It's All in the Game

Mark Overmars

I. Inleiding

De computer game industrie is in omvang de afgelopen jaren de filmindustrie gepasseerd. Het ontwikkelen van een computer game kost tegenwoordig vele miljoenen. Teams van tientallen programmeurs, animators, grafisch ontwerpers en geluidstechnici werken er vele jaren aan. Toch werd er tot recent in het onderwijs vrijwel geen aandacht aan besteed. Dit begint nu te veranderen. Aan een aantal universiteiten in het buitenland worden vakken over computer game ontwerp opgezet. Ook zijn er een aantal (commerciële) onderwijsinstellingen ontstaan die zich primair op game design richten. In Nederland kan men sinds twee jaar de opleiding *Design for Virtual Theatre and Games* volgen aan de hogeschool voor de kunsten in Utrecht. Deze opleiding richt zich sterk op de artistieke aspecten van game design. Aan de universiteit Utrecht wordt sinds afgelopen jaar binnen de opleiding Informatica door mij een vak *Game Design* verzorgd dat meer ingaat op de technische aspecten van het onderwerp. In dit artikel wil ik schetsen hoe dat vak is opgezet en de ervaringen van het eerste jaar geven. Hopelijk inspireert dit anderen om binnen het informatica-onderwijs op verschillende niveaus aandacht aan game design te geven.

Figuur 1 Een scène uit Black & White van Lionhead Studios

Traditioneel werden computer games door kleine groepjes programmeurs, bij voorkeur in assembly, op een zolderkamer in elkaar gehackt. Tegenwoordig is dat wel anders. Voor het ontwikkelen van een modern computerspel zijn specialisten op velerlei informaticagebieden nodig die goed moeten samenwerken. Daarnaast spelen ook de artistieke aspecten, zoals art-work, geanimeerde modellen, geluid en muziek en level-design een essentiële rol. Computer games gebruiken de modernste kennis op het gebied van computer graphics, kunstmatige intelligentie, mens-machine interactie, simulatie, gedistribueerde applicaties, beveiliging, en software engineering. Vaak stelt het gebruik van deze technieken binnen computer games extra eisen waar traditioneel niet zo veel aandacht voor is. Het bestuderen van deze technieken binnen de context van computer games is dan ook nuttig en leerzaam voor studenten. Het laat zien hoe kennis uit verschillende gebieden geïntegreerd kan worden en aangepast kan worden aan een specifiek toepassingsgebied.

Gezien het grote aantal onderwerpen dat een rol speelt bij het ontwerp van computer games is het nodig bij het opzetten van een vak op dit gebied keuzes te maken. Een voor de hand liggende keuze is om je te beperken tot de computer graphics aspecten. Hier is het meeste over bekend en er zijn goede tekstboeken op dit gebied [4,7]. Het geeft echter een erg beperkte blik op computer games. Hoewel belangrijk voor het succes van games zijn aspecten als ontwerp, speelbaarheid en kunstmatige intelligentie minstens zo belangrijk (en misschien wel

belangrijker). Vandaar dat ik in de opzet van het vak ook hiervoor de nodige ruimte heb gecreëerd. (Daar kwam bij dat de studenten reeds een vak computer graphics gevolgd hadden.) In de volgende secties ga ik in op de verschillende onderdelen in het vak.

II. Het Ontwerp

Wat is een goed computer game? Of misschien nog belangrijker: wat is een computer game? Dit is een vraag waarop een eenduidig antwoord ontbreekt. Er zijn hier vele, soms filosofische verhandelingen over geschreven (zie bijvoorbeeld [1,2]). Wanneer is iets een spel en wanneer is het speelgoed? Keywords die vaak genoemd worden zijn doelen en controle. Een spel moet een doel hebben en de speler moet controle over het verloop van het spel hebben. Maar dat is niet genoeg. Afhankelijk van het soort game spelen allerlei aspecten een belangrijke rol: verhaal, graphics, geluid, interactie, gedrag van tegenstanders, variatie, leercurve, mogelijkheden voor spelen via het internet, enzovoorts. Om een goed spel te kunnen ontwerpen is het van groot belang om te begrijpen wat de invloed van deze aspecten is. De beste manier om dit te doen is om een aantal succesvolle en minder succesvolle spelen te bekijken en te bediscussiëren waarom de spelen goed of slecht zijn en hoe ze verbeterd zouden kunnen worden.

