Efficiënte online virtuele werelden

Door Mike Hergaarden

Bachelor project Informatica 2009
Begeleider: Prof. dr. A. Eliëns
Tweede lezer: Dr. R. Siebes

Voorwoord

Dit verslag hoort bij mijn bachelor project waarmee ik dit jaar mijn drie jaar bachelor informatica aan de VU mee af ga ronden. Het doel van het bachelor project is om zelfstandig ervaring op te doen in een deelgebied van informatica. Ik heb er voor gekozen om online virtuele werelden te onderzoeken en om met die kennis er ook zelf een te implementeren. In dit verslag staat mijn gehele bachelor project beschreven dat bestaat uit een case study met daarnaast de bijbehorende theorie.

De case study is een onderdeel van een gezamenlijk project met Jos Hoebe; zijn bachelor project is getiteld “Procedural Generated online virtual worlds”. Gezamenlijk leggen we een basis voor een echte online virtuele wereld. Meer informatie hierover is te vinden op http://www.few.vu.nl/~mhn212/bp/

Dank aan prof. dr. A. Eliëns voor het begeleiden en het mogelijk maken van dit project. Verder wil ik dr. R. Siebes bedanken om de rol van tweede lezer op zich te nemen.

Mike Hergaarden

Inhoudsopgave

Samenvatting

Online virtuele werelden zijn ingewikkelde applicaties om te ontwerpen en te maken. In mijn bachelor project wordt een basis gelegd voor een online virtuele wereld. Zowel de theorie als de praktijk wordt behandeld. In een case study programmeer ik met behulp van Unity3D een echte virtuele 3D wereld, die te spelen is via de browser. In een gezamenlijk project wordt mijn multiplayer werk gecombineerd met het werk van Jos Hoebe. Hij concentreert zich op procedural content generation; een combinatie van onze werken levert daarmee een echt multiplayer wereld op.

In dit verslag wordt behandeld hoe mijn genetwerkde virtuele wereld werkt en waarom het zo werkt. Zo wordt de architectuur, besturing en networking rondom de spelers van deze wereld behandeld. Daarna worden ook alternatieve netwerk technieken behandeld die geen plaats hebben gekregen in mijn case study, maar die wel in andere implementaties erg belangrijk zijn.

Een belangrijk terugkomend punt is dat een effectief ontwerp van het spel veel kan veranderen voor de multiplayer prestaties. Hiermee wordt niet alleen het technische ontwerp bedoeld, maar vooral het ontwerp op gameplay niveau.

Inleiding

De laatste paar jaren worden online virtuele werelden steeds populairder. Het zijn niet langer meer alleen gamers die zich in online games verdiepen; vooral sinds Second Life[1] besteden universiteiten en het bedrijfsleven er (weer) serieus aandacht aan. Er wordt door sommige mensen serieus afgevraagd of een 3D wereld via de browser een nieuwe standaard op het web wordt.

Een probleem in een online wereld is de capaciteit van het netwerk. Er kan maar een beperkt aantal berichten tegelijk over het internet worden verstuurd, maar er moet een groot aantal gebruikers worden bediend. Om deze reden is het belangrijk de communicatie tussen alle gebruikers te optimaliseren.

In dit verslag behandel ik de optimalisatie van netwerk communicatie in genetwerkde virtuele werelden. Dit doe ik op basis van een eigen case study. Ik behandel een genetwerkde virtuele wereld die ik gemaakt heb voor dit project. Deze case study wordt geëvalueerd en vergeleken met andere bestaande genetwerkde virtuele werelden. De netwerk communicatie rondom de beweging van karakters is het belangrijkste onderwerp, daarnaast komt ook de beveiliging en multiplayer architectuur aan bod. De focus bij al deze onderwerpen ligt op het ontwerp niveau, en niet bij de "lage niveau implementatie" zoals het optimaliseren van netwerk pakketten.

1 Link: http://www.secondlife.com

Case study: implementatie van een virtuele wereld

Programma van eisen

Voordat ik kan beginnen met het maken van de genetwerkde wereld geef ik enkele eisen aan waaraan deze wereld moet voldoen. Hierop kan ik dan het ontwerp richten. Ik wil de mogelijkheid open houden om vanuit deze case study een echt spel te gaan maken, daarom zal doorgaans naar de virtuele wereld ook als “spel” gerefereerd worden. Hieronder volgt een lijst van de eisen die ik aan het spel stel:

1. Speelbaar in de browser:
Om een virtuele wereld populair te maken is het een groot voordeel om het gemakkelijk toegankelijk te maken. De browser is hier erg geschikt voor omdat bezoekers niks hoeven te installeren.

2. Het inloggen en registreren moet geïntegreerd zijn in het spel:
Net als de eerste reden, moet de drempel om te spelen zo laag mogelijk liggen.

