Avatar Animation in Networked Virtual Environments

Maarten Wijnants

2 juni 2003

Inhoudsopgave

Lijst van figuren

Voorwoord

Graag zou ik hier de mogelijkheid willen aangrijpen om enkele personen te bedanken voor de steun die ze mij hebben verleend bij de realisatie van mijn eindwerk. In de eerste plaats wil ik mijn promotor Prof. dr. Wim Lamotte en mijn begeleider Pieter Jorissen bedanken voor hun algemene coördinatie, richtlijnen, opmerkingen en suggesties. Tevens wil ik mijn ouders bedanken die mij de mogelijkheid gegeven hebben deze opleiding te volgen, en die mij steeds bijgestaan hebben met raad en daad. Tot slot zou ik zeker ook mijn vrienden willen bedanken voor hun niet-aflatende steun en voor het zorgen voor de nodige ontspanningsmomenten. Hierbij zou ik in het bijzonder Frederik Byl willen vermelden, die eigenhandig de volledige thesistekst nagelezen heeft.

Samenvatting

Een networked virtual environment is een systeem dat een interactieve virtuele wereld aanbiedt die gedeeld kan worden door geografisch verspreide gebruikers. Het is duidelijk dat een dergelijk systeem verschillende belangrijke subsystemen omvat. Zo zullen de gebruikers van het systeem bijvoorbeeld informatie moeten uitwisselen via het computernetwerk dat hen verbindt. Bovendien moet iedere gebruiker van een NVE op elk moment een min of meer consistente view op de aangeboden virtuele omgeving hebben. Tenslotte speelt ook awareness management een steeds belangrijkere rol in NVEs. NVE ontwerpers willen immers steeds grotere virtuele omgevingen aanbieden, en trachten alsmaar meer gelijktijdige gebruikers te ondersteunen. Een typische client machine is echter niet in staat om al de informatie te verwerken die gegenereerd wordt in dergelijke large-scale NVEs. Awareness management identificeert de informatie die van belang is voor een bepaalde gebruiker, en tracht zo de hoeveelheid data te beperken die gebruikers moeten verwerken.

Gebruikers van een 3D NVE worden in de virtuele omgeving voorgesteld door een zogenaamde avatar. Avatars vervullen enkele zeer belangrijke functies in de virtuele wereld, zoals bijvoorbeeld het visualiseren van de huidige interesse focus van gebruikers. Bovendien kunnen gebruikers via hun avatar interageren met de virtuele omgeving en met andere gebruikers.

Computer hardware heeft de afgelopen jaren een zodanige vooruitgang gekend dat het genereren van realistische beelden geen probleem meer vormt. Het is dan ook niet verwonderlijk dat de meeste recente NVEs visueel zeer realistische virtuele omgevingen aanbieden. Opdat dergelijke virtuele omgevingen hun geloofwaardigheid zouden behouden, moeten objecten in deze omgevingen echter eveneens op een realistische manier bewegen. Dit geldt in het bijzonder voor avatars. Daar interactiviteit één van de belangrijkste kenmerken van NVEs is, moeten deze animaties bovendien in real-time berekend worden. Skeletal animation, een animatietechniek die een animator in staat stelt figuren te animeren door enkel hun onderliggend skelet te manipuleren, is de laatste jaren uitgegroeid tot een standaard in dit gebied.

Hoofdstuk 1

Inleiding

Networked virtual environments zijn reeds een aantal jaar een zeer populair onderzoeksonderwerp. Hierbij moet echter opgemerkt worden dat er in verhouding nog maar weinig onderzoek gedaan werd naar de manier waarop gebruikers voorgesteld en geanimeerd kunnen worden in een NVE. In deze thesis wordt de beschikbare literatuur over deze twee onderwerpen kritisch besproken, en hun invloed op de belasting van het onderliggende computernetwerk bestudeerd. Alvorens deze onderwerpen aangesneden kunnen worden, is echter een goed begrip van networked virtual environments in het algemeen vereist. Daarom wordt er in hoofdstuk 2 dieper ingegaan op wat een networked virtual environment juist is. Bovendien worden er in dit hoofdstuk enkele mogelijke toepassingsgebieden en de belangrijkste samenstellende subsystemen van NVEs in detail behandeld.

De allereerste networked virtual environments waren tengevolge van het niet ter beschikking zijn van voldoende krachtige computer hardware volledig tekstgebaseerd. In dergelijke NVEs wordt er gebruik gemaakt van ongeformatteerde tekst om de virtuele omgeving en zijn objecten voor te stellen, en om de huidige toestand van geconnecteerde gebruikers te beschrijven. Het is duidelijk dat gebruikers van tekstgebaseerde NVEs niet gerepresenteerd moeten worden in de virtuele omgeving. Dankzij een gestage vooruitgang in computer hardware, kunnen meer recente networked virtual environments hun gebruikers echter een driedimensionale virtuele omgeving aanbieden. Opdat gebruikers op een correcte manier zouden kunnen interageren met elkaar, moeten ze in een 3D omgeving wel een virtuele belichaming hebben. Daarom worden gebruikers van een 3D NVE in de virtuele wereld voorgesteld door een zogenaamde avatar. In hoofdstuk 3 wordt er dieper ingegaan op het belang en het gebruik van avatars in networked virtual environments.

Eén van de hoofdbetrachtingen van NVE ontwerpers bestaat eruit gebruikers de illusie te geven dat ze zich daadwerkelijk in de aangeboden virtuele omgeving bevinden. Daarom wordt er vaak getracht deze omgevingen visueel zo realistisch mogelijk te maken. Het is echter niet moeilijk om in te zien dat de illusie van onderdompeling reeds snel zal verdwijnen indien de objecten in dergelijke virtuele omgevingen op een onrealistische manier bewegen. Dit geldt in het bijzonder voor de avatars van de geconnecteerde gebruikers. Daar interactiviteit één van de belangrijkste kenmerken van NVEs is, moeten avatars bovendien in *real-time* geanimeerd worden in de virtuele omgeving. In hoofdstuk 4 wordt er dieper ingegaan op deze problematiek. Dit hoofdstuk concentreert zich voornamelijk op skeletal animation, een vrij recente animatietechniek die in staat is om in real-time zeer realistische bewegingen te produceren voor complexe figuren.

Deze thesis zou uiteraard niet compleet zijn zonder enkele bestaande networked virtual environments in detail te beschouwen. Daarom wordt er in hoofdstuk 5 getracht een representatieve staal van bestaande NVEs aan te bieden, waarbij zowel pioniersystemen als meer moderne NVEs behandeld worden. Bovendien werd er in het kader van deze thesis een eenvoudige NVE in de programmeertaal C++ geïmplementeerd. Het doel van deze implementatie bestond uit het demonstreren van het gebruik van skeletal animation in een networked virtual environment, en het bestuderen van de invloed hiervan op de netwerkbelasting. In hoofdstuk 6 worden de resultaten van de implementatie gerapporteerd. In hoofdstuk 7 tenslotte worden de algemene resultaten van het eindwerk opgesomd, en wordt er op basis van deze eindresultaten een algemene conclusie geformuleerd.

Hoofdstuk 2

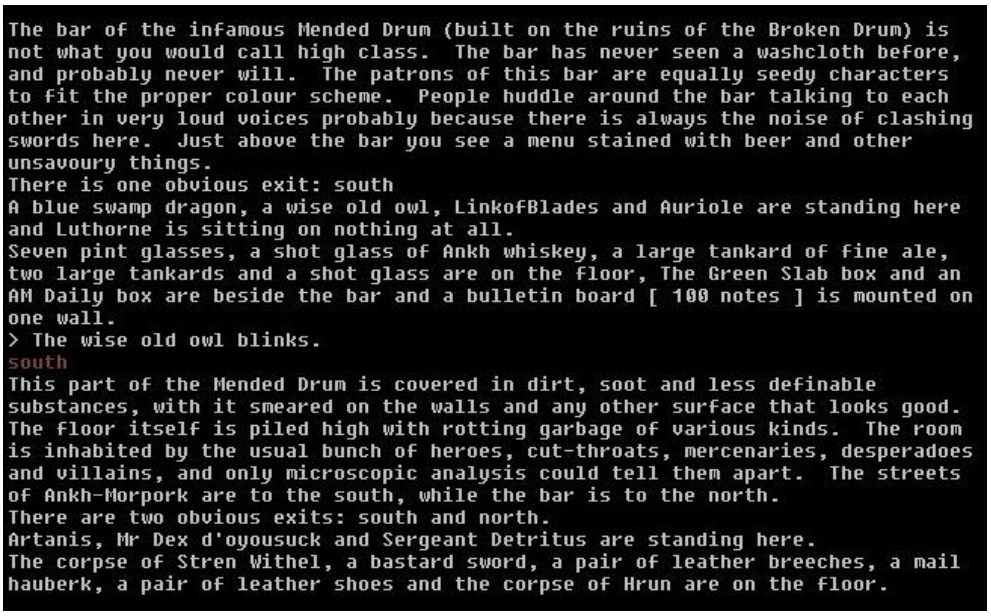
Networked Virtual Environments

2.1 Inleiding

Alvorens te definiëren wat een Networked Virtual Environment (NVE) juist is, lijkt het me nuttig kort hun ontstaan en evolutie te bespreken. NVEs zijn ontstaan doordat onderzoekers van enerzijds virtual reality en anderzijds computer networking de handen in elkaar sloegen. Maar ook de CSCW (Computer-Supported Cooperative Work) gemeenschap heeft een belangrijke invloed gehad op de ontwikkeling van NVEs ([1]). Het was immers reeds snel duidelijk dat NVEs ontzettend veel mogelijkheden bieden om de collaboratie tussen geografisch verspreide personen te verbeteren. En net zoals bij bijna alle grote uitvindingen (zoals bijvoorbeeld het Internet), was ook het Amerikaanse leger betrokken bij de ontwikkeling van NVEs. Eén van de allereerste NVEs was immers een militaire simulatie, genaamd het SIMNET systeem (zie sectie 5.3). Tenslotte hebben onderzoekers van NVE technologie ook veel inspiratie gehaald uit Multi-User Dungeons (MUDs, [2]).

MUDs ontstonden begin jaren 1980 en worden vooral gebruikt voor recreatieve doeleinden. Het zijn programma's die gebruikers toegang geven tot een gedeelde database die kamers en andere objecten bevat. Gebruikers kunnen bewegen in een kamer en andere kamers binnengaan. Hierbij zien ze enkel de objecten in de huidige kamer. MUDs gebruiken geen 3D graphics om de kamers en objecten voor te stellen, maar ongeformateerde tekst. Toch kan een MUD beschouwd worden als een NVE, daar meerdere gebruikers tegelijk geconnecteerd kunnen zijn met een MUD. Bovendien kunnen deze gebruikers zowel met de wereld als met de andere geconnecteerde gebruikers interageren.

Figuur 2.1 toont een typische situatie uit een moderne MUD, namelijk de Discworld MUD ([3]). De gebruiker bevindt zich in een bar die bestaat uit twee aparte



Figuur 2.1: Een typische situatie uit de Discworld MUD ([3])

kamers. De figuur geeft duidelijk weer hoe tekst gebruikt wordt om zowel de objecten als de andere gebruikers in de kamer te beschrijven. Ongeveer halfweg de figuur beslist de gebruiker in kwestie de huidige kamer te verlaten via de zuidelijke deur (hij heeft `south` ingetikt, wat als bruine tekst op het scherm verschijnt). Hierdoor betreedt hij de andere kamer van de bar, die opnieuw beschreven wordt met behulp van tekst. De gebruiker kan nu de bar verlaten, of terugkeren naar de vorige kamer (door respectievelijk `south` of `north` in te typen).

NVEs bestaan nog maar een twintigtal jaar, maar ze hebben in deze korte periode wel reeds een enorme evolutie ondergaan. Waar de eerste NVEs nog volledig tekstgebaseerd waren, maken bijna alle moderne NVEs nu reeds gebruik van geluid en 3D graphics. Bovendien worden NVEs fysisch ook steeds realistischer, en blijft ook het aantal gebruikers dat tegelijk met een NVE geconnecteerd kan zijn toenemen. Daar computers alsmaar krachtiger worden (de wet van Moore is nog steeds van toepassing) en gebruikers steeds meer netwerk bandbreedte ter beschikking hebben, is de kans groot dat deze evolutie zich zal doorzetten. Dit wil zeggen dat de mogelijkheid bestaat dat er binnen afzienbare tijd zeer realistische NVEs zullen ontstaan die tot duizenden gebruikers tegelijk kunnen ondersteunen.

2.2 Wat zijn Networked Virtual Environments?

Het is moeilijk om Networked Virtual Environments volledig te specificeren, omdat ze zoveel verschillende aspecten omvatten. Zo spelen bijvoorbeeld computer graphics en computer networking een belangrijke rol in NVEs, maar vaak is ook de collaboratie en communicatie tussen gebruikers van een NVE van belang. In de literatuur kan men dan ook verschillende definities van NVEs terugvinden. In [4] spreekt men bijvoorbeeld over "systemen die geografisch verspreide gebruikers toelaten te interageren in gedeelde virtuele werelden". In [5] wordt een NVE dan weer gedefinieerd als "een omgeving die gedeeld wordt door meerdere deelnemers, waarbij elke gebruiker geconnecteerd is via een verschillende host". Indien er in deze thesis over een NVE gesproken wordt, verstaan we hieronder *een systeem dat een interactieve virtuele wereld presenteert die gedeeld kan worden door geografisch verspreide gebruikers*. Deze beschrijving maakt meteen duidelijk dat NVEs gebaseerd zijn op twee belangrijke technologieën, namelijk Virtual Reality en computer networking. Gebruikers worden in de virtuele wereld voorgesteld door een grafische belichaming of *avatar*. Via zijn avatar kan een gebruiker niet enkel interageren met de virtuele omgeving, maar ook met andere gebruikers.

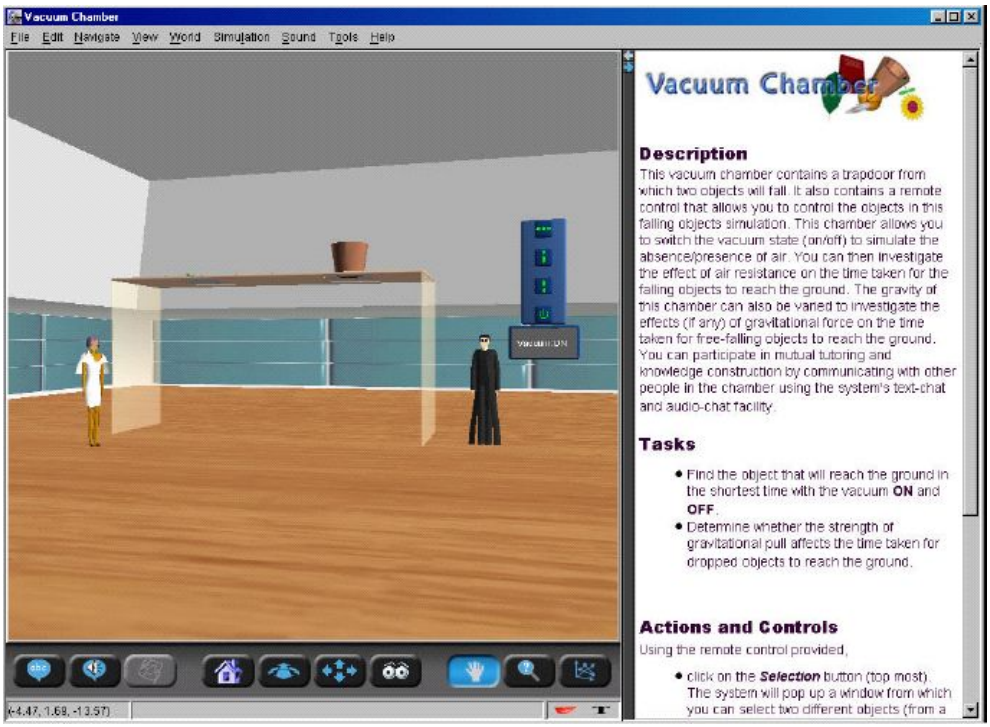
NVEs hebben ontzettend veel toepassingsgebieden. Momenteel wordt NVE technologie vooral gebruikt voor ontspanningsdoeleinden. Een mooi voorbeeld hiervan zijn multiplayer games, en in het bijzonder MMORPGs (Massively Multiplayer Online Role-Playing Games) zoals bijvoorbeeld Anarchy Online ([6]) en Dark Age Of Camelot ([7]). Een MMORPG biedt een 3D virtuele wereld aan waarin honderden gamers gelijktijdig via het Internet aanwezig kunnen zijn. Een speler moet bepaalde opdrachten (zogenaamde *quests*) uitvoeren, kan communiceren met andere spelers, kan zijn karakter laten evolueren, kan zich aansluiten bij een bepaalde spelersgroep in de virtuele wereld (zogenaamde *guilds*), enzovoort. MMORPGs kunnen met andere woorden beschouwd worden als de 3D opvolgers van de tekstuele MUDs. MMORPGs worden alsmaar populairder, en het is dan ook niet verwonderlijk dat er momenteel veel MMORPGs in ontwikkeling zijn, zoals bijvoorbeeld World of Warcraft ([8]) en Star Wars Galaxies ([9]).

Een andere toepassing uit de ontspanningssector zijn de virtuele chatruimtes. Dergelijke programma's bieden een virtuele wereld aan waarbinnen een gebruiker kan navigeren en communiceren met andere gebruikers. Dit communiceren gebeurt meestal via tekst, maar kan soms ook via spraak gebeuren. Active Worlds ([10]) is een voorbeeld van een dergelijke virtuele chatruimte die bovendien nog enkele extra functionaliteiten aanbiedt. Gebruikers kunnen in Active Worlds immers ook zelf

virtuele omgevingen aanmaken en winkelen in een virtueel winkelcentrum.

Behalve voor ontspanningsdoeleinden, kan NVE technologie eveneens gebruikt worden om de collaboratie tussen geografisch verspreide personen te ondersteunen en te verbeteren. In dit geval spreekt men meestal over Collaborative Virtual Environments (CVEs) in plaats van over NVEs. Een CVE kan bijvoorbeeld gebruikt worden om personen op te leiden. Zo bieden verschillende opleidingscentra voor piloten hun studenten een CVE aan waarmee ze virtuele vluchten kunnen uitvoeren. De Heathrow luchthaven in Londen bijvoorbeeld beschikt momenteel over 18 dergelijke CVEs die allemaal een bepaald type vliegtuig simuleren ([11]). Studenten moeten plaatsnemen in een exact nagebouwde cockpit die gemonteerd is op een beweegbaar platform. Dit platform reageert op een realistische manier op de acties van de student. Piloten kunnen aldus leren omgaan met een bepaald type vliegtuig zonder dat er mensenlevens op het spel staan, en kunnen zo bovendien leren hoe ze moeten reageren in etreme situaties die in de werkelijke wereld moeilijk of zelfs helemaal niet gesimuleerd kunnen worden. Maar CVEs kunnen bijvoorbeeld ook gebruikt worden als educatief hulpmiddel in het middelbaar onderwijs. Een mooi voorbeeld hiervan is het C-VISions systeem ([12], zie ook foto 2.2). C-VISions is een 3D CVE die studenten toelaat deel te nemen aan collaboratieve en interactieve virtuele simulaties. Door studenten te laten samenwerken in deze virtuele simulaties, probeert men ze een beter inzicht te geven in bepaalde begrippen uit de fysica, biologie en chemie.

NVEs blijken uitermate geschikt voor dergelijke educatieve doeleinden ([12]). Ten eerste ondersteunen NVEs proefondervindelijk leren. Studenten leren niet *over* iets, maar ervaren zelf hetgeen ze willen leren. Dit leidt normaal gezien tot een beter begrip van de dingen die men wil aanleren. Bovendien kunnen de studenten zelf hun leerervaring controleren. Ze bepalen bijvoorbeeld zelf aan welk tempo ze willen leren. Ten tweede kan een NVE verschillende views aanbieden op eenzelfde onderwerp. Met behulp van NVEs kunnen studenten een onderwerp vaak bekijken vanuit standpunten die in de echte wereld niet mogelijk zijn. Zo kan bijvoorbeeld een CVE die gebruikt wordt om bepaalde principes uit de chemie aan te leren een view *in* een molecule aanbieden, zodat de verschillende atomen en elektronen in de molecule zichtbaar worden. Ten derde is het mogelijk om in NVEs tabellen, statistieken, grafieken, en andere vormen van grafische informatie op te nemen. Deze kunnen de student helpen om hun virtuele ervaringen om te zetten in abstracties en regels over het domein waarover ze aan het leren zijn. Tenslotte mag men ook niet vergeten dat meerdere personen gelijktijdig in een NVE aanwezig kunnen zijn. Door samen te werken en te discussiëren met elkaar in de virtuele omgeving, kunnen studenten heel



Figuur 2.2: De "Vacuum Chamber", een virtuele simulatie uit het C-VISions systeem waarmee studenten ervaring kunnen opdoen over de fysische wetten die het vallen van objecten beïnvloeden ([12])

veel bijleren over het onderwerp dat ze *samen* aan het bestuderen zijn. Indien een persoon bijvoorbeeld een bepaalde observatie niet begrijpt, kan hij vragen stellen aan de andere gebruikers die op dat moment in dezelfde virtuele wereld aanwezig zijn en de gebeurtenis in kwestie dus ook geobserveerd hebben.

Het is onmogelijk om alle mogelijke applicaties van NVEs te bespreken, er zijn er gewoonweg te veel. De lezer zou nu echter ongeveer een beeld moeten hebben van wat NVEs zijn, en waar ze voor gebruikt kunnen worden. Het is nu dan ook tijd om dieper in te gaan op hoe NVEs werken.

2.3 Hoe werken Networked Virtual Environments?

Een Networked Virtual Environment is een complex software systeem dat uit verschillende belangrijke subsystemen bestaat. Zo moet een moderne 3D NVE bijvoorbeeld over een krachtige grafische engine beschikken die in staat is om in real-time scènes op het scherm van de gebruiker te tekenen. Hedendaagse 3D grafische engines

maken hiervoor onder andere gebruik van *polygon culling* (polygonen die niet zichtbaar zijn voor de gebruiker worden ook niet getekend) en *level of detail rendering* (indien een object ver verwijderd is van de gebruiker, worden er minder polygonen gebruikt om het object voor te stellen dan wanneer de gebruiker zich dicht bij het object bevindt). Naast visueel realisme speelt ook geluid een belangrijke rol in NVEs. Daarom is het wenselijk dat een moderne NVE ook een goede ondersteuning voor geluid heeft. Bovendien is het interessant wanneer bepaalde fysische wetten (zoals bijvoorbeeld de zwaartekracht en wrijving) gesimuleerd kunnen worden in een virtuele omgeving. Een NVE die claimt realistisch te zijn, kan dan ook niet zonder een goede physics engine die onder andere *real-time collision detection and response* ondersteunt.