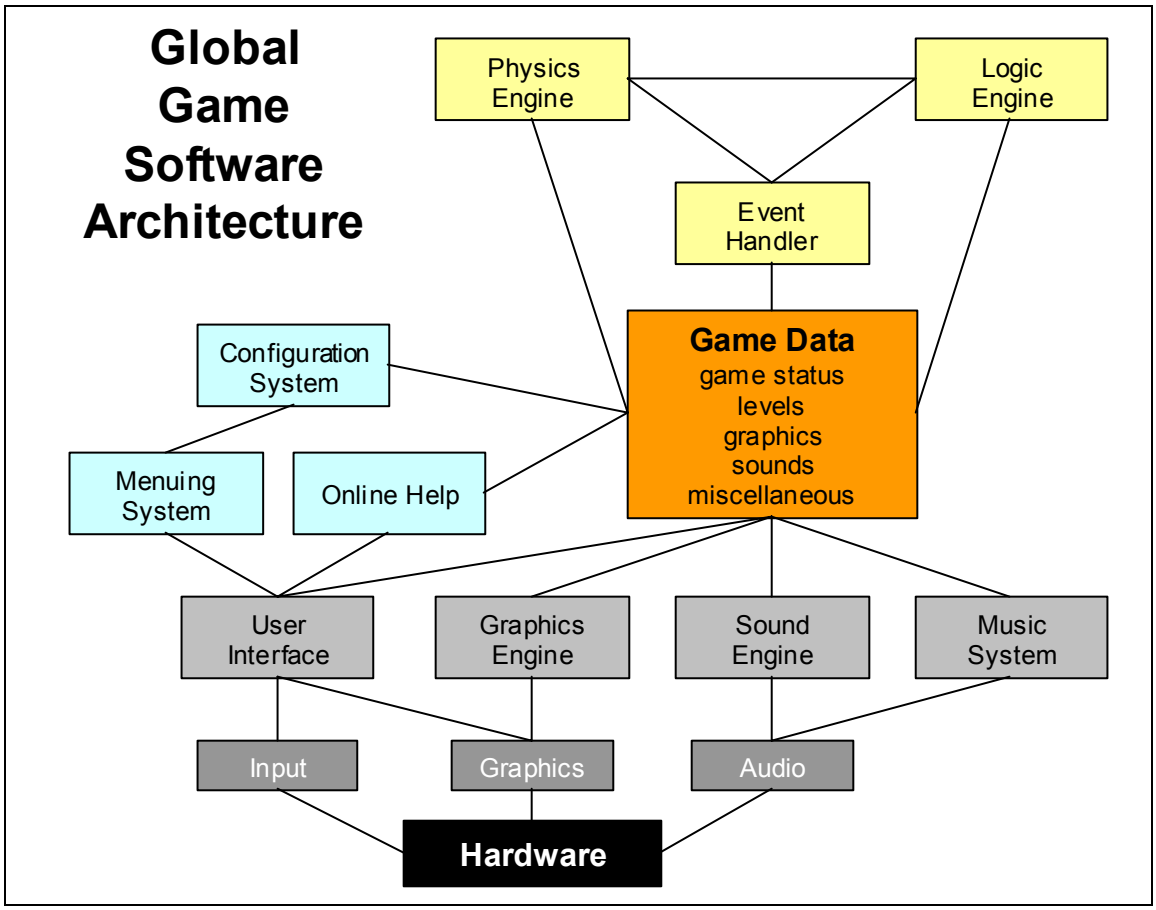
Een essentiële stap in het ontwikkelen van een computerspel is het ontwerp. Dit ontwerp, dat gemaakt dient te worden voordat het spel geïmplementeerd wordt, is een uitgebreide beschrijving van alle aspecten van het spel. Een ontwerpdocument voor een commercieel spel kan vele honderden pagina's dik zijn. Het beschrijft onder andere het verhaal, de units die in het spel voorkomen (bijvoorbeeld de tegenstanders, de militaire eenheden, de mogelijke gebouwen) met hun functie en relatie tot elkaar, de gebruikersinterface, en de niveaus. (Verwar dit document niet met het ontwerpdocument voor de software.)