3. Multiplayer:
Dit is een eis van het project.

4. Veilig (hacken/manipuleren van het spel moet voorkomen worden):
Een multiplayer omgeving verliest z'n waarde als spelers vals kunnen spelen.

5. Het spel moet op networking gebied zo efficiënt mogelijk zijn:
Dit is een eis van het project

6. Het spel moet 3D weergeven worden:
Het moet er zoveel mogelijk uit kunnen zien als onze eigen wereld.

Aan de hand van deze eisen kan ik het spel verder gaan uitwerken. De specificaties van de gameplay missen hier omdat enkel de basis voor een echt spel gelegd wordt.

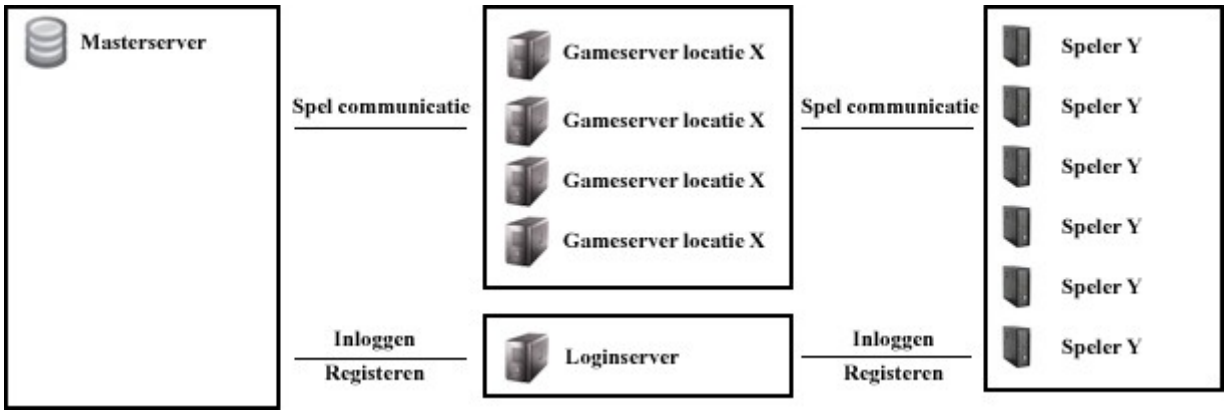
Vanwege de bovenstaande eisen kies ik voor Unity3D[2] als game engine, daarnaast heb ik ook al enige ervaring met deze engine. In Unity3D heb ik alle mogelijkheden om te doen wat ik wil (streaming assets, networking, 3D in de browser etc.). Daarnaast legt Unity3D ook geen beperkingen op aan de mogelijkheden omdat C en C++ plugins gebruikt kunnen worden. Met deze plugins is er alle vrijheid om eventuele missende mogelijkheden zelf toe te voegen.

Networking is een belangrijk onderdeel van dit project maar zoals al eerder vermeld werk ik niet op het lage niveau van netwerk pakketten etc. Ik maak gebruik van de ingebouwde basis netwerk functies van Unity3D. Dit biedt de mogelijkheid om simpel een verbinding te maken om de inhoud van webpagina's op te roepen en om netwerk berichten te versturen. Unity3D heeft deze functies geïmplementeerd met behulp van de open source netwerk library RakNet[3].

2 Link: http://www.unity3d.com
3 Link: http://www.jenkinssoftware.com

Multiplayer architectuur

Online werelden hebben allemaal een andere architectuur. Ik richt me op de efficiëntst mogelijke die binnen mijn richtlijnen valt. In grote lijnen zijn er twee mogelijke architecturen: De client-server architectuur en de peer to peer architectuur. In de case study ga ik gebruik maken van de client-server architectuur omdat de traditionele peer to peer architectuur niet voldoet aan mijn eisen. De peer to peer architectuur mist de veiligheid, controleerbaarheid en de controle van de client-server architectuur. Om die redenen was het logisch om voor een client-server architectuur te kiezen. Een alternatieve architectuur wordt behandeld in de case study evaluatie. De geïmplementeerde architectuur ziet er als volgt uit:



Figuur 1: Multiplayer architectuur van de case study

Alle data van de spelers wordt opgeslagen op één centrale master server. Voorbeelden van wat er opgeslagen wordt zijn bijvoorbeeld de positie van de spelers in de wereld en hoeveel virtueel geld ze bezitten. De speler maakt nooit direct contact met deze master server, maar doet dit via game servers of de login(en registratie) server. Zo worden alle veranderingen geverifieerd voordat ze daadwerkelijk toegepast worden.

