chat-tool over de jaren doen evolueren om in te stemmen met de wensen en verwachtingen van messenger gebruikers. Zo heeft messenger zijn chat component weten uit te breiden met onder andere: emoticons, customized emoticons, tekst font, tekst color, transfer van bestanden en een tekenprogramma.

Spraak

Één van de meest directe vormen van communicatie is spraak. Spraak communicatie in games is niet zo oud, maar dit heeft vooral te maken met de bandbreedte van internet verbindingen tussen gebruikers. Tegenwoordig is het eenvoudig en relatief goedkoop om over ruime internet bandbreedte te beschikken. Deze bandbreedte is noodzakelijk voor vloeiende gesprekken, zonder dat daarbij vertraging in het spel veroorzaakt wordt. Volgens T. Valich is spaak communicatie in de nieuwste teamwork games niet meer weg te denken.

Text-to-speech

Gedurende intensieve momenten in een spel is het makkelijk om belangrijke informatie, die bijvoorbeeld in een chat-box gepresenteerd kan worden, te missen. Deze informatie die met het oog gemist wordt, kan worden opgevangen door een text-to-speech component. Deze zet geschreven tekst om in spraak. Er bestaan voldoende commerciële text-to-speech engines die licenties verkopen voor hun product. Site-pal [2] is een bedrijf die zo een text-to-speech engine in haar product heeft verwerkt. Dezelfde text-to-speech engines worden toegepast in verschillende applicaties over verschillende platformen. Het omgekeerde is ook mogelijk (spraak naar tekst). Dit is handig om miscommunicatie te voorkomen. Het komt regelmatig voor dat bepaalde gesproken woorden geïnterpreteerd worden als andere woorden die er op lijken. Maar dit proces is nog tamelijk geavanceerd en werkt niet goed genoeg om in games te verwerken.

[1] *http://services.nl.msn.com/messenger/*
[2] *http://www.sitepal.com/*

Video/webcam

Een andere vorm van communicatie is video communicatie. Dit gebeurd via zogenaamde webcams. Het concept van video communicatie binnen games wordt weinig gebruikt. Dit komt omdat video streams relatief veel bandbreedte consumeert. Vooral wanneer je bedenkt dat bepaalde spellen 4 tot 16 spelers kan betrekken. Voor online kaartspellen zoals UNO en Texas Hold'm is video communicatie geoorloofd omdat de gameplay weinig bandbreedte vereist. Er wordt namelijk om de beurt een zet gedaan. Bij complexe spellen waarbij intensieve interactie praktisch tegelijk gebeurd, zoals bij de meeste teamwork games, is het niet geoorloofd om veel bandbreedte te missen, omdat er dan vertraging wordt veroorzaakt. Hierbij is de speler met de beste internet verbinding in het voordeel. Dit is het bekende fenomeen genaamd "lag". Een voorbeeld: Er zijn twee spelers die online tegen elkaar een voetbal spel spelen. De één heeft een goede verbinding en de andere een slechte. Degene met een goede verbinding heeft de bal, valt aan en doet een schotpoging. De andere speler weet pas dat er schotpoging is gedaan nadat de bal in het net is verdwenen. Het is duidelijk dat de speler met de goede verbinding in het voordeel is.

Ook al worden webcams vrij weinig in computer games gebruikt, zijn zij wel populair. De nieuwste video game consoles kunnen uitgebreid worden met een webcam. Naast communicatie zijn zij ook populair vanwege de nieuwe mogelijkheid van speler input. CamSpace demonstreert deze nieuwe mogelijkheid met hun programma CamTrax [1].

Figuur 34: CamSpace demonstreert speler input via webcam.

In de nabije toekomst wanneer consumenten over betere internet verbindingen (glasvezel) beschikken, zal video communicatie net zo onmisbaar zijn als spraak. Met de evolutie naar betere internet verbindingen zullen ook nieuwe in-game middelen van communicatie worden geïntroduceerd, die op dit moment nog onmogelijk zijn.

[1] *http://camspace.com/*

6.5 Teamwork AI (gedrag)

In teamwork games van tegenwoordig kun je de single player en/of co-op optie niet missen. Hier nemen één of meerdere spelers het op tegen de computer. De computer bestaat dan uit een groep van intelligente wezens. Kritiek op dit soort spellen is dat zij in praktijk helemaal niet zo intelligent zijn, of helemaal niet naar verwachting gedragen. Volgens Schwab [Sch04] wordt de kwaliteit van teamgedrag van de computer bepaald door gebruik van Artificial Intelligence (AI). AI bestaat simpel gezegd uit een set van regels die bepalen hoe een entiteit zich moet gedragen wanneer deze zich in een bepaalde situatie verkeerd. Deze regels en de interpretatie daarvan hoeven niet voor elk spel opnieuw uitgevonden te worden. Deze kunnen worden hergebruikt in meerdere computer spellen. Ontwikkelaars van teamwork games hebben speciale afdelingen die onderzoek doen naar AI voor hun games. Dit gaat gepaard met uitvoerige testen.

6.6 Experience system

Zoals eerder beschreven is een *experience system* is een systeem dat de groei en ervaring van AI karakters bepaalt. In dit systeem wordt een groot aantal variabelen in rekening gebracht en worden rekenformules geïntroduceerd, die exact bepalen hoe snel een AI karakter kan groeien.

Vooral *Role Playing Games* (RPG) maken gebruik van *experience systems*. Om een voorbeeld te geven zal ik gebruik maken van de RPG “Final Fantasy VIII” [1] uit de bekende “Final Fantasy” serie.

Figuur 35: Final Fantasy VIII maakt gebruik van een geavanceerde experience system

[1] *http://www.final-fantasyviii.com*

Vooral deze titel uit de serie staat bekend om zijn complexe *experience system* en besturing (vanwege de vele variabelen). In dit spel is magie en het vechten tegen monsters de normaalste zaak van de wereld, wat ook geld voor de meeste traditionele RPG's. Als speler neem je controle over een aantal karakters. Meestal is dit maximaal drie tot vijf karakters tegelijk. Elke karakter heeft zijn eigen sterke en zwakke punten, welke in variabelen worden uitgedrukt. Enkele van deze variabelen zijn fysieke kracht, defensie tegen fysieke aanvallen, magische kracht, defensie tegen magie. Een karakter dat fysiek sterk is, maar magisch zwak, zal een hogere score hebben in de fysieke aanval en fysieke defensie variabelen. De momenten wanneer een karakter kan groeien (hogere scores in variabelen ontvangen) is gedurende gevechten tegen monsters. Wanneer de strijd is gewonnen ontvangen de karakters zogenaamde "*experience points*". Afhankelijk van de acties die met de karakters zijn uitgevoerd, stijgen bepaalde variabelen in waarde. Heeft een karakter gedurende het gevecht alleen magische spreuken gebruikt, dan zal bij deze karakter de magische aanval variabele in waarde stijgen.

De *experience-system* kan ontwikkeld worden op een manier waarin de game designer zijn variabelen en rekenformules kan introduceren. Dit concept wordt gebruikt in de RPG genaamd "RPG Maker"waar de speler als het ware zijn eigen RPG kan maken [1].

6.7 Andere toepassingen van teamwork

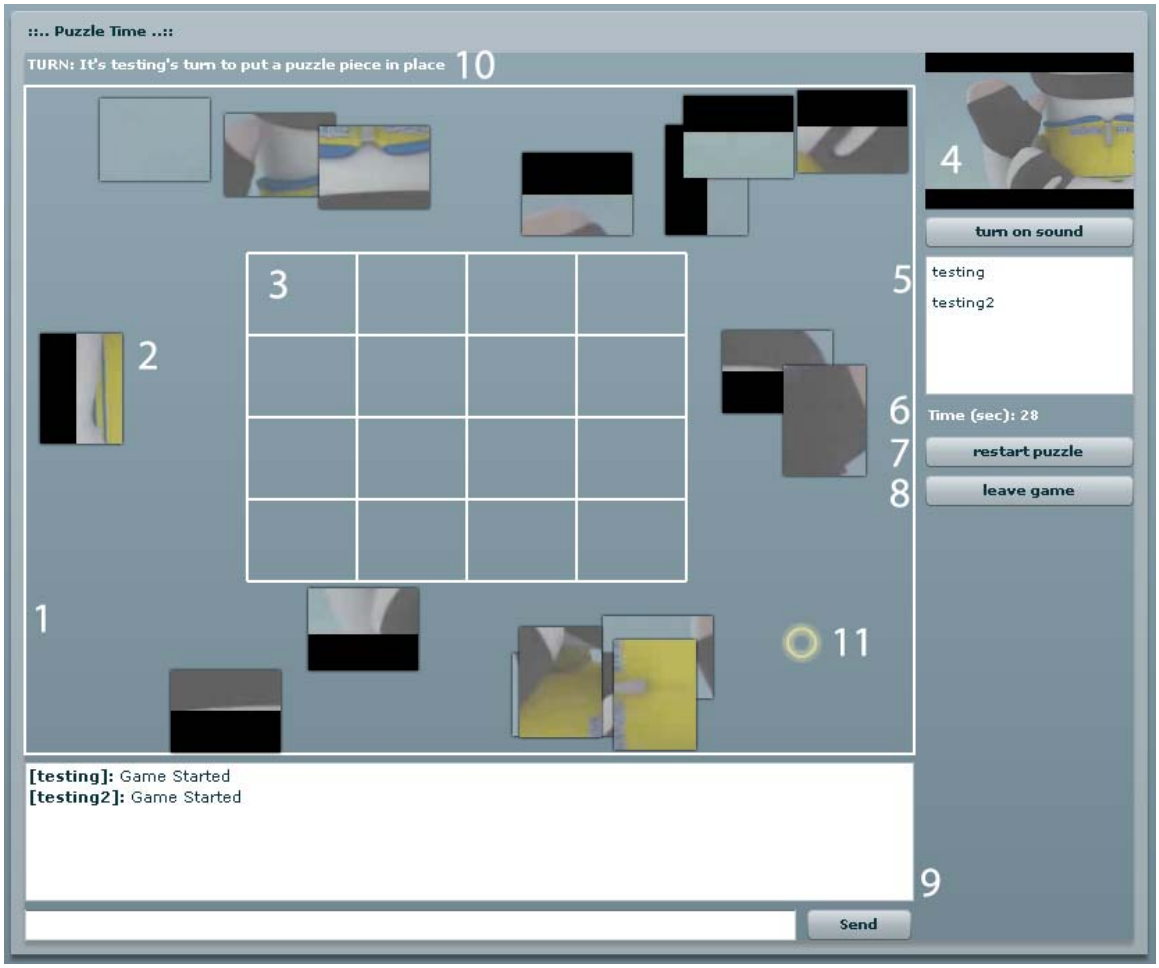
Er bestaan meer mogelijkheden om teamwork in games toe te passen, maar zij zullen al snel zo specifiek zijn, dat deze alleen toegepast kunnen worden in spellen van hetzelfde onderwerp. Het belangrijkste aspect van teamwork is communicatie. Een ander aspect van teamwork is het bieden van hulp. Het bieden van hulp is taakspecifiek. Taakspecifieke onderdelen van een spel zijn geschikte kandidaten om als herbruikbare componenten te ontwikkelen, maar zij zullen niet snel met teamwork geassocieerd worden.

Een voorbeeld. Er zijn twee spelers die naar een geografische map kijken. Zij zijn op zoek naar de locatie van een schatkist. De ene speler weet waar de schatkist ligt, maar de andere niet. Om de onwetende speler op de hoogte te brengen van de locatie kan de speler de plek aanwijzen. In dit voorbeeld is aanwijzen een vorm van communicatie, en een manier om hulp te bieden, maar wordt niet direct geassocieerd met teamwork.

6.8 Prototype: Video Puzzel

Om enkele toepassingen van teamwork te demonstreren heb ik een minigame ontwikkeld. De minigame is een video puzzel. Het principe is gelijkwaardig aan de bekende legpuzzel, maar in plaats van een plaatje wordt er gebruik gemaakt van een video clip. De video puzzel demonstreert, teamwork dat door het spel zelf wordt afgedwongen, chat communicatie, en een manier om een teamgenoot hulp te bieden. De handleiding voor het prototype is terug te vinden in appendix C.

[1] *http://rpgmaker.agetec.com/*



Figuur 36: screenshot van de multiplayer video puzzel

Teamwork

Het spel is ontworpen om gespeeld te worden met twee personen. De puzzelstukjes worden door de spelers om de beurt geplaatst op een rooster. Hiermee wordt teamwork door het spel geforceerd, zodat het niet mogelijk is dat één speler alle puzzelstukjes legt.

Communicatie

Communicatie is een belangrijk onderdeel in samenwerking. Aan spel is chat box toegevoegd, zodat de spelers met elkaar kunnen communiceren, strategieën kunnen bepalen, en elkaar hulp kunnen bieden.

Aanwijzer

Een meer praktische manier van hulp bieden gaat via aanwijzen. Een speler die niet aan de beurt is kan hulp bieden aan de speler die een puzzelstuk op de rooster moet plaatsen. Er kan gewezen worden waar een puzzelstuk geplaatst moet worden. Deze tool is in andere woorden een handige manier van voorzeggen.

Conclusie

Totstandkoming van goede samenwerking wordt gefaciliteerd door het aanbieden van voldoende en begrijpbare communicatie mogelijkheden en mogelijkheden om elkaar hulp te bieden. Dit geldt vooral wanneer de spelregels en/of gameplay complex zijn. De simpele gameplay van de videopuzzel maakt de chat-box met als doel teamwork te faciliteren bijna overbodig. Het om de beurt plaatsen van de puzzelstukjes en het kunnen aanwijzen van locaties is in principe voldoende. Communicatie via chat maakt het spel persoonlijker en brengt een bepaalde aantrekkingskracht met zich mee. Om deze reden kun je de chat-box of andere manier van directe communicatie (bijvoorbeeld spraak en webcam) terug vinden in de meeste multiplayer games, ook al is dit voor de totstandkoming van samenwerking overbodig.

De video puzzel is in een aantal opzichten gelimiteerd, maar het is mogelijk om het spel uit te breiden, zodat het gespeeld kan worden met meer dan twee personen. Ook is het mogelijk om nieuwe spelvariaties te introduceren. Zo bestaat de mogelijkheid om het spel uit te breiden met een webcam functie. In plaats van een video clip wordt er gebruik gemaakt van de ‘live webcam feed’, om de puzzelstukjes te maken. Met deze functie kunnen de spelers elkaar ‘live’ hulp bieden.

Een ‘serious’ toepassing van de video puzzel bestaat wanneer er gebruik gemaakt wordt van ‘losse’ videoclips om de speler gegeven informatie over te brengen. Bijvoorbeeld ter promotie van een nieuw product. In de eerste plaats wordt informatie overgebracht in de vorm van geluid, omdat de video clip nog in stukjes ligt. Naar mate de puzzelstukjes op de juiste plaats worden gezet zal het beeld materiaal een steeds grotere informatiewaarde aannemen.