Waarschijnlijk het belangrijkste aspect van het spelontwerp is de "gameplay". Dit beschrijft hoe het spel gespeeld wordt. Wat kan de speler doen en wat zijn de gevolgen van de acties. Welke units heeft hij op welk moment ter beschikking en wat is de functie van die units. Het is hierbij cruciaal dat een juiste balans gevonden wordt tussen de units. Dit speelt vooral bij strategiespellen een essentiële rol. Er mag geen unit zijn die alle andere units verslaat. Dat maakt het spel oninteressant. De kracht van de computer tegenstanders moet in balans zijn met de kracht van de speler. En als meerdere mensen tegen elkaar kunnen spelen (iets wat een steeds belangrijkere rol gaat spelen in computer games) dan dient geen van de spelers een duidelijk voordeel vooraf te hebben. Dit is zeer lastig te realiseren en in vele games zitten fouten op dit gebied (meestal omdat spelers units op andere wijzen gebruiken dan de ontwerper gedacht had). Zeer veel tijd gaat zitten in het vinden van de juiste balans. Meestal gebeurt dit door uitgebreid te testen, maar er is ook enige theorie over die de studenten kunnen leren (zie bijvoorbeeld [6]).

Ook is het van belang dat studenten inzicht krijgen in de rol van de diverse personen die betrokken zijn bij het maken van een computer game: de ontwerper, de grafische artiesten, de geluidsmensen, de software engineers, de programmeurs, en de testers. Een goede onderlinge afstemming van taken en verantwoordelijkheden is cruciaal voor het slagen van een project. De praktijk leert dat hier erg veel mis gaat. Het tijdschrift *Game Developer* bevat in elke aflevering een artikel over de totstandkoming van een commercieel computer game. Hierin wordt ook aangegeven wat er fout is gegaan. Bijna altijd, zelfs bij de succesvolle spelen, wordt opgemerkt dat er problemen waren met het ontwerp (men was te snel aan het implementeren geslagen) en met de taakverdeling binnen het team. Dit leidde vaak tot aanzienlijke tijd- en budgetoverschrijdingen.

III. Software Architectuur

Tot redelijk recent werden computer games op een ad-hoc wijze geprogrammeerd, bij voorkeur in assembly en soms in C. Het argument was dat moderne software engineering technieken, zoals objectgeoriënteerd programmeren en het gebruik van herbruikbare componenten leidde tot te trage code. En snelheid werd als het meest belangrijke aspect van game programming gezien. De laatste jaren is men echter in gaan zien dat de overhead bij het gebruik van talen als C++ en het gebruik van algemene software bibliotheken zoals DirectX nauwelijks tot tragere code leidt maar wel een belangrijke invloed heeft op de snelheid waarmee het project afgerond kan worden en, als gevolg daarvan, op de kosten. Sterker nog, met de gigantisch toegenomen complexiteit van computer games wordt het vrijwel onmogelijk om deze correct te implementeren zonder gebruik te maken van moderne software engineering technieken.