Alle onderdelen in Figuur 1 kunnen gezien worden als losse applicaties, dit hoeven niet per se aparte fysieke computers te zijn. In mijn uitvoering heb ik op één computer zowel alle game servers, de login server en de master server draaien. Dit was vanwege het gebrek aan computers en heeft verder ook geen invloed gehad op de case study.

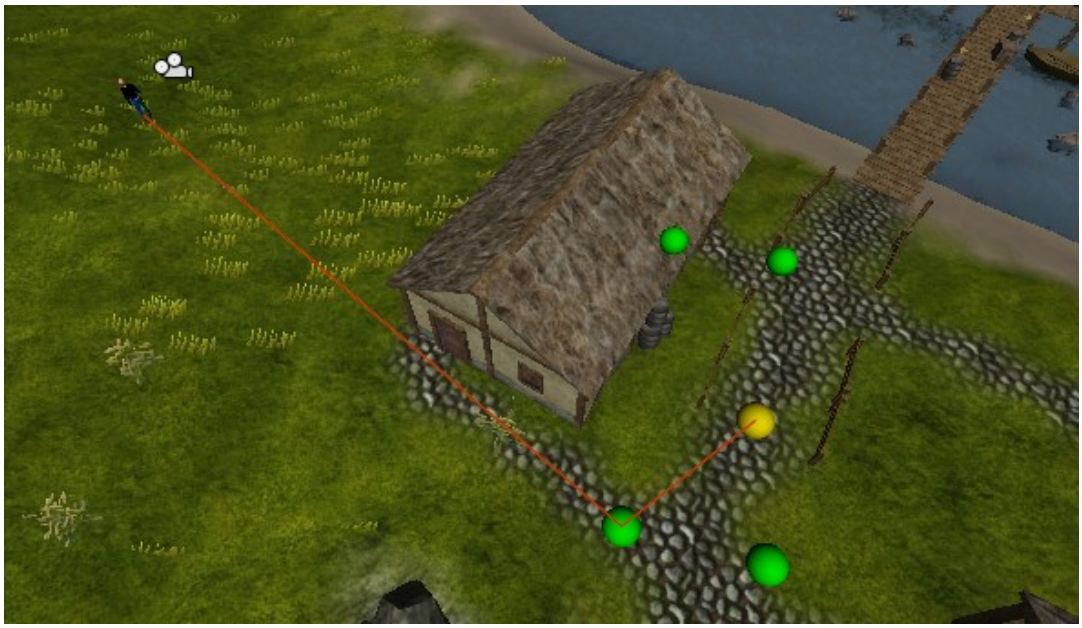
Ik heb ervoor gekozen om alles zo schaalbaar mogelijk te maken; in deze architectuur kan de “load” gemakkelijk gebalanceerd worden over één of meerdere servers om de kosten-performance verhouding te verkrijgen. Deze load balance is statisch; per gebied. De master server houdt een lijst bij van welke server(s) een bepaalde locatie beheren, en of ze wel of niet online zijn. Zo kunnen er voor één locatie meerdere game servers zijn, om de spelers te verdelen over verschillende servers. Ook kan het zijn dat één server meerdere locaties bediend.

De master server bestaat uit een MYSQL database. De master server heeft een web front-end voor de communicatie met de game servers en de login/registratie server(gebruik makend van apache en PHP). De game servers en clients draaien de Unity player en alle communicatie verloopt dan ook via de networking library van de unity player.

Netwerk communicatie

De avatar

Zoals al eerder is vermeld, is de verplaatsing van karakters in een multiplayer wereld een van de grootste bronnen van netwerkverkeer. Omdat network efficiency het belangrijkste onderwerp van dit project is, heb ik er voor gekozen om de besturing van de avatar te doen via muis klikken: Je klikt ergens op het gebied, en je avatar loopt er naar toe. Dit scheelt veel bandbreedte tegenover de gebruikelijkere besturing in andere spellen, maar introduceert een nieuw probleem: De besturing van de avatar is iets logger. Toch is dit laatste probleem makkelijker op te lossen dan die van de bandbreedte. De loggere besturing is te verbeteren door de camera goed in te stellen en de speler hierover ook controle te geven. Verder is simpele pathfinding toegepast zodat de avatar daadwerkelijk naar het punt loopt waar de speler klikte en zodat de avatar niet tegen muren aanloopt. Omdat de wereld toegankelijk is via de browser is het ook erg logisch om de avatar met de muis te besturen.



Figuur 2: Pathfinding in werking; de speler stond op de gele bol en klikte op de plek waar hij nu staat. Het berekende pad is hier rood weergeven.