6.9 Prototype: Game Trace

De gameplay van de video puzzel forceert samenwerking, maar stimuleert dit niet. Om dit te stimuleren is het nodig om samenwerking te belonen. Dit kan in de vorm van score. In de video puzzel wordt een tijd bijgehouden. Wanneer de puzzel snel opgelost wordt dan ontvangt de speler daar een hoge score voor. Deze samen behaalde score kan opgenomen worden in een rangelijst, waarin het best scorende duo bovenaan de lijst staat.

Behalve tijd kunnen ook andere gameplay elementen bij de score bepaling betrokken worden, zoals mate van chat-box gebruik, en mate van effectief aanwijzen van de puzzelstukjes. Om verschillende gameplay elementen te betrekken bij de score is het noodzakelijk om de acties van de speler te registreren. Dit is de zogenaamde “game trace”.

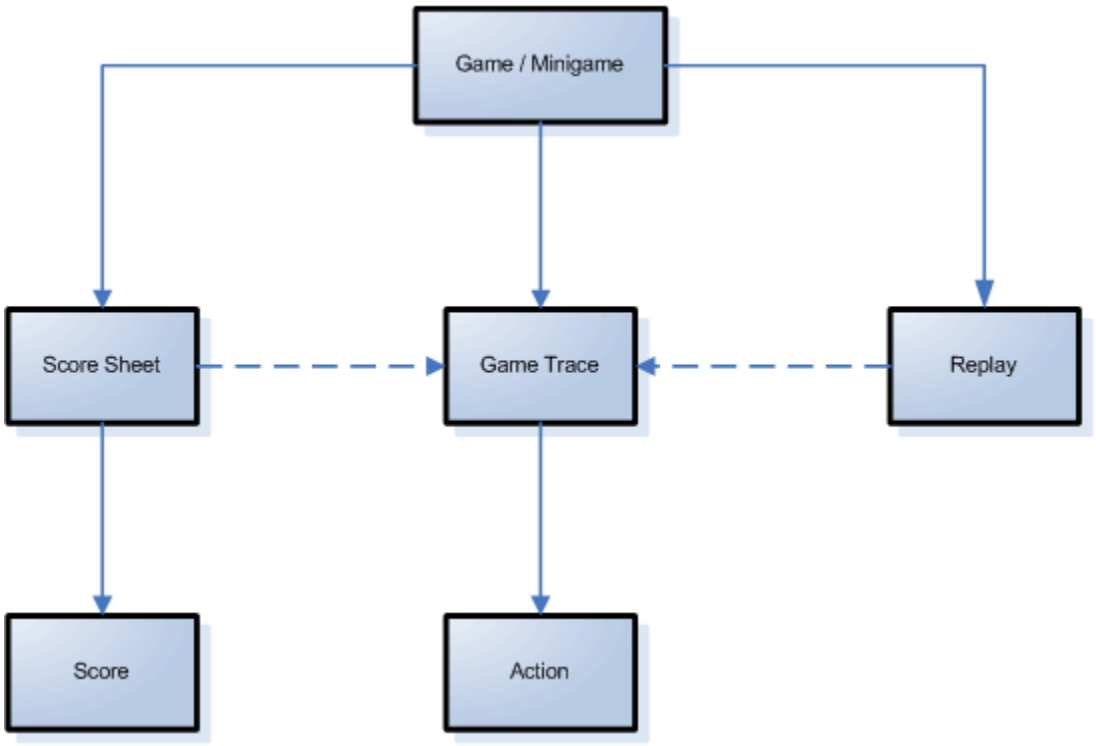
Uit de game trace is het mogelijk om gedetailleerde statistieken te genereren. In het geval van een voetbalspel is het mogelijk om met een game trace wedstrijdstatistieken te generen. De registratie van spelers input kan gelimiteerd worden tot enkele acties, welke voldoende zijn om de gewenste informatie uit te halen, maar kan ook zo gedetailleerd geregistreerd worden, zodat het mogelijk is om spel van de speler te reproduceren. Dit laatste staat bekend als de “replay” functie, en staat bekend als een hooggewaardeerde feature binnen een spel. Zo is het mogelijk om met de replay functie uit de race simulatie “Forza 2” [1], een gespeelde race na te beschouwen.

Figuur 37: uitgebreide replay functie uit de racesimulatie “Forza 2”

[1] *http://forzamotorsport.net*

Game Trace model

Om een game trace in de video puzzel te verwerken ben ik uitgegaan van het volgende zelf ontworpen model.



Figuur 38: Game Trace model

De game trace bestaat uit een serie van acties. De acties die door de speler genomen kan worden zijn afhankelijk van het spel dat gespeeld wordt. Dit betekent dat elk spel (game/minigame) een “vertaler” nodig heeft om de game trace uit te kunnen lezen. Voor elke actie moeten een aantal gegevens worden opgenomen:

- Naam van de actie
- Tijdstip waarop de actie heeft plaatsgevonden
- Parameters die nodig zijn voor de vertaler om de actie na te bootsen

Replay

De acties die beschreven staan in de game trace, worden uitgelezen door een replay object. Hierin wordt gespecificeerd hoe een actie geïnterpreteerd moet worden, en nagebootst kan worden.

Score

Wanneer een nieuwe actie aan de game trace wordt toegevoegd, dan wordt een event gegenereerd die dit aangeeft. De score sheet object vangt dit op, en bepaald of er voor de toegevoegde actie een score kan worden gegeven. In de score sheet staan de scores gespecificeerd. Dit object bepaald voor welke acties de speler punten ontvangt. Elke keer wanneer een score wordt gegeven, dan wordt dit bijgehouden door de score sheet. Dit gebeurd op vergelijkbare wijze als hoe de game trace object nieuwe acties toevoegt. Een score object bestaat uit de volgende onderdelen:

- Naam van de score. Dit is om verschillend type scores van elkaar te onderscheiden. Hierdoor kan aan de speler worden gecommuniceerd waarvoor hij precies beloond wordt.
- Waarde van de score. Dit is een numeriek getal, dat de waarde van de score representeert.

6.10 Conclusie

Toepassingen van teamwork in games kunnen in twee groepen worden ingedeeld, namelijk: faciliteiten en frameworks.

Onder faciliteiten worden hulpmiddelen verstaan die teamwork kunnen stimuleren. Communicatiemiddelen als chat en spraak zijn hier voorbeelden van. Dit zijn praktische oplossingen en kunnen in verschillende teamwork games worden toegepast. Zo bieden meerdere XBOX games communicatiemogelijkheden aan via de XBOX headset.

Figuur 39: XBOX headset

Frameworks zijn teamwork modellen. Zij geven aan op welke manier teamwork plaatsvindt in een spel. Enkele verschillende teamwork frameworks zijn: *experience system*, gespecialiseerde classes/rolverdeling en *commander system*. Zij vormen de basis voor een teamwork spel en moeten per spel specifiek worden uitgewerkt. Zo kunnen twee verschillende spellen gebaseerd zijn op de *experience-system* maar de gebruikte formules en variabelen verschillen van elkaar.

Teamwork AI is een complex systeem dat de samenwerking tussen agents bepaald. Zij behoort niet tot de faciliteiten groep, maar is onderdeel van een framework dat gebruik maakt van AI. Ook het bepalen van game scores die teamwork kunnen stimuleren zijn uit de frameworks af te leiden.

7 Proof of Concept

Dit hoofdstuk beschrijft de proof of concept voor een serious game met teamwork aspecten, gebruikmakende van informatie beschreven in voorgaande hoofdstukken. Verder zal de methodiek voor de ontwikkeling van de gemaakte POC gegeven worden.

7.1 Game concept

Het spel draait, zoals aangegeven in hoofdstuk 2, om teamwork. Hierbij wordt gefocust op samenwerking tussen de teamleden, en het oplossende vermogen bij het optreden van conflicten. Voordat het spel ontwikkeld kan worden is er een framework nodig om teamwork in het spel te introduceren. Hiervoor maak ik gebruik van het teamwork model uit hoofdstuk 5. Dit model is gebaseerd op het experience-system.

Het spel wordt opgebouwd uit projecten. Elk project is verschillend. Een project kan in de “game taal” vertaald worden naar “level”. Na het behalen van een level kan de speler even rust nemen, en ontvangt deze punten voor het voltooien van het level. De punten worden bepaald aan de hand van de genomen acties gedurende het project. Deze worden per speler opgeslagen in een “save game” bestand. In principe kan het spel oneindig uitgebreid worden met nieuwe projecten, maar de proof of concept zal één project (level) bevatten.

De karakters die in het spel voor komen kunnen zichzelf ontwikkelen in vaardigheden en relaties met andere karakters. De wijze waarop de karakters zichzelf kunnen ontwikkelen gaat volgens het ontwikkelde teamwork model.

Projecten

Elk project heeft een tijdlijn. De duur van een project kan per project verschillen. Maar dit betekent niet noodzakelijkerwijs dat het spelenderwijs voltooien van een lang project langer duurt dan een korter project.

Aan het begin van de tijdlijn wordt een project beschreven. Hierin staat beschreven wat er moet gebeuren, en wat er opgeleverd moet worden. Daarnaast wordt er in het begin een team samengesteld. De speler stelt aan de hand van de opdracht het best mogelijke team samen uit een aantal beschikbare werknemers. Bij een slechte keuze zal de speler minder punten ontvangen en zal het project er onder lijden. Gedurende het project is het mogelijk om leden van het team om te wisselen, maar ook dit zal natuurlijk punten gaan kosten.

Aan het eind van het project ontvangt de speler punten voor het voltooien van het project. Hierbij wordt gekeken in hoeverre de teamleden hun vaardigheden hebben verbeterd, hoe zij gepresteerd hebben en hoe de harmonie binnen het team is. Daarnaast wordt gekeken hoe de speler bepaalde “minigames” heeft gespeeld en hoe de speler problemen en conflicten binnen het team heeft opgelost.

Het onderwerp van een project is elke keer verschillend. Zo kan het eerste project over de ontwikkeling van een computer applicatie gaan, en het volgende project over de reparatie van een ingestorte brug.

Teamwork conflicten

Binnen een team kunnen er allerlei conflicten en problemen tussen de teamleden ontstaan. Wetenschappelijk is beschreven hoe een teamlid kan reageren in bepaalde situaties, maar voor het uitdrukken van zijn reactie zijn eindeloos verschillende manieren mogelijk. Daarom zullen er conflict scenario's in het spel worden geïntroduceerd. Het spel kan op deze manier uitgebreid worden met nieuwe scenario's, zodat het spel variatie kan bieden.

Elke scenario leidt tot een probleem, een conflict of een oplossing van een conflict. Vanuit de nieuwe situatie zijn vervolg scenario's mogelijk.

Gameplay

Het spel wordt gespeeld in een 2D omgeving, waarbij de speler input geeft met muis en toetsenbord. Binnen de 2D omgeving kunnen bepaalde acties worden uitgevoerd die altijd gedurende een project aanwezig zullen zijn. Zij dienen om problemen op te lossen en teamwork tussen de leden te verbeteren. Daarnaast komen per project een serie events binnen die van te voren zijn vastgesteld. Deze events kunnen minigames zijn, of bepaalde problemen en conflicten die kunnen ontstaan.

7.2 Tools

Voor het maken van de game zijn twee tools gemaakt, voor de efficiënte ontwikkeling van het spel. Ten eerste is een tool ontwikkeld om projecten aan te maken. Hierin worden de minigames, gebeurtenissen en verhaal in geplaatst. Dit is gebaseerd op mods en game editor technieken beschreven in hoofdstuk 3.

Daarnaast is een teamwork script tool ontwikkeld voor het aanmaken van conflict scenario's.

Project editor

De project editor kan gezien worden als een level editor, waarmee nieuwe levels ontworpen kunnen worden. In dit geval zal de project editor erg gelimiteerd zijn in vergelijking met commerciële level editors die veel meer mogelijkheden bieden.

Elk project heeft een eigen speelruimte, waarin de speler kan bewegen. Van hieruit kan de speler minigames starten, zijn score bekijken, etc. Het uiterlijk hiervan is voor elk project verschillend. De speelruimte voor een project waarbij software ontwikkelt moet worden ziet er bijvoorbeeld anders uit dan een project waarbij wegen gerepareerd moeten worden.

De project editor biedt de mogelijkheid om een project naar eigen wensen samen te stellen. Zo kan een speelruimte gekozen worden en kunnen events in een bepaalde volgorde worden vastgesteld. Deze events zijn zelfgekozen minigames en gebeurtenissen die in het project zullen voorkomen.

<u>Graphical User Interface</u>

De project editor maakt gebruik van een robuuste GUI waarin de gebruiker zijn keuzes en creativiteit in kan vast leggen. Dit gebeurd door het geven van input via de muis en door het invullen van formulieren.

XML

De project editor genereert een XML bestand, dat een beschrijving van het project is. XML is een taal dat op veel verschillende frameworks kan worden toegepast. Zo is het mogelijk om op eenvoudige wijze hetzelfde XML bestand te gebruiken over een verschillend aantal platformen. In dit geval zal de FLEX omgeving gebruikt worden om het XML bestand te interpreteren.

Teamwork AI Script/Interpreter

De teamwork AI tool kan gezien worden als de behaviors/Artificial Intelligence van een game. Er wordt gebruik gemaakt van de door Friedman-Hill [Fri03] beschreven rule-based scripting. Met de teamwork AI tool is het mogelijk om regels aan het spel toe te voegen die bepalen hoe teamwork verloopt binnen het spel. Hierbij zal ik de regels voor teamwork niet zelf bedenken, maar maak gebruik van resultaat uit het onderzoek naar samenwerking binnen teamverband, dat uitgevoerd is door Jansen [Jan02] en Sherif et al. [SHW+61].

De teamwork events die hieruit voortvloeien worden niet zoals in een project in een bepaalde volgorde getriggerd, maar hangt af van de relatie tussen de karakters en de situatie waarin zij zich verkeren. Deze events kunnen daarom tussen de events van een project in voorkomen

7.3 Project 1

Hier zal ik het eerste project beschrijven met haar inhoud en de gebeurtenissen die plaats zullen vinden.

Als speler wordt je gevraagd om een website te ontwikkelen voor een autofabrikant. De eisen die aan de website worden gesteld zijn de volgende:

- De voorpagina moet aantrekkelijk zijn voor de consument.
- Alle auto modellen moeten op de website te vinden zijn.
- Een consument moet via de website een dealer bij hem in de buurt kunnen vinden.

Begin

Voor het ontwikkelen van de website kunnen het beste twee personen worden gekozen met de volgende rollen: *webdesigner* en *webprogrammeur*. De speler kan ook andere rollen, en meer teamleden kiezen, maar dit zal gevolgen hebben voor het project.