De NVE ervaring wordt realistischer naarmate de hierboven aangehaalde subsystemen gesofistikeerder worden. Deze subsystemen zijn dus wel degelijk belangrijk, maar ze zijn niet *essentieel* voor de werking van een NVE. Zo is bijvoorbeeld de grafische engine van de meeste hedendaagse NVEs nog erg primitief (zie bijvoorbeeld de foto's in hoofdstuk 5). In de rest van deze sectie worden er twee subsystemen besproken die *wel* essentieel zijn voor de werking van een NVE. Bovendien wordt er ook dieper ingegaan op een subsysteem dat zeer belangrijk is voor NVEs die veel gelijktijdige gebruikers trachten te ondersteunen.

2.3.1 Netwerk communicatie

Dé kenmerkende eigenschap van een NVE is de aanwezigheid van een computernetwerk waarmee gebruikers informatie kunnen uitwisselen ([13]). Het is immers deze eigenschap die een NVE onderscheidt van een single-user Virtual Reality applicatie. De ontwerper van een NVE moet op netwerkniveau enkele belangrijke beslissingen nemen die de werking en de schaalbaarheid van de NVE enorm beïnvloeden.

Transport protocol

Een NVE ontwerper moet eerst en vooral bepalen welk transport protocol hij gaat gebruiken voor zijn applicatie ([13]). Een transport protocol voorziet diensten die door applicaties gebruikt kunnen worden om data te versturen over een computernetwerk. Het is algemeen aanvaard dat een hedendaagse NVE niet enkel bruikbaar moet zijn op een LAN, maar ook over het Internet. Op deze manier wordt de NVE immers voor veel meer personen toegankelijk. Het Internet heeft twee belangrijke protocollen in de transport laag, namelijk het *Transmission Control Protocol* (TCP)

en het *User Datagram Protocol* (UDP), die beide zowel voor- als nadelen hebben ([14]).

TCP is een connection-oriented protocol. Dit wil zeggen dat twee computers een connectie moeten opzetten alvorens ze kunnen communiceren met elkaar, en dat ze deze connectie terug moeten vrijgeven wanneer alle data uitgewisseld is. Bovendien is TCP een reliable protocol. TCP verstuurt immers automatisch acknowledgements voor ontvangen pakketten, en wanneer er een pakket verloren gaat (bijvoorbeeld door ruis op het netwerk), zal TCP het automatisch opnieuw versturen. TCP heeft tevens een ingebouwd congestiebeheersingsmechanisme dat tracht te voorkomen dat het onderliggende netwerk verzadigd geraakt, en het protocol maakt gebruik van flow control om te voorkomen dat een snelle zender een trage ontvanger kan overspoelen met pakketten. TCP biedt applicaties dus een betrouwbare end-to-end stroom van bytes aan over een mogelijk onbetrouwbaar netwerk.

UDP daarentegen is een connectionless en unreliable transport protocol. Dit wil zeggen dat er geen connecties opgezet of vrijgegeven moeten worden tussen computers, en dat het protocol geen automatische transmissie van acknowledgements of retransmissie van verloren pakketten voorziet, niet zorgt voor de ordening van pakketten, en geen gebruik maakt van flow control. UDP verstuurt enkel en alleen onafhankelijke pakketten (in deze context vaak *datagrammen* genoemd), zonder te garanderen dat pakketten betrouwbaar of in volgorde afgeleverd worden aan de geadresseerde computer. UDP biedt applicaties dus enkel een best-efforts datagram service aan.

TCP voorziet duidelijk veel meer diensten dan UDP, maar daar is (uiteraard) ook een prijs aan verbonden. TCP werkt met connecties, en het opzetten van deze connecties kost tijd en resources (er moet een pad gezocht worden van bron naar bestemming, een host moet enkele variabelen bijhouden voor elke TCP connectie die hij momenteel onderhoudt met andere hosts, routers moeten onthouden op welke uitgaande lijn ze binnenkomende pakketten moeten plaatsen voor elke connectie die door de router passeert, enzovoort). Bovendien kan het wachten op acknowledgements en het opnieuw versturen van verloren pakketten kostbare tijd in beslag nemen. Het onbetrouwbare UDP levert al deze diensten niet, en is dan ook veel sneller dan TCP. Dit wil zeggen dat TCP het meest geschikte protocol is om data te versturen die correct afgeleverd moet worden, terwijl UDP het aangewezen protocol is wanneer pakketten zo snel mogelijk of steeds binnen een bepaalde tijdsperiode moeten arriveren bij de bestemming. Indien overigens slechts een beperkt niveau van betrouwbaarheid gewenst is, wordt TCP vaak te inefficiënt bevonden. Een NVE ontwerper kan in dit geval terugvallen op UDP, en zelf mechanismen implementeren

om het protocol betrouwbaarder te maken ([13]). Zo kan een zender bijvoorbeeld verplicht worden een uniek sequence number in ieder datagram te plaatsen. Dit stelt de ontvanger in staat verloren datagrammen te detecteren, en ontvangen datagrammen te ordenen. Uiteraard, hoe meer diensten men bovenop UDP bouwt, hoe minder efficiënt UDP wordt.

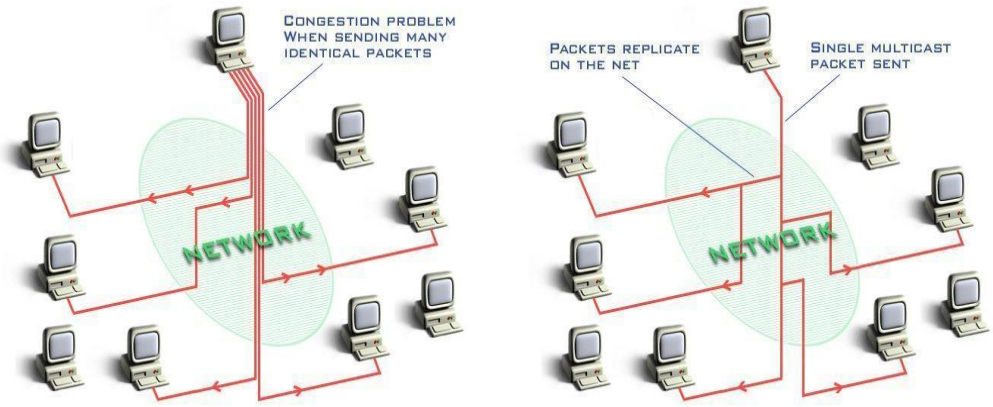
Niets belet de NVE ontwerper overigens van beide protocollen te combineren in zijn applicatie. Dit is uitermate handig, daar er in een moderne NVE vaak verschillende *soorten* informatie uitgewisseld moeten worden, die elk verschillende eisen stellen wat betreft betrouwbaarheid en snelheid van aflevering. Login informatie bijvoorbeeld moet steeds correct ontvangen worden door de bestemming, en kan dus best verzonden worden met TCP. Maar andere soorten data, zoals bijvoorbeeld gedigitaliseerde spraak, moeten steeds op tijd afgeleverd worden (late data is in dit geval erger dan incorrecte data). Voor zulke informatie is UDP duidelijk het meest geschikte transport protocol. De meeste hedendaagse NVEs gebruiken dan ook verschillende transport protocollen om verschillende soorten data te verzenden.

Distributie van berichten

De tweede beslissing die een NVE ontwerper moet nemen betreft de manier waarop berichten uitgewisseld worden tussen gebruikers. De NVE ontwerper heeft de keuze uit drie distributieschema's, namelijk *unicast*, *broadcast* en *multicast* ([13], [14]). Unicast communicatie gebeurt steeds tussen één zender en één ontvanger. Dit wil zeggen dat een unicast bericht enkel ontvangen wordt door de computer voor wie het bericht bestemd is. Broadcasting daarentegen levert elk verzonden bericht af aan *alle* computers op een netwerk. Multicasting tenslotte stelt een zender in staat een bericht naar een groep van specifieke computers te sturen ([16]). Multicasting werkt met *multicast groepen*, die elk een bepaald multicast adres hebben. Computers kunnen multicast groepen *joinen* en terug verlaten, en wanneer een zender een bericht naar een multicast groep stuurt, zullen enkel de computers die tot deze groep behoren het bericht ontvangen. Zowel TCP als UDP kunnen gebruikt worden om unicast berichten te versturen over een computernetwerk, maar broadcasting en multicasting is enkel mogelijk met UDP datagrammen.

Het komt vaak voor dat een bericht naar meerdere bestemmingen tegelijk verstuurd moet worden. Wanneer bijvoorbeeld een gebruiker zich verplaatst in een virtuele omgeving, moet dit normaliter meegedeeld worden aan verschillende andere gebruikers van de NVE. Indien de NVE gebruik maakt van unicasting, zal de zender in dit geval hetzelfde bericht meerdere keren moeten versturen, namelijk één

maal naar elke bestemming (zie figuur 2.3(a)). Dit wil zeggen dat de zender redelijk wat bandbreedte ter beschikking moet hebben, en dat hij op elk moment een lijst moet bijhouden van gebruikers die geïnteresseerd zijn in zijn data. Het is duidelijk dat dit distributieschema snel voor problemen kan zorgen wanneer er veel gebruikers gelijktijdig in de virtuele omgeving aanwezig zijn. Broadcasting daarentegen is een meer schaalbaar distributieschema. Een broadcast bericht wordt immers door iedere computer op een netwerk ontvangen. Dit wil zeggen dat een zender elk bericht steeds slechts één keer moet versturen, zelfs wanneer meerdere gebruikers in het bericht geïntereseerd zijn. Spijtig genoeg zijn er twee grote nadelen verbonden aan broadcasting. Ten eerste forwarden gateways broadcast berichten *niet*. Dit impliceert dat zowel de zender als alle mogelijke bestemmingen tot hetzelfde netwerk moeten behoren. Internet NVEs kunnen dus *geen* gebruik maken van broadcasting. Ten tweede is broadcasting een inefficiënt distributieschema. Elke computer op een lokaal netwerk zal immers elk broadcast bericht voor dat netwerk ontvangen en inspecteren, zelfs als de computer in kwestie niet geïnteresseerd is in het bericht. Dit ontvangen en inspecteren van "ongewenste" broadcast berichten neemt kostbare tijd en resources in beslag.



Figuur 2.3: Distributie van een bericht tussen computers: (a) unicast; (b) multicast ([16])

Multicast biedt, net zoals broadcast, multihost aflevering van berichten aan (zie figuur 2.3(b)), maar heeft de twee nadelen die verbonden zijn aan broadcasting *niet*. Multicasting kan namelijk wel degelijk gebruikt worden om WAN NVEs te implementeren, en een multicast bericht wordt steeds enkel ontvangen door computers die daadwerkelijk in het bericht geïnteresseerd zijn. Momenteel is er echter nog één groot probleem geassocieerd met multicasting: het wordt nog niet door alle netwer-

ken en operating systems ondersteund. Hoewel de meeste recente routers multicast wel degelijk ondersteunen, is dit spijtig genoeg niet het geval voor het gros van de oudere routers. Dit wil bijvoorbeeld zeggen dat multicasting over het Internet momenteel nog onmogelijk is, daar veel routers die in het huidige Internet in gebruik zijn reeds een lange tijd meegaan. Desalniettemin wordt multicasting als het meest geschikte distributieschema voor NVEs beschouwd ([13]). Indien een NVE ontwerper beslist multicasting te gebruiken voor zijn applicatie, moet hij wel nog bepalen hoe de informatie uit de virtuele wereld verdeeld moet worden over verschillende multicast groepen. Meestal gebeurt dit op basis van spatiale informatie. De locales uit Spline zijn hier een mooi voorbeeld van (zie sectie 2.3.3 voor meer uitleg).

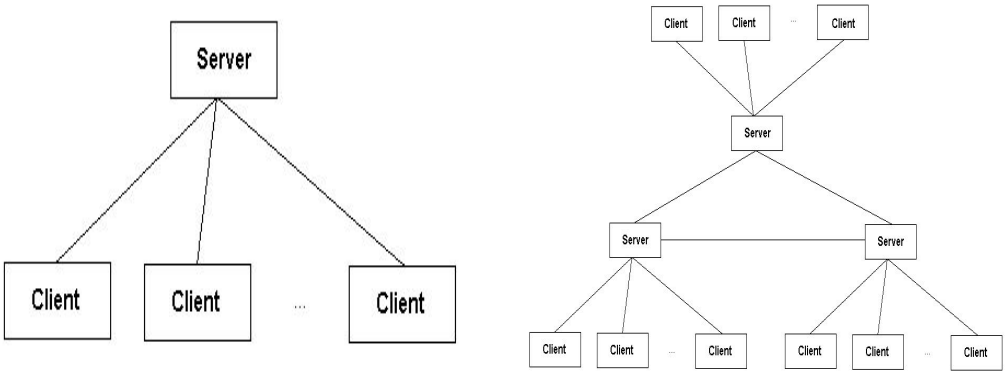
De drie distributieschema's kunnen bovendien gecombineerd worden in een applicatie. Zo kan een NVE bijvoorbeeld een beroep doen op unicasting om de communicatie tussen een gebruiker en een bepaalde server van het systeem te verzorgen, en tegelijkertijd multicasting gebruiken om berichten te versturen die bestemd zijn voor meerdere computers (zoals bijvoorbeeld positie-updates van gebruikers). Broadcasting wordt echter bijna niet meer gebruikt in moderne NVEs, omdat het een ineffeciënt distributieschema is dat bovendien aanleiding geeft tot applicaties die enkel bruikbaar zijn op een LAN.

Netwerk topologie

Tenslotte moet een NVE ontwerper nog beslissen welke netwerk topologie hij gaat gebruiken voor zijn applicatie. Deze beslissing bepaalt hoe berichten gerouteerd zullen worden in het systeem. Er bestaan twee belangrijke netwerk topologieën voor NVEs, namelijk de *client-server architectuur* en de *peer-to-peer topologie* ([13], [17]).

Een client-server systeem bestaat uit een aantal client machines en één of meer servers (zie figuur 2.4). De karakteristieke eigenschap van deze topologie bestaat eruit dat clients niet rechtstreeks met elkaar communiceren, maar via de servers. Wanneer een client een bericht naar een andere client wil versturen, moet hij dit bericht naar een bepaalde server sturen. Indien deze server de geadresseerde client rechtstreeks kan bereiken, zal hij het bericht naar die client forwarden. Indien dit niet het geval is, zal de server het bericht forwarden naar een andere server die de geadresseerde client wel kan bereiken. Het grote voordeel van de client-server architectuur ligt in het feit dat servers binnenkomende berichten kunnen inspecteren (en eventueel bewerken) alvorens ze verder te sturen. Op basis van de aldus verkregen informatie kan een server bijvoorbeeld beslissen dat een ontvangen bericht enkel van belang is voor een beperkt aantal clients, en het bericht vervolgens enkel

naar deze clients doorsturen. Dit spaart zowel bandbreedte als processortijd van clients, daar ze op deze manier geen "irrelevante" berichten moeten ontvangen en verwerken. Maar een server kan bijvoorbeeld ook redundantie elimineren in opeenvolgende berichten, of meerdere berichten samenvoegen tot één enkel bericht. Het tweede voordeel van deze topologie is dat servers clients interessante diensten kunnen aanbieden, zoals bijvoorbeeld interaction detection en voice bridging ([18]). Het idee hierachter is dat servers normaal gezien meer bandbreedte en processorkracht ter beschikking hebben dan client machines, en dat een server dus meer geschikt is om dergelijke rekenkundig intensieve operaties uit te voeren.



Figuur 2.4: De client-server architectuur: (a) 1 server; (b) meerdere servers ([13])

De client-server topologie heeft echter ook twee nadelen. Ten eerste introduceert het routeren van berichten via servers extra delay. Een server moet immers elk binnenkomend bericht inspecteren, en vervolgens beslissen naar welke servers en/of clients het bericht doorgestuurd moet worden. Dit neemt kostbare tijd in beslag. Indien er meerdere servers in het systeem aanwezig zijn, is het zelfs mogelijk dat een bericht meerdere servers moet passeren alvorens het bij de geadreseerde client arriveert. Het is duidelijk dat de netwerkconnectie tussen de servers in dit geval zeer snel moet zijn om de delay aanvaardbaar te houden. Ten tweede vormen de servers de bottleneck van het systeem. Elke server heeft slechts een beperkte hoeveelheid bandbreedte en processorkracht ter beschikking. Dit wil zeggen dat er een bovengrens is op het aantal gebruikers dat een server kan ondersteunen. Indien deze bovengrens overschreden wordt, daalt de prestatie van het systeem zeer snel (delay neemt toe, berichten gaan verloren omdat de buffers van de servers vol zitten, enzovoort). Dit probleem kan enkel opgelost worden door extra servers aan het systeem toe te voegen.

Een zuiver peer-to-peer systeem daarentegen bevat geen servers, enkel clients (in deze context vaak *peers* genoemd). Elke client kan rechtstreeks communiceren met elke andere client van het systeem. Het grote voordeel hiervan is dat berichten niet langs servers moeten passeren, en dat de delay dus normaal gezien kleiner zal zijn dan in een client-server systeem. Bovendien is deze topologie meer schaalbaar dan de client-server topologie. Zoals reeds vermeld plaatsen servers een limiet op het aantal gelijktijdige gebruikers van het systeem. Een peer-to-peer topologie is ook niet oneindig schaalbaar, maar ervaring en onderzoek heeft toch aangetoond dat deze limiet sneller bereikt wordt in een client-server systeem dan in een peer-to-peer systeem ([51]). Het nadeel van de peer-to-peer topologie is dan weer dat er veel verantwoordelijkheid op de clients geplaatst wordt. Zo zijn het in een client-server systeem bijvoorbeeld de servers die beslissen welke clients een bepaald bericht moeten ontvangen. In een peer-to-peer topologie moeten clients deze beslissing zelf nemen. Dit wil zeggen dat clients in een peer-to-peer systeem normaal gezien meer processorkracht en bandbreedte ter beschikking moeten hebben dan in een client-server systeem. Maar aangezien computers voor thuisgebruik alsmaar krachtiger en goedkoper worden, en breedband Internetaansluitingen stilaan ingeburgerd raken, vormt dit steeds minder een probleem.

Een NVE ontwerper kan er overigens ook voor kiezen de client-server en peer-to-peer topologie te combineren met elkaar. Men kan in een peer-to-peer systeem bijvoorbeeld enkele special purpose servers invoegen die bepaalde taken uitvoeren voor clients. Een dergelijk systeem buit de voordelen van beide topologieën uit, zonder dat er extra nadelen geïntroduceerd worden. Merk tenslotte ook op dat transport protocol, distributieschema en netwerk topologie niet losstaan van elkaar. Een zuiver client-server systeem maakt meestal gebruik van TCP of UDP unicasting, terwijl een zuiver peer-to-peer systeem bijna altijd multicasting of broadcasting zal gebruiken (om de bandbreedte die clients ter beschikking moeten hebben aanvaardbaar te houden).

2.3.2 Gedeelde informatie consistent houden

De virtuele omgeving die aangeboden wordt door een NVE wordt gedeeld door verschillende gebruikers, en bevindt zich op elk moment in een bepaalde toestand. Deze toestand beschrijft onder andere de positie en oriëntatie van alle gebruikers en objecten in de omgeving. Wanneer de toestand aangepast wordt, moeten alle gebruikers in de NVE hiervan op de hoogte gebracht worden, zodat ze hun model van de wereld kunnen aanpassen. Iedere gebruiker van een NVE moet met andere

woorden op elk moment een min of meer consistente view op de gedeelde virtuele omgeving hebben. Dit is nodig om een gebruiker het gevoel te geven samen met andere gebruikers in eenzelfde omgeving aanwezig te zijn. Onderstel bijvoorbeeld dat gebruiker A bepaalde acties uitvoert in een NVE, maar dat de andere gebruikers hier niet van op de hoogte gebracht worden. Dit wil zeggen dat andere gebruikers nooit zullen reageren op een actie van gebruiker A. Het gevolg hiervan is dat deze gebruiker reeds snel de illusie zal verliezen in een multiuser omgeving aanwezig te zijn, daar hij zich geïsoleerd voelt van de andere gebruikers. Dit voorbeeld toont aan dat het beheren van de toestand van de gedeelde virtuele wereld een zeer belangrijk onderdeel van elke NVE vormt.

Aanpassingen aan deze toestand worden aan de verschillende gebruikers van de NVE meegedeeld via het computernetwerk dat hen verbindt. Het versturen van berichten over een computernetwerk introduceert echter steeds een zekere delay, daar elektronen en licht (in het geval van glasvezel verbindingen) niet oneindig snel zijn. Dit wil zeggen dat de informatie in een bericht reeds verouderd kan zijn op het moment dat een bepaalde gebruiker het bericht ontvangt. Onderstel bijvoorbeeld dat een gebruiker een object in de virtuele wereld verplaatst. De NVE zorgt ervoor dat alle andere gebruikers op de hoogte gebracht worden van de nieuwe positie van het object door een bericht te versturen over het onderliggende netwerk. Op het moment dat een bepaalde gebruiker dit bericht ontvangt, kan de positie van het object in kwestie echter alweer veranderd zijn. Bovendien is het mogelijk dat verschillende gebruikers eenzelfde updatebericht op verschillende tijdstippen ontvangen. Dit kan ertoe leiden dat verschillende gebruikers verschillende modellen van de virtuele omgeving onderhouden (mogelijk slechts voor een zeer korte tijd). Men kan aantonen dat het onmogelijk is te garanderen dat elke gebruiker een consistente view heeft op een gedeelde virtuele omgeving die snel kan veranderen ([13]). Deze stelling staat bekend als de *consistency-throughput tradeoff*, en heeft als gevolg dat een NVE ontwerper een afweging zal moeten maken tussen consistentie en dynamiek.

In [13] worden er drie basisbenaderingen geïdentificeerd die door een NVE ontwerper aangewend kunnen worden om de toestand van de gedeelde omgeving te beheren. Een NVE ontwerper kan ten eerste gebruik maken van een *repository* die op elk moment de meest recente informatie over alle gedeelde "voorwerpen" (objecten, gebruikers, weersvooruizichten, enzovoort) in de virtuele omgeving bevat. Telkens een gebruiker informatie over een gemeenschappelijk voorwerp nodig heeft, moet hij deze informatie uit de repository halen. In het geval van een *passieve repository* ligt de controle over de communicatie tussen gebruiker en repository bij de gebruiker: het is de gebruiker die de communicatie initieert door een request naar

de repository te sturen. In het geval van een *actieve repository* daarentegen houdt elke gebruiker een lokale kopie bij van de informatie in de repository, en stuurt de repository nieuwe of aangepaste informatie automatisch door naar gebruikers, zodat ze hun lokale kopie kunnen updaten. Het grote voordeel hiervan is dat een gebruiker informatie over de gedeelde virtuele omgeving rechtstreeks uit zijn lokale kopie kan halen[1]. Maar een gebruiker kan uiteraard niet enkel informatie uit de repository opvragen, hij kan de informatie in de repository ook bijwerken. Het is duidelijk dat de toestand van een gemeenschappelijk voorwerp nooit door twee (of meer) gebruikers tegelijk aangepast mag worden. Dit impliceert dat een repository steeds gebruik zal moeten maken van een lockingmechanisme om dergelijke situaties te voorkomen. Merk tenslotte op dat de repository niet gecentraliseerd hoeft te zijn. De repository kan immers verspreid worden over de verschillende gebruikers van de NVE, waarbij elke gebruiker een deel van de gemeenschappelijke informatie bijhoudt.