Voor de beweging om langs een huis te lopen wordt er maar één pakket gestuurd van de speler naar de server en de server stuurt maar één pakket door naar alle andere spelers; Simpelweg de coördinaten waar de speler nu staat (op de server) en de coördinaten waar de speler naartoe loopt. Omdat alle code op precies dezelfde manier wordt uitgevoerd op alle computers is er geen andere data nodig. De begin positie word meegestuurd omdat de huidige positie van de speler op de verschillende computers af kan wijken vanwege de tijd die het kost om de berichten door te sturen. Technieken zoals dead reckoning, interpolatie en extrapolatie zijn hier verder overbodig. Deze worden later in dit verslag nog wel kort behandeld.

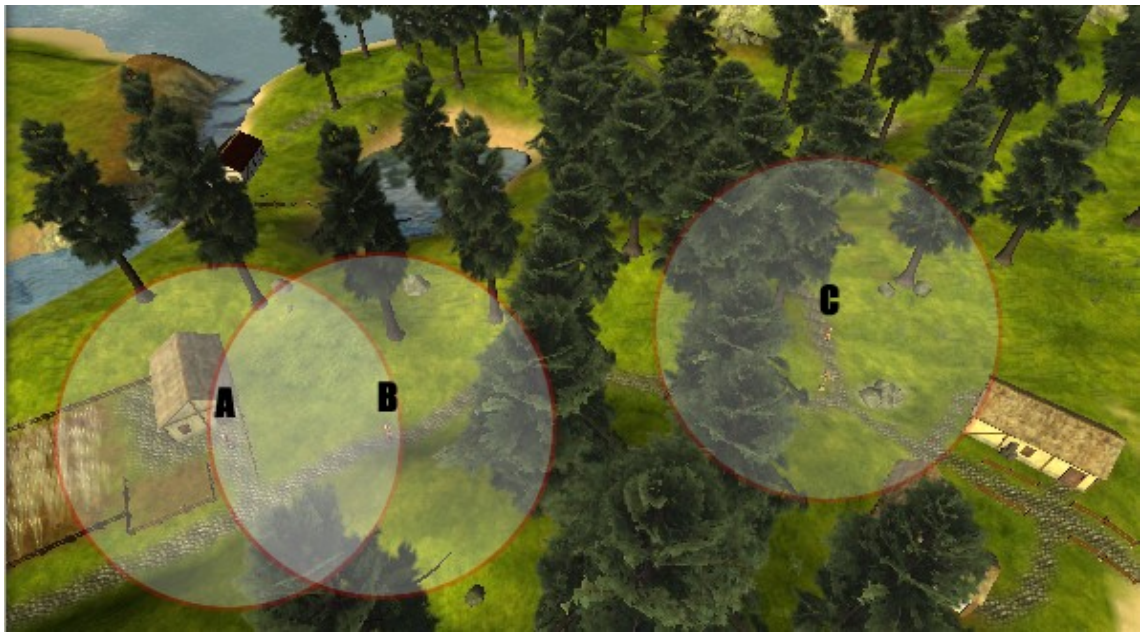
Animaties en avatar uiterlijk

Er zijn nog twee soorten netwerk berichten niet behandeld: Het versturen van eigenschappen van een object en het versturen van informatie over gebeurtenissen. Om die reden is het interessant om naar de eigenschappen zoals het uiterlijk en de animaties van de avatar te kijken.

In de case study kunnen de spelers niet hun eigen avatar bewerken, maar krijgen elke keer bij het inloggen een avatar met een andere kleur kleren. Deze eigenschappen van de avatar worden naar de andere clients gestuurd zodra de spelers voor het eerst in elkaars buurt komen. Er wordt een voorgeprogrammeerd getal overgestuurd waardoor beide spelers precies weten hoe de avatar eruit moet zien. De loop animaties van de avatars worden automatisch gestart of gestopt tijdens het lopen. Een uitzondering is de dans animatie. Zodra een speler op de spatiebalk drukt zal zijn of haar avatar gaan dansen. Ook hier wordt weer een getal verstuurd wat in dit geval staat voor het starten van een animatie. Uit een lijst van getallen met de bijbehorende animaties wordt ook meteen bepaald hoe deze animatie zich moet gedragen in combinatie met andere animaties en of de animatie eenmalig of herhalend afgespeeld moet worden.

Verdere optimalisatie

Nu het bewegen van avatars mogelijk is gemaakt kunnen we dit nog verder optimaliseren. We willen natuurlijk niet dat we in een bepaalde omgeving berichten van andere spelers ontvangen die ver buiten het bereik zijn. Daarom berekent de server de afstand tussen alle spelers, en schakelt communicatie aan of uit afhankelijk van de Area Of Interest(AOI) afstand. Een harde grens van bijvoorbeeld 50 meter is niet altijd verstandig omdat een speler deze grens meerdere malen achter elkaar kan oversteken. Het activeren/deactiveren van spelers kan extra performance kosten met zich meebrengen. Daarom heb ik er voor gekozen om de communicatie uit te schakelen na 60 meter afstand, en pas weer aan te zetten onder de 50 meter afstand. Deze waardes zijn natuurlijk afhankelijk van implementatie en schaal van de wereld, maar kunnen gemakkelijk aangepast worden.