Events
De gebeurtenissen die gedurende het project plaatsvinden, zijn gefocust op samenwerking en niet speciaal de inhoud van het project.

- (minigame) De webdesigner en webprogrammeur spreken met de klant over uiterlijk en functionele details. Hierbij is het belangrijk dat de ontwikkelaars elkaar aanvullen en goed communiceren met de klant om miscommunicatie te voorkomen (zie bijlage D).
- (probleem) De computer van de webdesigner loopt te traag. De webontwikkelaar en webprogrammeur raken hierdoor gefrustreerd, omdat de webdesigner moeilijk kan werken en de webprogrammeur op het ontwerp wacht.
- (minigame) De webdesigner ontwerpt de voorpagina van de website. Dit gaat via een puzzel, waarbij het beste gescoord wordt wanneer de puzzel snel wordt opgelost (zie bijlage E).

Eind
Aan het eind krijgt de speler een score, die gebaseerd is op de handelingen van de speler. De relaties en ontwikkelingen die tussen de teamleden zijn ontstaan, worden bewaard en zullen ook van belang zijn bij mogelijke toekomstige projecten.

7.4 Methodologie

De ontwikkeling van een serious game is een software project waarbij goede organisatie en aanpak van groot belang is voor efficiëntie en kwaliteit aspecten. Dit is het principe van software engineering; een discipline die zich bezighoudt met alle aspecten van het bouwen van complexe maar betrouwbare software.
Elk project dat gebruik maakt van een software engineering discipline, handhaaft een software ontwikkeling methode. Er bestaan meerdere methodes, met elk zijn voordelen en nadelen. De software ingenieur kiest de methode die het beste bij zijn project past.
Omdat de proof of concept niet een zeer uitgebreid spel is, en omdat snelheid gewenst is bij de ontwikkeling van de serious game, heb ik gekozen voor de *Rapid Application Development* methode.

Rapid Application Development
In reactie op de niet flexibele processen van de watervalmethode is de *Rapid Application Development* methode bedacht. De RAD methode is een iteratief ontwikkelingsproces waarbij prototypes worden gebouwd [Vli00].

De RAD levenscyclus bestaat uit vier fasen:
1. requirements planning
2. user design
3. constuction/prototyping
4. cutover

Timeboxing

Aan elke levenscyclus is een *time box* verbonden. Dit is de duur van een levenscyclus. De duur van een levenscyclus wordt meestal uitgedrukt in weken in plaats van maanden, zoals dat het geval is bij de watervalmethode. Deze time box is onveranderbaar. Als de time box onvoldoende tijd geeft voor het implementeren van alle functies in het prototype, dan zullen een paar functionaliteiten worden opgeofferd.

Iteratie en uitbreiding

Wanneer een doorlopen levenscyclus een onvoltooid product levert, zal de levencyclus van de RAD methode worden herhaald totdat het gewenste eindproduct gerealiseerd is.

Naast het iteratieve aspect van de RAD methode bevat de RAD methode ook een uitbreidingsaspect. Dit betekend dat verschillende kleine (sub)delen van het systeem over verschillende time boxes worden ontwikkeld.

De RAD methode kent een aantal voordelen en nadelen

Voordelen

- Verhoogde ontwikkelingssnelheid doormiddel van methodes als *rapid prototyping*, virtualisatie van systeemgerelateerde routines, en andere technieken.
- Verlaagde eindgebruiker functionaliteit (door nauwere design focus), dus minder complexiteit.
- Grotere nadruk op simpelheid en bruikbaarheid van de GUI.

Nadelen

- Verminderde schaalbaarheid en features wanneer een door RAD ontwikkelde applicatie start als een prototype en naar een afgeronde applicatie evolueert.
- Verminderde features door timeboxing wanneer features opgeschoven worden naar latere versies om een release af te ronden binnen een korte periode.

De RAD methode produceert, compleet ontwikkelde en geteste features (maar een heel klein (sub)deel van het geheel) om de paar weken of maanden. De nadruk ligt in het verkrijgen van het kleinste stukje werkende functionaliteit, om in een vroeg stadium business waarde te leveren, en deze continu te verbeteren en uit te breiden gedurende de levensduur van het project.

Voor het ontwikkelen van de serious game POC met herbruikbare teamwork componenten is de Rapid Application Methode geschikt, omdat deze open staat voor flexibiliteit, herbruikbaarheid, en is het mogelijk om in een vroeg stadium prototypes te leveren.

7.5 Gebruikte techniek

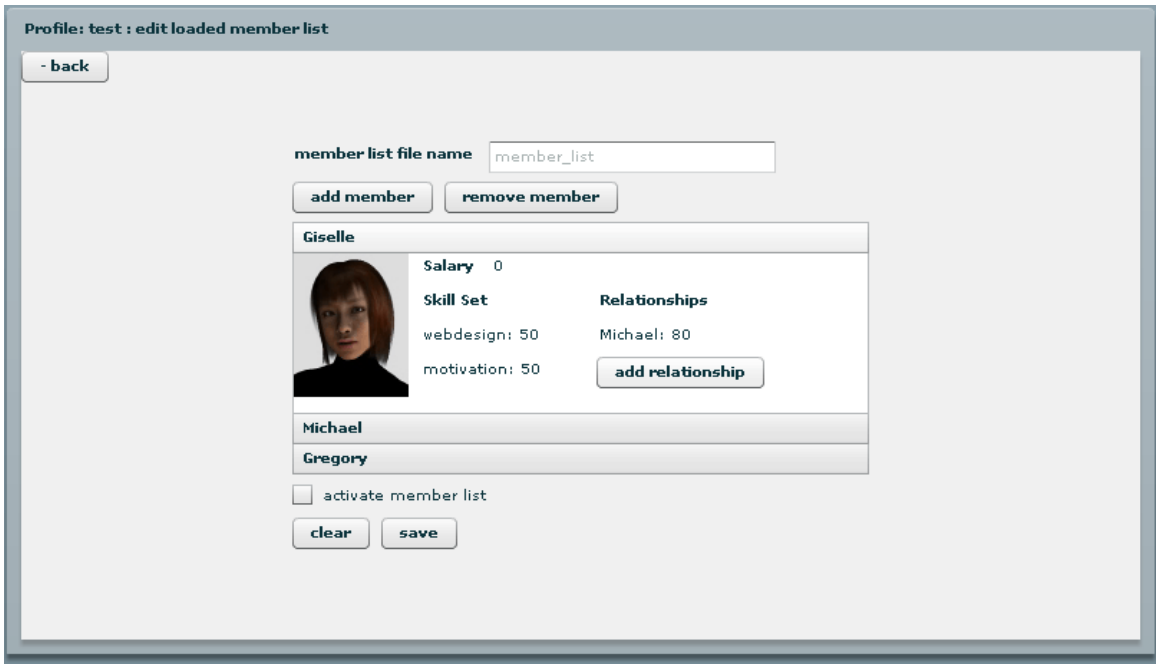
Voor het bouwen van de teamwork game is gebruik gemaakt van het FLEX framework in combinatie met PHP. De Flex Player en een PHP server zijn vereist om het spel te kunnen spelen. Verder is er gebruik gemaakt van andere programma's waarvoor ik verwijs naar bijlage in F.

Gedurende de ontwikkeling van de game ben ik enkele technische problemen tegengekomen, welke beschreven staan in bijlage G.

7.6 Resultaat

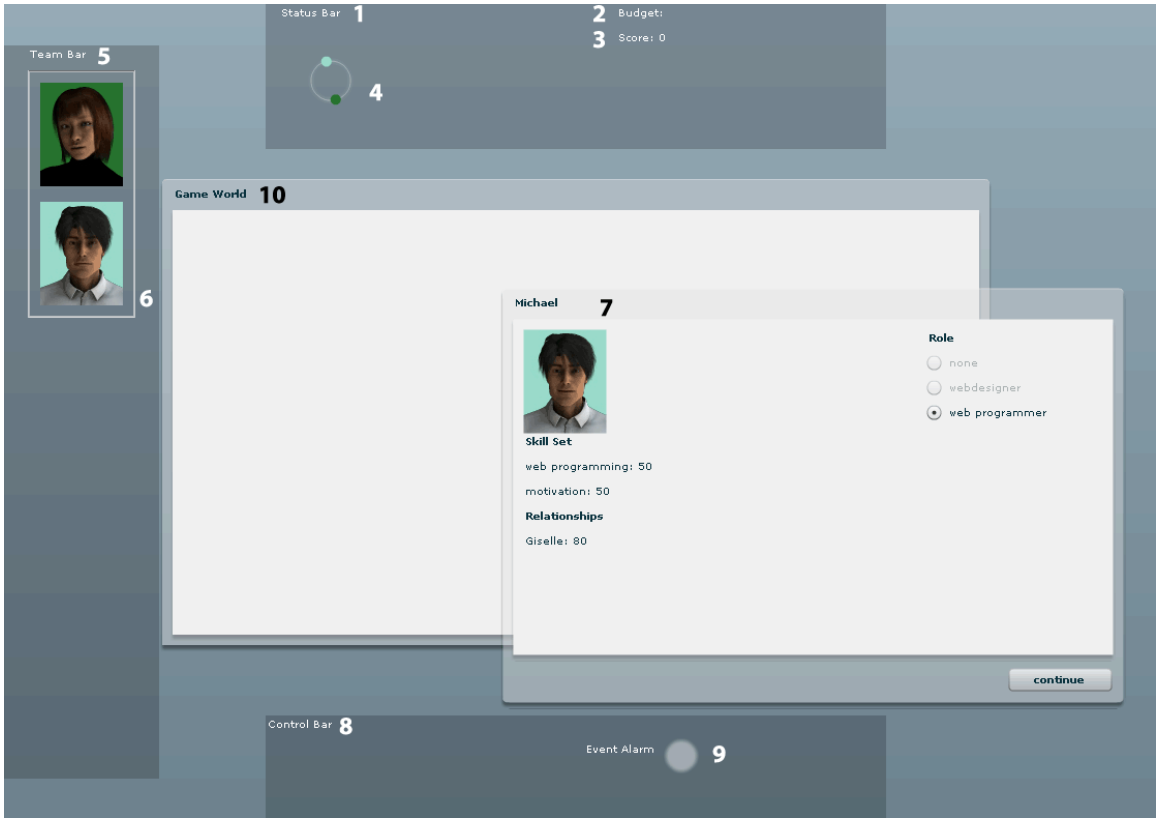
De PoC levert een spel op, dat volledig aan te passen is. Met de project editor is deze uit te breiden met nieuwe verhaal elementen, minigames, en teamwork scenario's.

Het plaatje hieronder laat een gedeelte van de project editor zien, waarmee nieuwe karakters aan het spel toegevoegd kan worden. De wijzigingen worden bij voltooiing opgeslagen in een XML bestand.



Figuur 40: onderdeel van de editor waarmee nieuwe karakters aangemaakt kan worden.

Hieronder is een kale (niet grafisch aangekleed) versie te zien van het spel.



Figuur 41: kale versie van het teamwork spel

1. Dit is de *status bar* waarin de vooruitgang van het spel te zien is.
2. Dit is het *budget* dat de speler heeft. Het budget dient onder andere voor het uitkeren van het salaris van de gekozen teamleden.
3. De *score* wordt met een getal uitgedrukt. Dit is de totale score van de speler.
4. Dit is een grafische representatie van *teamwork* gedurende een project. Dit is dezelfde representatie, dat gebruikt is voor het eerste prototype. De handleiding voor van het prototype is te vinden in appendix B.
5. De *team bar* geeft de geselecteerde teamleden voor een project weer.
6. Dit een grafische representatie van een *teamlid.*
7. Het klikken op een teamlid uit de team bar opent het *teamlid informatie venster.* De huidige rol, vaardigheden en relaties zijn hierin beschreven.
8. Dit is de *control bar*, en is bedoeld voor het aanbieden van extra interactie mogelijkheden in bepaalde situaties.
9. De *event alarm* knippert een nieuwe event klaar staat op afgespeeld te worden. Dit kunnen minigames, verhaalelementen of teamwork conflicten zijn.
10. De *game world* bevat de speelruimte voor een project.

8 Conclusie

Het is mogelijk om het concept van teamwork her te gebruiken in verschillende games. De veelzijdigheid van teamwork maakt het mogelijk om teamwork op verschillende manieren in games toe te passen. Deze verschillende manieren zijn in een model vast te leggen en dienen als basis voor teamwork in een spel. Deze modellen zijn her te gebruiken.

Het doel van een teamwork model is om een team en de variabelen die daar van invloed op zijn, in kaart te brengen. Zij zijn te vergelijken met *design patterns* uit de software engineering wereld. Een *design pattern* is een terugkerende structuur van communicerende componenten dat een oplossing biedt voor een *software design* probleem [Vli00]. Een *design pattern* is geen afgerond ontwerp dat direct naar code te transformeren is. Het is een *template* die aangeeft hoe een probleem opgelost kan worden.

Hetzelfde geld voor een teamwork model. Zij biedt een oplossing om teamwork in een game te verwerken. Enkele teamwork modellen zijn: *experience-system*, *ranking-system* en *commander-system*. Deze teamwork modellen staan los van de uitwerking van het model. Zo zijn de Proof of Concept, “Final Fantasy VIII” [1] en “Lost Odyssey” [2] gebaseerd op de *experience-system*, maar zijn ze alle drie inhoudelijk compleet verschillend van elkaar.

Naast hergebruik van teamwork modellen is het mogelijk om “teamwork componenten” her te gebruiken. Dit zijn uitgewerkte onderdelen van een teamwork model. Zij bevatten dezelfde eigenschappen als software componenten (wordt meerdere malen gebruikt, niet context specifiek, samen te stellen met andere componenten, ingekapseld). Enkele teamwork componenten die ook in de PoC voorkomen zijn: team, teamleden, relaties, vaardigheden en chat-box.

Deze componenten zijn bepalend voor de inhoud van een spel en zijn daarom deel van de *soft architecture*. Ondanks dat zij dezelfde eigenschappen hebben als software componenten uit de *hard architecture* worden zij niet net zo vaak gebruikt. De *sound engine* uit de *hard architecture* bijvoorbeeld, zorg voor de geluidsweergave van een spel. Bijna elk spel maakt gebruik van muziek en/of geluidseffecten. Dit betekent dat een *sound engine* voor bijna elk spel gebruikt kan worden. Een team component (component dat de samenstelling van een team bijhoudt) kan echter alleen gebruikt worden in spellen waar teams en teamwork in voorkomen.

[1] *http://www.final-fantasyviii.com*
[2] *http://www.xbox.com/en-US/games/splash/l/lostodyssey/*

Aanbevelingen

Hier volgen een aantal aanbevelingen voor Getronics PinkRoccade.