De tweede benadering, *frequent state regeneration*, vereist dat ieder gemeenschappelijk voorwerp juist één eigenaar heeft, en verplicht elke host regelmatig updateberichten te versturen voor alle gedeelde voorwerpen waarvan hij eigenaar is. Elk updatebericht bevat een *volledige* beschrijving van de toestand van een bepaald voorwerp, en moet steeds naar alle gebruikers van de NVE verstuurd worden. Men kan dus best gebruik maken van multicasting (of eventueel broadcasting) om deze berichten te distribueren. Wanneer een host een updatebericht ontvangt, gebruikt hij de informatie in het bericht om zijn lokaal model van de virtuele omgeving bij te werken. Merk op dat updateberichten bij deze benadering onbetrouwbaar verstuurd kunnen worden. Indien een host een updatebericht voor een bepaald voorwerp niet ontvangt, heeft dit immers geen dramatische gevolgen, daar de eigenaar spoedig een nieuw updatebericht voor dit voorwerp zal versturen. Waarom moet nu elk gedeeld voorwerp juist één eigenaar hebben? Onderstel dat een bepaald voorwerp twee (of meer) eigenaars heeft. Dit zou als gevolg hebben dat meerdere hosts updateberichten versturen voor hetzelfde voorwerp. Het is niet moeilijk om in te zien dat dit tot inconsistenties kan leiden. Onderstel nu daarentegen dat een bepaald object geen eigenaar heeft. Dit zou het ongewenste effect hebben dat het object pas zichtbaar wordt voor nieuwe gebruikers wanneer een bepaalde gebruiker het object in bezit neemt. Dit wil zeggen dat er, net zoals bij de repository benadering, een lock manager nodig is die eigendom over elk gedeeld voorwerp toekent aan juist één host. Indien geen enkele gebruiker eigenaar is van een bepaald voorwerp, moet de lock

[1]Er is dus geen netwerk communicatie nodig met de repository. Zoals reeds vermeld introduceert netwerk communicatie steeds een zekere delay.

manager dit object zelf in bezit nemen[2].

De derde benadering tenslotte heet *dead reckoning*, en verplicht hosts eveneens updateberichten te versturen voor alle gedeelde voorwerpen waarvan ze eigenaar zijn (waarbij ieder gedeeld object weer een unieke eigenaar moet hebben). Elk dead reckoning protocol bestaat uit twee delen, namelijk een *prediction algoritme* en een *convergence algoritme*. Bij de twee vorige benaderingen gebruikte een host de informatie in een updatebericht voor een bepaald voorwerp enkel en alleen om zijn lokaal model van de virtuele omgeving bij te werken. Bij de dead reckoning benadering daarentegen gebruikt een host deze informatie om de toekomstige toestand van dat voorwerp te voorspellen. Het prediction algoritme is verantwoordelijk voor het maken van deze voorspellingen. Dit algoritme kan bijvoorbeeld rekening houden met de huidige snelheid en versnelling van een object om de toekomstige positie van het object te schatten. Dit vereist uiteraard wel dat deze informatie in elk updatebericht opgenomen wordt. Over het algemeen geldt de regel dat de schattingen nauwkeuriger worden naarmate het prediction algoritme met meer factoren rekening houdt. De keerzijde van de medaille is echter dat updateberichten in dat geval ook steeds meer informatie moeten bevatten, en dat de berekeningen van het prediction algoritme alsmaar complexer worden en dus steeds meer tijd in beslag zullen nemen. Daarom wordt er meestal voor gekozen het prediction algoritme niet te complex te maken. Wanneer een host nu een nieuw updatebericht voor een bepaald gedeeld voorwerp ontvangt, is het mogelijk dat de geschatte toestand van dit object verschilt van de toestand die aangegeven wordt in het updatebericht. Indien dit het geval is, moet het convergence algoritme ervoor zorgen dat de huidige, incorrecte toestand van het object overgaat in de juiste toestand. Het convergence algoritme kan bijvoorbeeld simpelweg de huidige toestand vervangen door de correcte toestand. Deze aanpak kan echter aanleiding geven tot onrealistische situaties waarbij objecten bijvoorbeeld ineens naar een andere positie verspringen. Daar dit meestal niet gewenst is, zal het convergence algoritme meestal voor een soepelere overgang zorgen. Een NVE ontwerper moet echter rekening houden met het feit dat hier uiteraard ook een rekenkundige kost aan verbonden is.

Een NVE ontwerper zal een keuze moeten maken uit deze drie mogelijke benaderingen. Welke benadering het meest geschikt is, hangt af van de vereisten en eigenschappen van de beoogde applicatie. Indien absolute consistentie gewenst is, is de repository benadering de aangewezen keuze. Het is immers veel moeilijker om absolute consistentie te garanderen wanneer er gebruik gemaakt wordt van één van de twee andere benaderingen. Maar deze twee benaderingen zijn dan weer, in te-

[2]De lock manager zal zelf updateberichten voor deze objecten moeten versturen.

genstelling tot de repository benadering, uitermate geschikt om de toestand van een dynamische virtuele omgeving te beheren. Een extra pluspunt van dead reckoning ten opzichte van de twee andere benaderingen is dat deze aanpak de netwerktrafiek aanzienlijk kan verminderen door updateberichten onregelmatig te versturen. Bij een dead reckoning systeem kunnen zenders immers rekening houden met het gebruikte prediction algoritme, en enkel updateberichten versturen wanneer de voorspelde toestand op remote hosts te fel verschilt van de daadwerkelijke toestand van een bepaald object. Het is duidelijk dat op deze manier het aantal verstuurde updateberichten enorm gereduceerd kan worden.

2.3.3 Awareness management

In theorie moet een gebruiker van een NVE steeds verwittigd worden wanneer er iets verandert in de virtuele wereld, zodat hij zijn lokaal model van de wereld kan aanpassen. Dit verwittigen gebeurt met behulp van berichten die informatie over de verandering in kwestie bevatten. Wanneer bijvoorbeeld een bepaalde gebruiker bewogen heeft in een virtuele omgeving, moeten in principe alle andere gebruikers, die op dat moment in dezelfde omgeving aanwezig zijn, op de hoogte gebracht worden van de nieuwe positie van deze gebruiker. Met de huidige netwerktechnologie en processorkracht is dit enkel haalbaar wanneer er slechts enkele gebruikers (hoogstens een vijftigtal) tegelijk met de NVE geconnecteerd zijn.

Veel NVEs worden echter pas echt interessant wanneer er honderden tot zelfs duizenden gebruikers gelijktijdig in de virtuele wereld aanwezig zijn. Bovendien is de virtuele ruimte die een NVE aanbiedt vaak erg uitgestrekt en bezaaid met verschillende objecten waarmee gebruikers kunnen interageren. Dit wil zeggen dat een gebruiker van dergelijke NVEs ontzettend veel informatie moet bijhouden over de virtuele wereld (zoals bijvoorbeeld de positie en de toestand van de objecten en gebruikers in de virtuele wereld, grafische informatie over de virtuele ruimte, enzovoort) en veel updateberichten zal moeten ontvangen en verwerken. Deze updateberichten worden normaal gezien via het onderliggende netwerk naar de gebruiker in kwestie verstuurd. De netwerk bandbreedte die een doorsnee gebruiker tot zijn beschikking heeft, volstaat echter meestal *niet* om een dergelijke hoeveelheid informatie af te handelen.

Het doel van awareness management in NVEs bestaat uit het beperken van de hoeveelheid informatie die een bepaalde gebruiker moet verwerken. Onder informatie verstaan we hier bijvoorbeeld de grafische representatie, positie en toestand van gebruikers en objecten in de NVE, de geluiden die gebruikers en objecten produce-

ren, de acties die gebruikers uitvoeren, enzovoort. Awareness management tracht dit doel te bereiken door de informatie te identificeren die van belang is voor een bepaalde gebruiker. Enkel en alleen deze informatie wordt verstuurd naar de gebruiker in kwestie. Een gebruiker zal dus met andere woorden geen "irrelevante" data ontvangen en ook niet moeten verwerken. Indien het hier over grafische data gaat, wil dat zeggen dat deze data bovendien niet gerenderd moet worden door de client PC. Dit kan van belang zijn voor gebruikers die niet beschikken over een krachtige grafische computer.

Awareness management vormt een zeer belangrijk onderdeel van NVEs. Door de hoeveelheid informatie te beperken die gebruikers moeten verwerken, zorgt awareness management er immers voor dat gebruikers met een minder goede netwerkverbinding en/of minder krachtige PC ook kunnen deelnemen aan networked virtual environments. Bovendien kan awareness management de collaboratie tussen gebruikers van een NVE verbeteren door alle informatie uit de NVE te onderdrukken die niet relevant is voor hun huidige collaboratieve taak ([19]).

Er bestaan verschillende soorten awareness management modellen voor NVEs. Ik besteed in deze thesis voornamelijk aandacht aan de naar mijn mening twee meest krachtige modellen die momenteel in gebruik zijn, namelijk modellen gebaseerd op *locales* en het *spatial model of interaction*. Alvorens dieper in te gaan op deze twee modellen, is het misschien nuttig om op te merken dat awareness management ook in de werkelijke wereld voorkomt. Niemand is zich immers bewust van *alles* wat er in de echte wereld gebeurt. We weten vrij goed wat er zich in onze onmiddellijke omgeving (bijvoorbeeld het land waarin we leven) afspeelt, maar we weten slechts zeer weinig of zelfs helemaal niets over de gebeurtenissen in ver afgelegen gebieden (bijvoorbeeld een land in een ander continent).

Locales

De notie locale werd geïntroduceerd in het Spline systeem ([20], zie ook sectie 5.2). Een locale is een beperkt virtueel gebied zoals bijvoorbeeld een kamer of een (kleine) open ruimte, en kan virtuele objecten en avatars bevatten. Door verschillende locales samen te voegen, kan men een virtuele omgeving creëren. Of, anders gezegd, een virtuele wereld kan opgedeeld worden in een aantal locales. Belangrijk hierbij is dat deze opdeling transparant is voor de gebruiker. Dit wil zeggen dat de gebruiker geen naden kan zien tussen de verschillende locales die samen een virtuele omgeving vormen. Locales kunnen een willekeurige vorm hebben, en elk object in de opgesplitste NVE bevindt zich steeds in exact één locale. Bovendien definieert

elke locale zijn eigen coördinatensysteem.

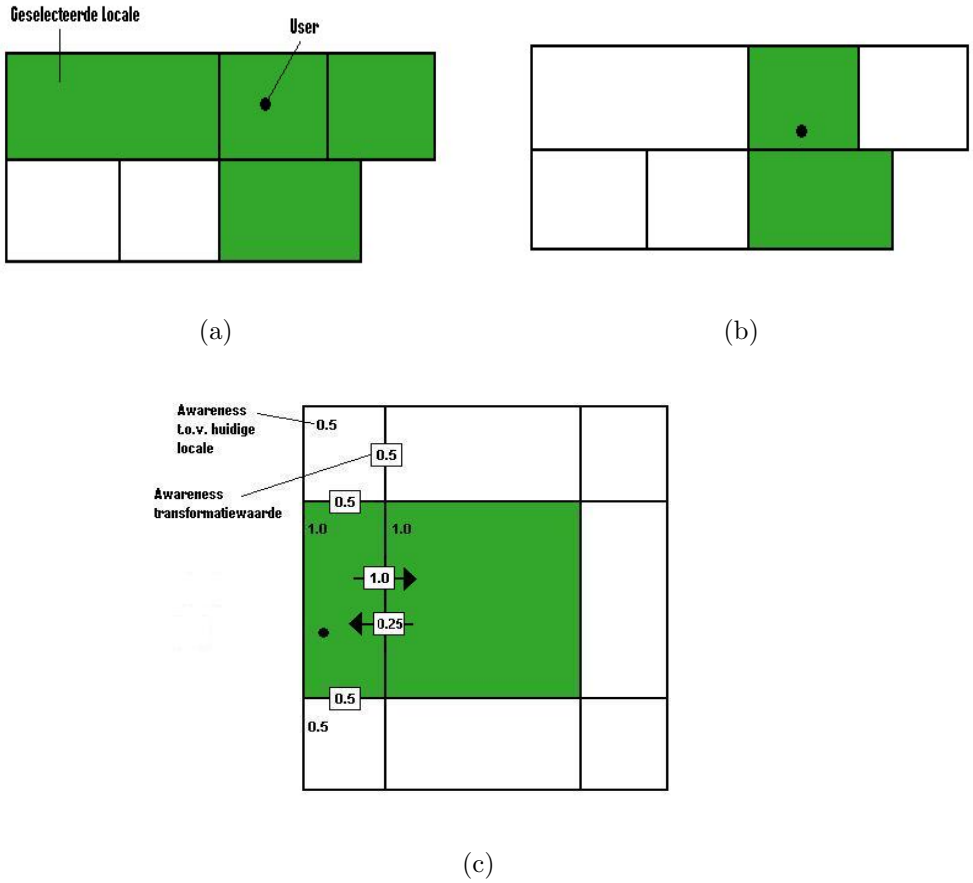
Hoewel een virtuele wereld uit verschillende locales kan bestaan en erg uitgestrekt kan zijn, kan een bepaalde gebruiker steeds enkel waarnemen wat er in zijn onmiddellijke omgeving gebeurt. Dit wil zeggen dat een gebruiker, die zich in locale X bevindt, enkel op de hoogte moet zijn van wat er zich afspeelt in de locales die zich in de buurt van locale X bevinden. De gebruiker in kwestie moet enkel data over deze locales ontvangen, en enkel deze locales (en de objecten die erin aanwezig zijn) moeten gerenderd worden.

Elke locale is in Spline geassocieerd met een bepaald communicatiekanaal. Dit communicatiekanaal wordt gebruikt om berichten te versturen en te ontvangen over de locale en de objecten die zich in de locale bevinden. Indien bijvoorbeeld de positie van een object in locale X wijzigt, wordt dit meegedeeld via het communicatiekanaal dat geassocieerd is met X. Elk communicatiekanaal omvat typisch een aantal multicast adressen, één adres voor elk van de verschillende soorten data die aanwezig zijn in de NVE (zoals bijvoorbeeld visuele data, audio data, ...). Door nu enkel te connecteren met bepaalde multicast adressen (bijvoorbeeld de adressen van de locale waar de gebruiker zich momenteel in bevindt, en de adressen van de buurlocales van deze locale), ontvangt een gebruiker enkel informatie over een (klein) gedeelte van de virtuele wereld. De gebruiker ontvangt *geen* data over locales die ver van hem verwijderd zijn of waar hij niet in geïnteresseerd is. Dit heeft als voordeel dat het netwerk van de gebruiker minder belast wordt, dat de gebruiker zijn model van de virtuele wereld klein kan houden, en dat de gebruiker minder informatie moet verwerken en renderen.

Dit toont aan hoe locales gebruikt kunnen worden als basis voor awareness management. Een gebruiker moet zich immers enkel bewust zijn van de huidige locale en zijn directe omgeving. Maar over welke locales uit die directe omgeving moet een gebruiker nu informatie ontvangen en over welke niet? Hier bestaan verschillende strategieën voor. Enkele mogelijke (en eenvoudige) locale selection policies zijn ([21]):

- **Nearest Neighbor Selection**: Nearest Neighbor Selection selecteert enkel de locale waar de gebruiker zich momenteel in bevindt, en de onmiddellijke buurlocales van deze locale (zie figuur 2.5(a))
- **N Nearest Selection**: N Nearest Selection selecteert de N locales die zich het dichtst bij de huidige positie van de gebruiker bevinden (zie figuur 2.5(b) voor N = 2)

- **N Most Aware Selection**: N Most Aware Selection selecteert de huidige locale, en de N - 1 locales die de hoogste awareness hebben ten opzichte van de huidige locale. Dit werkt als volgt. De huidige locale heeft awareness 1, en met elke grens van een locale is er een awareness transformatiewaarde geassocieerd. Onderstel bijvoorbeeld dat de grens tussen de huidige locale en een buurlocale awareness transformatiewaarde 0.9 heeft. Dan heeft deze buurlocale een awarenesswaarde van 0.9 ten opzichte van de huidige locale (zie figuur 2.5(c) voor N = 2; deze figuur stelt een virtueel stadion voor dat opgedeeld is in locales)



Figuur 2.5: Locale Selection Policies: (a) Nearest Neighbor Selection; (b) 2 Nearest Selection; (c) 2 Most Aware Selection ([21])

Locales kunnen echter niet enkel gebruikt worden als basis voor awareness management, ze bieden ook nog enkele andere interessante mogelijkheden voor NVEs

([20]). Zo kunnen ze bijvoorbeeld gebruikt worden om problemen in verband met precisie over lange afstanden op te lossen. Elke locale definieert immers zijn eigen coördinatensysteem. Dit wil zeggen dat de objecten die zich in een locale bevinden gepositioneerd worden ten opzichte van de oorsprong van deze locale. Precisie kan dus behouden worden door de locales voldoende klein te houden. Locales maken het verder ook mogelijk dat meerdere personen gelijktijdig verschillende delen van een virtuele omgeving ontwerpen. Iedere locale heeft immers zijn eigen coördinatensysteem, en de relatie tussen een locale en zijn buren wordt steeds lokaal gespecifieerd (er is geen globaal coördinatensysteem voor de virtuele omgeving). Om vervolgens de afzonderlijk ontworpen locales samen te voegen tot één virtuele omgeving, moet men enkel de relaties aangeven die gelden tussen elke twee locales die buren worden in de virtuele wereld. Tenslotte kunnen locales ook gebruikt worden om enkele interessante effecten te creëren. Een mooi voorbeeld hiervan is het "Desert House" in Diamond Park (zie sectie 5.2 voor meer informatie over Diamond Park). Het interieur van dit huis is groter dan het exterieur. Met behulp van locales kunnen dus niet-Euclidische werelden gecreëerd worden.

Spatial model of interaction

Het doel van het spatial model of interaction ([22]) bestaat uit het controleren van de interactie tussen objecten in een gedeelde virtuele ruimte. Objecten kunnen in deze context zowel voorwerpen als gebruikers zijn. Om dit doel te bereiken, maakt het model gebruik van de volgende concepten:

- **Medium**: Elke interactie tussen objecten gebeurt steeds via een bepaald communicatiemedium. Voorbeelden van mogelijke media zijn het visuele medium, het audio medium, het tekstuele medium, enzovoort.
- **Aura**: De aura van een object definieert een deelruimte waarin interactie met andere objecten mogelijk is. Wanneer een object zich verplaatst in de virtuele omgeving, draagt het zijn aura's met zich mee.
- **Focus**: De focus van een observerend object geeft aan waarop het object momenteel zijn aandacht gevestigd heeft. Hoe meer een object A zich in de focus van een observerend object B bevindt, hoe bewuster B zich is van A.
- **Nimbus**: De nimbus van een geobserveerd object beschrijft zijn aanwezigheid en manifestatie in de virtuele omgeving. Hoe meer een object A zich in de nimbus van een geobserveerd object B bevindt, hoe bewuster A zich is van B.

- **Awareness**: De awareness van een object A ten opzichte van een ander object B drukt uit hoe belangrijk B voor A is.
- **Adapter**: Een adapter is een object in de virtuele wereld dat een invloed heeft op de aura, focus en/of nimbus van andere objecten in de wereld.

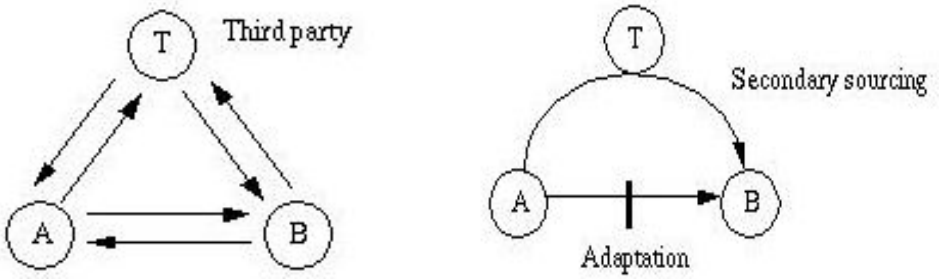
Ieder object heeft een aura, focus en nimbus voor elk medium waarin het kan interageren. Bovendien kunnen de verschillende aura's, foci en nimbi van een object verschillen in zowel vorm als grootte. Zo kan bijvoorbeeld de visuele aura van een object groter zijn dan zijn auditieve aura. Interactie tussen twee objecten via een bepaald medium is enkel mogelijk wanneer hun aura's voor dit medium overlappen. Een object is zich enkel bewust van de objecten waarmee interactie mogelijk is, en kan ook enkel van deze objecten informatie en updateberichten ontvangen. Het concept aura beperkt met andere woorden de aanwezigheid van een object in een virtuele omgeving, en heeft dus een gelijkaardige functie als de zopas besproken locales. De NVE houdt toezicht op de aura's van alle objecten in de virtuele wereld, en verwittigt objecten wanneer interactie met een ander object mogelijk wordt. Indien dit het geval is, wordt er een connectie opgezet tussen de twee objecten. Deze connectie kan door de objecten in kwestie gebruikt worden om onderling informatie uit te wisselen. De objecten (en dus niet de NVE) zijn zelf verantwoordelijk voor het opzetten, onderhouden en beëindigen van dergelijke connecties.

Nadat de connectie aangemaakt is, wordt de awareness bepaald die beide objecten van elkaar hebben in de verschillende media. Hiervoor wordt er gebruik gemaakt van de concepten focus en nimbus. Meer concreet: de awareness die object A heeft van object B in medium M, is een functie van de focus van A op B in M en de nimbus van B op A in M. Awareness is dus, net zoals de concepten aura, focus en nimbus, medium-afhankelijk, en de awareness die twee objecten van elkaar hebben moet niet noodzakelijk symmetrisch zijn (de awareness die A heeft van B kan verschillen van de awareness die B heeft van A). Eens de awareness tussen twee objecten met overlappende aura's berekend is, wordt deze gebruikt om de interactie tussen de objecten te controleren. Zo kan de awareness tussen twee objecten bijvoorbeeld het level of detail van de grafische representatie van de objecten bepalen, of het volume van een audio kanaal tussen de objecten regelen. De berekende awareness bepaalt met andere woorden de *kwaliteit* van de informatie die via de opgezette connectie uitgewisseld wordt.