Figuur 3: Spelers ontvangen alleen berichten van spelers die dichtbij zijn.

In figuur 3 is als voorbeeld enkel de area of interest grens weergeven. Speler A en B ´zien´ enkel elkaar en speler C ziet geen enkele speler. Van de spelers die niet onder het area of interest gebied vallen krijgen de spelers geen enkele berichten. Wel houden alle spelers een lijst bij van de spelers die “ergens” op de nabije wereld rondlopen, om ze weer snel zichtbaar te maken als dat nodig is.

Beveiliging

Naar eigen onderzoek en gerelateerde bronnen[R7] onderscheid ik de volgende relevante beveiligingsproblemen:

1. Het onderscheppen en veranderen van netwerk berichten
2. Het manipuleren van het programma

Beide problemen heb ik aangepakt. Ik gebruik van de in Unity3D ingebouwde RakNet netwerk beveiliging, dit voegt de volgende veiligheidsmaatregelen toe aan de netwerk communicatie:AES encryption, SYN cookies en CRC's. Dit garandeert nog geen totale veiligheid, maar maakt het onderscheppen en manipuleren van het netwerkverkeer al stukken lastiger. In het ontwerp en de uitvoering van de case study is er ten alle tijde rekening gehouden met het feit dat de gebruiker alle netwerk pakketten en het lokale programma volledig kan veranderen; alsof de volledige bron code openbaar is. Daarom wordt geen enkele gebruikers input ´vertrouwd´ en wordt alles eerst gecontroleerd door de server. Om de servers performance en bandbreedte te besparen wordt er niet altijd controle gedaan daar waar het niet uitmaakt of iets veranderd wordt (denk aan chat berichten) of waar het enkel in het nadeel van de speler zou zijn die vals probeert te spelen.

Evaluatie van de case study

Nu de case study is behandeld worden vervolgens de resultaten hiervan op een rijtje gezet. Ik bekijk of de case study goed uitgepakt heeft en of het z´n doel bereikt heeft. De implementatie van mijn case study is erg efficiënt op het netwerk gebied, maar dat is meer te danken aan de verandering op het gebied van gameplay dan door die op het netwerk gebied zelf. Ten slotte is het ook nog belangrijk om te behandelen hoe mijn implementatie zich verhoudt tot andere bestaande genetwerkde virtuele werelden.

De resultaten

De case study heeft een stabiele multiplayer architectuur opgeleverd die dankzij de schaalbaarheid gemakkelijk gebruikt kan worden om een grote wereld in stappen op te zetten. Terwijl ik mij eerst bezig dacht te gaan houden met het implementeren van dead reckoning en interpolatie van de avatars in de wereld kwam hier uiteindelijk niks van terecht. Dit was te danken aan de verandering in de besturing van de avatars. Dit bracht zeer goed resultaten, maar was dit eigenlijk geen vals spelen? Alle bestaande virtuele werelden gebruiken namelijk pijltjestoetsen (of de WASD knoppen) om een avatar te bewegen. De beweging limiteren tot muis klikken maakt het een heel ander spel. Toch denk ik dat sommige toepassingen deze kant op moeten gaan, de virtuele werelden waar de laatste tijd veel aandacht naar uit gaat vanuit het bedrijfsleven en universiteiten zijn niet gericht op gamers maar juist op alle mensen die weinig ervaring hebben met computerspellen. Het is daarom erg belangrijk om virtuele werelden erg laagdrempelig te maken. Nu met engines zoals Unity3D er de mogelijkheid is om 3D werelden via de browser te weergeven is daarmee al een grote stap gezet. Doordat de wereld via de browser bekeken wordt is het niet meer dan logisch om over te stappen op de gemakkelijkere muisbesturing.

Cijfers van netwerk gebruik

Harde cijfers zijn moeilijk te geven omdat het netwerk gebruik afhangt van de hoeveelheid spelers, de dichtheid van die spelers en het gedrag van de spelers [R3][R4]. Omdat spelers totaal niet te zijn voorspellen heb ik een gemiddeld gebruik te proberen simuleren door 30 seconden lang actief te lopen in mijn case study, dat resulteerde in 30 verstuurde berichten, en zo'n 0.1kb/s verkeer. Bij 16 spelers zou een speler daarom zo'n 1.6kb/s netwerkverkeer hebben. Een gemiddeld schietspel met 16 spelers, waar de bewegingen erg nauwkeurig moeten zijn, kost tegenwoordig zo'n 6kb per seconde. Maar ook deze waarde varieert sterk.