- Mocht Getronics PinkRoccade zelf serious games willen ontwikkelen als toegevoegde waarde voor haar producten, dan is het van belang daar een *template*/framework voor te ontwikkelen. Met deze *template*/framework worden dezelfde type games gemaakt. Omdat de serious game als toegevoegde waarde van een product dient, is originaliteit niet van belang. Dit komt omdat de doelgroep van de game steeds verschilt. Een game voor een product dat geleverd is aan de UWV is alleen voor UWV medewerkers bestemd, en een game voor een product dat geleverd is aan de Belastingdienst is alleen bestemd voor Belastingdienst medewerkers.
  De *template*/framework dient om serious games aan de lopende band te ontwikkelen. Aan te raden is om te beginnen met een *template*/framework dat gebaseerd is op de *experience-system*, omdat in dit systeem het doel is dat de speler groeit en meer kennis en vaardigheden ontwikkeld, waardoor het systeem geschikt is voor verschillende serious games.

- De games industrie is inmiddels groot en redelijk “volwassen”. Het serious gaming aandeel hiervan is nog redelijk jong. Een periode van één tot twee jaar is nodig om Getronics PinkRoccade te laten groeien tot een belangrijke speler in de serious gaming markt. In deze periode is het van belang om vier tot zeven serious games te ontwikkelen en vrij te geven. Dit zal Getronics PinkRoccade naambekendheid geven binnen de serious gaming markt.

- Getronics PinkRoccade kan een begin maken met een game over zichzelf. De game geeft informatie over GPR, kan een rondleiding van verschillende gebouwen en minigames bevatten. Deze game kan op de website van GPR geplaatst worden en dient als een leuke manier van kennis maken met GPR. Vervolgens kunnen spellen ontwikkeld worden voor interne producten binnen GPR. Producten waarvoor GPR trainingen geeft aan haar eigen werknemers. Na de ontwikkeling van deze serious games heeft GPR voldoende kennis vergaard om games te ontwikkelen voor producten die geleverd worden aan externe klanten.

- Naast het geven van een toegevoegde waarde aan GPR producten, zijn er andere kansen die met serious games gegrepen kunnen worden. Educatie en gezondheidszorg zijn enkele voorbeelden. Onder basis en middelbare school scholieren zijn *handhelds* (mobiele gaming consoles zoals Playstation Pocket [1] en Nintendo DS [2]) erg populair. Zij kunnen als platform worden ingezet voor serious games om bijvoorbeeld een nieuwe taal te leren (Engels, Frans, Duits en Spaans zijn de standaarden op de middelbare school). In de gezondheidszorg kunnen games voor de Nintendo Wii [3] worden ontwikkeld die zorgen voor nodige lichamelijke inspanning.

Toekomst

Naar aanleiding van dit onderzoek zijn een aantal vervolgonderzoeken mogelijk.

- In deze scriptie is een teamwork model dat gebaseerd is op de *experience-system* ontwikkeld. Het is interessant om verschillende teamwork modellen in kaart te brengen. Zij moeten flexibel en uitbreidbaar zijn, zodat zij in verschillende projecten toegepast kunnen worden. Door het in kaart brengen van de meest voorkomende teamwork mogelijkheden is het voor een spel ontwikkelaar eenvoudiger om teamwork in een spel toe te passen.

- In deze scriptie is voor het high-level concept, teamwork, eerst een model ontwikkelt voordat het in een spel kan worden toegepast. Vervolg onderzoek kan modellen ontwikkelen voor andere high-level concepten dan teamwork.

- Ontwikkeling van een spel kost tijd. Een game framework dat alleen maar “aangekleed” hoeft te worden kan het proces versnellen. Er bestaan verschillende type games en zien er op software niveau verschillend uit. Verder onderzoek kan frameworks ontwikkelen voor verschillende type games (zoals race game, *role playing game*, *first person shooter* en platform games). Deze frameworks kunnen vervolgens omgezet worden in level/game editors, die de ontwikkeling van een game versimpelen. Deze frameworks kunnen de originaliteit van een spel limiteren, maar dit is voor de serious games van GPR (nog) geen punt.

[1] *http://nl.playstation.com/psp/*
[2] *http://ms.nintendo-europe.com/dslite/*
[3] *http://nl.wii.com/*

9 Bronnen

[Bau03] Baumann, T.M.
Effective teamwork in online games.
2003, SAE College.

[BS04] Bourg, D.M.; Seeman, G.
AI for game developers.
2004, O'Reilly.

[FS05] Flynt, J. P.; Salem, O.
Software Engineering for Game Developers.
2005, Thomson

[Fri03] Friedman-Hill, E.
Jess in action.
2003, Manning Publications Co, ISBN 1-930110-89-8.

[GHJV94] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. M.
Design Patterns, Elements of Reusable Object-Oriented Software.
1994, Addison-Wesley Professional.

[Gar96] Gartner, B.
High performance through teamwork.
1996, Rosen Publishing Group.

[Gol98] Goleman, D.
Working with emotional intelligence.
1998, New York: Bantam Books.

[GSC+04] Greenfield, J.; Short, K.; Cook, S.; Kent, S.; Crupi, J.
Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools.
2004, Wiley, ISBN 0-471-20284-3.

[GVD08] Grote Van Dale.
Groot woordenboek.
2008, Van Dale, ISBN: 9789066484740.

[HH96] Harris, P.R.; Harris, K.G.
Managing effectively through teams.
1996, MCB UP Ltd.

[Jan02] Jansen, P.G.W.
Organisatie en Mensen. Inleiding in de bedrijfspsychologie voor economen en bedrijfskundigen.
2002, Uitgeverij Nelissen, Soest.

[JJ95] Johnson, D.W.; Johnson, R.T.
Active Learning: Cooperation in the College Classroom.
1995, Edina, MN: Interaction Book Co.

[JJ99] Johnson, D.W.; Johnson, R.T.
Learning together and alone: Cooperative, competitive, and individualistic learning (5th ed.).
1999, Needham Heights: Massachusetts: Allyn and Bacon.

[Joh97] Johnson, R. E.
Frameworks = (components + patterns).
1997, Communications of the ACM, volume 40 (issue 10) pagina 39 - 42.

[KS93] Katzenbach, J.; Smith, D.K.
The Wisdom of Teams.
1993, New York: HarperCollins.

[Lan00] Lanser, E.G.
Why you should care about your emotional intelligence.
2000, Healthcare Executive, (Nov/Dec), 6-11.

[LT01] Luca, J.; Tarricone, P.
Does emotional intelligence affect successful teamwork?
2001, School of Communications and Multimedia.

[MC05] Michael, D.; Chen, S.
Serious games: Games that educate, train, and inform.
2005, Thompson Publishing.

[Mil06] Millington, I.
Artificial Intelligence for Games.
2006, Elsevier Inc.

[NB03] Nash, S.; Bolin, C.
Teamwork from the Inside Out Fieldbook: Exercises and Tools for Turning Team Performance Inside Out.
2003, Davies-Black Publishing, ISBN:089106172X.

[PWC] PriceWaterhouseCoopers.
Dutch Entertainment & Media Outlook towards 2011: How to stay in touch with tomorrow's consumer. - Trends in the Netherlands 2007 – 2011*.

[Pur07] Purdy, J. A.
Serious games getting serious about digital games in learning.
2007, http://www.corpu.com/newsletter wi07/sect2.asp

[Rie00] Riehle, D.
Framework Design: A Role Modeling Approach.
2000, PhD thesis, Swiss Federal Institute of Technology, Zurich.

[RM04] Rollings, A.; Morris, D.
Game Architecture and Design: A New Edition.
2004, New Riders Publishing

[Rou01] Rouse, R.
Game design: theory & practice.
2001, Wordware Publishing Inc.

[SZ03] Salen, K.; Zimmerman, E.
Rules of Play: Game Design Fundamentals.
2003, The MIT Press, ISBN 978-0-262-24045-1.

[Sch04] Schwab, B.
AI Game Engine Programming.
2004, Charles River Media.

[SHW+61] Sherif, M., Harvey, O.J.; White, B.J.; Hood, W. R.; Sherif, C.W.
Intergroup conflict and cooperation: The Robbers Cave experiment.
1961, University of Oklahoma Book Exchange.

[SK08] Shuja, A. K.; Krebs, J.
IBM Rational Unified Process Reference and Certification Guide: Solutions Designer.
2008, Pearson Publishing.

[Szy02] Szyperski, C.
Component Software: Beyond Object-Oriented Programming. 2nd ed.
2002, Addison-Wesley Professional, ISBN 0-201-74572-0.

[Val07] Valich, T.
In-game voice communication comes of age.
2007, the Inquirer.

[Vli00] Vliet, H. van.
Software Engineering: Principles and Practice.
2000, John Wiley & Sons Ltd.

[Zim07] Zimmerman, J.
Toepassen van "serious games" ter ondersteuning van de opleiding binnen het bedrijfsproces ICT service management
2007, Getronics PinkRoccade.

10 Bijlagen

A. Gehanteerde definities

Hieronder bevindt zich een lijst van gebruikte termen met haar definities.

3D engine

Zie *graphics engine*.

3D Objecten

Driedimensionale weergave van een object, gebruikmakende van computertechnieken.

Actionscript

De scripttaal van Adobe Flash.

Add-on

Een uitbreiding op een computerspel of computerprogramma die extra mogelijkheden toevoegt.

Adobe Flash

Een computerprogramma waarmee animaties, webvideo's en webapplicaties (zoals spelletjes en gehele websites) gemaakt kunnen worden.

API

Een Application Programming Interface (API) is een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel (meestal in de vorm van bibliotheken).

Artificial Intelligence (AI)

De wetenschap die zich bezighoudt met het creëren van een artefact dat een vorm van intelligentie vertoont.

Bandbreedte

Geeft aan hoeveel data per seconde door een verbinding verstuurd kan worden.

Behavior

Artificiële gedragsvertoning.

Bug

Een fout in een computerprogramma, waardoor het zijn functie niet (geheel) volgens specificaties vervult.

C

Zeer efficiënte, flexibele en in hoge mate machineonafhankelijke hogere programmeertaal.

C++

Objectgeoriënteerde programmeertaal, gebaseerd op C.

Chat

Het voeren van een gesprek door het over en weer typen van tekst tussen twee of meer gebruikers van computers die zich meestal op verschillende locaties bevinden en die tegelijkertijd in het zelfde netwerk werken.

Chatbox

Een virtuele ruimte op het internet waar men kan chatten.

Compileren

Overbrengen van instructies geschreven in een hogere programmeertaal naar machinetaal of assembleertaal.

Computer code

Een, in de computer wetenschap, serie van statements en declaraties geschreven in een voor mensen leesbare computer programmeertaal.

Computer configuratie

Geheel van apparaten en programmatuur dat samen een computersysteem of -netwerk vormt.

Computer game

Zie *computerspel*.

Computer graphics

Breed gezien is het de manipulatie van visuele en geometrische informatie gebruikmakende van computertechnieken.

Computerspel

Spel dat via of met de computer wordt gespeeld.

Concept Art

Een vorm van illustratie waar het hoofddoel is om een visuele representatie van een ontwerp, idee en/of sfeer over te brengen voor gebruik in films, video games of stripalbums, voordat het in het eindproduct wordt gezet.

Data engine

Handelt het inladen en uitladen van data af voor een game.

Demo

Demonstratieversie van een computerprogramma.

Design document

Een geschreven schets van de ontwikkeling van een softwareproduct, geschreven door een software ontwerper om een software ontwikkeling team een globale begeleiding te geven van de architectuur van een software project.

Design pattern

Een generiek opgezette softwarestructuur, die een bepaald veelvoorkomend type software-ontwerpprobleem oplost. Het patroon geeft geen concrete oplossing, maar biedt een soort sjabloon, waarmee het ontwerpprobleem kan worden aangepakt. Het ontwerppatroon ziet er uit als een klassendiagram, waar de relatie tussen de verschillende klassen en objecten wordt weergegeven.

DirectX

DirectX is een verzameling van APIs die het voor programmeurs eenvoudiger maakt computerspellen te programmeren op Windows.

Drag & Drop

Een bewerking die gebruikt wordt in computerprogramma's. De bewerking houdt in dat er met de muis een onderdeel op het scherm wordt vastgepakt (klikken en de muisknop ingedrukt houden) en naar een andere plaats wordt gesleept door met de muis te bewegen.

Dynamics system

Zie *physics engine*.

Emoticon

Symbolen die emoties weergeven door middel van een plaatje of een combinatie van lees- en lettertekens.

Engine

Een engine is de softwarematige basis van een computerprogramma. Het genereert de werking van het programma.

Entertainment- en mediasector

Nederlandse entertainment- en mediasector bestaande uit televisie, radio, internet, videogames, kranten, tijdschriften, boeken, sport, themaparken en out-of-home adverteren segmenten (publicatie PriceWaterhouseCoopers).

Event

Een gebeurtenis in een computerprogramma waarop een programma kan reageren.

Event handler

Een asynchrone callback subroutine die de input afhandelt dat door een programma wordt ontvangen.

Experience-points

Vaak afgekort als xp, exp of ep, wordt veel gebruikt in rollenspellen en computer role playing games. Experience points zijn een verafspiegeling van hoe ver men staat in het spel.

Feature

Functionaliteit dat aangeboden wordt door een software applicatie.

First Person Shooter (FPS)

Een computerspel genre. Het speltype wordt vooral gebruikt bij shooters (schietspellen). De speler bekijkt de wereld vanuit de ogen van het personage dat hij/zij speelt.

Framework

Een herbruikbaar ontwerp voor een software systeem (of subsysteem). Een software framework kan support programma's, code libraries, scripting taal, of andere software bevatten, die de verschillende componenten van een software project bij elkaar brengt.

Game

Zie *computerspel*.

Game architectuur

De structuur van een game dat de onderdelen, datastroom en interacties tussen de componenten van het systeem definieert.

Game component

Een subsysteem binnen de game architectuur, dat voor een specifieke taak is ontworpen.

Game configuratie systeem

Een computerprogramma waarmee game variabelen (moeilijkheidsgraad, geluidvolume, keyboard configuratie, etc.) kunnen worden aangepast naar de wensen van de speler.

Game console

Computer die bestemd of geschikt is voor het spelen van computerspelletjes.

Game design

Het ontwerpproces van de inhoud en regels van een game.

Game development

Game element

Een instantie van een game component.

Game logic

Een game component dat de regels van een game omvat.

Gameplay

De manier waarop een gebruiker een computerspel ervaart of de elementen van het spel zelf. Hierbij kan worden gedacht aan de besturing maar ook de controle over het weergegeven visuele standpunt.

Geluidskaart

Uitbreidingskaart in een pc die zorgt voor de weergave van het geluid.

Grafische kaart

De interface tussen een computer en het beeldscherm en is een uitbreidingskaart voor de besturing van het beeldscherm.