Adapters zijn objecten in de virtuele wereld die de aura, focus en/of nimbus van objecten versterken of verzwakken. Stel u bijvoorbeeld voor dat iemand in de werkelijke wereld een podium betreedt. Dit heeft normaliter twee gevolgen. Ten

eerste worden hierdoor meer mensen zich bewust van deze persoon, en ten tweede manifesteert deze persoon zich meer in wereld. Deze effecten kunnen gesimuleerd worden met het spatial model of interaction door het podium als een adapter object te beschouwen dat zowel de aura als de nimbus vergroot van de gebruikers op het podium.

Het spatial model of interaction is een krachtig en flexibel model, maar heeft twee grote tekortkomingen ([23]). Ten eerste houdt het model - het erg beperkte concept van adapter objecten buiten beschouwing gelaten - geen rekening met contextuele factoren bij het bereken van de awareness tussen objecten. De awareness die twee objecten van elkaar hebben hangt immers enkel af van de focus en de nimbus van de objecten in kwestie. Specifieke informatie uit de virtuele omgeving (zoals bijvoorbeeld de regio waar de objecten zich in bevinden) wordt genegeerd bij de awareness berekening. Ten tweede functioneert het model niet goed wanneer er veel gebruikers gelijktijdig in de virtuele wereld aanwezig zijn. Hoewel het concept aura het aantal mogelijke interacties dat beschouwd moet worden drastisch vermindert, is interactie steeds slechts mogelijk tussen *twee* objecten. Interactie tussen een object en een groep van objecten wordt niet ondersteund door het model. Dit wil zeggen dat een gebruiker vaak veel connecties met objecten zal moeten onderhouden, en in een drukbevolkte NVE vaak nog te veel informatie zal ontvangen. Om deze twee problemen te verhelpen, werd het spatial model of interaction uitgebreid met een nieuw concept, namelijk third party objects.



Figuur 2.6: (a) Third party objects; (b) De effecten van third party objects ([24])

Third party objects zijn onafhankelijk objecten die de awareness tussen andere objecten beïnvloeden, en kunnen beschouwd worden als een uitbreiding van adapters ([23]). Waar vroeger alle interactie steeds tussen slechts twee objecten gebeurde, is het nu mogelijk dat er hier nog een derde object bij betrokken is. In dat geval zijn er individuele awareness relaties tussen zowel de twee interagerende objecten, als

tussen het third party object en deze twee objecten (zie figuur 2.6(a)). Third party objects kunnen twee soorten effecten hebben op awareness relaties, namelijk *adaptation* en *secondary sourcing* (zie figuur 2.6(b)). Adaptation slaat op de manipulatie van bestaande awareness relaties tussen twee objecten, en is dus identiek aan het effect van adapter objecten. Maar het is echter ook mogelijk dat informatie, afkomstig van een object of een groep van objecten, via het third party object afgeleverd wordt aan een ander object. Dit heet secondary sourcing. Het third party object kan de ontvangen informatie op de één of andere manier transformeren alvorens ze verder te sturen. Met behulp van secondary sourcing is het dus mogelijk een alternatieve representatie van een object of een groep van objecten aan te bieden aan een ander object. Een voorbeeld van een alternatieve representatie is een "abstractie" (een vereenvoudigde en goedkopere voorstelling) van een menigte van gebruikers. De effecten adaptation en secondary sourcing kunnen overigens op verschillende manieren gecombineerd worden in verschillende communicatiemedia.

De introductie van third party objects heeft het spatial model of interaction nog veel krachtiger gemaakt. Met behulp van secondary sourcing kan een third party object verschillende individuele awareness relaties tussen een object en andere objecten vervangen door één enkele awareness relatie. Het gevolg hiervan is dat dat object veel minder connecties moet onderhouden, en dat er veel minder informatie uitgewisseld wordt via het onderliggende netwerk. Third party objects kunnen op deze manier bijvoorbeeld gebruikt worden om menigtes van gebruikers in NVEs te ondersteunen ([24]). Maar third party objects kunnen hun effecten ook recursief op elkaar toepassen. Dit heeft als gevolg dat ze bijvoorbeeld ook gebruikt kunnen worden om een virtuele wereld op te delen in regio's, die mogelijk verschillende awareness manipulaties geassocieerd hebben met hun grenzen ([23]).

Andere awareness management modellen

Locales en het spacial model of interaction zijn zeker niet de enige awareness management modellen die momenteel in gebruik zijn. In NPSNET-IV bijvoorbeeld wordt de virtuele wereld opgedeeld in hexagonale cellen met een vaste grootte, en heeft elke gebruiker een zogenaamde *area of interest*. Een gebruiker van NPSNET-IV is zich steeds enkel bewust van de cellen die in zijn area of interest liggen. Het awareness management model van het RING systeem is dan weer gebaseerd op line-of-sight visibility. In het RING systeem is elke gebruiker zich immers enkel bewust van de objecten en gebruikers die hij daadwerkelijk kan zien. Zowel NPSNET als RING en hun respectievelijke awareness management modellen worden in meer detail bespro-

ken in hoofdstuk 5.

Het awareness management model van het PARADISE (Performance ARchitecture for Advanced Distributed Interactive Simulation Environments) systeem houdt rekening met zowel spatiale als organisatorische factoren via zogenaamde *projection aggregations*. Elk projection aggregation bevat objecten en gebruikers die tot dezelfde organisatie behoren en zich in dezelfde regio van de virtuele omgeving bevinden ([25]). Zo kan een projection aggregation bijvoorbeeld een groep tanks voorstellen. Projection aggregations versturen regelmatig updateberichten die informatie bevatten over hun leden, zoals bijvoorbeeld het totaal aantal leden, een positie die de locaties van de leden samenvat, en de distributie van de leden ten opzichte van deze positie. Deze informatie kan gebruikt worden om een vereenvoudigde representatie te construeren van de leden van de projection aggregation in kwestie. Het is duidelijk dat de netwerktrafiek gereduceerd kan worden door projection aggregations te gebruiken voor entiteiten die ver verwijderd zijn of niet van belang zijn voor een bepaalde gebruiker.

2.4 Conclusie

Networked virtual environments zijn reeds een aantal jaar een zeer populair onderzoeksonderwerp. Het is dan ook niet verwonderlijk dat NVEs in hun korte bestaan reeds een indrukwekkende vooruitgang gekend hebben. In dit hoofdstuk werden de drie belangrijkste subsystemen van NVEs in detail behandeld, en enkele minder belangrijke subsystemen kort aangehaald. De prestaties van al deze subsystemen zijn in de afgelopen jaren met rasse schreden vooruitgegaan. Daar de interesse in het onderwerp niet lijkt af te nemen en onderzoekers geruggensteund worden door steeds krachtigere computer hardware, is de kans groot dat deze evolutie zich zal doorzetten.

Bovendien werden er in dit hoofdstuk een aantal mogelijke toepassingsgebieden van NVEs besproken. Er werd aangetoond dat NVEs onder andere gebruikt kunnen worden voor ontspanningsdoeleinden, om personen op te leiden, en om de collaboratie tussen geografisch verspreide gebruikers te verbeteren. Ongeacht het toepassingsgebied, moeten gebruikers van een 3D NVE echter op de één of andere manier gerepresenteerd worden in de virtuele omgeving. Deze problematiek wordt behandeld in het volgende hoofdstuk.

Hoofdstuk 3

Avatars in Networked Virtual Environments

3.1 Inleiding

De allereerste networked virtual environments waren volledig tekstgebaseerd. Tekstgebaseerde NVEs gebruiken ongeformatteerde tekst niet alleen om de virtuele omgeving en zijn objecten voor te stellen, maar ook om de huidige positie en toestand van de gebruikers te beschrijven. Dit wil zeggen dat gebruikers van dergelijke NVEs geen belichaming nodig hebben in de virtuele omgeving. Door enorme vooruitgang in computer hardware werd het echter reeds snel mogelijk om 3D graphics te gebruiken in een NVE. In een 3D omgeving moet een gebruiker wel een belichaming hebben. Daarom worden gebruikers van een 3D NVE in de virtuele wereld voorgesteld door een *avatar*. Het woord avatar is afkomstig uit het hindoeïsme, en betekent letterlijk "de incarnatie van een godheid" ([26]).

Avatars zijn er in alle maten en gewichten, gaande van simplistisch tot complex en van onrealistisch tot mensachtig (zie figuur 3.1). Simplistische avatars bestaan uit eenvoudige geometrische figuren zoals bijvoorbeeld bollen en kubussen. Complexere avatars daarentegen kunnen opgebouwd zijn uit honderden polygonen, en zijn meestal ook voorzien van een texture. Hoewel voor sommige applicaties eenvoudige avatars volstaan, worden realistische avatars meestal toch geprefereerd. Zo suggereren [27], [28] en [29] bijvoorbeeld dat mensachtige avatars aanleiding geven tot een groter gevoel van *presence* en *co-presence* dan eenvoudige of onrealistische avatars. Presence in een NVE kan gedefinieerd worden als het subjectieve gevoel echt in de virtuele omgeving aanwezig te zijn (de gebruiker vergeet als het ware de realiteit

en denkt dat hij zich in de omgeving bevindt die door de NVE gesuggereerd wordt, [27]). Met co-presence daarentegen wordt het gevoel samen met andere, menselijke gebruikers in eenzelfde omgeving aanwezig te zijn bedoeld ([27]). Ik concentreer me daarom in deze thesis op realistische en mensachtige avatars.

Figuur 3.1: Avatars in networked virtual environments: (a) simplistisch; (b) onrealistisch; (c) mensachtig ([28])

3.2 Het belang van avatars in Networked Virtual Environments

Avatars spelen een cruciale rol in 3D NVEs. Dit wordt duidelijk wanneer we bedenken welke belangrijke functies het menselijk lichaam in de echte wereld allemaal vervult. Het lichaam van een bepaald persoon verschaft informatie over onder andere de spatiale positie en oriëntatie, gemoedstoestand, belangstelling en sociale status van deze persoon. Bovendien is het lichaam ons medium van interactie en communicatie met de buitenwereld. Het is immers via ons lichaam dat we kunnen interageren met de wereld en kunnen communiceren met andere personen (zowel via spraak als via lichaamstaal). Aangezien een avatar de belichaming van een gebruiker in een NVE voorstelt, vervullen avatars in de virtuele omgeving soortgelijke functies ([30], [31]):

Aanwezigheid: Een avatar geeft de aanwezigheid van een bepaalde gebruiker in de virtuele omgeving aan. Het is erg belangrijk dat een gebruiker van een NVE op elk moment kan waarnemen of er in de virtuele omgeving andere gebruikers in zijn nabijheid aanwezig zijn of niet. Indien de NVE gebruik maakt van realistische, mensachtige avatars, kunnen gebruikers deze constatatie snel

en eenvoudig maken. Belichamingen van gebruikers kunnen in dit geval immers gemakkelijk onderscheiden worden van andere objecten in de virtuele omgeving.

Lokalisatie: Een avatar verschaft informatie over de positie, houding en oriëntatie van een gebruiker in de virtuele wereld. Deze informatie kan andere gebruikers hints geven over de huidige (gemoeds)toestand en bedoelingen van deze gebruiker. Zie sectie 3.4 voor meer uitleg.

Identificatie: Indien een persoon steeds dezelfde avatar gebruikt in de virtuele omgeving, zullen andere gebruikers hem gaan associëren met deze belichaming. Dit wil zeggen dat gebruikers herkend kunnen worden aan hun virtuele representatie. Dit is uiteraard zeer belangrijk in een NVE. Veronderstel bijvoorbeeld dat je op zoek bent naar een bepaalde gebruiker in de virtuele omgeving. Weten welke avatar de persoon gebruikt, kan deze zoektocht enorm vergemakkelijken.

Visualisatie van de interesse focus en acties van anderen: De avatar van een bepaalde gebruiker geeft aan waar het aandachtspunt van deze persoon in de virtuele omgeving ligt. Meestal is dit het punt waar de avatar naar kijkt. Bovendien kan men via de avatar te weten komen welke acties een gebruiker aan het uitvoeren is in de virtuele omgeving. Wanneer bijvoorbeeld een avatar een object opraapt, het vervolgens een bepaalde tijd met zich mee draagt en het tenslotte weer loslaat, weten we dat een bepaalde gebruiker juist een object in de virtuele wereld verplaatst heeft. Deze informatie is zeer belangrijk wanneer collaboratie of communicatie tussen gebruikers in de NVE van belang is.

Sociale representatie van de gebruiker via decoratie: Een gebruiker kan zijn avatar een persoonlijke *look* geven door de avatar te decoreren met bijvoorbeeld juwelen of bepaalde kledingstukken. Deze decoraties kunnen bovendien een sociale betekenis hebben, in de zin dat ze de taak of sociale status van een gebruiker in de virtuele wereld kunnen aangeven. Zo zal een virtuele politieagent bijvoorbeeld steeds een uniform dragen. Het is dan ook niet moeilijk om in te zien dat de decoratie van de avatar de interactie tussen gebruikers kan beïnvloeden.

Beschikbaarheid: Avatars kunnen aangeven of gebruikers beschikbaar zijn voor interactie of niet door voldoende informatie te tonen over de huidige activiteit van de gebruiker, of via een expliciete aanwijzing zoals bijvoorbeeld een “Do

not disturb" bordje. Veronderstel bijvoorbeeld dat gebruiker A niet reageert op een vraag van een andere gebruiker. Dit kan twee oorzaken hebben: ofwel negeert A de andere gebruiker, ofwel is A niet beschikbaar voor interactie (hij is te druk bezig, hij zit niet meer achter zijn PC, enzovoort). Het is uiteraard ontzettend belangrijk dat gebruikers een onderscheid kunnen maken tussen deze twee radicaal verschillende situaties.

3.3 Interactietechnieken voor Networked Virtual Environments

Via zijn avatar kan een gebruiker interageren met de virtuele omgeving. Dit impliceert echter dat een gebruiker zijn avatar op de één of andere manier moet kunnen controleren. [31] en [33] identificeren drie mogelijke benaderingen. In het geval van *directly controlled avatars* past de gebruiker de toestand van de avatar rechtstreeks aan. Er wordt in dit geval meestal gebruik gemaakt van sensoren om de bewegingen van de gebruiker te registreren, waarna de gewrichten en ledematen van zijn avatar op basis van deze informatie gepositioneerd worden. Het grote voordeel van deze benadering is dat er een realistische mapping is van bewegingen van de gebruiker naar resulterende bewegingen van de avatar. Anders gezegd, er is in dit geval een goede overeenkomst tussen proprioceptie en zintuiglijke informatie over het virtuele lichaam ([32]). Proprioceptie kan gedefinieerd worden als de onbewuste stroom van prikkels die in het menselijk lichaam ontstaat, en die ons op elk moment een idee geeft waar de verschillende delen van ons lichaam (ledematen, hoofd, romp, enzovoort) zich bevinden. Dankzij proprioceptie kunnen we bijvoorbeeld onze neus aanraken met gesloten ogen.

Een goede overeenkomst tussen proprioceptie en zintuiglijke informatie heeft een zeer gunstige invloed op het gevoel van presence. De kans is immers groot dat gebruikers zich in dit geval gaan identificeren met hun avatar ([32]). In het geval van *guided avatars* daarentegen specificeert de gebruiker enkel welke taken of acties zijn avatar moet uitvoeren, in plaats van rechtstreeks de toestand van de avatar aan te passen. Dit specificeren van taken gebeurt typisch met behulp van toetsenbord en/of muis. Zo kan een gebruiker een guided avatar laten wandelen door bijvoorbeeld de pijltjestoetsen van zijn toetsenbord in te drukken. Om daarentegen een directly controlled avatar te laten wandelen, zal een gebruiker bijvoorbeeld zelf ter plaatse moeten stappen. Dit wil zeggen dat er bij guided avatars geen rechtstreekse overeenkomst is tussen de bewegingen die een gebruiker uitvoert en de resulterende

bewegingen van zijn avatar, zoals bij directly controlled avatars wel het geval is. Het is dan ook niet verwonderlijk dat guided avatars normaal gezien aanleiding geven tot een minder gevoel van presence dan directly controlled avatars.

Autonomous avatars tenslotte handelen op basis van high-level doelstellingen en de informatie die ze ontvangen over de toestand van de virtuele omgeving. Een gebruiker kan dergelijke avatars "sturen" door de doelen van de avatar aan te passen. Een voorbeeld van een high-level doel zou bijvoorbeeld kunnen zijn "tracht zo weinig mogelijk te botsen met andere avatars en virtuele objecten". Om informatie over de virtuele omgeving te verkrijgen, maken autonomous avatars meestal gebruik van zogenaamde virtuele sensoren. Via deze virtuele sensoren kunnen autonomous avatars de virtuele omgeving zien, horen en voelen.

Autonomous avatars lijken de vreemde eend in de bijt, maar ze hebben wel degelijk toepassingsgebieden in NVEs ([29]). Zo kunnen ze bijvoorbeeld gebruikt worden om ervoor te zorgen dat er steeds een bepaald aantal avatars in de virtuele omgeving aanwezig is. Dit is vooral belangrijk wanneer men druk bevolkte omgevingen tracht te simuleren, zoals bijvoorbeeld voetbalstadia of luchthavens. Een andere mogelijke toepassing van autonomous avatars is de virtuele substituut. Een virtuele substituut is een autonome agent die in de virtuele omgeving handelt in naam van een bepaalde gebruiker. Indien een gebruiker bijvoorbeeld in de virtuele omgeving aanwezig wil blijven terwijl hij zelf met iets anders bezig is (en hij zijn avatar dus niet zelf kan besturen), kan hij een beroep doen op een autonomous avatar met de gewenste high-level doelstellingen.

3.4 Gebaren en niet-verbale communicatie

De avatar is voor een gebruiker niet enkel het medium van interactie in de virtuele omgeving, maar ook het medium van communicatie. Menselijke communicatie is gebaseerd op zowel spraak als niet-verbale communicatiemiddelen. Met niet-verbale communicatie bedoelen we het gebruik van het lichaam om informatie mee te delen tijdens de omgang met anderen. Zo vallen onder andere gebaren, de houding van het lichaam en gelaatsuitdrukkingen onder de noemer niet-verbale communicatie. Deze niet-verbale communicatiemiddelen kunnen in combinatie met spraak gebruikt worden om te herhalen, benadrukken, aan te vullen, tegen te spreken of te controleren wat er gezegd wordt. We gebruiken bijvoorbeeld vaak gebaren tijdens het spreken om de nadruk te leggen op bepaalde woorden, of om een visuele voorstelling te geven bij mondelinge beschrijvingen. Maar niet-verbale communicatie kan ook volledig losstaan van spraak. Zo kunnen we bijvoorbeeld uit de houding of de gelaatsuit-

drukkingen van een persoon afleiden wat de huidige emotionele toestand van deze persoon is, zonder dat hij expliciet moet zeggen hoe hij zich momenteel voelt. Meer zelfs, verschillende auteurs ([34] en [35] sommen er enkele op) beweren dat houding en gelaatsuitdrukkingen efficiënter en betrouwbaarder zijn dan spraak om emoties uit te drukken. Efficiënter in de zin dat deze niet-verbale communicatiemiddelen dingen kunnen uitdrukken die zeer moeilijk uit te drukken zijn met woorden, en betrouwbaarder in de zin dat ze belangrijker geacht worden dan spraak indien er een contradictie is tussen deze twee communicatiekanalen. Indien bijvoorbeeld iemand die er belabberd en vermoeid uitziet zegt dat hij zich fantastisch voelt, gaat men deze uitspraak automatisch in vraag stellen.

Studies hebben aangetoond dat meer dan 65 procent van de informatie die uitgewisseld wordt tijdens menselijke communicatie uitgedrukt wordt op een niet-verbale manier ([34]). Deze vaststelling, gecombineerd met de bovenstaande discussie, maakt duidelijk dat niet-verbale communicatie een zeer belangrijke rol speelt in de alledaagse interactie tussen mensen. Het is dan ook niet moeilijk om in te zien dat het integreren van niet-verbale communicatiemechanismen in NVEs de kwaliteit van de NVE ervaring voor gebruikers drastisch kan verbeteren. Gebaren en houdingen kunnen op twee manieren in NVEs geïntroduceerd worden ([33]). Ten eerste kan men gebruik maken van sensoren om de gebaren en de houding van een gebruiker vast te leggen, en deze informatie vervolgens gebruiken om de avatar dezelfde gebaren te laten uitvoeren en dezelfde houding te laten aannemen. Ten tweede kan de NVE een verzameling voorgedefinieerde gebaren en houdingen aanbieden waaruit gebruikers op elk moment een keuze kunnen maken (bijvoorbeeld via een grafische user interface). Merk op dat we er ons niet altijd bewust van zijn dat we een bepaald gebaar gebruiken of dat we een bepaalde houding aannemen. Het nadeel van de tweede benadering is dat gebruikers in dit geval zelf verantwoordelijk worden voor het dupliceren van dergelijke onbewuste gebaren en houdingen in de virtuele omgeving. Bovendien kan een NVE steeds slechts een eindige verzameling gebaren en houdingen aanbieden. Het is dus mogelijk dat een gebruiker op een zeker moment een gebaar of houding wil gebruiken die niet in deze verzameling is opgenomen. Hij kan in dit geval ofwel beslissen geen gebaar te gebruiken, ofwel terugvallen op een ander gebaar dat wel aangeboden wordt.

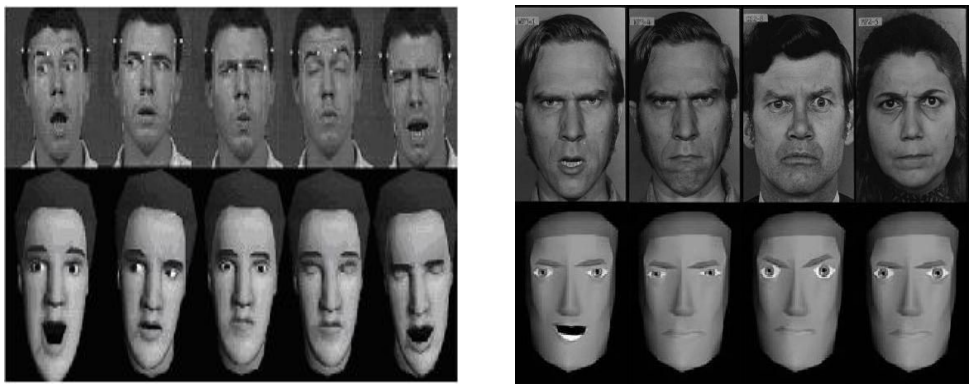
Om daarentegen gelaatsuitdrukkingen te integreren in NVEs, heeft een NVE ontwerper de keuze uit vier benaderingen ([33]). Ten eerste kan men een camera gebruiken om het gezicht van de gebruiker vast te leggen, en de geregistreerde videobeelden vervolgens als een texture op het gezicht van de avatar plaatsen (zie figuur 3.2). Deze benadering vereist dat de gebruiker steeds voor de camera blijft

zitten, en men zal een beroep moeten doen op beeldverwerkingstechnieken om het gezicht in de beeldenstroom te scheiden van de achtergrond. Het voordeel van deze methode is dat op deze manier de gelaatsuitdrukkingen van de gebruiker perfect gedupliceerd worden in de virtuele wereld, maar de benadering is spijtig genoeg ook ontzettend belastend voor het computernetwerk. De camera registreert immers meerdere beelden per seconde (meestal een vijftiental), en elk beeld moet naar alle andere gebruikers van de NVE gestuurd worden via het onderliggende computernetwerk. Dit wil zeggen dat deze methode niet geschikt is voor large-scale virtual environments met veel gelijktijdige gebruikers.