Figuur 2 De globale structuur van de game software

De globale structuur van een game (vanuit het perspectief van de software) is gegeven in Figuur 2 (vrij naar [6]). In dit model speelt de game data de centrale rol. Aan de onderkant bevindt zich de interactie van die data met de hardware. Hiervoor zijn tegenwoordig goede softwarebibliotheken beschikbaar, hoewel sommige games deze verder uitbreiden. Aan de bovenkant bevindt zich de eigenlijke spelsoftware die reageert op events (gebruikershandelingen, botsingen, etc.) in de wereld. Fysische simulatie aan de ene kant en logische (vaak op AI gebaseerde) processen aan de andere kant bepalen de gevolgen van die events. In het vak worden de verschillende componenten en hun rol besproken.

IV. Gedrag en AI

In de game wereld komen vele objecten voor. Een aantal zijn statisch en doen niets, maar andere objecten bewegen en verrichten bepaalde handelingen (zoals op je schieten). Ook zijn er meer abstracte entiteiten, zoals de computertegenstander die het vijandelijke leger bestuurt. Ook deze dient bepaald strategisch en tactisch gedrag te vertonen. Binnen de game design wereld wordt dit alles onder de noemer kunstmatige intelligentie (AI) geplaatst. De laatste paar jaar is de aandacht voor AI binnen de game industrie aanzienlijk toegenomen. Zowel een groter deel van het budget als een groter deel van de computerrekentijd wordt hieraan toegewezen [8].

Het meest eenvoudige gedrag is reactief gedrag. Een bal stuitert als hij de grond raakt. Een pacman monster slaat af als hij bij een splitsing komt. Een tegenstander schiet als hij je ziet. Dit is vaak eenvoudig te specificeren met een regel-gebaseerd systeem. Ook speelt fysische simulatie hierbij een rol. Bijvoorbeeld in het spel Black & White (zie Figuur 1) kan de speler rotsblokken oppakken en weggooien. Deze rollen vervolgens van bergen af naar beneden.

Een moeilijker probleem is het plannen van bewegingen. In veel strategiespelen bijvoorbeeld moeten eenheden een route vinden van hun huidige positie naar een gewenste nieuwe positie. Hierbij mag de unit niet tegen obstakels of andere units opbotsen. Het vinden van dergelijke paden is een zeer lastig probleem, zeker als het om meerdere units tegelijkertijd gaat. Je ziet dan ook nog regelmatig in spelen dat units vast komen te zitten.

Technieken uit de robotica kunnen hier toegepast worden. Ook technieken uit de kunstmatige intelligentie kunnen gebruikt worden.

Strategisch gedrag is nog veel lastiger. Hoe kan een computertegenstander het beste aanvallen? Waar zijn de zwakke punten? Hoe kan men leren van het gedrag van een speler en toekomstige acties daarop aanpassen? Dit vereist het kunnen redeneren over motieven, intenties, en kennis. Hier kan agent-technologie, zoals ontwikkeld binnen de kunstmatige intelligentie, een rol spelen. Ook ziet men het gebruik van neurale netwerken in deze context. Toch blijft veel nog ad-hoc oplossingen, gecombineerd met heel veel testen en het aanpassen van parameters.

Een andere belangrijke ontwikkeling binnen de game AI is wat wel artificial life, afgekort als A-life, genoemd wordt. Hierbij probeert men populaties wezens zich the laten ontwikkelen door met elkaar en met hun omgeving te interageren. Een belangrijk recent voorbeeld hiervan is The Sims (zie Figuur 3).