Granulariteit

Granulariteit (letterlijk korreligheid) is een term die aangeeft in welke mate er detail-gegevens van entiteiten aanwezig zijn.

Graphical User Interface (GUI)

Een grafische gebruikersomgeving is een manier van interactie met een computer waarbij grafische beelden, widgets en tekst gebruikt worden.

Graphics

Zie *computer graphics*.

Graphics engine

Subsysteem van een computerprogramma dat de visuele weergave van een game voor zijn rekening neemt.

Cross-platform

Een programmeertaal, softwaretoepassing of hardwareapparaat dat werkt op meer dan één systeemplatform (bijvoorbeeld UNIX, Windows en Macintosh).

Hard architecture

De “fysieke” architectuur, zoals subsystemen die zorgen voor de interactie tussen computer hardware en de speler (de input/output architectuur).

Hardware

De elektronische en mechanische delen in en om computersystemen.

Horizontal solution

Zie *hard architecture*.

In-game

Momentopname gedurende het spelen van een computerspel.

Interface

Een interface is een intermediair waarmee twee systemen met elkaar communiceren.

Interpreter

Een speciaal computerprogramma dat programma's verwerkt die in een bepaalde programmeertaal geschreven zijn. Dit gebeurt door de instructies in de broncode van zo'n programma met de hulp van een parser te interpreteren - te vertalen in voor de processor begrijpelijke code - terwijl het wordt uitgevoerd. Dat is een andere benadering dan een compiler, die een programma al voor uitvoering (beter) geschikt maakt voor de processor.

Level

D omgeving waarin de speler speelt.

Level editor

Een computerprogramma waarmee levels gemaakt kunnen worden voor een computerspel.

Library

Een verzameling code (functies/routines) die door programma's kunnen worden gebruikt.

Megabyte

Een megabyte, afgekort MB, is 1000 kilobyte ofwel 1.000.000 byte. Een byte bestaat in deze context uit 8 bits.

Menusysteem

Een lijst van mogelijkheden die normaal gesproken niet direct zichtbaar is. Het is een onderdeel van de grafische gebruikersinterface.

Minigame

Een klein computerspel binnen een groot spel. Een minigame is altijd kleiner of eenvoudiger dan het spel waarin het zich bevindt.

Mod

Aangepaste versie van een bestaand spel.

Module

Zie *game component*.

Motion capture

De techniek die men gebruikt om bewegingen van mensen te kopiëren naar realistische animaties.

Multiplayer

De verzamelnaam voor het spelen van een computerspel met meerdere spelers.

Muziek systeem

Een computerprogramma dat muzikale noten, ritmes, etc., kan interpreteren.

Online Community

Een virtuele gemeenschap, internetgemeenschap of online gemeenschap is een groep mensen die communiceren en/of samenwerken, met behulp van vooral het internet of een andere informatietechnologie, in plaats van elkaar in levende lijve te ontmoeten.

Open-architectuur

Een type architectuur waarbij het mogelijk is om componenten toe te voegen, te upgraden, en uit te wisselen.

Patch

Softwareonderdeel dat wordt geïnstalleerd om fouten in een computerprogramma te verhelpen, prestaties te verbeteren e.d.

Physics engine

Een computer programma dat de natuurkundige modellen van Newton simuleert, gebruikmakende van variabelen als massa, snelheid, wrijving en wind resistentie.

Platform

Een hardware architectuur of software framework waar software op gedraaid kan worden.

Porten

Het aanpassen van software, met als doel deze te laten draaien op een ander operating system of CPU dan waarvoor de software oorspronkelijk voor was ontwikkeld.

Prerelease

Een gecompileerde versie van ontwikkelingsbestanden met het doel fouten er uit te halen.

Requirement

Een gedocumenteerde eis van wat een bepaald product of service moet zijn of doen.

RIA

Een Rich Internet Application (RIA) is een term die gebruikt wordt voor interactieve internetapplicaties, die het gevoel geven van een desktop programma (bijvoorbeeld een tekstverwerker of een agenda).

Role Playing Game (RPG)

Een overkoepelende term voor een bonte verzameling onderling nogal uiteenlopende games. Wat zij allen gemeenschappelijk hebben, is dat de spelers zich een imaginaire wereld voorstellen, waarin zij personages besturen en tijdens het spelen een verhaal vertellen.

Run-time

Beschrijft de operatie van een computer programma, de duur van zijn uitvoering, van begin tot beëindiging.

Save game

Een stuk digitale informatie over de voortgang van een speler in een computerspel.

Scripttaal

Een programmeertaal die geschikt is voor het schrijven van scripts, kleine programmaatjes om veel voorkomende taken (bijv. systeembeheertaken) te automatiseren, of om een grote maar eenmalige taak te verrichten.

SDK

Een Software Development Kit is een verzameling hulpmiddelen die handig zijn bij het ontwikkelen van computerprogramma's voor een bepaald besturingssysteem of type hardware, of voor het maken van software die een speciale techniek gebruikt.

Serious game

Een software applicatie die ontwikkeld is met game technologie en principes van game design voor een doel anders dan puur entertainment.

Soft architecture

De softwarematige architectuur waar een game van gemaakt is.

Softwarefabriek

Methodiek om software te produceren, met technieken die te vergelijken zijn met standaard fabrieken, zoals een auto fabriek.

Software systeem

Een systeem gebaseerd op software dat deel uitmaakt van een computersysteem (een combinatie van hardware en software).

Sound engine

Subsysteem van een computerprogramma dat het geluid van een game verwerkt.

Source code

De code die door de programmeur in een formele programmeertaal is geschreven. Dit staat tegenover de uitvoerbare code of machinetaal voor de processor zoals die door een compiler of interpreter vanuit de broncode gegenereerd wordt.

Speelruimte

Zie *level*.

Spel

Zie *computerspel*.

Streaming media

Verzamelterm voor de technologie om audio en video rechtstreeks via computernetwerken (zoals het internet) te distribueren.

String

Reeks bits of tekens.

Subsysteem

Zie *game component*.

Texture

In grafische computerprogramma's worden texturen (doorgaans aangeduid met het Engelse textures) gebruikt om 2D of 3D objecten te voorzien van een bepaalde structuur en uiterlijk.

Tool

Hulpmiddel, gereedschap voor het ontwikkelen van een game.

Triggeren

Het afvuren, in gang zetten, opwekken, van een proces binnen een computerprogramma.

Vertical solution

Zie *soft architecture*.

Video game

Zie *computerspel.*

Virtuele catalogus

Term voor een verzameling 3de partij software componenten, die gebruikt kunnen worden voor het ontwikkelen van games.

XML

Een standaard voor het definiëren van formele markup-talen voor de representatie van gestructureerde gegevens in de vorm van platte tekst.

B. Handleiding: teamwork score prototype

Start scherm:

Bij het openen van het programma zijn 3 balletjes te zien die met verschillende snelheden in een cirkelbaan voortbewegen.

- Bal: Elke bal representeert een team component
- Snelheid: De snelheid waarmee de bal beweegt representeert de prestatie die een team component levert. Des te hoger de snelheid, des te hoger de prestatie.
- Cirkelbaan: De straal van elke cirkelbaan representeert de betrokkenheid van een team component binnen een team. Des te kleiner de straal is, des te beter de relaties zijn met de andere team componenten.
- Overall teamwork: De overall teamwork waarde geeft een beeld van de samenwerking tussen de team componenten binnen het team. De waarde kan minimaal 0, en maximaal 100 zijn. Een hoge waarde betekent dat de samenwerking binnen het team goed is.
- Overall team performance: De overall team performance waarde geeft een beeld van de prestatie van het team. De waarde kan minimaal -100, en maximaal 100 zijn. Een hoge waarde betekent dat het team goed presteert.

Team component focus

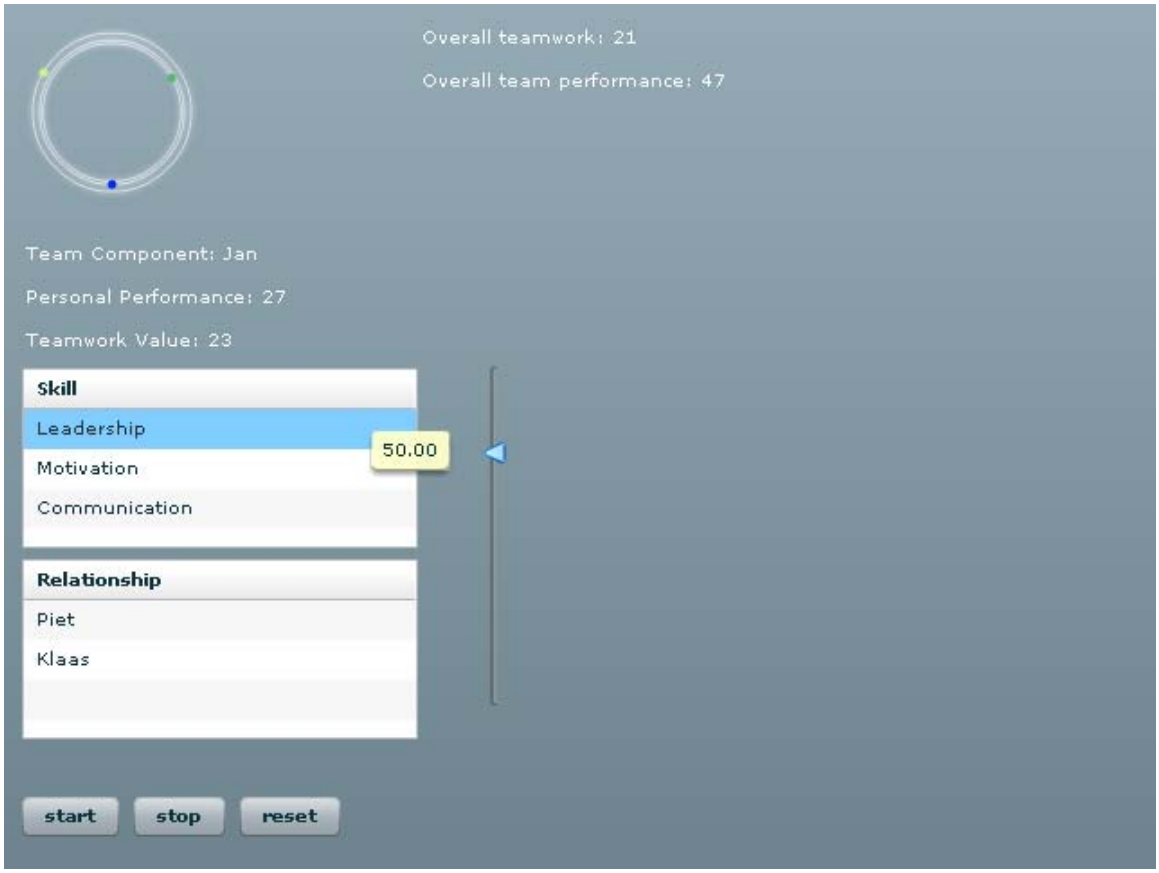
- stop: Door op de stop knop te drukken, wordt de cirkelbeweging gepauzeerd.

Wanneer de balletjes zich in een stilstaande situatie verkeren is het mogelijk om op één van de balletjes te klikken om meer informatie over een team component te vragen.

- Team Component: Dit is de naam van de gekozen team component.
- Personal Performance: Dit is een waarde tussen -100 en 100, welke de persoonlijke prestatie representeert. Een hoge waarde betekent dat de gekozen team component goed presteert.
- Teamwork Value: Dit is een waarde tussen 0 en 100, welke de persoonlijke teamwork waarde representeert. Een hoge waarde betekent dat de gekozen team component goed kan samenwerken met de andere team componenten.
- Skill: Dit is een lijst van persoonlijke vaardigheden van de gekozen team component, die de “personal performance” waarde en/of de “teamwork value” waarde kunnen beïnvloeden.
- Relationship: Dit is een lijst van relaties die de gekozen team component heeft. De waardes die aan de relaties zijn gekoppeld bepalen de “teamwork value” waarde.

Nieuwe waardes toekennen aan vaardigheden en relaties

Door op één van de items uit de skill lijst of uit relationship lijst te klikken, komt er een slider te voorschijn.



- Slider: Door te slider omhoog of omlaag te bewegen wordt aan de gekozen item een nieuwe waarde toegekend. De nieuwe waarde heeft direct invloed op de “personal performance” of de “teamwork value” waarde.

- start: Door op de start knop te drukken, is het start scherm te zien. De balletjes zullen weer in de cirkelbaan voortbewegen.

- reset: De reset knop levert een gepauzeerde start scherm op waarbij de balletjes de start positie aannemen.

C. Handleiding: video puzzel prototype

De multiplayer video puzzel is de video puzzel die met zijn tweeën tegelijk gespeeld wordt. De spelers leggen om de beurt een puzzelstuk op de juiste plek.

Flash beveiliging
De multiplayer video puzzel maakt gebruik van SmartFoxServer om meerdere spelers met elkaar te verbinden. Daarnaast maakt de video puzzel momenteel gebruik van locale media die dienen als inhoud voor de puzzel. Dit betekent dat de video clips niet van een video server worden gestreamd, maar van een locale plek worden opgevraagd.

De beveiliging van flash laat netwerk verbinding gezamenlijk met toegang tot locale data niet zomaar toe. Voor de multiplayer video puzzel betekent dit dat de video clips niet worden ingeladen.

Om de video puzzel toegang te geven tot locale data, dan zal de video puzzel tot een “vertrouwde flash applicatie” benoemd moeten worden.

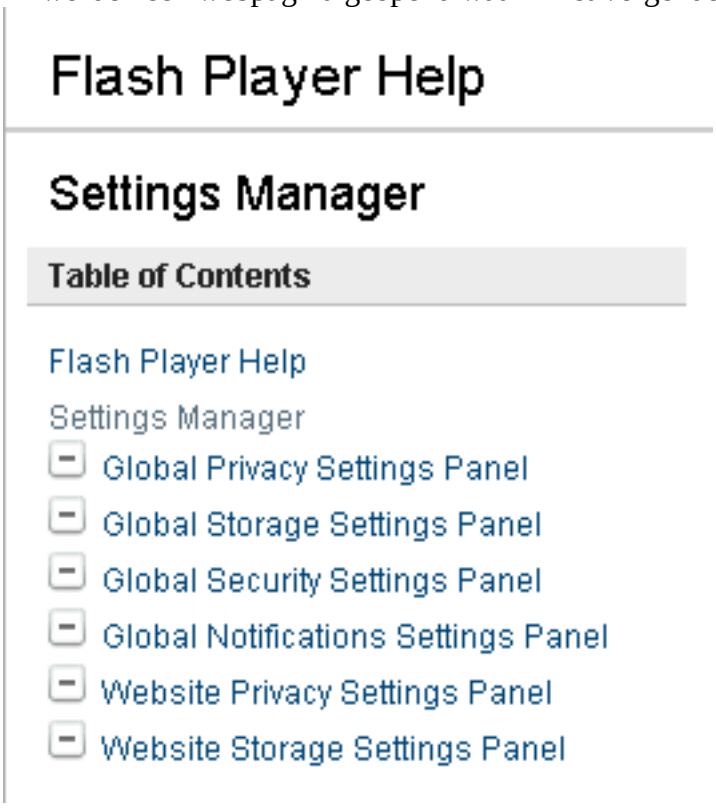
Dit gaat als volgt:

- Terwijl de flash player open staat, klik op de rechter muis knop en kies “Settings”.