Figuur 3.2: Video-texturing van het gezicht van de gebruiker op de avatar ([33])

De tweede benadering heet model-based coding, en maakt eveneens gebruik van een camera. Er worden in dit geval echter geen videobeelden verstuurd over het computernetwerk, maar parameters die de huidige gelaatstrekken van de gebruiker beschrijven. Om deze parameters te bekomen, wordt de beeldenstroom afkomstig van de camera geanalyseerd. Het doel van deze analyse bestaat uit het lokaliseren van bepaalde onderdelen van het gezicht (ogen, lippen, wenkbrauwen, enzovoort) in het beeld. Op basis van de informatie die de analysefase oplevert, kan men bepalen wat de huidige opening van de mond is, wat de afstand tussen en de kromming van de wenkbrauwen is, waar de ogen zich bevinden in de oogholtes, enzovoort. Deze parameters worden vervolgens via het netwerk naar de andere gebruikers verstuurd, die de informatie gebruiken om het gezicht van de avatar in kwestie te animeren. Het is duidelijk dat deze benadering het netwerk minder belast dan de eerste benadering, maar ze is ook minder nauwkeurig. Bovendien zijn de beeldverwerkingstechnieken die in de analysefase gebruikt worden complexer dan deze uit de eerste benadering. Dit kan men gedeeltelijk verhelpen door markers op het gezicht van de gebruiker

te plaatsen. Markers vereenvoudigen het extraheren van gelaatstrekken uit de beeldenstroom immers aanzienlijk. ([36]). Figuur 3.3 toont enkele mogelijke resultaten van deze benadering.



Figuur 3.3: Model-based coding: het gezicht van de gebruiker en het resulterende gezicht van de avatar ([33], [35])

De derde benadering concentreert zich uitsluitend op het animeren van de mond van de avatar. Door de spraak van een gebruiker te analyseren, kan men immers informatie verkrijgen over de beweging van zijn lippen. Deze informatie kan men vervolgens gebruiken om de lippen van de avatar te animeren. Een zeer eenvoudige implementatie van deze methode is bijvoorbeeld de volgende: open de mond van de avatar wanneer de gebruiker spreekt, en sluit de mond weer wanneer de gebruiker stopt met spreken. Dit geeft geen realistisch resultaat, maar op deze manier weten de gebruikers van een NVE wel steeds wie aan het spreken is en wie niet. Uiteraard is een meer gesofistikeerde aanpak waarbij de lippen van de avatar de spraak van de gebruiker correct volgen eveneens mogelijk. Tenslotte kan een NVE, net zoals voor gebaren en houdingen, een verzameling voorgedefinieerde gelaatsuitdrukkingen aanbieden. Een mogelijk nadeel van deze benadering is dat ze gebruikers in staat stelt te liegen over hun huidige gemoedstoestand. Dit kan leiden tot ongewenste situaties waarbij gebruikers de gemoedstoestand van anderen verkeerd inschatten. Het is dan ook niet moeilijk om in te zien dat deze aanpak de interactie tussen gebruikers danig in de war kan sturen.

3.5 Avatars en netwerk trafiek

In de vorige sectie werd de integratie van gebaren en niet-verbale communicatie in een NVE behandeld. Deze integratie is echter enkel mogelijk indien de NVE gebruik maakt van *gelede avatars*. Een gelede avatar bestaat uit verschillende onderdelen (armen, benen, romp, hoofd, enzovoort) die allemaal afzonderlijk kunnen bewegen. De avatars in figuur 3.1(a) bijvoorbeeld kunnen geen gebaren uitvoeren, omdat ze simpelweg niet over armen en handen beschikken. Gelede avatars introduceren echter nieuwe moeilijkheden op het gebied van netwerkcommunicatie. Herinner u uit het vorige hoofstuk dat alle gebruikers op de hoogte gebracht moeten worden wanneer de toestand van een gemeenschappelijk voorwerp in de virtuele omgeving verandert. Aangezien de avatar zo een gemeenschappelijk voorwerp is, moet er een bericht over het netwerk verstuurd worden naar alle gebruikers telkens de toestand van een avatar aangepast wordt. De toestand van een niet-gelede avatar kan enkel wijzigen via een positie- of richtingverandering van de avatar. Voor gelede avatars daarentegen moet er een bericht verstuurd worden telkens de houding van de avatar wijzigt[1]. Dit wil zeggen dat er voor gelede avatars normaliter vaker updateberichten verstuurd moeten worden dan voor niet-gelede avatars.

Bovendien heeft men grotere berichten nodig om de houding van een gelede avatar te beschrijven dan om de toestand van een niet-gelede avatar te beschrijven. Meer concreet kan de houding van een gelede avatar op vier veschillende manieren gerepresenteerd worden ([37]). Ten eerste kan men de translatie en rotatie van alle gewrichten van de avatar expliciet opnemen in elk updatebericht. Wanneer een gebruiker een dergelijk bericht ontvangt, moet hij de oude translaties en rotaties enkel overschrijven met deze in het bericht. Dit wil zeggen dat remote hosts in dit geval niet zelf moeten zorgen voor de animatie van de avatar in kwestie. Deze aanpak resulteert echter ook in grote berichten, daar een realistische avatar al snel een twintigtal gewrichten heeft, en daar er redelijk veel data nodig is om de transformaties van deze gewrichten te representeren. Ten tweede kan men enkel de hoeken die de gewrichten van de avatar aannemen doorsturen. Deze aanpak resulteert in kleinere berichten, maar de zender zal deze hoeken wel steeds moeten berekenen uit de huidige translaties en rotaties van de gewrichten. Bovendien moeten remote hosts de ontvangen hoeken opnieuw omzetten naar transformaties, zodat de avatar getekend kan worden op het scherm. Dit omzetten van en naar translaties en rotaties neemt uiteraard kostbare tijd in beslag. Ten derde kan men ervoor opteren

[1]Met houding wordt hier de positie en oriëntatie van de verschillende onderdelen van de avatar bedoeld.

enkel de transformatie van de belangrijkste gewrichten van de avatar door te sturen. Ontvangers zullen in dit geval zelf de transformatie van de andere gewrichten moeten berekenen met behulp van inverse kinematics (zie het volgende hoofdstuk). Daar een inverse kinematics probleem meestal meerdere oplossingen heeft, kan deze benadering op remote hosts onrealistische houdingen produceren, en is het eveneens mogelijk dat de houding van de avatar op verschillende remote hosts niet hetzelfde is. Ten vierde kan men enkel een high-level beschrijving van de huidige toestand van de avatar doorsturen naar de andere gebruikers. Wanneer een gebruiker een dergelijk updatebericht ontvangt, moet hij zelf zorgen voor de animatie van de avatar in kwestie.

Welke benadering het meest geschikt is, hangt af van de vereisten en eigenschappen van de beoogde applicatie. Indien een nauwkeurige duplicering van animaties op remote hosts belangrijk is, kan men best een beroep doen op de eerste of de tweede benadering. Deze twee benaderingen belasten het netwerk echter ook het meest. Indien men daarentegen een large-scale applicatie met veel gelijktijdige gebruikers voor ogen heeft, kan men best opteren voor de derde of laatste benadering. Deze benaderingen sparen het netwerk, maar ze zijn minder nauwkeurig en ze plaatsen een grotere computationele last op remote hosts dan de eerste twee benaderingen. Verschillende benaderingen kunnen uiteraard ook gecombineerd worden in één enkele applicatie. Zo kan men bijvoorbeeld high-level beschrijvingen gebruiken voor eenvoudige animaties zoals wandelen en springen, en één van de andere benaderingen voor complexere of samengestelde animaties, zoals bijvoorbeeld wandelen en wuiven tegelijk.

3.6 Conclusie

Het is duidelijk dat avatars een cruciale rol spelen in 3D NVEs. Avatars vervullen immers enkele zeer belangrijke functies in de virtuele omgeving, zoals bijvoorbeeld het aangeven van de aanwezigheid van een gebruiker. Bovendien is de avatar voor gebruikers het medium van interactie met de virtuele wereld. Er werd vermeld dat directly controlled avatars gebruikers normaliter meer onderdompelen in de virtuele omgeving dan guided avatars. Toch maken de meeste NVEs gebruik van guided avatars en niet van directly controlled avatars. De reden hiervoor is dat er speciale invoerapparaten nodig zijn om directly controlled avatars te sturen, zoals bijvoorbeeld gloves en sensoren. Guided avatars daarentegen kunnen perfect gecontroleerd worden met behulp van een toetsenbord en een muis.

Via zijn avatar kan een gebruiker eveneens communiceren met andere gebruikers

van de NVE. Er werd in dit hoofdstuk dieper ingegaan op het belang van niet-verbale communicatie en de integratie ervan in NVEs. De huidige NVEs schenken echter zeer weinig aandacht aan niet-verbale communicatie. Sterker nog, het merendeel van de hedendaagse NVEs biedt zelfs geen ondersteuning voor spraak. Dit kan enerzijds verklaard worden door het feit dat spraak het computernetwerk mogelijk zwaar kan belasten. Anderzijds moeten we echter ook vaststellen dat NVE ontwerpers zich vaak te zeer concentreren op het visuele communicatiekanaal, en daardoor de andere menselijke zintuigen een beetje verwaarlozen.

Tenslotte werd het begrip gelede avatar geïntroduceerd, en hun implicaties op de netwerk trafiek besproken. Er werd aangetoond dat gelede avatars het computernetwerk zwaarder belasten dan niet-gelede avatars. Toch maken de meeste NVEs gebruik van gelede avatars, omdat ze, in tegenstelling tot niet-gelede avatars, op een realistische manier geanimeerd kunnen worden. De real-time animatie van gelede avatars vormt het onderwerp van het volgende hoofdstuk.

Hoofdstuk 4

Real-time character animation

4.1 Inleiding

In het vorige hoofdstuk werd aangehaald dat realistische, mensachtige avatars een positieve invloed hebben op het gevoel van presence. Indien dergelijke belichamingen echter op een onrealistische manier bewegen, zal het gevoel van onderdompeling reeds snel verdwijnen. Veronderstel bijvoorbeeld dat een gebruiker van een NVE doorheen de virtuele omgeving beweegt. Daar avatars gemeenschappelijke voorwerpen zijn, worden andere gebruikers via updateberichten op de hoogte gebracht van de nieuwe positie van deze gebruiker. Indien remote hosts de informatie in dergelijke updateberichten enkel gebruiken om de avatar in kwestie te translateren in de virtuele omgeving, geeft dit aanleiding tot onrealistische taferelen. De avatar zal zich immers ineens op een andere plaats bevinden, zonder dat hij eigenlijk een lichaamsdeel bewogen heeft. Indien de avatar daarentegen naar zijn nieuwe positie wandelt of loopt, komt dit veel geloofwaardiger over. Dit eenvoudige voorbeeld toont aan dat avatars ook op een realistische manier geanimeerd moeten worden opdat het gevoel van presence bewaard zou blijven.

Interactiviteit is één van de kenmerkende eigenschappen van NVE systemen. Gebruikers van een NVE moeten immers in *real-time* kunnen interageren met de virtuele omgeving en met andere gebruikers. Dit impliceert dat de avatars van gebruikers eveneens in *real-time* geanimeerd moeten worden in de virtuele wereld, en dat een NVE ontwerper hiervoor dus geen beroep kan doen op offline animatietechnieken. Real-time character animation heeft de laatste jaren een enorme vooruitgang gekend, en men kan momenteel in real-time resultaten produceren die vroeger enkel offline haalbaar waren ([40]). Toch zal een NVE ontwerper steeds een balans moeten trachten te vinden tussen geproduceerd realisme enerzijds en computationele

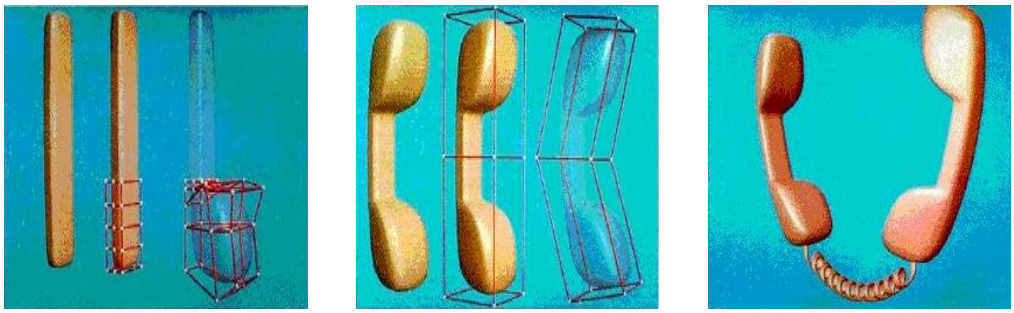
complexiteit en animatiesnelheid anderzijds ([41]). Bij offline animatie daarentegen spelen tijd en computationele complexiteit helemaal geen rol.

Dit is overigens niet de enige afweging die men moet maken bij het real-time animeren van 3D modellen. De computationele complexiteit van een animatietechniek kan immers gereduceerd worden door meer informatie over de animaties op te slaan op de harde schijf ([40]). De processor zal in dit geval minder berekeningen moeten uitvoeren om hetzelfde resultaat te bekomen. Daar de laatste jaren de capaciteit van harde schijven spectaculair is toegenomen, vormen grote files tegenwoordig geen probleem meer. Voordat de processor de inhoud van files kan gebruiken, moet deze data echter naar het geheugen van de computer getransporteerd worden, en geheugengebruik is, in tegenstelling tot file storage, wel een kostbare resource. Dit wil zeggen dat men bij real-time character animation eveneens een evenwicht moet trachten te vinden tussen geheugengebruik en processorbelasting.

Het gezicht van een mensachtige avatar heeft specifieke animatievereisten, en wordt daarom vaak onafhankelijk van de rest van het lichaam geanimeerd ([41]). Zoals duidelijk zal worden in sectie 4.2, wordt het lichaam van een avatar meestal geanimeerd door de stand van zijn gewrichten aan te passen. Het gezicht heeft echter geen gewrichten, en kan dus niet op een dergelijke manier geanimeerd worden. Daarom wordt er meestal een deformation model gebruikt om het gezicht te animeren. Een voorbeeld van zo een deformation model is Free-Form Deformation (FFD, [42]). In de FFD benadering wordt het object dat vervormd moet worden in een eenvoudige controlestructuur zoals bijvoorbeeld een kubus geplaatst. Wanneer deze controlestructuur vervormd wordt, wordt het ingesloten object op een natuurlijke wijze mee vervormd. Figuur 4.1 toont hoe de FFD benadering gebruikt kan worden om, vertrekkend van een eenvoudige balk, een telefoonhoorn te modelleren. Door nu controlestructeren rond bepaalde gebieden van het gezicht (de ogen, de mond, de wenkbrauwen, enzovoort) te plaatsen en deze controlestructuren te vervormen, kan men het gezicht animeren. Ik concentreer mij in de rest van dit hoofdstuk op de animatie van het lichaam (zonder het hoofd).

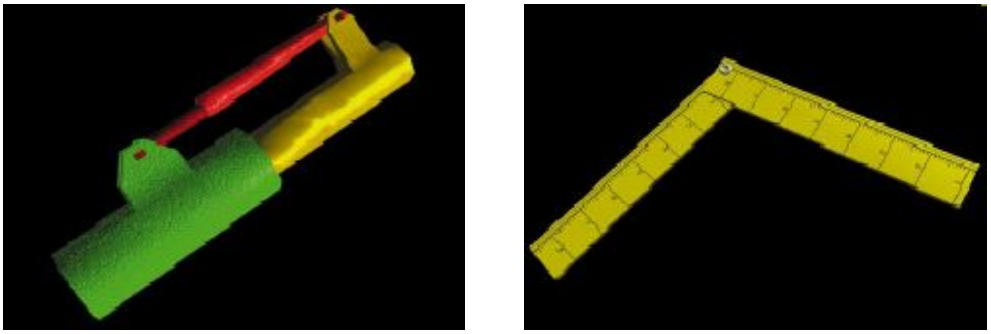
4.2 Layered modeling en skeletal animation

Een mensachtige figuur kan op verschillende manieren geanimeerd worden. De benadering die momenteel veruit het meest gebruikt wordt, is skeletal animation. Skeletal animation is gebaseerd op een modelleertechniek die layered modeling heet. In layered modeling wordt de figuur die geanimeerd moet worden opgedeeld in een aantal lagen, waarbij elke laag zijn eigen geometrische eigenschappen heeft ([43]). De on-



Figuur 4.1: Modelleren van een telefoonhoorn met de FFD benadering ([42])

derste laag van een gelaagd model bestaat steeds uit een aantal eenvoudige objecten (botten) die verbonden zijn door gewrichten. De botten en gewrichten vormen samen een hiërarchisch skelet, en deze laag wordt dan ook, niet verwonderlijk, de skelet laag genoemd. De gewrichten kunnen de botten van het skelet roteren en eventueel ook verplaatsen. Meer concreet bepaalt elk gewricht het aantal vrijheidsgraden (degrees of freedom, DOFs) van het volgende bot in de hiërarchie ([44]). Figuur 4.2(a) toont een gewricht dat de positie van het volgende object in de hiërarchie (de gele staaf) slechts in één dimensie kan aanpassen. Dit wil zeggen dat dit gewricht overeenkomt met één translational degree of freedom. Figuur 4.2(b) daarentegen toont een gewricht dat het volgende object in de hiërarchie rond juist één as kan roteren. Een dergelijk gewricht komt overeen met één rotational degree of freedom.

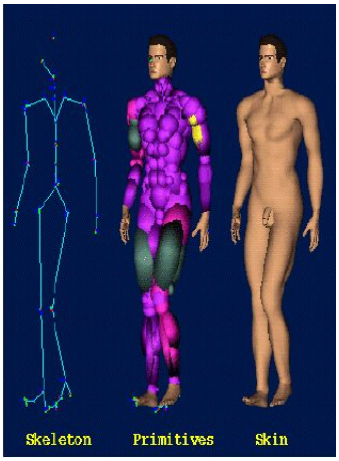


(a) 1 *translational* degree of freedom (b) 1 *rotational* degree of freedom

Figuur 4.2: Elk gewricht in de skelet laag komt overeen met een aantal degrees of freedom ([44])

Bovenop de skelet laag kunnen vervolgens andere lagen geplaatst worden. Figuur 4.3 toont een model van een mannelijke figuur dat bestaat uit drie lagen. Het

model bevat, naast een skelet laag, nog een laag die spierweefsel voorstelt en een laag die de huid van de figuur voorstelt. Belangrijk hierbij is dat de verschillende lagen van een gelaagd model niet onafhankelijk zijn van elkaar. Er kunnen immers relaties tussen de verschillende lagen gespecificeerd worden. Deze relaties zorgen ervoor dat, wanneer een laag beweegt, de bovenliggende lagen zullen vervormen en mee bewegen. Voor het model in figuur 4.3 wil dit bijvoorbeeld zeggen dat de geometrische primitieven die het spierweefsel voorstellen verbonden zijn met het onderliggende skelet, en dus mee zullen bewegen wanneer het skelet geanimeerd wordt. Bovendien zal dit spierweefsel vervolgens invloed uitoefenen op de huid laag, zodat de huid eveneens zal vervormen en mee bewegen.



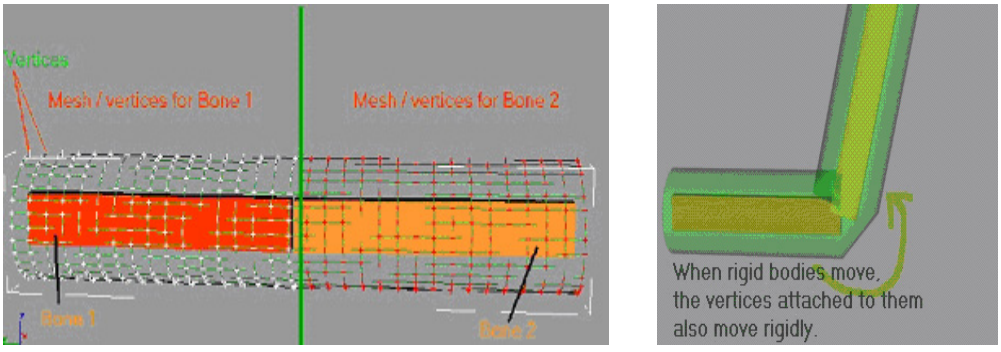
Figuur 4.3: Voorbeeld van een model dat uit drie lagen bestaat ([43])

Het zou ondertussen reeds duidelijk moeten zijn wat juist de relatie is tussen layered modeling en skeletal animation. Layered modeling stelt een animator in staat om de beweging van een complex model te controleren via een vrij eenvoudige structuur, namelijk een skelet. Dankzij de relaties die gelden tussen de verschillende lagen, moet immers enkel de skelet laag geanimeerd worden om het volledige model te doen bewegen. Skeletal animation vergemakkelijkt het werk van een animator aanzienlijk. Het animeren van een beperkt aantal gewrichten is immers veel eenvoudiger dan alle vertices uit de huid laag (vaak meer dan honderd) rechtstreeks te moeten animeren. Merk ook op dat skeletal animation files relatief klein zullen zijn, daar alle objecten uit de verschillende lagen (gewrichten, huidvertices, geometrische primitieven, enzovoort) slechts één keer opgeslagen moeten worden. De animatie zelf kan opgeslagen worden als een opeenvolging van transformaties voor de verschillende gewrichten van het model.