Uitbreidbaarheid

In de huidige implementatie zijn alle assets in het programma “ingebakken”. Dat houdt in dat alle assets geladen worden tijdens het streamen van de webplayer. Om het spel uit te breiden zouden alle gebruikers het spel (geforceerd) opnieuw moeten openen zodat het compleet opnieuw gedownload wordt. Dit is in de praktijk geen groot probleem zolang het niet te vaak gebeurt. De meeste multiplayer online role-playing games(MORPG) zijn 's nachts 4-6 uur per week offline vanwege onderhoud of updates.

Vergelijking met andere virtuele werelden

Ik onderscheid drie verschillende spellen om te vergelijken met mijn case study: Runescape[4], Second Life en World of Warcraft[5]. In de vergelijkingen is gameplay niet meegenomen.

Runescape

Runescape is een enorm populair online role playing game. Het spel is gedeeltelijk gratis, 3D en te spelen via de browser. Deze feiten hebben ervoor gezorgd dat het spel nu miljoenen spelers telt.

Runescape vs de case study

De gelijkenis van Runescape en mijn case study qua multiplayer architectuur en besturing is erg groot. De besturing en architectuur zijn nagenoeg gelijk: Ook in Runescape wordt de avatar door muisklikken(en met pathfinding) bestuurd. Verder is er voor de wereld een groot aantal servers waarover de spelers verdeeld worden via één centrale loginserver. Er zijn twee grote verschillen tussen Runescape en mijn case study. Ten eerste is Runescape is één grote seamless streaming wereld, terwijl mijn case study telkens één bepaald gebied laadt. Ten tweede streamed Runescape z'n assets mee, terwijl mijn case study alle assets geladen moet hebben voordat het level geladen kan worden.

Zijn er voor runescape of mijn case study nog verbeter punten?

Runescape is vriendelijker voor de gebruikers dan mijn case study omdat tijdens het spelen geen echte laadschermen weergeven worden, wel wordt bij het overschrijden van een onzichtbare grens af en toe een nieuw gebied geladen. Toch merkt de gebruiker hier weinig van. Bij mijn case study *kan* een laadscherm weergeven worden als een gebruiker naar een gebied gaat wat nog niet geladen is. Wel worden alle gebieden tijdens het spelen al geladen, zodat er niet altijd een laadscherm weergeven hoeft te worden. Verder gaat runescape slimmer met de bandwidth om; het voordeel van de case study die alle levels tijdens het spelen laadt is ook een nadeel omdat hiermee (vaak) onnodige assets geladen worden. Ook al is dit niet van te voren niet te voorspellen.

Second Life

Ook al heb ik Second Life altijd als hype ervaren, de aandacht en discussies rondom dit spel zijn wel degelijk serieus. Second Life's succes is onder andere te danken aan de mogelijkheid dat spelers zelf de wereld kunnen vormen door het maken van gebouwen, kleren etc.

Second Life vs de case study

Mijn case study en Second Life lijken op het eerste gezicht erg verschillend. Toch is de server architectuur niet erg verschillend[R8], wél is Second Life natuurlijk veel uitgebreider en heeft daarom een grotere architectuur. De gebruikelijke pijltjestoetsen besturing van Second Life is het eerste grote verschil met mijn case study, wat de veelbesproken netwerk veranderingen met zich meebrengt. Verder is het volgende grootste verschil het dynamisch streamen van de benodigde assets in SL van het gebied waarin de gebruiker zich bevind.

Zijn er voor Second Life of mijn case study nog verbeter punten?

Omdat je in Second Life je je overal kunt bewegen(je kunt zelfs vliegen en onderwater lopen) en omdat Second Life draait op *user created content* staan er een aantal netwerk gerelateerde dingen meteen vast. Ten eerste is de karakters besturing via de pijltjestoetsen besturing een must. Hier zou bewegen via muisklikken geen oplossing zijn. Ten tweede dwingt de *user created content* ook af dat er ten alle tijden content gestreamed kan worden. De verschillen op het networking gebied

4 Link: http://www.runescape.com
5 Link: http:///www.wow-europe.com

worden door de gameplay van Second Life afgedwongen en zijn voor Second Life de ideale eigenschappen. Nu Second Life steeds meer publiekelijk bezit wordt[6], zou het toepassen van peer to peer technieken een sterke performance verbetering met zich mee kunnen brengen. Zo zouden de textures bijvoorbeeld via de clients gedeeld kunnen worden. Op dit moment moeten spelers vaak nog lang laden bij het binnenkomen van een nieuw gebied; ze kunnen wel meteen spelen, maar in een "lelijke" wereld. Resources zouden veilig via P2P gedistribueerd kunnen worden door middel van resource checksums die de server uitdeeld.