Het onderstaands scherm is te zien:

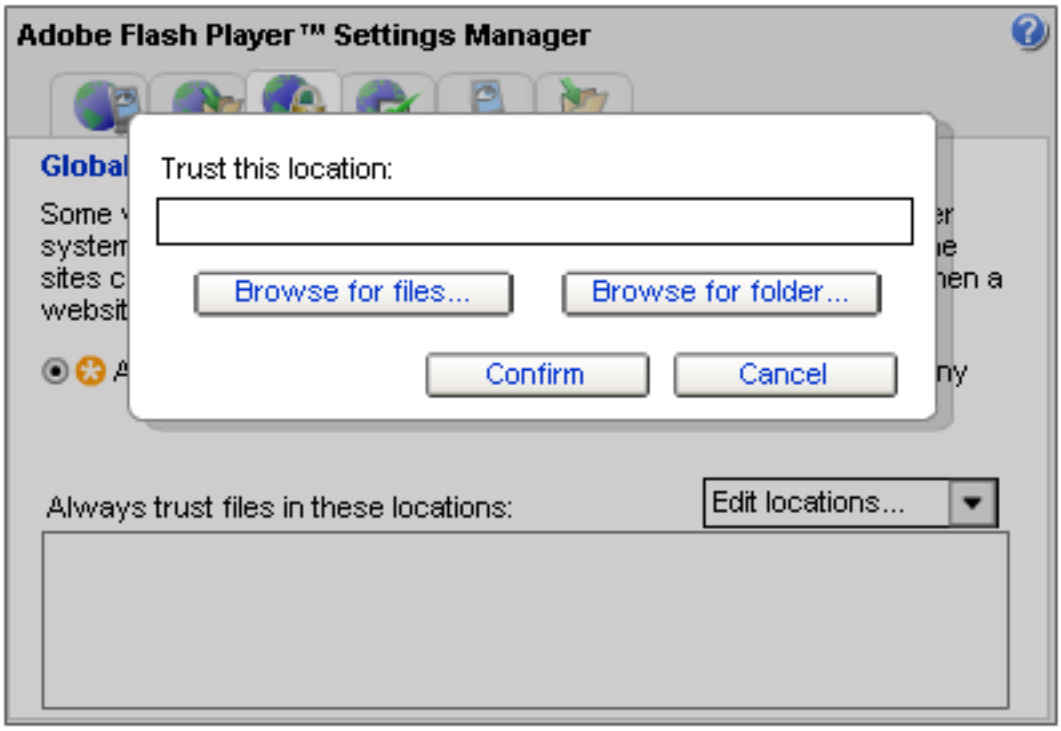
- Klik op de “Advanced” knop.

Er worden een webpagina geopend waarin het volgende scherm te zien is:



- kies “Global Security Settings” uit de lijst

- klik in het bovenstaande scherm eerst op “Edit locations…” en vervolgens op “Add location…”



- klik op “Browse for folder …” en kies de folder waar de video puzzel zich bevindt om de video puzzel bestanden tot vertrouwde bestanden te benoemen.

Na deze beveiligingsaanpassing kan de video puzzel probleemloos gestart worden.

In een toekomstige versie van de video puzzel zal er in plaats van locale media, streaming media gebruikt worden, waardoor deze beveiligingsaanpassing niet meer nodig is.

Login

Wanneer de video puzzel wordt opgestart, wordt er direct een verbinding gemaakt met de “SmartFoxServer”. Als een succesvolle verbinding met de server gelegd is, wordt het login scherm gepresenteerd.

In dit scherm moet een unieke gebruikersnaam worden ingevuld.

Na het klikken op de “Enter” knop, wordt het chat-scherm gepresenteerd.

Chat Scherm

Wanneer het chat-scherm gepresenteerd wordt, wordt de speler direct aan de “Lobby” room toegevoegd. De Lobby room is de enige ruimte waar ge-chat kan worden zonder een spel te spelen.

1. In dit kader worden alle chat teksten van de Lobby ruimte gepresenteerd.

2. Dit is een lijst van alle aanwezige ruimtes. De Lobby ruimte kan maximaal 50 gebruikers bevatten. Alle andere ruimtes zijn game ruimtes en kunnen maximaal 2 gebruikers bevatten.
   De gebruiker bevindt zich in de ruimte die geel opgelicht is. Een gebruiker kan een andere ruimte betreden door op een ongeselecteerde ruimte uit de lijst te klikken.
   Wanneer er op een “Game ruimte” wordt geklikt waar zich 1 gebruiker in bevindt, dan wordt de gebruiker naar de spelruimte geleidt en wordt er een spel gestart.

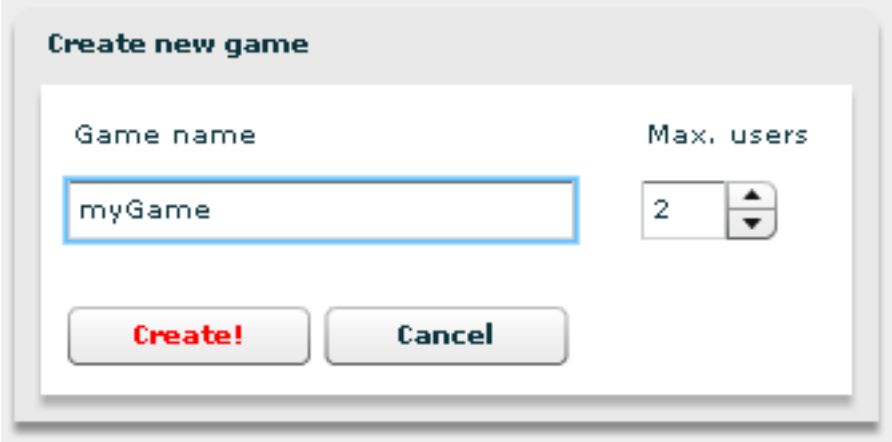
3. Met deze knop kan een spelruimte gecreëerd worden. Wanneer een gebruiker een spelruimte creëert, dan wordt de gebruiker daar automatisch aan toegevoegd. De spelruimte wordt gepresenteerd en de gebruiker wacht op een tweede speler.

4. Dit is een lijst van gebruikers die aanwezig zijn in dezelfde ruimte.
   Een persoonlijke boodschap naar een andere gebruiker kan worden gestuurd door op de andere gebruiker uit de lijst te klikken.

5. Met deze “tekst input” ruimte en de “send knop”, kan een publieke boodschap verzonden worden. De boodschap wordt in het chat-kader gepresenteerd aan alle gebruikers in dezelfde ruimte.

6. Met de “logout” knop kan een gebruiker zich van de server afmelden. Het logout-scherm wordt gepresenteerd.

Spel creëren
Door op de “Create Game” knop te klikken in het “Chat Scherm”, wordt het volgende pop-up scherm gepresenteerd



- Om een nieuwe spelruimte te creëren moet een naam voor de ruimte ingevuld worden.
- Vervolgens kan er op de “Create!” knop geklikt worden om de spelruimte te creëren.
- Om het aanmaken van een nieuwe spelruimte te annuleren kan er op de “Cancel” knop geklikt worden.
- Het “Max. users”, refereert naar het maximale aantal spelers, dat gezamenlijk de video puzzel kunnen spelen in de aangemaakte spelruimte. Het maximale aantal spelers staat momenteel vast op 2 spelers, en kan niet worden veranderd. In een toekomstige versie van de video puzzel is het mogelijk om meer dan 2 spelers in een spelruimte uit te nodigen.

Nadat een gebruiker een spelruimte heeft aangemaakt, wordt deze direct naar de spelruimte geleid. Omdat de gebruiker op dat moment de enige speler in de ruimte is, wordt het volgende pop-up scherm gepresenteerd.

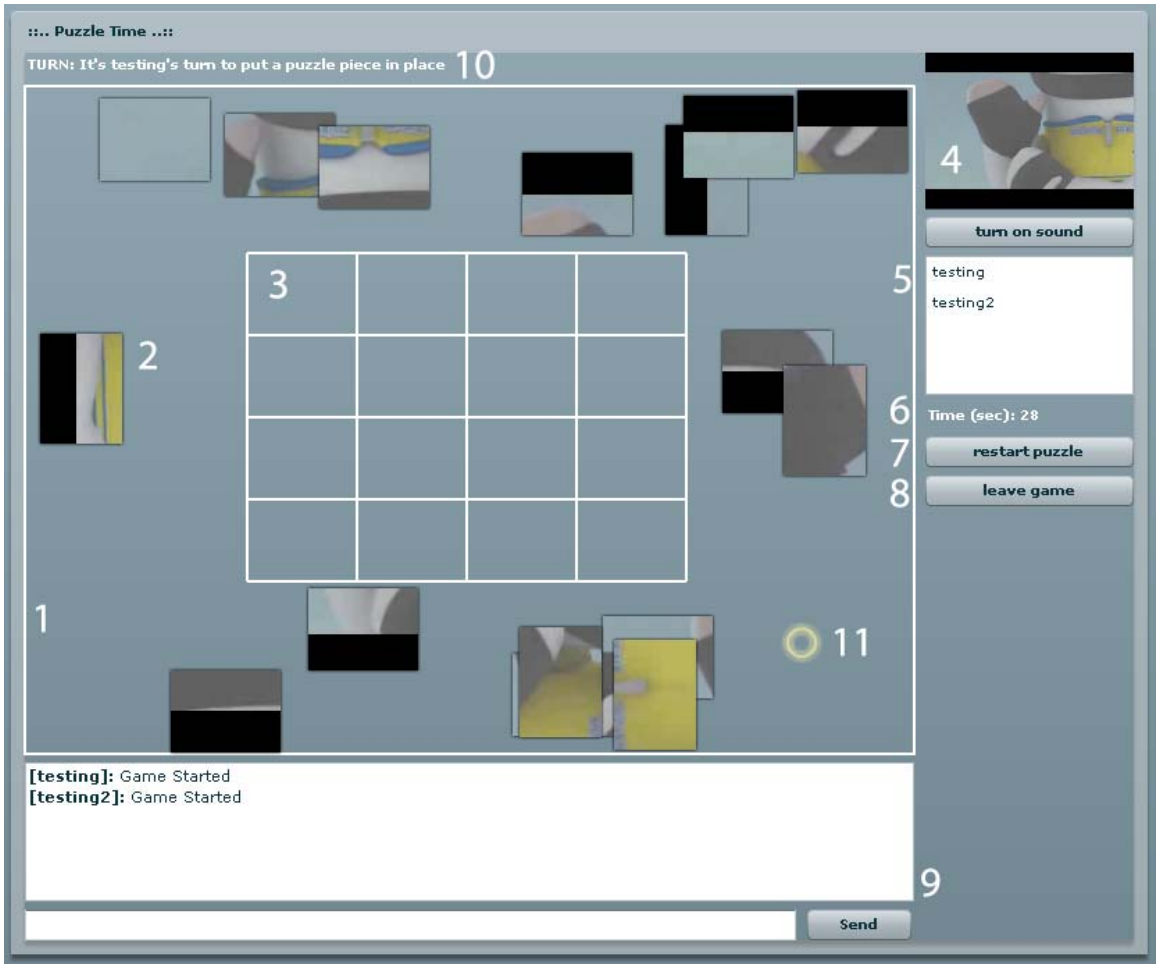
- Als een gebruiker niet langer wil wachten op een tweede speler, dan kan deze op de “Cancel” knop klikken. De gebruiker wordt dan naar het “Chat-scherm” geleid en de aangemaakte ruimte wordt verwijderd.
- Mocht een tweede speler de ruimte betreden, dan zal dit scherm zich verbergen en wordt het spel gestart.

Spel betreden
Nadat een gebruiker een spel gecreëerd heeft, is de naam van de ruimte te zien in de “Room List” van het “Chat-scherm”.

Als er maar één speler in een spelruimte aanwezig is, dan kan een gebruiker op de spelruimte in de “Room List” klikken.

De gebruiker wordt automatisch naar de spelruimte geleid en het spel wordt gestart.

Puzzle Time
Wanneer de puzzel gestart is, wordt het volgende scherm gepresenteerd.



1. Restrictiekader: Het witte kader om de puzzel is het restrictiekader en definieert de speelruimte. De puzzelstukjes kunnen niet buiten dit kader geplaatst worden.
2. Puzzelstuk: Wanneer de puzzel gestart wordt, worden alle puzzel stukjes op een willekeurige plek binnen het restrictiekader, en buiten de puzzelrooster geplaatst.
3. Puzzelrooster: De rooster dat zich in het midden van het restrictiekader bevindt is de puzzelrooster. Deze rooster definieert de posities waar de puzzelstukjes geplaatst moeten worden.
4. Preview: Dit kader geeft een preview van de puzzel. In geval dat de inhoud van de puzzel een video is, kan het geluid aan en uit worden gezet met de bijbehorende knop.
5. Gebruiker lijst: Dit is een lijst van de spelers die aanwezig zijn in de spelruimte. Deze lijst heeft dezelfde karakteristieken als de gebruiker lijst van het “Chat-scherm”.
6. Tijd: Hier wordt bijgehouden hoe lang het duurt voordat de puzzel opgelost wordt.
7. Restart knop: Met deze knop kan de puzzel opnieuw gestart worden. Alle puzzelstukjes worden op een willekeurige plek binnen het restrictiekader, en buiten de puzzelrooster geplaatst.

8. Spel verlaten: Met deze knop kan de spelruimte verlaten worden. Het spel wordt automatisch gestopt en beide spelers worden automatisch naar het “Chat-scherm” geleid.
9. Chat: In deze ruimte kunnen de spelers met elkaar chatten. De chat functie komt overeen met de chat functie van het “Chat-scherm”.
10. Speler beurt: In dit spel moeten de spelers om de beurt een puzzelstuk op de juiste plek plaatsten. In deze ruimte wordt bijgehouden wie er aan de beurt is.
11. Pointer: Wanneer een speler niet aan de beurt is, kan de speler de andere speler helpen door met de linker muis knop binnen het restrictiekader te klikken. Op het aangeklikt punt wordt tijdelijk een cirkel gepresenteerd. Hiermee kan de speler, die niet aan de beurt is, de andere speler helpen door puzzelstukjes en posities aan te wijzen.