Er is in principe geen beperking op het aantal lagen waaruit een gelaagd model kan bestaan. Voor real-time applicaties wordt er echter meestal een beroep gedaan op modellen met slechts twee lagen, namelijk een skelet laag en een huid laag. Het aantal berekeningen dat uitgevoerd moet worden tijdens de animatie van een model stijgt immers met het aantal lagen waaruit het model bestaat. Het is dan ook niet moeilijk om in te zien dat complexe modellen reeds snel voor performance problemen kunnen zorgen in NVEs met veel gelijktijdige gebruikers. Modellen met enkel een skelet en een huid laag kunnen opgedeeld worden in twee categorieën: *rigid skinned characters* en *soft skinned characters*.

4.2.1 Rigid skinned characters

Bij een rigid skinned character is elke vertex uit de huid laag verbonden met juist één bot van het onderliggende skelet ([40]). Dit wil zeggen dat wanneer een bepaald bot beweegt, alle vertices die ermee geassocieerd zijn op een identieke manier mee zullen bewegen. Het grote nadeel van deze benadering is dat de huid in de buurt van gewrichten op een onrealistische manier kan vervormen wanneer het model geanimeerd wordt. Dit wordt geïllustreerd in figuur 4.4. Figuur 4.4(a) toont twee botten en een vertex mesh die de huid voorstelt. Alle vertices links van de groene streep zijn enkel verbonden met het linkse bot, terwijl alle vertices rechts van de streep enkel verbonden zijn met het rechtse bot. Wanneer de twee botten geherpositioneerd worden zodat ze een hoek van ongeveer 90 graden vormen, krijgen we de situatie uit figuur 4.4(b). Let vooral op de scherpe hoek die de huid vormt op de plaats waar de twee botten samenkomen (aangegeven door de groene pijl).



Figuur 4.4: De huid van een rigid skinned character kan mogelijk op een visueel onrealistische manier vervormen in de buurt van gewrichten ([46])

4.2.2 Soft skinned characters

De onrealistische vervormingen die bij rigid skinned characters kunnen optreden, zijn een rechtstreeks gevolg van het feit dat elke huidvertex bij deze benadering steeds met juist één bot verbonden is. Soft skinned characters proberen hier een mouw aan te passen door toe te laten dat huidvertices tegelijkertijd door meerdere botten uit het onderliggende skelet beïnvloed kunnen worden ([40]). Voor elke vertex wordt er bijgehouden welke botten met deze vertex geassocieerd zijn, en hoe groot de invloed van elk bot is (het zogenaamde skin weight). Indien de som van alle skin weights voor een bepaalde huidvertex gelijk is aan 1, kan de uiteindelijke positie van deze vertex als volgt berekend worden:

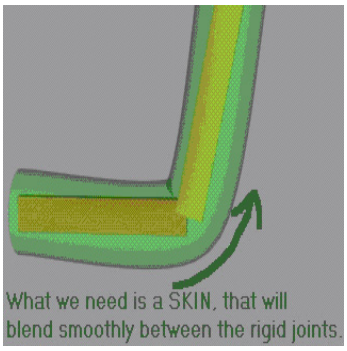
Listing 4.1:

```
Vector3D final_position(0,0,0); /* initialize at (0,0,0) */
int i = 0;
Voor elk bot b dat met vertex v geassocieerd is doe
{
        Vector3D temp = concateneer b.transform met v.position;
        final_position += temp * v.weight[i];
        i++;
}
```

Figuur 4.5 toont het resultaat van de soft skinning benadering, toegepast op de situatie uit figuur 4.4(b). Merk op hoe glad de huid is op de plaats waar de twee botten samenkomen. Het is overduidelijk dat deze benadering in staat is om meer realistische vervormingen te produceren dan rigid skinning. Hier is echter ook een prijs aan verbonden. Soft skinned characters gebruiken namelijk meer geheugen dan rigid skinned characters. Er moet immers voor elke vertex een lijst van botten bijgehouden worden in plaats van slechts één bot, en ook de skin weights moeten opgeslagen worden. Bovendien maakt listing 4.1 duidelijk dat er er meer berekeningen nodig zijn om soft skinned characters te animeren dan om rigid skinned characters te animeren.

4.3 Alternatieven voor skeletal animation

Dankzij enorme vooruitgang in computer hardware is skeletal animation momenteel bruikbaar voor real-time toepassingen. Een vijftal jaar geleden echter beschikten computers voor thuisgebruik nog niet over voldoende grafische kracht, en was het gebruik van skeletal animation nog beperkt tot offline animatie. Dit wil zeggen dat



Figuur 4.5: Soft skinning kan zeer gladde en realistische vervormingen produceren in de buurt van gewrichten ([46])

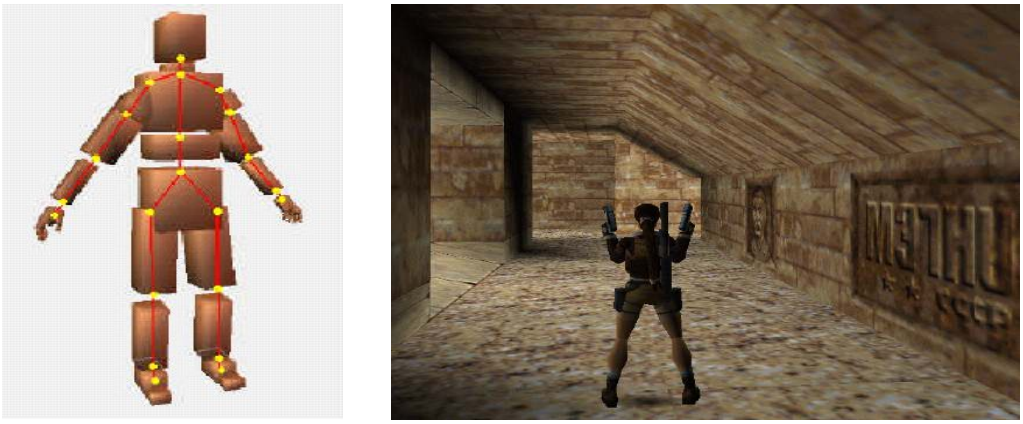
men vroeger een beroep moest doen op andere benaderingen om figuren in real-time te animeren. In deze sectie worden twee van deze oudere benaderingen besproken. Het zal echter snel duidelijk worden dat skeletal animation op alle vlakken superieur is aan deze twee alternatieven. Meer zelfs, skeletal animation combineert de voordelen van beide benaderingen, zonder de nadelen over te nemen.

4.3.1 Hiërarchie van aparte objecten

Bij deze benadering worden de verschillende lichaamsdelen van het 3D model (zoals bijvoorbeeld de bovenarm, de onderarm, de hand, enzovoort) voorgesteld door aparte objecten. Deze aparte objecten worden verbonden met elkaar, en vormen aldus een hiërarchie ([47]). Dit wordt geïllustreerd in figuur 4.6(a). Merk op dat deze benadering, in tegenstelling tot wat de rode lijnen en gele punten in deze figuur suggeren, *geen* gebruik maakt van een onderliggend skelet. Dergelijke hiërarchische modellen kunnen geanimeerd worden door de verschillende objecten uit de hiërarchie te roteren en te translateren. Deze benadering gebruikt weinig geheugen, daar alle vertices slechts één keer opgeslagen moeten worden. De animatie zelf kan opgeslagen worden als een sequentie van transformaties voor de verschillende onderdelen van het model. Bovendien kunnen er voor hiërarchische modellen, net zoals bij skeletal animation overigens, animaties on the fly gegenereerd worden via bijvoorbeeld inverse kinematics.

Het grote nadeel van deze benadering is dat er gaten tussen de verschillende lichaamsdelen kunnen ontstaan wanneer het model geanimeerd wordt. Hoewel dit tot op een zeker niveau verdoezeld kan worden door de verschillende lichaamsdelen te laten overlappen, zullen dergelijke hiërarchische modellen toch steeds een ietwat

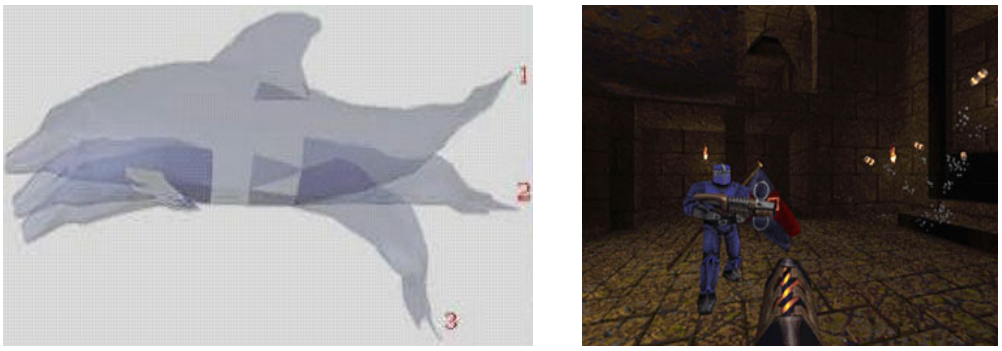
onrealistische indruk geven. Deze animatietechniek werd onder andere gebruikt in het computerspel Tomb Raider van Core Design (zie figuur 4.6(b)).



Figuur 4.6: Hiërarchische modellen: (a) Schematisch ([40]); (b) Lara Croft

4.3.2 Single mesh characters

Single mesh characters bestaan, in tegenstelling tot hiërarchische modellen, niet uit verschillende onderdelen, maar uit één polygonmesh. Dergelijke modellen kunnen geanimeerd worden door rechtstreeks de positie van de vertices in de mesh aan te passen ([47]). De animatie wordt opgeslagen als een sequentie van meshes, waarbij elke mesh het model in een andere houding toont (zie figuur 4.7(a)). Om de animatie af te spelen, moet men enkel de corresponderende vertexposities van elke twee opeenvolgende meshes interpoleren, of kan men zelfs gewoon alternerend de opeenvolgende meshes tonen. Het is duidelijk dat deze benadering computationeel zeer goedkoop is, maar ook ontzettend veel geheugen verbruikt. De vertices uit de mesh moeten immers meerdere keren opgeslagen worden, één keer voor elke voorgedefinieerde houding in de animatie. Bovendien is het onmogelijk om voor single mesh characters animaties on the fly te genereren, daar alle animaties voorgedefinieerd moeten zijn. Een pluspunt ten opzichte van hiërarchische modellen is wel dat single mesh characters nooit gaten zullen vertonen wanneer ze geanimeerd worden. Het computerspel Quake van id Software (zie figuur 4.7(b)) maakt onder andere gebruik van single mesh characters.



Figuur 4.7: Single mesh characters: (a) Voorbeeld uit de DirectX 8.0 SDK ([40]); (b) Quake

4.4 Overzicht van de verschillende animatietechnieken

Het grote voordeel van de layered modeling benadering is dat enkel het skelet van een model geanimeerd moet worden om het volledige model te doen bewegen, en dit dankzij de relaties die gelden tussen de verschillende lagen. In deze sectie worden er een aantal basistechnieken besproken die gebruikt kunnen worden om de skelet laag van een gelaagd model te animeren[1] ([43], [48]). De eerste basistechniek heet *motion capture*, en wordt voornamelijk gebruikt in computergames en in de filmindustrie. Bij motion capturing worden de bewegingen van een echte acteur vastgelegd via sensoren en opgeslagen in een bepaald file formaat. De informatie in deze files wordt vervolgens gebruikt om computermodellen dezelfde bewegingen te laten uitvoeren. Motion capturing kan uiterst realistische en visueel nauwkeurige animaties produceren, en is vooral geschikt voor applicaties waarbij modellen zeer specifieke animaties moeten kunnen afspelen, zoals bijvoorbeeld een trapbeweging in een voetbalspel. Het is erg moeilijk om dergelijke bewegingen op een realistische manier na te bootsen met één van de andere basistechnieken.

Bij de tweede basistechniek, *keyframing* genaamd, wordt de animatie opgeslagen als een opeenvolging van zogenaamde keyframes. Elk keyframe specificeert de houding van het model op een bepaald tijdstip in de animatie. Via een interpolatieschema worden vervolgens de frames berekend die tussen elke twee opeenvolgende keyframes in liggen. Eens deze tussenliggende frames bepaald zijn, kan het model

[1]Deze technieken zijn overigens niet enkel bruikbaar voor skeletal animation, maar kunnen ook op andere animatiebenaderingen toegepast worden.

geanimeerd worden door de verschillende frames voldoende snel na elkaar te tonen. Keyframing is een eenvoudige en krachtige techniek die een animator veel tijd kan besparen. De animator moet immers niet alle frames van de animatie expliciet zelf construeren, maar kan zich volledig concentreren op een beperkt aantal keyframes. De kwaliteit van de resulterende animatie hangt echter af van de bekwaamheid van de animator.

Ten derde zijn er de zogenaamde *kinematic* technieken die beweging specificeren onafhankelijk van de krachten die de beweging veroorzaakten. Kinematic technieken kunnen ingedeeld worden in twee categorieën, namelijk *forward kinematics* en *inverse kinematics*. Bij forward kinematics moet de animator expliciet de positie en oriëntatie van objecten specificeren op bepaalde tijdstippen in de animatie. Om bijvoorbeeld bij forward kinematics de hand van een mensachtig model te positioneren, zal de animator eerst het schoudergewricht moeten roteren, vervolgens de ellenboog en als laatste pas de pols. Het adjectief "forward" slaat dus op het feit dat een gewrichtenketting steeds van het uiteinde dat verbonden is met het lichaam (bijvoorbeeld de schouder) naar het losse uiteinde (een zogenaamde *end-effector* zoals bijvoorbeeld de hand) doorlopen moet worden om het losse uiteinde te positioneren. Het is duidelijk dat deze techniek aardig wat werk van de animator vereist. Inverse kinematics probeert hier een oplossing voor te bieden. Bij inverse kinematics kan de animator de end-effector immers rechtstreeks positioneren, en zal het animatiesysteem zelf berekenen hoe de tussenliggende gewrichten uit de gewrichtenketting geroteerd moeten worden opdat de end-effector op de gewenste positie terechtkomt. Het is duidelijk dat inverse kinematics het animeren van de ledematen van een model aanzienlijk vereenvoudigt ten opzichte van forward kinematics.

Zowel forward als inverse kinematics vereisen veel werk van de animator indien realistische en fysisch correcte animaties gewenst zijn. De reden hiervoor is dat deze technieken geen rekening houden met de krachten die de beweging veroorzaakten. De laatste basistechniek, *dynamics* genaamd, doet dit wel. Dynamic technieken zijn immers gebaseerd op de tweede wet van Newton, F = ma, die de versnelling a van een object met massa m relateert aan de kracht F die op het object uitgeoefend werd. Dit wil zeggen dat er bij een dynamic techniek krachten uitgeoefend moeten worden op de verschillende onderdelen van het model om het model te doen bewegen. Gegeven de uitgeoefende kracht F en de massa m kan immers de nieuwe versnelling van het onderdeel berekend worden. Door deze versnelling één respectievelijk twee keer te integreren, bekomt men de nieuwe snelheid en positie van het onderdeel in kwestie. Dynamic technieken kunnen, net zoals kinematic technieken, opgedeeld worden in twee categorieën, namelijk *forward dynamics* en *inverse dy-*

namics. Bij forward dynamics moet de animator expliciet zelf krachten uitoefenen op de verschillende onderdelen van het model om het model te animeren. Bij inverse dynamics daarentegen is het het animatiesysteem dat krachten uitoefent op het model opdat een bepaald gespecificeerd doel bereikt wordt. Dit doel kan zowel een volledige beschrijving van de gewenste beweging zijn, als een meer high-level constraint of opgave.

Dynamic technieken produceren animaties die fysisch correct zijn. Dergelijke animaties komen zeer realistisch over, en zijn zeer moeilijk na te bootsen met een kinematic techniek ([48]). Dit realisme komt echter ook met een prijs. Dynamic technieken zijn immers computationeel veel zwaarder dan kinematic technieken. Normaal gezien zal er met elke degree of freedom in het model namelijk een bewegingsvergelijking geassocieerd zijn. Dit wil zeggen dat er voor complexe modellen grote verzamelingen vergelijkingen opgelost moeten worden wanneer het model geanimeerd wordt. Bovendien moeten de relaties die gelden tussen de verschillende onderdelen van een geleed model ook op de één of andere manier opgenomen worden in de bewegingsvergelijkingen. Dit zorgt voor een koppeling van de bewegingsvergelijkingen, en maakt de controle over het model bij forward dynamics uitermate moeilijk ([48]). Het uitoefenen van een kracht op een bepaald onderdeel van een geleed model zal ten gevolge van deze koppeling immers niet alleen de beweging van dit onderdeel beïnvloeden, maar ook de beweging van naburige onderdelen.

4.5 Real-time inverse kinematics

Het concept inverse kinematics werd reeds in de vorige sectie geïntroduceerd, maar ik ga er hier dieper op in. Herinner u dat bij een *forward* kinematics probleem de transformaties van alle gewrichten in een gewrichtenketting gegeven zijn, en dat de resulterende positie en oriëntatie van de end-effector die bij deze ketting hoort berekend moet worden. In formulevorm kan dit geschreven worden als

$$x = f(q) \tag{4.1}$$

waarbij x de positie en oriëntatie van de end-effector voorstelt, en q de vector van gewrichttransformaties[2]. Bij een *inverse* kinematics probleem daarentegen is de positie en oriëntatie van de end-effector gekend, en zijn we op zoek naar de transformaties

[2]Zoals zal duidelijk worden in hoofdstuk 6, komt de functie f bij skeletal animation overeen met een eenvoudige matrix concatenatie.

van de tussenliggende gewrichten. Dit kan als volgt in formulevorm geschreven worden:

$$q = f^{-1}(x) \tag{4.2}$$

Er bestaan twee soorten oplossingen voor het probleem uit formule 4.2, namelijk *closed form oplossingen* en *numerieke oplossingen* ([44]). Closed form oplossingen worden analytisch bepaald via niet-iteratieve berekeningen. Het grote voordeel van closed form oplossingen is dat ze snel berekend kunnen worden, en dat ze zeer gemakkelijk omgezet kunnen worden naar programmeercode. Spijtig genoeg bestaat er geen algemene analytische oplossing voor willekeurige gewrichtenkettingen ([48]). Dit wil zeggen dat men voor elk inverse kinematics systeem een specifieke oplossing zal moeten berekenen, en dat de complexiteit van de oplossing zal toenemen met het aantal degrees of freedom in het systeem. Jeff Lander geeft in [44] en [45] zowel een algebraïsch als een geometrische closed form oplossing voor een eenvoudig inverse kinematics systeem bestaande uit twee gewrichten die elke corresponderen met één degree of freedom. Om tot deze oplossingen te komen, heeft hij gebruik gemaakt van verschillende basisformules uit de trigonometrie en de meetkunde. De complexiteit van de oplossingen voor dit zeer eenvoudig inverse kinematics probleem maakt echter meteen duidelijk dat deze benadering niet geschikt is voor ingewikkelde systemen met veel degrees of freedom.

Om meer complexe inverse kinematics problemen op te lossen, moet men een beroep doen op een numerieke methode. Numerieke methoden gebruiken iteratieve berekeningen en passen de stand van de gewrichten in elke iteratie een beetje in de goede richting aan. Dit geeft meteen een belangrijk verschilpunt aan tussen de closed form en de numerieke benadering: indien de gewenste positie van de end-effector niet bereikbaar is, zal een closed form benadering geen oplossing opleveren; een numerieke benadering daarentegen zal de gewenste positie zo dicht mogelijk trachten te benaderen. Iteratief convergeren naar de gewenste end-effector toestand heeft echter ook een nadeel. De gewrichtenketting zal bij een numerieke methode namelijk meerdere keren doorlopen moeten worden, terwijl dit bij een closed form benadering slechts één keer moet gebeuren. Closed form methoden zijn dus over het algemeen sneller dan numerieke benaderingen.

De meeste numerieke methoden zijn ofwel gebaseerd op matrix inversie, ofwel op optimaliseringstechnieken ([48]). Hierbij wordt de tweede mogelijkheid meestal geprefereerd, daar het inverteren van een matrix een computationeel complexe opdracht is. Bovendien kunnen er bij de matrix inversie benadering problemen ontstaan wanneer de matrix in kwestie niet inverteerbaar blijkt te zijn. Numerieke methoden gebaseerd op optimalisering vermijden om deze reden de matrix inversie

stap volledig, en trachten het inverse kinematics probleem op te lossen door een bepaalde error functie te minimaliseren via iteratieve optimaliseringstechnieken. Een goed voorbeeld van een mogelijke error functie is de afstand tussen de huidige en de gewenste positie van de end-effector. Het is duidelijk dat deze error functie beïnvloed wordt door de stand van de gewrichten, en dat het minimaliseren van deze functie het inverse kinematics probleem oplost.

4.6 Conclusie

Dankzij de opkomst van goedkope grafische beeldkaarten heeft real-time character animation de afgelopen jaren een enorme vooruitgang gekend. Enkele jaren geleden moest een NVE ontwerper nog een beroep te doen op eenvoudige methoden zoals bijvoorbeeld single mesh characters om avatars in de virtuele omgeving te animeren. Momenteel zijn computers voor thuisgebruik echter reeds in staat om in real-time gelaagde modellen via skeletal animation te animeren. Skeletal animation is een krachtige animatiemethode die zeer realistische animaties kan produceren. Bovendien zijn skeletal animation files relatief klein, en kunnen er voor gelaagde modellen nieuwe animaties on the fly gegenereerd worden via inverse kinematics. Het is dan ook niet verwonderlijk dat bijna alle recente First Person computergames (zoals bijvoorbeeld Half-Life van Valve en Unreal Tournament van Epic Megagames) gebruik maken van skeletal animation om zowel de tegenstanders als het hoofdpersonage te animeren.

In dit hoofdstuk werden de verschillende manieren besproken waarop het skelet van een gelaagd model geanimeerd kan worden. Hierbij moet opgemerkt worden dat de meeste hedendaagse animatiepakketten zich niet beperken tot één bepaalde animatietechniek, maar meerdere technieken aanbieden. Zo stellen veel hedendaagse modelers gebruikers bijvoorbeeld in staat via forward kinematics te interageren met het skelet van een gelaagd model, waarbij de geproduceerde animatie vervolgens kan opgeslagen worden als een sequentie van keyframes. Pure dynamic technieken worden momenteel nog niet veel gebruikt, omdat ze computationeel nog te zwaar zijn.

Hoofdstuk 5

Voorbeelden van bestaande NVEs

5.1 Inleiding

Na het bespreken van zowel NVE technologie in het algemeen als avatars en hun animatie, is het nu tijd om enkele bestaande NVE systemen in detail te beschouwen. Sinds hun ontstaan een twintigtal jaar geleden zijn er reeds ontzettend veel NVEs ontwikkeld. Ik probeer in dit hoofdstuk een representatieve staal van bestaande NVEs aan te bieden, waarbij zowel pioniersystemen als moderne NVEs besproken worden.