World of Warcraft

WoW, World of Warcraft, is de grootse online game van dit moment met in oktober 2008 nog 11 miljoen betalende spelers [L1]. Het spel weet al sinds de lancering in november 2004 grote aantallen spelers te boeien.

WoW vs de case study

Omdat dit spel volledig staat geïnstalleerd op de pc van de gebruikers dit het enige spel in het rijtje wat geen assets en werelddelen streamed. Wel is er hier ook een seamless wereld, wat inhoud dat de spelers zonder laadschermen over de hele wereld kunnen reizen. De besturing verloopt hier via de pijltjestoetsen, verder is er ook een AOI en een vergelijkbare server architectuur als in mijn case study.

Zijn er voor WoW of mijn case study nog verbeter punten?

Net als bij Second Life zijn de verschillen tussen mijn case study en WoW te verklaren omdat ze vanuit de specificaties van het spel worden afgedwongen; WoW wilt spelers een ervaring kunnen bieden die de spelers laat opgaan in de fantasie wereld. Het feit dat er amper laadschermen zijn en dat er meer controle is over de besturing van je karakter, helpt hier aan mee. Buiten de genoemde verschillen zijn er geen opmerkelijke verschillen. WoW past al P2P toe in de updater van het spel, zodat spelers meehelpen bij het distribueren van een nieuwere versie van het spel. P2P verder toepassen in het spel zelf zal geen grootste verbeteringen opleveren omdat er in WoW niet vaak meer dan 40 spelers in hetzelfde gebied staan (enkel in georganiseerde gevechten). Mocht WoW toch grotere groepen spelers tegelijk in een gevecht kunnen plaatsen dan zou P2P toevoeging een uitkomst kunnen zijn.

Alternatieve netwerk technieken

Er zijn belangrijke netwerk technieken voor online games die toch onbehandeld bleven in mijn case study. Sommige van deze technieken zouden mijn case study nog verder kunnen verbeteren en andere waren in mijn implementatie overbodig. In dit hoofdstuk worden deze alternatieve netwerk technieken alsnog besproken.

Dead reckoning / Extrapolatie

Een probleem die in mijn case study niet ter sprake kwam was packet loss; het verliezen van netwerk berichten. Dead reckoning is een techniek die bijhoudt wat de oude positie, richting en snelheid van een object was, om met deze informatie de beweging van het object voor te zetten als er geen netwerkberichten meer binnen komen. Op deze manier kan packet loss gemaskeerd worden voor de gebruiker. Ook kan deze techniek gebruikt worden om te besparen op netwerk pakketten[R5]. Met het gebruik van onderzoek uit de paper "*Cheat-Proofing Dead Reckoned Multiplayer Games*" kan Dead reckoning zelfs ingezet worden tegen cheating [R6].

6 http://www.computable.nl/artikel/ict_topics/ictbranche/1950893/2379258/server-second-life-wordt-open-source.html

Dead reckoning was om twee redenen niet van belang in mijn case study. Ten eerste is packet loss pas echt een probleem in multiplayer games waarin de communicatie erg snel en precies moet zijn. In mijn case study zou packet loss geen probleem zijn omdat er weinig berichten verstuurd worden en er mag een kleine vertraging zijn. Ten tweede zorgt de gebruikte networking library in Unity3D er automatisch voor dat de berichten die verstuurd worden gegarandeerd aankomen.

Interpolatie

Om het probleem van packet loss nog verder tegen te gaan kan interpolatie worden toegepast. Deze techniek lijkt sterk op extrapolatie, maar bij interpolatie wordt een missende waarde berekend die tussen een rijtje bekende waarden staat. Bij interpolatie wordt de besturing en communicatie van een gehele game bijvoorbeeld 100ms vertraagt. Als er door packet loss "gaten" ontstaan in bijvoorbeeld de beweging van een speler, dan kunnen deze gaten goed opgevangen worden.

Peer to peer (P2P)

Peer to peer technologie wordt niet vaak gebruikt in games omdat de client-server architectuur betrouwbaarder wordt geacht op het gebied van performance en veiligheid. Toch zijn er innovatieve P2P technieken ontwikkeld die P2P steeds betrouwbaarder maken. Zo kunnen in P2P netwerken zogenaamde nodes/supernodes aangewezen worden die binnen het P2P netwerk als server werken en meer autoriteit hebben dan andere clients. Op die manier kan al het werk op het gebied van networking worden gedistribueerd over het P2P netwerk. De network library NetDog[7] heeft dit bijvoorbeeld al geïmplementeerd en claimt record aantallen spelers te kunnen dragen op deze manier. Harde bewijzen in vorm van afgeronde spellen zijn er echter nog niet.