Figuur 3 Een scène uit The Sims van Maxis

V. Graphics

Het meest in het oog springende aspect van computer games is altijd de bewegende graphics geweest. Zoals hierboven aangegeven is dit zeker niet het enige belangrijke aspect maar het speelt wel een grote rol (al is het alleen maar om het spel te verkopen). Tegenwoordig tonen de meeste games een 3-dimensionale wereld (op wat puzzelspelen na). Dat betekent niet dat ze ook 3-dimensionale graphics gebruiken. Nog steeds wordt voor veel games gebruik gemaakt van zogenaamde sprites. Hierbij wordt een figuur of voorwerp voorgesteld door een of meerdere bitmaps. Door deze bitmaps achter elkaar te tonen gaat het figuur bewegen. Voor verschillende handelingen of gezichtspunten zijn verschillende plaatjes (zie Figuur 4). Het voordeel hiervan is dat er een veel grotere mate van detail mogelijk is. Het nadeel is dat de speler de wereld niet vanuit een willekeurig gezichtspunt kan bekijken.

Figuur 4 Een verzameling sprites voor een figuur

Om een 3-dimensionaal beeld te krijgen gebruikt men vervolgens een isometrische projectie waarbij men de wereld schuin van boven bekijkt. Zulke spelen worden dan ook vaak isometrisch genoemd. Bekende spelen als Age of Empires gebruiken deze techniek (zie Figuur 5). Kennis van de mogelijkheden (en onmogelijkheden) van sprites en isometrische technieken zijn dan ook nog altijd zeer belangrijk voor game ontwerpers.

Figuur 5 Isometrisch beeld uit Age of Empires 2 van Microsoft

Echte 3-dimensionale spelen modelleren de wereld en de figuren daarin als 3-dimensionale objecten. Om deze voldoende detail te geven worden texture maps gebruikt. Sinds een paar jaar hebben vrijwel alle computers grafische kaarten die deze 3-dimensionale graphics zeer snel weer kunnen geven. Bovendien zijn er softwarebibliotheken zoals DirectX en OpenGL die de programmeur een groot deel van het werk uit handen nemen. Toch is gedegen kennis van 3-dimensionale graphics van groot belang voor het ontwerpen en programmeren van computer games. Hierbij spelen vooral ook modellering van de objecten en de werelden, en animatie een cruciale rol.

Figuur 6 De 3-dimensional wereld van Quake 3 van Id Software

VI. Gedistribueerde Games

In de beginjaren van de computer games was het een individuele bezigheid en speelde men tegen de computer. Al gauw kwamen er spelen waarbij twee of soms zelf meer spelers op dezelfde computer tegen elkaar speelden. Nu tegenwoordig vrijwel iedereen een internetverbinding heeft speelt men steeds vaker tegen anderen via het internet. Bij sommige spelen, zoals Quake 3, is dit zelf de belangrijkste manier van spelen geworden. Bij sommige games speelt men tegen een paar tegenstanders, maar bij andere games speelt men met vele honderden tegelijk. Aangezien elke speler een deel van de software op zijn eigen computer heeft draaien heeft men dus te maken met een grootschalige gedistribueerde applicatie. Hierbij treden alle standaardproblemen van gedistribueerde systemen op. Maar de toepassing voor games introduceert een aantal nieuwe problemen [5].

Het meest voor de hand liggende probleem is dat het cruciaal is dat alle spelers op elk moment dezelfde informatie hebben. Het mag niet voorkomen dat de wereld er voor de ene speler iets anders uitziet dan voor een andere speler. Dit lijkt makkelijker dan het is. Spelers hebben verschillende computers. Snelle moderne computers reageren vaak sneller en kunnen meer weergeven dan oudere computers. Bij trage computers laat men vaak detail weg. Dit detail kan echter cruciaal zijn als een tegenstander zich bijvoorbeeld verstopt achter een boom. De trage computer zou wel eens kunnen besluiten om de boom niet te tekenen waardoor de speler plotseling zichtbaar wordt. Een soortgelijk probleem treedt op bij de communicatie. Als de communicatie traag is kan het zijn dat een speler een tegenstander veel te laat in de gaten krijgt en daardoor in het nadeel is. Goede communicatieprotocollen zijn nodig om dit soort problemen te vermijden.