5.2 Spline en Diamond Park

Het Spline (Scalable Platform for Large Interactive Network Environments) platform, ontwikkeld in het Mitsubishi Electric Research Laboratory (MERL), voorziet een eenvoudige architectuur waarmee multi-user interactieve virtuele omgevingen geïmplementeerd kunnen worden ([20]). Deze architectuur is gebaseerd op een gedeeld *world model*, een object-georiënteerde repository die informatie over alle voorwerpen in de virtuele omgeving bevat. Spline applicaties kunnen interageren met elkaar en de virtuele omgeving door dit world model aan te passen ([50]). Opdat dit op een vlotte manier zou kunnen verlopen, distribueert Spline het world model over de verschillende applicaties. Elke applicatie houdt lokaal met andere woorden steeds een min of meer consistente kopie bij van een gedeelte van het world model.

De virtuele omgeving wordt in Spline opgedeeld in een aantal locales. Elke locale heeft zijn eigen verzameling multicast adressen die gebruikt worden om berichten te versturen en te ontvangen over de locale en de objecten in deze locale. Een gebruiker

is zich steeds enkel bewust van de locale waar hij zich momenteel in bevindt, en de onmiddellijke buurlocales van deze locale (zie figuur 2.5(a) in sectie 2.3.3). Dit wil zeggen dat een gebruiker enkel over deze locales updateberichten kan ontvangen. Het is duidelijk dat locales de hoeveelheid informatie die gebruikers moeten verwerken drastisch kunnen reduceren, maar ze introduceren echter ook een probleem. Hoe kan een gebruiker immers informatie verkrijgen over een voorwerp zonder dat hij weet in welke locale dit voorwerp zich bevindt? Om dit probleem op te lossen, werden er zogenaamnde *beacon* objecten in Spline opgenomen ([20]). Indien een gebruiker de naam van een beacon object kent, kan hij steeds informatie over dit object opvragen, ook al bevindt het object zich niet in één van de locales waarvan de gebruiker zich momenteel bewust is.

Om de delay te minimaliseren en om bottlenecks te voorkomen, maakt Spline gebruik van een peer-to-peer netwerktopologie ([49]). Wanneer een gebruiker zijn lokale kopie van het world model aanpast, verstuurt hij een multicast bericht naar de andere gebruikers zodat deze hun lokale kopie eveneens kunnen bijwerken. Maar het Spline platform bevat ook enkele special purpose servers. Zo is er onder andere een *session manager* server die binnenkomende connectie aanvragen afhandelt, en een beacon server die informatie over de beacon objecten in de virtuele omgeving bijhoudt.

Diamond Park is een large-scale NVE die geïmplementeerd werd met behulp van het Spline platform. Het park bestaat uit een vierkante mijl virtueel landschap en bevat een vijftiental gebouwen ([20], zie figuur 5.1(a)). Gebruikers van Diamond Park kunnen hun avatar op twee manieren controleren, namelijk via een computer gecontroleerde stationaire fiets of via een standaard toetsenbord en muis, en worden respectievelijk voorgesteld door een fietser (zie figuur 5.1(c)) of een éénwieler (zie figuur 5.1(d)). Bovendien bevat Diamond Park een aantal autonome avatars, zoals bijvoorbeeld de scheidsrechter in figuur 5.1(e). Dit wil zeggen dat elk van de drie soorten avatars die besproken werden in sectie 3.3 (directly controlled avatars, guided avatars en autonomous avatars) aanwezig zijn in Diamond Park. Alle avatars zijn geleed en kunnen voornamelijk eenvoudige bewegingen uitvoeren zoals bijvoorbeeld fietsen.

5.3 SIMNET, DIS en NPSNET

Het Simulator Networking (SIMNET) project werd ontwikkeld door het Defense Advanced Research Projects Agency (DARPA), de centrale onderzoeksorganisatie van het Amerikaanse Department of Defense (DoD), en kan beschouwd worden als

Figuur 5.1: Enkele foto's van Diamond Park, een NVE geïmplementeerd met behulp van het Spline platform

de eerste succesvolle implementatie van een large-scale 3D NVE ([4]). SIMNET werd gebruikt voor militaire training en het houden van militaire simulaties. Het meest opmerkelijke aan het SIMNET systeem is waarschijnlijk dat het gebruik maakt van dead-reckoning. Elke gebruiker houdt lokaal informatie bij over *alle* objecten in de virtuele omgeving[1], en broadcast met regelmatige tussenpozen de toestand van de objecten die in zijn bezit zijn naar *alle* andere gebruikers van het systeem. Wanneer een gebruiker een dergelijk bericht ontvangt, gebruikt hij de informatie in dit bericht om het gedrag van het "geupdate" object lokaal te simuleren.

Het Amerikaanse DoD beschouwde SIMNET als een enorm succes, en besloot een standaard voor gedistribueerde simulaties te ontwikkelen ([4]). Deze standaard kreeg de naam DIS mee, wat staat voor Distributed Interactive Simulation. De DIS standaard maakt eveneens gebruik van dead-reckoning, en specificeert een aantal zogenaamde *Protocol Data Units* (PDUs) die gebruikt kunnen worden om events (zoals bijvoorbeeld een ontploffing, een positieverandering van een object, enzovoort) mee te delen aan de deelnemers van de simulatie. Toen DIS ontwikkeld werd had men voornamelijk militaire toepassingen voor ogen, maar de standaard kan eveneens ge-

[1]Er is dus geen centrale repository.

bruikt worden om niet-militaire simulaties te implementeren. Hoewel DIS een vrij robuuste standaard is, zijn er toch enkele problemen geassocieerd met het protocol op het gebied van schaalbaarheid ([51]). Zo zal een gebruiker van een DIS simulatie bijvoorbeeld steeds updateberichten over *alle* objecten in de virtuele omgeving ontvangen en moeten verwerken.

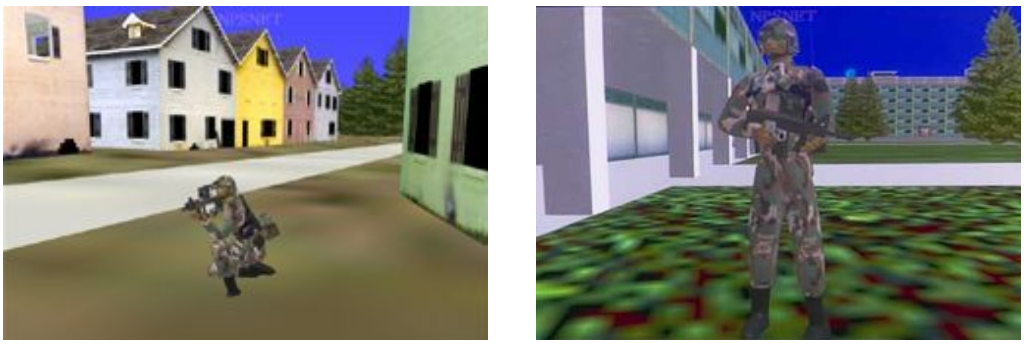
NPSNET (Naval Postgraduate School Networked Vehicle Simulator), ontwikkeld door de NPSNET onderzoeksgroep van de Naval Postgraduate School in California, was de eerste NVE die geïmplementeerd werd met behulp van de DIS standaard. Om de schaalbaarheidsproblemen die met het DIS protocol geassocieerd zijn te voorkomen, hebben de ontwikkelaars van NPSNET zelf een awareness management model geïmplementeerd ([51]). In NPSNET wordt de virtuele wereld opgedeeld in hexagonale cellen met identieke grootte, en heeft elke gebruiker een zogenaamde *area of interest*[2]. Er is een uniek multicast adres geassocieerd met elke cel, en gebruikers connecteren steeds enkel met de multicast adressen van de cellen die in hun area of interest liggen, en kunnen dus enkel over deze cellen informatie ontvangen via het onderliggende computernetwerk. Gebruikers houden ook enkel over de objecten in deze cellen lokaal informatie bij. De spatiale opdeling van de virtuele omgeving in cellen is vergelijkbaar met de locales benadering van het Spline platform, maar is minder flexibel. De locales van Spline kunnen immers een willekeurige vorm hebben, en moeten niet noodzakelijk allemaal even groot zijn.

Naast deze spatiale opdeling wordt de inhoud van de virtuele omgeving eveneens opgedeeld in functionele klassen die ook allen hun eigen multicast adres hebben. Deze klassen groeperen objecten volgens hun functie. Voorbeelden van functionele klassen zijn een radio communications klasse en een air control klasse. Gebruikers van de NVE kunnen connecteren met de multicast adressen van dergelijke functionele klassen, en zullen op deze manier informatie ontvangen over de objecten die tot de klasse behoren, zelfs wanneer deze objecten zich niet in hun huidige area of interest bevinden. Zo zal bijvoorbeeld een virtuele vliegtuigpiloot die via een radio kan communiceren met de vluchtleiding zowel met de radio communications als de air control functionele klasse moeten connecteren. Tenslotte voorziet NPSNET ook zogenaamde timing klassen. Alle objecten die tot eenzelfde timing klasse behoren, versturen steeds met identieke tussenpozen updateberichten naar het multicast adres van deze klasse, zelfs wanneer de toestand van het object sneller verandert dan deze opgelegde updaterate. Sommige entiteiten hebben immers geen real-time updates nodig van de objecten waarin ze geïnteresseerd zijn. Zo moet een satelliet

[2]Deze area of interest komt meestal overeen een cirkel die gecentreerd is rond de huidige positie van de gebruiker.

die rond een planeet draait bijvoorbeeld enkel bij benadering weten wat er op het planeetoppervlak gebeurt. Dergelijke entiteiten kunnen in NPSNET met een timing klasse connecteren die aan hun noden voldoet.

NPSNET was de eerste NVE die gebruikers representeerde als virtual humans in de virtuele omgeving ([39]). Deze virtual humans zijn geleed, en kunnen verschillende animaties uitvoeren (zie figuur 5.2). Gebruikers kunnen hun belichaming controleren via een toetsenbord en een muis, of op een immersieve manier via body tracking sensoren. Bovendien werden er in NPSNET-V, de meest recente versie van het NPSNET project, autonome avatars geïntroduceerd. Dit wil zeggen dat elk van de drie soorten avatars uit sectie 3.3 aanwezig zijn in NPSNET-V.



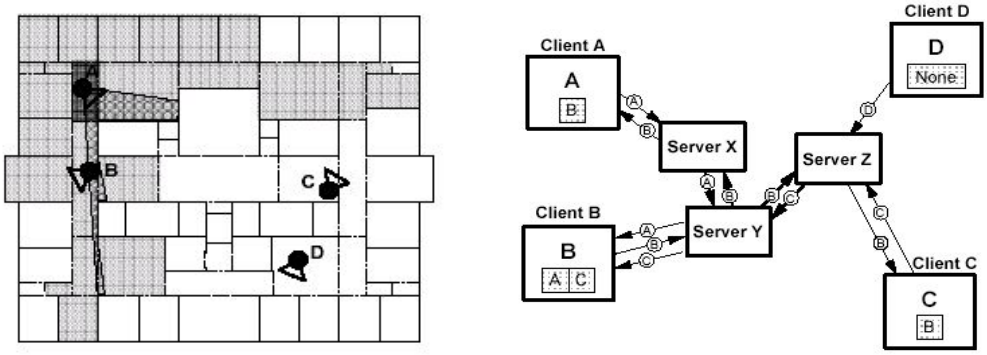
Figuur 5.2: Het NPSNET systeem wordt gebruikt voor het houden van militaire simulaties ([51])

5.4 RING

Het RING (Real-time Interactive Networked Graphics) systeem, ontwikkeld door Thomas Funkhouser van AT&T Bell Laboratories, ondersteunt interactie tussen een groot aantal gebruikers in virtuele omgevingen met veel visuele barrières, zoals bijvoorbeeld gebouwen en steden ([52]). RING maakt gebruik van een zuivere unicast client-server netwerktopologie met meerdere servers. De servers bevatten op elk moment de meest recente informatie over alle gemeenschappelijke voorwerpen in de virtuele omgeving. De clients daarentegen houden steeds enkel informatie bij over de objecten die de lokale gebruiker daadwerkelijk kan waarnemen. Wanneer een gebruiker de toestand van een gemeenschappelijk voorwerp aanpast, verstuurt hij een unicast updatebericht naar de server die met deze client geassocieerd is. Deze server zal er vervolgens voor zorgen dat het updatebericht bij de gewenste gebruikers

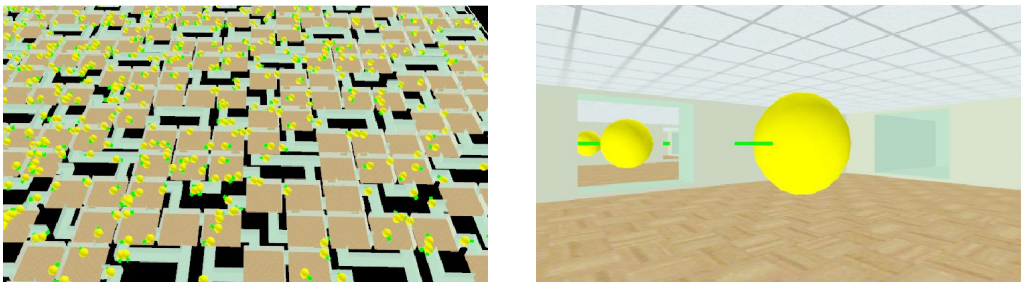
terechtkomt.

Het awareness management model van RING is gebaseerd op line-of-sight visibility. Alvorens de multi-user simulatie van start gaat, wordt de virtuele wereld opgedeeld in een aantal cellen die begrensd zijn door statische polygonen uit de virtuele omgeving ([52]). Vervolgens wordt er voor elke cel bepaald welke cellen zichtbaar zijn vanuit deze cel. Dit wordt geïllustreerd in figuur 5.3(a). Alle cellen die zichtbaar zijn vanuit de zwarte cel, werden in deze figuur grijs ingekleurd. De servers van het RING systeem hebben toegang tot deze op voorhand berekende visibility informatie, en forwarden binnenkomende updateberichten enkel naar gebruikers die zich in een cel bevinden die zichtbaar is vanuit de cel met het geupdate object. Enkel deze gebruikers zouden de verandering immers eventueel kunnen waarnemen. Veronderstel bijvoorbeeld dat gebruiker C uit figuur 5.3(a) zich verplaatst heeft in de virtuele omgeving, en dat de netwerktopologie van de NVE eruit ziet zoals aangegeven in figuur 5.3(b). Daar gebruiker C verbonden is met server Z, stuurt hij een updatebericht met zijn nieuwe positie naar deze server. Server Z forward dit bericht naar server Y, die het vervolgens doorstuurt naar gebruiker B. Server Z forward het updatebericht echter niet naar gebruiker D en ook niet naar server X. Zowel gebruiker D als alle clients die door server X beheerd worden (in dit geval enkel client A) bevinden zich immers in cellen die niet zichtbaar zijn vanuit de cel van gebruiker C. De labels van de pijlen in figuur 5.3(b) geven aan hoe updateberichten over de vier gebruikers gerouteerd zullen worden in het systeem. Bovendien geven de gestippelde kaders in deze figuur voor elke client aan over welke gebruikers hij lokaal informatie bijhoudt. Tussen updateberichten in simuleert elke client zelf het gedrag van deze gebruikers ([52]).



Figuur 5.3: RING: (a) Top-down overzicht van een virtuele omgeving met vier gebruikers; (b) Routering van updateberichten ([52])

De originele versie van het RING systeem gebruikte een statische client-server netwerktopologie waarbij elke client steeds verbonden was met dezelfde server. In de tweede versie van RING daarentegen is elke server verantwoordelijk voor een bepaalde regio van de virtuele omgeving, en communiceren clients met verschillende servers op basis van de regio waarin ze zich bevinden ([17]). Indien gebruikers hun huidige regio verlaten, schakelen ze automatisch over naar de server die verantwoordelijk is voor de nieuwe regio die ze betreden. Het grote voordeel hiervan is dat er minder updateberichten tussen servers uitgewisseld moeten worden, daar er in de virtuele omgeving meestal enkel visuele interactie mogelijk is tussen de gebruikers die zich in dezelfde regio bevinden. Gebruikers worden in RING overigens voorgesteld door een gele bol en een groene streep die de huidige kijkrichting van de gebruiker aangeeft (zie figuur 5.4). De avatars van het RING systeem zijn dus niet geleed, en kunnen geen animaties uitvoeren.



Figuur 5.4: Voorbeeld van een RING virtual environment met 400 kamers en 512 gelijktijdige gebruikers ([52])

5.5 DIVE

DIVE, een acroniem voor Distributed Interactive Virtual Environment, is een software platform waarmee multi-user virtuele omgevingen geïmplementeerd kunnen worden ([54]). Het DIVE platform werd ontwikkeld door het Swedish Institute of Computer Science (SICS), en concentreert zich voornamelijk op schaalbaarheid en interactiviteit. DIVE maakt om deze reden gebruik van een gedistribueerde repository, waarbij elke gebruiker van de NVE lokaal informatie bijhoudt over een gedeelte van de virtuele omgeving. Gebruikers kunnen interageren met elkaar en de virtuele omgeving door de inhoud van deze gedistribueerde repository aan te passen. Hierbij worden aanpassingen eerst lokaal toegepast en vervolgens via een updatebericht

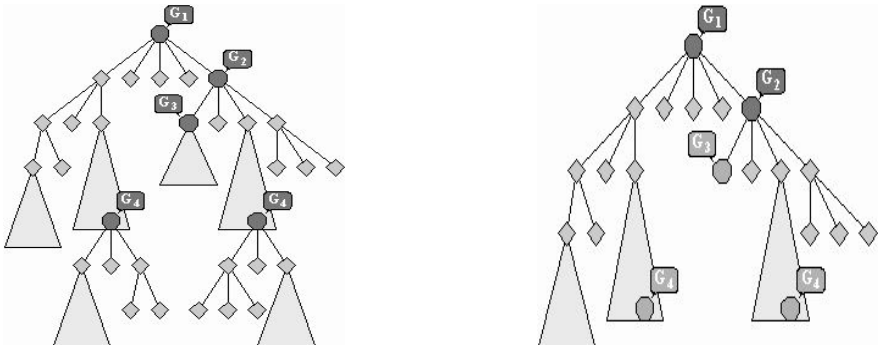
meegedeeld aan de andere gebruikers. Deze updateberichten worden via multicast verstuurd, zodat het lijkt alsof er zich ergens op het computernetwerk één gecentraliseerde repository bevindt. Maar dit is dus niet het geval, de informatie over de virtuele wereld wordt wel degelijk over de verschillende gebruikers van de NVE verdeeld.

Bovenstaande uiteenzetting maakt meteen ook duidelijk dat DIVE op het gebied van netwerkcommunicatie gebruik maakt van peer-to-peer multicasting. Dit is niet verwonderlijk, daar snelle interactie tussen gebruikers en de virtuele omgeving één van de belangrijkste doelstellingen van het DIVE platform is. Zoals besproken in sectie 2.3.1 introduceert een client-server architectuur immers steeds een zekere delay, wat een negatieve invloed heeft op de interactiesnelheid. Updateberichten worden om deze reden via multicasting naar alle (geïnteresseerde) gebruikers van de NVE tegelijk verstuurd. Wanneer een gebruiker een dergelijk updatebericht ontvangt, gebruikt hij de informatie in het bericht om zijn lokaal model bij te werken. Merk op dat updateberichten in DIVE betrouwbaar verstuurd worden opdat de lokale modellen van de verschillende gebruikers consistent zouden blijven. Tussen updateberichten in gebruikt iedere gebruiker overigens dead-reckoning om het gedrag van de objecten in zijn model lokaal te simuleren. Bovendien bevat de DIVE netwerktopologie enkele special purpose servers, zoals bijvoorbeeld de DIVE name server, die verantwoordelijk is voor het afhandelen van binnenkomende connectie aanvragen[3].

De inhoud van de virtuele omgeving wordt in de gedistribueerde repository opgeslagen als een hiërarchie van entiteiten (objecten, gebruikers, enzovoort). Het awareness management model van DIVE voorziet een mechanisme waarmee deze hiërarchie opgedeeld kan worden in sub-hiërarchieën ([54], zie figuur 5.5(a)). Met de bovenste entiteit van de hiërarchie en met elke subhiërarchie is er een uniek multicast adres geassocieerd. Deze subhiërarchieën worden in figuur 5.5(a) voorgesteld door een achthoek. Wanneer er een updatebericht over een bepaalde entiteit verstuurd moet worden, wordt de hiërarchie, vertrekkend van de entiteit in kwestie, naar boven toe doorlopen totdat er een multicast adres gevonden wordt. Het updatebericht wordt vervolgens naar dit multicast adres verstuurd. Elke DIVE applicatie beslist zelf met welke multicast adressen er geconnecteerd wordt, en houdt lokaal enkel informatie bij over de subhiërarchieën die met deze multicast adressen geassocieerd zijn ([54]). Indien een applicatie bijvoorbeeld enkel connecteert met multicast adressen G1 en G2 voor de hiërarchie uit figuur 5.5(a), ziet het lokaal model van deze

[3]Deze DIVE name server heeft dus exact dezelfde functie als de session manager server van het Spline platform.

applicatie eruit zoals aangegeven in figuur 5.5(b). De applicatie zal enkel over de entiteiten die getoond worden in deze figuur informatie ontvangen.

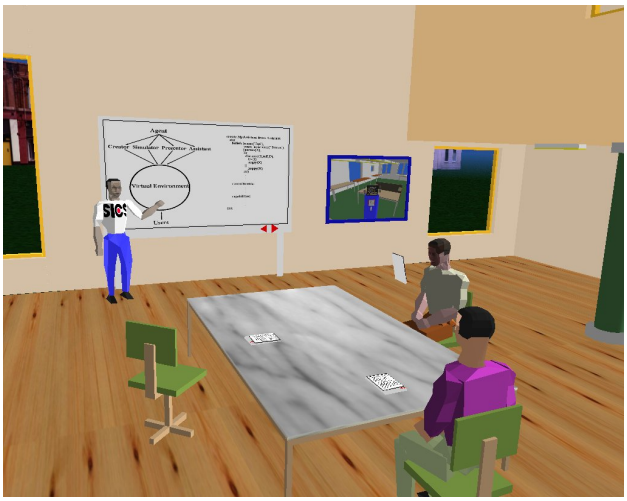


Figuur 5.5: Awareness management in DIVE: (a) Hiërarchische voorstelling van de volledige virtuele wereld; (b) Lokaal model van een applicatie die enkel met multicast adressen G1 en G2 geconnecteerd is ([54])

Het DIVE platform biedt gebruikers zowel simplistische als meer realistische virtuele belichamingen aan. Sommige avatars, zoals deze uit figuur 5.6 bijvoorbeeld, zijn geleed en kunnen verschillende animaties uitvoeren (wandelen, springen, knielen, enzovoort). Op de DIVE website ([53]) is er een videofragment te vinden dat de zogenaamde walkman avatar in actie toont. Gebruikers kunnen hun avatar controleren via een toetsenbord en een muis, of op een immersieve manier via body tracking sensoren. Bovendien ondersteunt DIVE autonomous avatars ([54]).