Puzzel interactie
De puzzel wordt opgelost wanneer alle puzzelstukjes op de juiste plaats, met de correcte rotatie, binnen de puzzelrooster geplaatst worden. De puzzelstukjes moet om de beurt gelegd worden.

Wanneer een speler aan de beurt is om een puzzelstukje op de juiste positie te plaatsen, kan deze de volgende acties uitvoeren.

Roteren
De puzzelstukjes kunnen worden geroteerd door er met de linker muis knop op te klikken. Bij elke rotatie wordt het puzzelstukje met 90 graden naar rechts gedraaid.

Slepen
De puzzelstukjes kunnen gesleept worden door de linker muis knop op een puzzelstukje ingedrukt te houden, en vervolgens de muis binnen het restrictiekader te bewegen. Het puzzelstukje wordt op de nieuwe positie geplaatst wanneer de linker muis knop wordt losgelaten.

Plaatsen
Wanneer een puzzelstukje met de juiste rotatie naar het correcte kader binnen de puzzelrooster wordt gesleept, dan wordt het puzzelstukje in de puzzelrooster geplaatst. De plaatsing wordt met een visueel effect weergegeven. Het geplaatste puzzelstukje kan niet meer geroteerd of gesleept worden.

Wanneer een speler niet aan de beurt is, kan de speler de volgende actie uitvoeren.

Aanwijzen:
De speler kan met de linker muis knop binnen het restrictiekader te klikken. Op het aangeklikt punt wordt tijdelijk een cirkel gepresenteerd. Hiermee kan de speler, die niet aan de beurt is, de andere speler helpen door puzzelstukjes en posities aan te wijzen.

Server en Media aanpassen
Dit spel is afhankelijk van de “SmartFoxServer”. Om het IP adres, poort en zone van server te wijzigen, kan het “serverData.xml” bestand aangepast worden, dat zich in de “assets” folder bevindt. Daarnaast kan er ook verwezen worden naar een andere video bestand

D. Minigame 1: communicatie

Dit spel is een communicatie minigame, waarbij je als speler de rol inneemt van de werknemer(s) en met de klant communiceert. Het is de bedoeling dat de speler op deze manier meer details over het project weet te bemachtigen en vragen van de klant correct weet te beantwoorden.

Gameplay

Communicatie gaat via chat, waarbij de speler vragen kan stellen en antwoorden kan geven via een chat box.
De speler kan op 4 manieren punten behalen.

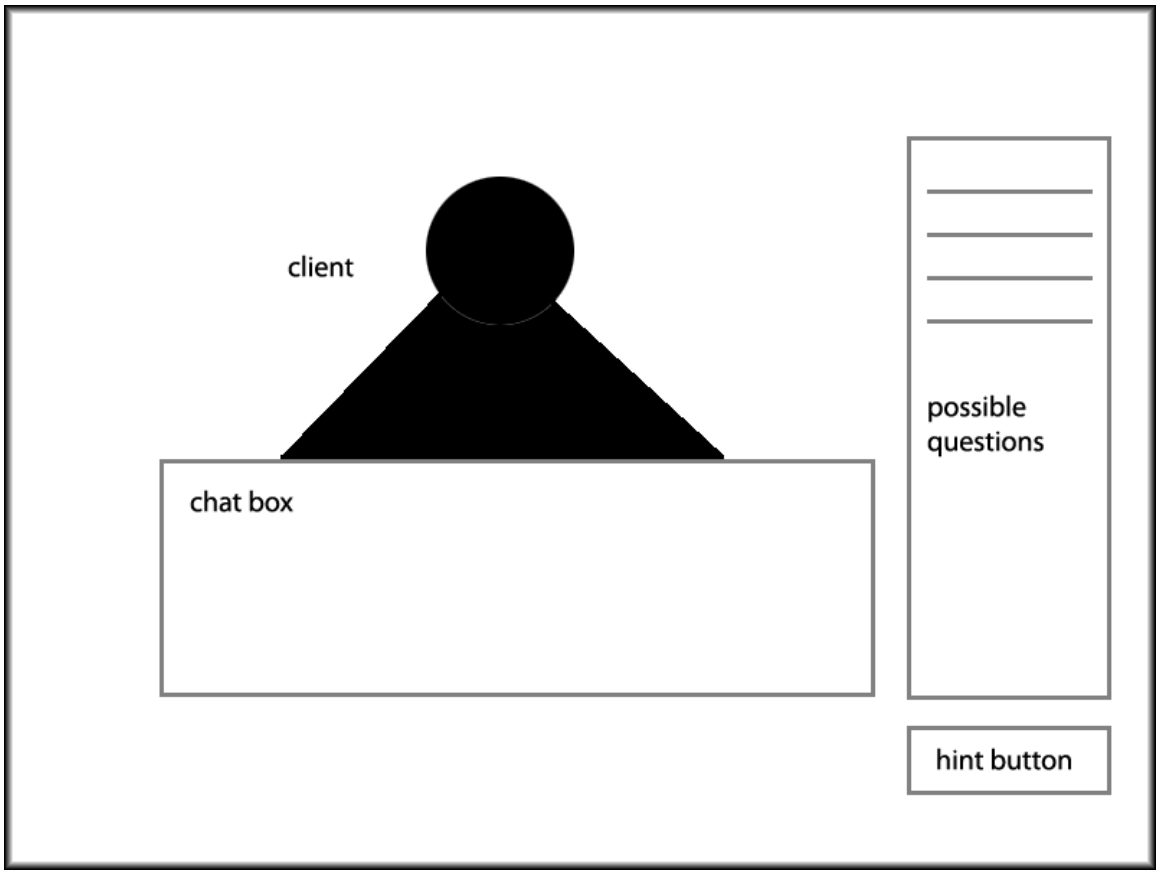
1. Door vragen te stellen waarop de klant meer details vrijgeeft over de opdracht.
2. Door vragen van de klant correct te beantwoorden (bonus punten).
3. Door snel correcte vragen te stellen en correcte antwoorden te bieden (tijd bonus).
4. Door vragen te stellen en antwoorden te geven die de klant niet begrijpt, of al aan de orde zijn geweest (minpunten).
5. Door het vragen van hints, voor nog onbeantwoorde vragen. Dit kost natuurlijk punten.

Het spel is tijdgebonden. De hoeveelheid tijd dat de speler krijgt om het spel af te ronden is afhankelijk van het aantal punten dat de speler kan winnen.
Er zullen vooral 3 lijsten worden opgesteld.

1. Een lijst van correcte vragen, die de speler kan stellen, met het bijbehorende antwoord van de klant. Ook wordt in deze lijst de hints verwerkt.
2. Een lijst van vragen die de klant kan stel, met het bijhorende antwoord van de speler.
3. Een lijst van verboden woorden, waar de speler extra op bestraft kan worden. (b.v. bepaalde terminologie).

Tijdens het spelen van het spel is in het scherm altijd een aantal regels te zien. Het aantal correspondeert met het aantal vragen van lijst 1. Wanneer de speler een correcte vraag stelt, dan zal één van de regels worden ingevuld. Op deze manier weet de speler hoeveel vragen hij nog te gaan heeft. Als de speler dezelfde vragen stelt, dan zal deze er natuurlijk voor bestraft worden.
Ook wordt de tijd weergegeven, dat aangeeft hoeveel seconden de speler nog heeft om punten te behalen.

Het plaatje hieronder geeft een abstracte voorstelling weer van de visuele representatie van het spel.



Technisch

Het bouwen van een model dat natuurlijke zinnen kan interpreteren is complex en ingewikkeld. Ik zal het daarom simpel houden en controleren op steekwoorden. Wanneer de steekwoorden in één zin worden geleverd, dan zal dit als een correcte vraag of antwoord worden beschouwd. Dit betekend dat de speler met slechte grammatica en verkeerde zinsopbouw toch optimaal kan scoren. In een toekomstige versie van dit spel is het mogelijk om deze functie te verbeteren.

XML

Hieronder wordt een simpele XML weergave gegeven van het formaat waarmee de lijsten gedefinieerd worden.

Lijst 1: Correcte vragen die de speler kan stellen

```
<questions>
   <question>
      <id></id>
      <sentence></sentence>
      <answer_to_question></answer_to_question >
      <keywords>
         <keyword></keyword>
      </keywords>
      <hints>
         <hint></hint>
      </hints>
   </question>
</questions>
```

Lijst 2: Correcte antwoorden die speler kan geven

```
<answers>
   <answer>
      <id></id>
      <sentence></sentence>
      <question_for_answer></question_for_answer >
      <keywords>
         <keyword></keyword>
      </keywords>
      <hints>
         <hint></hint>
      </hints>
   </answer>
</answers>
```

Lijst 3: Verboden woorden

```
<forbidden_words>
   <forbidden_word>
      <word></word>
      <weight></weight>
      <reason></reason>
   </forbidden_word>
</forbidden_words>
```

Webdesign Project

Aan de hand van het beschreven formaat, zijn hieronder 3 lijsten samengesteld voor het webdesign project.

Lijst 1: Correcte vragen die de speler kan stellen

- Question 1
- Id: resolution
- Sentence: “For which screen resolution do you want the website to be optimized?”
- Answer to question: “I would like the website to be optimized for visitors who may have smaller displays. I believe a resolution of 800x600 should be maintained.”
- Keywords: screen, resolution
- Hints: resize, big screen, small screen, pixels

- Question 2
- Id: web browser
- Sentence: “Do you want the website to be optimized for a specific web browser?”
- Answer to question: “I’d like the website to be optimized for all possible web browsers. Of cource.”
- Keywords: web browser
- Hints: Opera, Firefox, Mozilla, Netscape, Explorer.

- Question 3
- Id: database
- Sentence: “Should the content for the website be stored in a database?”
- Answer to question: “Yes. Information may change over time. In that case changes only need to be made in the database.”
- Keywords: database
- Hints: information storage, dynamic, SQL

- Question 4
- Id: programming language
- Sentence: “Is there a specific programming language you want to include or exclude from the website?”
- Answer to question: “What’s important to us as a company is that the website can be viewed by the majority of internet users. Programming languages that acquires the user to download third party software should be omitted, with the exception of the broadly used flash player plug-in.”
- Keywords: programming language
- Hints: HTML, JAVA, PHP, code.

- Question 5
- Id: loading time
- Sentence: “What is the maximum loading time you want to maintain for a user with a relative slow connection?”
- Answer to question: “Loading time should be maintained at a maximum of 1 minute.”
- Keywords: loading time
- Hints: waiting, slow connection, initialize

- Question 6
- Id: Language
- Sentence: “Do you want the website to support various languages?”
- Answer to question: “We are an international company; therefore we would like the website to be viewed in the language the viewer prefers.”
- Keywords: language
- Hints: countries, foreign, communication

- Question 7
- Id: front page
- Sentence: “Do you perhaps already have a concrete idea of what kind of information should be displayed on the front page?”
- Answer to question: “The front page should be identified with our company image. Our logo and content should be displayed in an elegant and professional manner. We would also like to use the front page as promotion for new products.”
- Keywords: front page
- Hints: face, company image, logo, main page

- Question 8
- Id: web server
- Sentence: “Do you already have a web server for your website”
- Answer to question: “Yes. I will give you the access codes and server details after the interview.”
- Keywords: web server
- Hints: storage, web space

- Question 9
- Id: domain name
- Sentence: “Do you have a registered domain name for your website?”
- Answer to question: “Yes, we have.”
- Keywords: domain name
- Hints: web address, DNS

- Question 10
- Id: content
- Sentence: “Is there a document containing specific data about the content you want to present on your website?”
- Answer to question: “Yes, we have prepared some documents for you, containing information about our company, products and other information we want to see on the website.”
- Keywords: content
- Hints: company info, product details, information, data

Lijst 2: Correcte antwoorden die speler kan geven

- Answer 1
- Id: applications
- Sentence: “Which applications do you use for creating websites?”
- Question for answer: “We use a range of various programs depending on the task at hand. Some programs are Flash, Photoshop and Dreamweaver.”
- Keywords: various
- Hints: Flash, Photoshop, Dreamweaver, depends

- Answer 2
- Id: possible
- Sentence: “Is it possible show 3D content on the website?”
- Question for answer: “Yes it is, but because of the complexity of 3D it can take more time to develop than other content.”
- Keywords: yes it is.
- Hints: possibilities are endless

- Answer 3
- Id: database
- Sentence: “What kind of database will you be using for the website?”
- Question for answer: “There are several technologies that can be used, but we prefer using MySQL.”
- Keywords: MySQL
- Hints: open source, My Query

E. Minigame 2: design

In deze minigame is het de taak van de webdesigner om de voorpagina te ontwikkelen. Hiervoor heeft hij een aantal teksten en plaatjes ontvangen van de klant. Deze items moet de speler in een gegeven ruimte plaatsen. De speler krijgt elke keer een nieuw object om in de ruimte te plaatsen.
Hiervoor krijgt de speler aanwijzingen in woorden, die hints geven naar de optimale plaats in de ruimte. Deze woorden representeren de creatieve gedachtes van de webdesigner. Bij plaatsing van een object in de ruimte zal de speler direct feedback ontvangen van de webdesigner. Die zal zich uitdrukken met termen als: Genius, Good, Excellent, Bad, Ugly, I want to puke, etc.
De speler zal niet direct na plaatsing het object in de ruimte zien. Deze wordt verborgen gehouden. Aan het eind van het spel zal een compilatie worden gemaakt van alle geplaatste objecten. Ook hiervan wordt een eind oordeel gegeven.