Als men via internet tegen andere spelers speelt kent men deze meestal niet. Dit betekent automatisch dat men moet waken voor malafide spelers die op illegale manier proberen het spel te winnen. Dit is een zeer lastig probleem. Immers, een belangrijk deel van het spelplezier zit erin dat de spelers bepaalde dingen niet weten (bijvoorbeeld hoe de wereld er uit ziet). Echter, de software weet dit meestal wel (tenzij er gebruik wordt gemaakt van een globale server die deze informatie bijhoudt, maar dat is vaak te traag). Dat betekent dat een speler die in staat is zijn eigen versie van het spel te hacken op die manier de tegenstanders makkelijker kan verslaan. Ook kan een speler de communicatie afluisteren of zichzelf door een sneller reagerend computer programma vervangen. In al deze gevallen is het spel voor de andere spelers niet leuk meer. De game ontwerpers proberen die op te lossen door een combinatie van standaard software beveiligingstechnieken en het monitoren van de acties van de spelers om te zien of ze niet te snel reageren of gebruik maken van kennis die ze niet kunnen hebben.

VII. Ervaringen

Een nieuw vak opzetten en geven is altijd een spannende aangelegenheid, zeker als het een onderwerp als game design betreft. De belangstelling van de studenten was groot. Gezien het experimentele karakter en het feit dat ik groepsinteractie wilde heb ik het aantal deelnemers beperkt tot 14. De studenten waren enthousiast over het onderwerp. Ze waardeerden het feit dat niet alleen op de technische aspecten werd ingegaan.

Goede tekstboeken die het onderwerp in de volle breedte behandelen zijn er niet. Uiteindelijk heb ik voor het boek van Watt en Policarpo [7] gekozen omdat dat een aantal belangrijke aspecten behandeld. Daarnaast heb ik een groot aantal artikelen van het web gehaald, in het bijzonder uit het tijdschrift *Game Developer*, ISSN 1073-922X, uitgegeven door CMP Media Inc. (zie ook de website http://www.gdmag.com). Ook heb ik veel materiaal uit het boek van Rolings en Morris [6] gehaald en heb ik zelf aantekeningen gemaakt (zie de web-site van het vak: http://www.cs.uu.nl/docs/vakken/gds). Hier treft u ook vele links naar andere interessante sites, een overzicht van beschikbare literatuur, en een aantal interessante ontwikkelomgevingen voor games.

Om gevoel te krijgen voor de aspecten die belangrijk zijn bij het ontwerp van computer games moesten de studenten twee games bestuderen. Over de een moesten ze een korte voordracht houden. Over de tweede moesten ze een uitgebreide review schrijven. De voordrachten waren niet zo succesvol omdat men vaak niet veel verder kwam dan het laten zien van wat mooie plaatjes. De volgende keer zal ik dat onderdeel dan ook laten vallen. De review was echter zeer nuttig en de studenten hebben daar veel aandacht aan besteed.

Aan het onderdeel over gedistribueerde games ben ik niet toegekomen. Ook heb ik 3D-graphics voor gedrag en AI behandeld. Dat laatste was geen goede keus aangezien men voor de opdrachten meer kennis over gedrag en AI nodig had dan over 3D graphics (zeker gezien het feit dat de studenten daar al het nodige van afwisten).

De studenten moesten twee opdrachten maken. Het probleem hierbij is dat het volledig uitprogrammeren van een spel heel veel werk is. Vandaar dat gekozen is voor het gebruik van twee ontwikkelomgevingen, speciaal bedoeld voor het maken van computer games. Deze pakketen nemen veel van het programmeerwerk uit handen van de ontwikkelaar die zich daardoor meer kan concentreren op de belangrijke aspecten van het spel.

Individueel moesten de studenten met het pakket *Game Maker* dat door mijzelf geschreven is (http://www.cs.uu.nl/people/markov/gmaker) een eenvoudig 2-dimensionaal spel maken. Hierbij diende eerst een uitgebreide ontwerpdocument geschreven te worden. Dit laatste was essentieel. De resultaten waren over het algemeen erg goed. De resultaten en ontwerpdocumenten zijn te vinden op de website van het vak (http://www.cs.uu.nl/docs/vakken/gds).