Dynamische load distributie

Dit laatste onderwerp kwam bij Peer to peer ook al aan bod: Het dynamisch aanpassen van de server load. Hiermee worden dus bepaalde gebieden(en daarmee ook de spelers) aan één of meerdere servers verdeeld. Mijn implementatie gebruikt statische load distributaie omdat alleen de vaste gebieden over één of meer servers kunnen verdeeld, maar dit kan niet bijgeschaald worden per vierkante meter. Dynamisch load distributie kan volgens de paper "L*ocality Aware Dynamic Load Management for Massively Multiplayer Games*"[R2] 6 tot 8 keer betere prestaties opleveren. Voor verdere informatie over dit onderwerp verwijs ik dan ook naar deze paper.

7 Link: http://www.pxinteractive.com

Conclusie

Een punt wat vaak in dit verslag genoemd is, is dat het erg loont om een spel hier en daar aan te passen in het voordeel van de multiplayer prestaties. Dit kan niet altijd. Een andere vuistregel is dat voor elk netwerk bericht een afweging gemaakt worden of het bericht echt nuttig is en welke van de spelers het (nog) niet per se zou hoeven ontvangen.

Gelukkig worden de computers en internet verbindingen steeds sneller en goedkoper, wat het mogelijk maakt om nog meer gebruikers tegelijkertijd in een virtuele wereld te plaatsen. Maar hiermee neemt de vraag voor efficiënte multiplayer implementaties ook steeds meer toe. Ik verwacht dat we, voor betere prestaties, de load van servers moeten proberen te verdelen over de talloze clients. Hier duid ik dan op de combinatie van client-server en de peer-to-peer architectuur, zoals ook al gebruikt door NetDog of Skype[8].

De case study is geslaagd omdat ik heb kunnen implementeren wat ik voor ogen had, en omdat ik het aan mijn eisen heb kunnen laten voldoen. Deze basis kan gebruikt worden om een echte multiplayer wereld te maken, zonder iets te hoeven toe te voegen op het gebied van networking. Een mogelijke toevoeging zou nog kunnen zijn om assets te streamen “on-demand”, wat met Unity3D goed mogelijk is.

Voor verder onderzoek in de diverse deelgebieden verwijs ik door naar de literatuurlijst. Daarin zijn zowel verdiepende als algemenere papers opgenomen.

8 Link: http://www.skype.com/help/guides/p2pexplained/

Literatuurlijst

[R1] Blizzard entertainment, oktober 2008,
URL: http://eu.blizzard.com/en/press/081028.html

[R2] Locality Aware Dynamic Load Management for Massively Multiplayer Games,
URL: http://www.cs.toronto.edu/~jinchen/ppopp.pdf

[R3] Analysis of MultiPlayer Online Game Traffic Based on Selfsimilarity,
URL: http://www2.itu.edu.tr/~erolme/english/published%20papers/ng42.pdf

[R4] Analyzing Unreal Tournament 2004 Network Traffic Characteristics,
URL: http://doc.tm.uka.de/2008/ut2004traffic.pdf

[R5] A Review on Networking and Multiplayer Computer Games,
URL: http://staff.cs.utu.fi/~jounsmed/papers/TR454.pdf

[R6] Cheat-Proofing Dead Reckoned Multiplayer Games,
URL: http://warriors.eecs.umich.edu/games/papers/adcog03-cheat.pdf

[R7] Aspects of Networking in Multiplayer Computer Games,
URL: http://staff.cs.utu.fi/~jounsmed/papers/AspectsOfMCGs.pdf

[R8] Second Life server architectuur documentatie,
URL: http://wiki.secondlife.com/wiki/Server_architecture

Bijlagen

Beschrijving van de wereld die samen met Jos Hoebe gemaakt is

Door een combinatie van mijn en Jos Hoebe's werk hebben we een echte genetwerkde virtuele wereld gecreëerd. We hebben eerst ieder onze eigen bachelor projecten afgemaakt, waarna we onze case studies samenvoegden. Mijn multiplayer code is gecombineerd met Jos' code die eilanden kan genereren door maar enkele nummers te gebruiken. Zo kost dit spel maar enkele megabytes om te downloaden, maar is er een eindeloze grote wereld vol met eilanden mogelijk. Gebruikers kunnen door mijn multiplayer code gezamenlijk deze wereld verkennen.

Jos' bachelor project getiteld "*Procedural Generated online virtual worlds*" kan worden gevonden op http://www.few.vu.nl/~jhe310/bp/.

Meer informatie over het gezamelijke werk kan worden gevonden via http://www.few.vu.nl/~mhn212/bp/