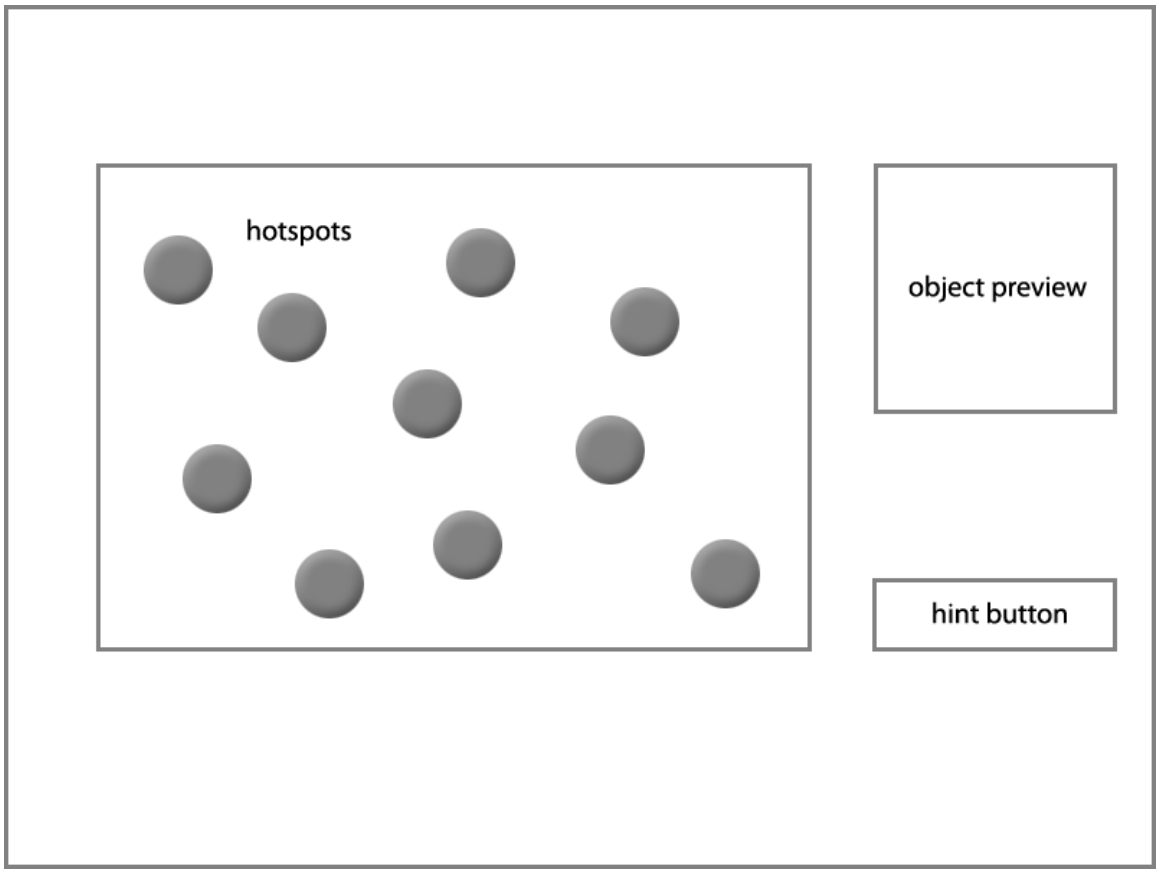
Hotspots
De ruimte waar de objecten in geplaatst moeten worden, is verdeeld in zogenaamde hotspots. Elke hotspot is een plaats waar de speler een object kan plaatsen.

Moeilijkheidsgraad
Het aantal objecten dat geplaatst kan worden is gelimiteerd, maar het aantal hotspots waar de objecten geplaatst kunnen worden hangt af van de moeilijkheidsgraad instelling. Een hoge moeilijkheidsgraad betekent dat de speler een keuze moet maken uit relatief meer hotspots.

Score
Bij plaatsing van een object krijgt de speler direct feedback. Wanneer de speler een object de meest optimale plaats geeft in de ruimte, dan zal hij daarvoor het meest beloond worden. Anders zal de score lager zijn, afhangend van de mate waarin de plaatsing afwijkt van de meest optimale hotspot. Ook zal de tijd worden bijgehouden. Bij snelle plaatsing van een object, kan de speler bonus punten ontvangen.

Hints
Mocht de speler er niet uit komen, dan kan deze een hint vragen, welke een aanwijzing geeft naar de beste plek in de ruimte. Deze verhelderende hints kosten de speler punten.

Het plaatje hieronder geeft een abstracte voorstelling weer van de visuele representatie van het spel.



Technisch
Voor het spel zijn een aantal initiële waardes nodig, namelijk:

- Resolutie: dit is de grootte van de webpagina
- Moeilijkheidsgraad

Elk object zal met een aantal kenmerken moeten worden gedefinieerd.
Object

- Id: unieke naamgeving voor het object
- Coördinaten: dit is de x, en y waarde van de meest optimale plek in de ruimte.
- Bron: dit is een verwijzing naar de visuele representatie van het object. Bijvoorbeeld een verwijzing naar een jpeg bestand.
- Hints: Dit zijn de verhelderende aanwijzingen naar de meest optimale plek in de ruimte. Deze hints zouden door het spel kunnen worden gegenereerd aan de hand van de vooraf gedefinieerde coördinaten. Maar in dat geval is het niet mogelijk om inhoudelijke hints te geven die alleen betekenis hebben voor specifieke objecten. Ook biedt het zelf instellen van hints de mogelijkheid voor een uniekere ervaring.

F. Gebruikte programma's

Hieronder wordt een lijst van programma's weergegeven die ik gebruikt heb voor het maken van de prototypes en proof of concept.

After Effects [1]

After Effects is een *digital motion graphics* en compositie software en is bedoeld voor de postproductie van film en video.

- Toepassing: converteren van de gemaakte video clips in Poser 3D naar FLV (flash video) formaat. Flash video laat transparantie toe, waardoor de video clips in lagen over elkaar geplaatst kunnen worden.
- Alternatieven: Combustion [2], Motion [3]

EditPlus [4]

Editplus is een editor dat hulpmiddelen bevat voor programmeurs, zoals: syntax highlight en spelling check.

- Toepassing: gebruikt voor het schrijven van PHP en XML bestanden.
- Alternatieven: PHP Edit [5], emacs [6].

Flash [7]

Authoring tool voor het maken van flash bestanden (SWF).

- Toepassing: Omzetten van FLV (flash video) bestanden naar flash (SWF) bestanden.
- Alternatieven: Flex Builder, Flex SDK.

Flash Player [8]

Flash Player is een multimedia en applicatie player. Flash Player speelt SWF bestanden af die gemaakt kunnen worden met onder ander Adobe Flash en Adobe Flex.

- Toepassing: De prototypes en proof of concept worden met de Flash Player afgespeeld.
- Alternatieven: Shockwave Player [9], Gnash [10], Microsoft Silverlight [11].

---

[1] *http://www.adobe.com/nl/products/aftereffects/*
[2] *http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=5562397*
[3] *http://www.apple.com/finalcutstudio/motion/*
[4] *http://www.editplus.com*
[5] *http://www.waterproof.fr*
[6] *http://www.gnu.org/software/emacs/emacs.html*
[7] *http://www.adobe.com/products/flash/*
[8] *http://www.adobe.com/products/flashplayer/*
[9] *http://www.adobe.com/products/shockwaveplayer/*
[10] *http://www.gnashdev.org/*
[11] *http://www.microsoft.com/silverlight/default.aspx*

Flex Builder [1]

Adobe Flex is een gratis open-source framework voor de ontwikkeling van cross platform *Rich Internet Applications* dat gebaseerd is op het eigen Adobe Flash [2] platform. platform. Flex applicaties kunnen gebouwd worden door gebruik te maken van de gratis Flex SDK. Adobe Flex Builder, een integrated development environment (IDE), kan de ontwikkeling van RIA's dramatisch versnellen.

- Toepassing: De prototypes en proof of concept zijn flex applicaties die ontwikkeld zijn met de Flex Builder.
- Alternatieven: Flex SDK [3], Flash.

Photoshop [4]

Photoshop is een programma voor het bewerken van digitaal beeldmateriaal (bijvoorbeeld foto's).

- Toepassingen: bewerken van digitaal beeldmateriaal die in de prototypes en proof of concept gebruikt worden.
- Alternatieven: The Gimp [5], Paint Shop Pro [6].

Poser [7]

Poser is een 3D rendering software pakket voor het poseren, animeren en renderen van 3D mens en dier figuren.

- Toepassing: Gebruikt voor de visuele representatie van teamleden in de proof of concept applicatie.
- Alternatieven: DAZ Studio [8], Avimator [9].

SmartFoxServer [10]

SmartFoxServer is een platform voor de snelle ontwikkeling van *multiuser* applicaties en games met Macromedia Flash MX, MX 2004, 8, Flex 2 and Flash CS3.

- Toepassing: Gebruikt voor de multiplayer versie van de video puzzel. SmartFoxServer dient als server waar spelers op in kunnen loggen om samen de video puzzel te spelen.
- Alternatieven: --

[1] *http://www.adobe.com/products/flex/*
[2] *http://www.adobe.com/products/flash/*
[3] *http://labs.adobe.com/technologies/flex/sdk/flex2sdk.html*
[4] *http://www.adobe.com/nl/products/photoshop/photoshop/*
[5] *http://www.gimp.org/*
[6] *http://www.corel.com/servlet/Satellite/nl/nl/Product/1184951547051*
[7] *http://my.smithmicro.com/win/poser/index.html*
[8] *http://www.daz3d.com/studio*
[9] *http://www.avimator.com*
[10] *http://www.smartfoxserver.com*

WAMP [1]

WAMP is een acroniem voor een combinatie van softwarepakketten waar een dynamische website op kan draaien. Windows (besturingssysteem), Apache (webserver), MySQL (databaseserver), PHP (scripttaal) of Perl of Python.

- Toepassing: Uitlezen en opslaan van XML bestanden. Deze XML bestanden zijn save files en configuratie bestanden voor de proof of concept applicatie.
- Alternatieven: Appserv [2], EasyPHP [3].

[1] *http://www.wampserver.com*
[2] *http://www.appservnetwork.com*
[3] *http://www.easyphp.org*

G. Technische problemen

Deze bijlage geeft een opsomming van gevonden technische problemen met Adobe Flex.

enterFrame [1]

enterFrame is een *event*, dat aangeeft dat de flash player zich in een nieuwe frame bevindt. Dit is net als een frame van een film. Een PAL film draait op 25 frames per seconde. Dit betekent voor de *enterFrame event* dat deze 25 keer in één seconde wordt *getriggered* (afgevuurd). Dit houdt niet op totdat de film stopt. Hetzelfde geld voor een Flex applicatie. De *enterFrame event* wordt alleen gebruikt als deze wordt ingezet. De event kan dus aan en uitgezet worden.

- Probleem: De teamwork score prototype, dat ook in de proof of concept wordt gebruikt, maakt gebruik van de *enterFrame event* om in elke frame een nieuwe positie voor de balletjes (bewegen in een cirkelbeweging en representeren de teamleden van een team) te berekenen. Wanneer een nieuw scherm in de PoC wordt geladen dan blijft de teamwork score op de achtergrond draaien. Het afvuren van de *enterFrame event* stoort de verwerking van nieuwe *events*. Dit resulteert in schermen die niet compleet worden ingeladen (er missen bijvoorbeeld een aantal kaders/menu's of er wordt helemaal niets ingeladen) en incorrecte layout (kaders/menu's worden niet op de vooraf ingestelde positie geplaatst of teksten worden niet gecentreerd).
- Oplossing/workaround: De storingen door de *enterFrame* event worden opgelost door het uitzetten van de *enterFrame event* wanneer het scherm met de teamwork score wordt verlaten. Wanneer het scherm met de teamwork score wordt aangeroepen, moet er eerst gewacht worden totdat het scherm compleet is ingeladen (wachten totdat alle *events* verwerkt zijn), voordat de *enterFrame event* kan worden aangezet.

SWFLoader [2]

SWFLoader laad en geeft een opgegeven SWF bestand weer. Typisch wordt *SWFLoader* gebruikt om een Flex applicatie te laden in een host Flex applicatie.

- Probleem: De PoC applicatie is de host applicatie en gebruikt SWFLoader om externe minigames (ook Flex applicaties) in te laden. De folderstructuur van de host applicatie ziet er als volgt uit:

```
- root/
  - game_host.swf          (PoC applicatie)
  - assets/                (folder met externe bestanden)
    - minigames/           (minigames floder)
      - puzzel/            (folder waar de puzzel minigame zich in bevindt)
        - puzzel.swf       (puzzel minigame applicatie)
        - assets/          (folder met externe bestanden voor de minigame)
          - *assets*       (videoclip, configuratie bestand, etc.)
```

[1] *http://livedocs.adobe.com/flex/2/langref/flash/display/DisplayObject.html#event:enterFrame*
[2] *http://livedocs.adobe.com/flex/2/langref/mx/controls/SWFLoader.html*

Binnen de puzzel.swf bestand worden de externe bestanden die nodig zijn voor de puzzel opgevraagd met de volgende link: “assets/*assets*”. De verwachting is dat de bestand gehaald worden uit: “root/assets/minigames/puzzel/assets/”. Maar de host applicatie vertaald het verzoek van de puzzel minigame naar: “root/assets/*assets*”. De *SWFLoader* “isoleert” de minigame niet van de host applicatie. Het resultaat is dat de puzzel minigame de bestanden niet kan vinden en er een foutmelding wordt gegenereerd. Het apart afspelen van de minigame, los van de host applicatie, geeft geen problemen.

- Oplossing/workaround: Om het probleem op te lossen heeft de minigame kennis nodig van de folderstructuur van de host applicatie. Met andere woorden. De link in de puzzel minigame moet verwijzen naar: “root/assets/minigames/puzzel/assets/”. Deze informatie wordt door de host applicatie aan de minigame gegeven doormiddel van een LocalConnect [1]. Op deze manier kan de minigame binnen en buiten de host applicatie zonder foutmelding gespeeld worden.

Flash Cache

Net als bekende web browsers zoals Internet Explorer en Mozilla Firefox gebruikt de Flash Player een cache om ingeladen bestanden (video, xml, tekst, plaatjes, etc.) tijdelijk op een locale plek (geheugen of harde schijf) te bewaren. Wanneer de Flash applicatie bijvoorbeeld een al ingeladen foto nodig heeft, dan gebruikt deze het bestand uit de cache. In geval dat de Flash applicatie op een externe web server draait, hoeft de foto niet nogmaals gedownload te worden.

- Probleem: Met de PoC applicatie is het mogelijk om configuratie bestanden (XML bestanden) in te laden en te wijzigen. Het aanbrengen van wijzigingen of het aanmaken van geheel nieuwe bestanden wordt gedaan met behulp van PHP. Bij het inladen van een configuratie bestand wordt deze in de cache van de Flash Player opgeslagen. Na het maken van de wijzigingen wordt de configuratie opgeslagen doormiddel van PHP. De configuratie bestand is nu gewijzigd, maar in de cache van Flash Player bestaat alleen het eerste ingeladen ongewijzigde configuratie bestand. Bij het laden van het bestand wordt het bestand uit de cache gehaald. Dit betekent een ongewijzigde versie van het spel. Om het gewijzigde configuratiebestand in te laden moet de applicatie eerst afgesloten en daarna weer worden opgestart.
- Oplossing/workaround: Gewijzigde configuraties worden opgeslagen in een *work directory* onder een nieuwe bestandsnaam (bijvoorbeeld: originele bestandsnaam+versienummer). De wijzigingen worden ook in het originele bestand doorgevoerd, maar voor het inladen van de configuraties wordt gebruik gemaakt van het laatste bestand uit de *work directory*. Op deze manier “denkt” de Flash Player dat het een nieuwe bestand aan het inladen is en bewaart deze elke keer weer in zijn cache.

[1] http://livedocs.adobe.com/flex/2/langref/flash/net/LocalConnection.html