Figuur 7 Twee spelen gemaakt door de studenten

Als tweede opdracht moesten de studenten in teams van 4 of 5 werken aan een 3-dimensionaal spel. Hiervoor werd als ontwikkelomgeving Jamagic van ClickTeam gebruikt (http://www.clickteam.com/English/). Deze geïntegreerde omgeving maakt het mogelijk om op eenvoudige wijze 3-dimensionale spelen te maken. De gebruikte taal is gebaseerd op Java Script, uitgebreid met vele objecten en functies voor games. Helaas was de omgeving, ondanks toezeggingen van de makers, niet op tijd beschikbaar. De betaversie die de studenten moesten gebruiken bevatte vele bugs en de documentatie was nog erg onvolledig. Het gevolg was dat de resultaten tegenvielen. Nu Jamagic eindelijk officieel beschikbaar is gaat dit hopelijk beter.

VIII. Conclusies

Game design is een boeiend onderwerp waarbij een groot aantal aspecten van de Informatica een rol spelen. Het is ook een onderwerp dat van toenemend belang is voor het bedrijfsleven. Niet alleen is de game industrie zelf de laatste jaren uitgegroeid tot een aanzienlijke bedrijfstak, ook in andere toepassingen spelen de technieken uit de game wereld een toenemende rol. Computers worden bijvoorbeeld meer en meer voor instructie gebruikt. Deze instructie vindt tegenwoordig vaak plaats in een virtuele wereld waarin de gebruiker rondloopt en handelingen verricht die veel gelijkenis vertonen met games. Ook samenwerking met andere mensen via de computer maakt steeds vaker gebruik van zulke virtuele werelden.

Game design kan op vele plaatsen in het onderwijs gebruikt worden. De cursus die ik hier besproken heb was een vergevorderde universitaire cursus die veel voorkennis veronderstelde. Maar ook in een eerder stadium van de opleiding en zelfs op middelbare scholen kan er prima aandacht aan besteed worden, indien gebruik gemaakt wordt van eenvoudig te bedienen ontwikkelomgevingen zoals *Game Maker*. Het is een onderwerp dat leerlingen en studenten aanspreekt en aan de hand waarvan veel moderne aspecten van de Informatica zijn toe te lichten.

Referenties

1. Greg Costikyan, *I have no words and I must Design*, Interactive Fantasy #2, 1994. (te downloaden via http://www.costik.com/)
2. Greg Costukyan, *Where Stories End and Games Begin*, Game Developer, Sept 2000, pp. 44-53.
3. Mark DeLoura, *Game Programming Gems,* Charles River Media, 2000, ISBN 1-58450-049-2.
4. David H. Eberly, *3D Game Engine Design,* Morgan Kaufmann, 2001, ISBN 1-55860-593-2.
5. Matt Pritchard, *How to Hurt the Hackers*, Game Developer, June 2000, pp. 28-40.
6. Andrew Rollings, Dave Morris, *Game Architecture and Design,* Coriolis, 2000, ISBN 1-57610-425-7.
7. Allen Watt, Fabio Policarpo, *3D Games, Real-time Rendering and Software Technology,* ACM Press (Addison-Wesley), 2001, ISBN 0201-61921-0.
8. Steven Woodcock, *Game AI, The State of the Industry*, Game Developer, August 2000, pp. 24-32.

Auteur

Prof. Dr. M.H. Overmars is hoogleraar informatica bij het Instituut voor Informatica en Informatiekunde van de Universiteit Utrecht. Hij leidt het Centrum voor Geometry, Imaging en Virtual Environments.
Adres: Postbus 80.089, 3508 TB Utrecht. URL: http://www.cs.uu.nl/people/markov