5.6 MASSIVE

MASSIVE (Model, Architecture and System for Spatial Interaction in Virtual Environments) is een gedistribueerd multi-user virtual reality systeem dat ontwikkeld werd door de Computer Science Department van de University of Nottingham. De MASSIVE onderzoeksgroep concentreert zich op awareness management in NVEs. Het originele spatial model of interaction werd voor het eerst volledig geïmplementeerd in MASSIVE-1 via peer-to-peer unicasting ([23]). Zijn opvolger, MASSIVE-2, implementeert het uitgebreide spatial model of interaction via peer-to-peer multicasting ([23], [24]). De meest recente versie van het systeem, MASSIVE-3, tenslotte gebruikt locales als basis voor awareness management, maar breidt dit concept uit met onder andere zogenaamde *aspects*, functionele en organisatorische scoping en abstracties ([21]).



Figuur 5.6: Een DIVE virtuele omgeving kan bijvoorbeeld gebruikt worden voor het houden van virtuele conferenties ([53])

5.7 VLNET

Het VLNET (Virtual Life NETwork) systeem, ontwikkeld door de MIRALab onderzoeksgroep van de University of Geneva, heeft voornamelijk aandacht voor de integratie van mensachtige avatars in virtuele omgevingen. VLNET bevat zowel directly controlled avatars, guided avatars als autonomous avatars. Deze avatars worden via skeletal animation geanimeerd en kunnen realistische bewegingen uitvoeren, zoals bijvoorbeeld wandelen en grijpen ([39]). Bovendien zijn er verschillende niet-verbale communicatiemechanismen opgenomen in VLNET. Zo kunnen de avatars in VLNET bijvoorbeeld enkele houdingen aannemen en verschillende gebaren uitvoeren ([34]). Gebruikers kunnen deze houdingen en gebaren zowel via real-time tracking als via een grafische user interface specificeren. Daarenboven integreerde de originele versie van VLNET gelaatsuitdrukkingen in de virtuele omgeving door het gezicht van de gebruiker te extraheren uit een continue beeldenstroom afkomstig van een camera, en het vervolgens als een texture op het gezicht van de avatar te plaatsen ([39], zie ook sectie 3.4). In latere versies van het systeem kunnen de gezichten van de avatars zelfs expliciet geanimeerd worden.

5.8 Multiplayer games

Computergames die gebruikers in staat stellen te spelen tegen menselijke tegenstanders zijn momenteel ontzettend populair. Het merendeel van deze multiplayer games (zoals bijvoorbeeld Counter-Strike van Valve Software) plaatst een limiet op het totaal aantal gebruikers dat gelijktijdig in de virtuele omgeving aanwezig kan zijn[4]. Bovendien bevatten de virtuele omgevingen die dergelijke computergames aanbieden meestal weinig objecten waarmee gebruikers kunnen interageren. Deze "gewone" multiplayer games kunnen wel degelijk als NVE systemen beschouwd worden, maar het is duidelijk dat ze veel problemen die verbonden zijn aan het implementeren van een NVE omzeilen door verschillende restricties op te leggen. Dankzij deze restricties hebben dergelijke multiplayer games bijvoorbeeld geen awareness management model nodig. MMORPGs vormen hier echter een uitzondering op. De meeste MMORPGs bieden immers rijke virtuele omgevingen aan waarin honderden tot zelfs duizenden geografisch verspreide gebruikers gelijktijdig aanwezig kunnen zijn. Het is dan ook niet moeilijk om in te zien dat awareness management in MMORPGs wel een cruciale rol speelt.

Daar computergames meestal een zeker realisme nastreven, is het niet verwonderlijk dat de meeste multiplayer games realistische, mensachtige avatars gebruiken om hun gebruikers in de virtuele omgeving voor te stellen. Spelontwikkelaars moesten vroeger een beroep doen op minderwaardige technieken zoals bijvoorbeeld single mesh characters om deze belichamingen in real-time te animeren. Dankzij de opkomst van snellere processors en goedkope grafische hardware zijn computers voor thuisgebruik momenteel echter in staat om complexe avatars op een realistische manier in real-time te animeren, bijvoorbeeld via skeletal animation. Zoals reeds aangehaald in hoofdstuk 4 overtreft skeletal animation de oudere animatiebenaderingen op alle vlakken. Het is dan ook niet verwonderlijk dat de meeste hedendaagse multiplayer computergames gebruik maken van skeletal animation om de virtuele belichamingen van gebruikers te animeren. Hoewel skeletal animation een uitermate krachtige animatiemethode is die zeer realistische bewegingen kan produceren, moet het echter toch opgemerkt worden dat de avatars in het merendeel van de huidige computergames enkel voorgeprogrammeerde acties kunnen uitvoeren, en dat dynamische interactie tussen avatar en virtuele omgeving nog maar zeer zelden mogelijk is. Zo is er bijvoorbeeld voor het oppikken van een voorwerp meestal slechts één voorgedefinieerde animatie voorzien. Dit wil zeggen dat steeds dezelfde animatie afgespeeld zal worden wanneer een gebruiker een object opraapt, wat ook de hoogte

[4]De meeste multiplayer computergames ondersteunen maximaal 32 of 64 gelijktijdige gebruikers.

is waarop dit object zich bevindt. Inverse kinematics laat een meer dynamische interactie tussen avatar en virtuele omgeving toe, en zou in dergelijke situaties veel realistischere resultaten kunnen produceren.

Computergames trachten een zo breed mogelijk publiek te bereiken, en voorzien om deze reden bijna allemaal Internet-ondersteuning. Daar multicasting over het Internet momenteel nog onmogelijk is, wil dit zeggen dat bijna alle huidige multiplayer games een client-server netwerktopologie gebruiken.

5.9 Conclusie

In dit hoofdstuk werden er enkele bestaande NVE systemen besproken. Het is duidelijk dat veel NVEs, en dan vooral de academische, zich concentreren op één bepaald aspect van de NVE problematiek. Zo concentreren RING en MASSIVE zich bijvoorbeeld op awareness management en schaalbaarheid, terwijl DIVE vooral een zo hoog mogelijke interactiesnelheid nastreeft. VLNET heeft dan weer voornamelijk aandacht voor de integratie van realistische, mensachtige avatars in virtuele omgevingen en de problemen die dit met zich meebrengt. Het gevolg van dergelijke specialisaties is dat het merendeel van de bestaande NVEs uitblinkt in één bepaald gebied, maar spijtig genoeg ook vaak te weinig aandacht besteedt aan de andere aspecten van de NVE problematiek waar niet op geconcentreerd wordt. Het MASSIVE systeem bijvoorbeeld maakt nog steeds geen gebruik van mensachtige avatars om gebruikers in de virtuele wereld te representeren, hoewel het toch reeds enige tijd duidelijk is dat mensachtige avatars een enorm gunstige invloed hebben op het gevoel van presence en co-presence in NVE systemen.

Merk ook op dat er een aantal opmerkelijke verschilpunten tussen commerciële en academische NVEs vastgesteld kunnen worden. De meeste commerciële NVE systemen beschikken bijvoorbeeld over een krachtige 3D grafische engine, terwijl het merendeel van de academische toepassingen een eerder elementaire grafische engine gebruikt. De reden hiervoor is eenvoudig. Visueel realistische virtuele omgevingen zijn aantrekkelijk voor gebruikers, en kunnen het verkoopcijfer van commerciële toepassingen enorm doen toenemen. Academische onderzoeksgroepen daarentegen streven geen financiëel succes na, en nemen dan ook vaak genoegen met een eenvoudige grafische engine en concentreren zich op andere NVE aspecten zoals bijvoorbeeld schaalbaarheid en ondersteuning voor spraak. Ook op het gebied van avatars en avatar animatie staan de commerciële NVEs veel verder dan hun academische tegenhangers. De avatars van de meeste multiplayer computergames kunnen een enorm gamma van bewegingen uitvoeren, terwijl academische NVEs meestal niet

al te veel aandacht besteden aan avatar animatie (het VLNET systeem vormt hier duidelijk een uitzondering op). Hierbij moet echter wel opgemerkt worden dat veel academische NVEs, en dit in tegenstelling tot commerciële NVEs, directly controlled avatars ondersteunen. Dit kan verklaard worden door het feit dat er speciale invoerapparatuur nodig is om directly controlled avatars te besturen, die wegens zijn hoge kostprijs momenteel nog maar enkel in onderzoeksinstellingen in wijdverspreid gebruik is. Samengevat kunnen we dus stellen dat het merendeel van de commerciële NVEs een zo hoog mogelijk visueel realisme nastreeft, terwijl academische NVEs over het algemeen meer aandacht hebben voor andere aspecten van de NVE problematiek zoals bijvoorbeeld schaalbaarheid en het ondersteunen van veel gelijktijdige gebruikers.

Hoofdstuk 6

Implementatie

6.1 Inleiding

In het kader van deze thesis werd er een eenvoudige NVE in de programmeertaal C++ ontwikkeld die enkele principes uit de voorafgaande hoofdstukken in de praktijk omzet. Er werd voor een zuivere client-server netwerk architectuur gekozen omwille van de conceptuele eenvoud van deze topologie. Bovendien was het niet meteen de bedoeling om een large-scale NVE te implementeren die veel gelijktijdige gebruikers ondersteunt. De server fungeert als een centrale repository die op elk moment de meest recente informatie over alle gemeenschappelijke voorwerpen in de virtuele omgeving bevat. Wanneer een gebruiker de toestand van een gemeenschappelijk voorwerp verandert, past hij eerst zijn lokaal model van de virtuele wereld aan, en verstuurt hij vervolgens een bericht met informatie over het geupdate object naar de server. De server slaat de informatie in dergelijke updateberichten lokaal op, en forward ze vervolgens simpelweg naar alle andere gebruikers die op dat moment met de NVE geconnecteerd zijn. De geïmplementeerde NVE gebruikt met andere woorden geen specifiek awareness management model.

Bovenstaande uitweiding maakt duidelijk dat er niet al te veel aandacht besteed werd aan de drie subsystemen die in sectie 2.3 als essentieel voor de werking van NVEs bestempeld werden (netwerk communicatie, het consistent houden van gemeenschappelijke informatie en awareness management). De nadruk van de implementatie lag dan ook volledig op de real-time animatie van de belichamingen van gebruikers in de virtuele wereld. De ontwikkelde applicatie biedt gebruikers een aantal avatars aan die onder andere kunnen wandelen, lopen en springen. Dit wil zeggen dat de avatar van een bepaalde gebruiker steeds op een realistische manier zal bewegen wanneer deze gebruiker doorheen de aangeboden virtuele omgeving na-

vigeert. Het controleren van de avatars gebeurt overigens via het toetsenbord en de muis.

6.2 Skeletal animation

Zoals reeds aangehaald in hoofdstuk 4 is skeletal animation een krachtige animatiemethode die zeer realistische bewegingen kan produceren, en die op alle vlakken superieur is aan de oudere animatiebenaderingen. Het is dan ook niet verwonderlijk dat er gekozen werd voor skeletal animation om de belichamingen van gebruikers in de virtuele omgeving te animeren. De verschillende avatarmodellen en hun animaties werden aangemaakt met een animatiepakket dat beschreven wordt in de volgende sectie, en werden vervolgens opgeslagen in een file. Wanneer de ontwikkelde applicatie opgestart wordt, wordt de data in deze files ingelezen in een aantal datastructuren. Meer concreet zijn er datastructuren voorzien die vertices, driehoeken, meshes, materialen, gewrichten, keyframes en animaties voorstellen ([55], [56]). De gebruikte avatarmodellen bestaan overigens uit twee lagen, namelijk een skelet laag en een huid laag.

De vertex datastructuur houdt zowel de positie van de vertex als het bot waarmee deze huidvertex geassocieerd is bij. Elke huidvertex is steeds met juist één bot geassocieerd, wat wil zeggen dat de gebruikte avatarmodellen rigid zijn (zie sectie 4.2.1). De huidvertices worden vervolgens gegroepeerd tot driehoeken. Deze driehoeken worden in het programma voorgesteld door de driehoek datastructuur, die zowel texturecoördinaten als normalen bijhoudt voor zijn drie samenstellende vertices. Driehoeken worden op hun beurt dan weer gegroepeerd tot meshes. De mesh datastructuur bewaart naast referenties naar zijn samenstellende driehoeken ook een referentie naar de materiaaleigenschappen van deze mesh. Een mesh is dus niets meer dan een collectie driehoeken die allemaal dezelfde materiaaleigenschappen hebben. Materiaaleigenschappen tenslotte worden in het programma voorgesteld door de materiaal datastructuur, die zowel informatie over de belichting als de gebruikte texture bijhoudt.

De hierboven besproken datastructuren volstaan wanneer men de avatarmodellen enkel wil inladen en renderen. Indien men de modellen echter ook daadwerkelijk wil animeren, heeft men een aantal bijkomende datastructuren nodig. Eerst en vooral moeten de verschillende gewrichten die samen het skelet van het model vormen eveneens gerepresenteerd worden in het programma. Hiervoor zorgt de joint datastructuur. In deze datastructuur wordt de volgende informatie bewaard:

- De naam van de joint.
- Een referentie naar het oudergewricht van deze joint. De verschillende gewrichten vormen samen immers een hiërarchie.
- Een translatie en rotatie (twee 3D vectoren) die aangeven waar dit gewricht zich in de ruimte bevindt ten opzichte van zijn oudergewricht.
- Een *relative* transformatiematrix die dit gewricht relateert aan zijn oudergewricht.
- Een *absolute* transformatiematrix die de positie van het gewricht weergeeft ten opzichte van de oorsprong van de virtuele wereld.
- Een *final* transformatiematrix die gebruikt wordt bij het renderen van het model.

Het is duidelijk dat met deze representatiewijze het skelet van het avatarmodel geanimeerd kan worden door de final transformatiematrix van de verschillende gewrichten van het model te variëren in de tijd. Hiervoor wordt er een beroep gedaan op de keyframing techniek die besproken werd in sectie 4.4. Dit wil zeggen dat de verschillende animaties die de modellen kunnen uitvoeren opgeslagen werden als een opeenvolging van keyframes. Deze keyframes worden in het programma gerepresenteerd door de keyframe datastructuur, die de positie *of* de oriëntatie van één bepaald gewricht van het skelet op een bepaald tijdstip in een animatie bijhoudt. De animaties zelf tenslotte worden in het programma voorgesteld door de animatie datastructuur. Deze datastructuur bewaart onder andere de volgende informatie:

- De naam van de animatie (bijvoorbeeld “run” of “idle”).
- De totale tijd (in milliseconden) die nodig is om deze animatie volledig af te spelen.
- Een variabele die aangeeft of de animatie steeds herhaald moet worden of slechts één keer afgespeeld moet worden.
- Een timer die telt hoe lang de animatie reeds aan het afspelen is.
- Voor elk gewricht van het model een aantal translational en rotational keyframes die het algemene verloop van de animatie bepalen.

Wanneer een bepaalde animatie gestart wordt, wordt de final transfomatiematrix van elk gewricht van het model gelijkgesteld aan de absolute transformatiematrix van dit gewricht, en worden de huidige translational en rotational keyframes van elk gewricht gelijkgesteld aan de eerste translational, respectievelijk rotational keyframe voor dit gewricht. Bovendien wordt de timer van de animatie gereset. Gedurende de animatie wordt er vervolgens voortdurend de huidige tijd opgevraagd, en de final transformatiematrix van de verschillende gewrichten aangepast zodat de houding van het model zich ergens tussen het huidige keyframe en het volgende keyframe in bevindt, afhankelijk van de tijd die reeds verlopen is sinds het huidige keyframe. Dit gebeurt overigens afzonderlijk voor de translational en rotational keyframes van de animatie in kwestie. Naarmate de tijd vordert worden de verschillende keyframes doorlopen, wat wil zeggen dat de avatar de gewenste animatie zal afspelen. Om het avatarmodel tijdens een dergelijke animatiecyclus te renderen, volstaat het elke vertex van het model continu te transformeren met de final transformatiematrix van het gewricht waarmee deze vertex geassocieerd is. Via de driehoek datastructuur kan het model vervolgens uitgetekend worden op het scherm als een collectie driehoeken.

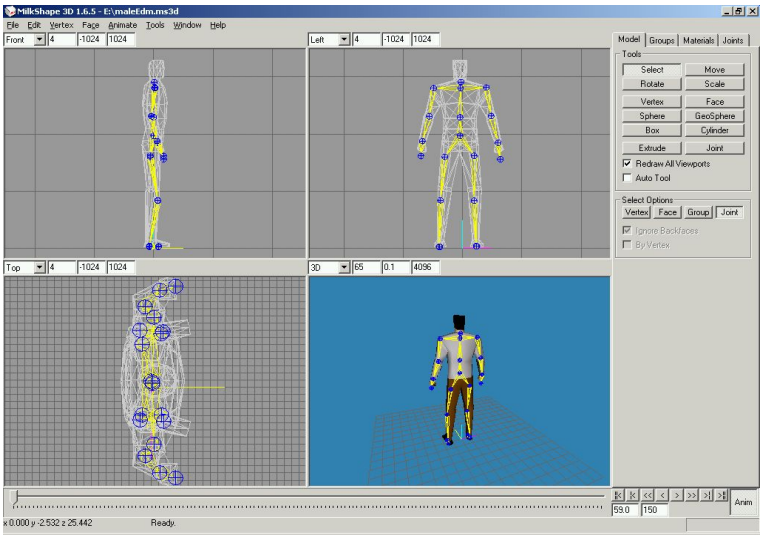
6.3 MilkShape 3D

6.3.1 De modeler

Er werd een beroep gedaan op MilkShape 3D om de avatarmodellen en de animaties die deze modellen kunnen uitvoeren aan te maken. MilkShape 3D is een eenvoudige doch krachtige modeler die ontwikkeld werd door chUmbaLum sOft ([57], zie ook figuur 6.1). Gebruikers kunnen met behulp van MilkShape 3D zowel skeletten als modellen construeren, skeletten integreren in modellen, en deze modellen vervolgens animeren via skeletal animation. Interactie met het skelet van een gelaagd model is in MilkShape 3D enkel mogelijk via forward kinematics, en de gecreëerde animaties moeten steeds opgeslagen worden als een sequentie van keyframes.

6.3.2 Het MS3D file formaat

Het file formaat van de MilkShape 3D modeler is vrijgegeven en ontzettend goed gedocumenteerd. Dit maakte het inladen van MS3D files in de datastructuren die besproken werden in de vorige sectie veel eenvoudiger, en is meteen ook één van de belangrijkste redenen waarom er gekozen werd voor MilkShape 3D en niet voor een ander animatiepakket zoals bijvoorbeeld 3D Studio Max. Het MS3D file formaat



Figuur 6.1: Een screenshot van de MilkShape 3D modeler van chUmbaLum sOft

heeft echter één beperking: elke MS3D file kan namelijk slechts één animatie bevatten. De avatars van de beoogde applicatie moesten echter meerdere animaties kunnen uitvoeren. Een mogelijke oplossing voor dit probleem bestaat eruit verschillende MS3D files aan te maken voor ieder avatarmodel, waarbij elke file juist één animatie bevat. Het is niet moeilijk om in te zien dat deze aanpak aanleiding geeft tot een grote collectie MS3D files. Hoewel hier in principe niets mis mee is, is het veel overzichtelijker indien alle animaties voor een bepaald model gegroepeerd worden in één en dezelfde file. Bovendien moet men er rekening mee houden dat een MS3D file niet enkel de animatie bevat die door een bepaald model uitgevoerd kan worden, maar tevens ook dit model zelf. Dit wil zeggen dat de modeldata van de avatars ook onnodig in verschillende files gekopieerd zou moeten worden indien er voor deze aanpak gekozen werd.

Om deze twee problemen te vermijden, werd er besloten de verschillende animaties voor een bepaald model achter elkaar te plaatsen in de tijd, en ze vervolgens op te slaan als één lange animatie. Op deze manier moet er steeds slechts één MS3D file aangemaakt worden voor elk avatarmodel. Dergelijke files bevatten naast het model zelf immers ook alle animaties die door dit model uitgevoerd kunnen worden. Maar hoe weten we nu welke keyframes in de MS3D file bij welke animatie thuishoren[1]? Daar deze informatie niet opgenomen kan worden in de MS3D file zelf, werd ervoor gekozen om de MS3D files voor de verschillende modellen te accompagneren met een

[1]Herinner u dat MilkShape 3D animaties steeds opslaat als een sequentie van keyframes.

eenvoudige tekstfile. Deze tekstfiles houden het totaal aantal animaties bij dat is opgeslagen in de bijbehorende MS3D file, en bevatten bovendien een verzameling tripletten die elk een animatie afbakenen in deze file. Het eerste element van het triplet geeft de keyframes aan waaruit de animatie bestaat, het tweede element specificeert de naam van de animatie, en het derde element tenslotte bewaart het totaal aantal frames van de animatie (hiermee kan de totale tijdsduur van de animatie berekend worden). Daarnaast bevatten deze tekstfiles ook de naam, een korte beschrijving, en de URL van een bitmap preview van het corresponderende avatarmodel. Deze informatie wordt aan de gebruikers van de ontwikkelde applicatie getoond wanneer ze een avatar moeten selecteren. De tekstfiles houden tenslotte ook nog de namen bij van de gewrichten die on the fly geanimeerd kunnen worden via inverse kinematics (zie sectie 6.5).

Figuur 6.2 illustreert de layout van de zojuist besproken ondersteunende tekstfiles (de tekst in blauw is commentaar). De MS3D file die hoort bij de tekstfile die in deze figuur getoond wordt, bevat vier verschillende animaties, waarbij de eerste 30 keyframes in de MS3D file de run animatie specificeren, de 70 volgende de idle animatie, enzovoort. Bovendien blijkt in dit voorbeeld het totaal aantal frames van elke animatie gelijk te zijn aan het aantal keyframes van deze animatie. Dit wil zeggen dat in dit geval elk frame van de animatie een keyframe is. Normaal gezien zal het totaal aantal frames van een animatie echter groter zijn dan het aantal keyframes van deze animatie.

```
4 – total number of animations in the MS3D file
EDM Male – name of the model
A standard male avatar – short description of the model
maleEdm.bmp – URL of a .bmp image preview of the model
Joint6 Joint7 Joint8 Joint9 – Names of the IK joints
1-30     run      1-30 – list of (keyframenumber, animationname,
31-100   idle     31-100         framenumber) triplets
101-130  walk     101-130
131-150  jump     131-150
```

Figuur 6.2: Elke MS3D file wordt vergezeld door een tekstfile die extra informatie over het model en zijn animaties bijhoudt

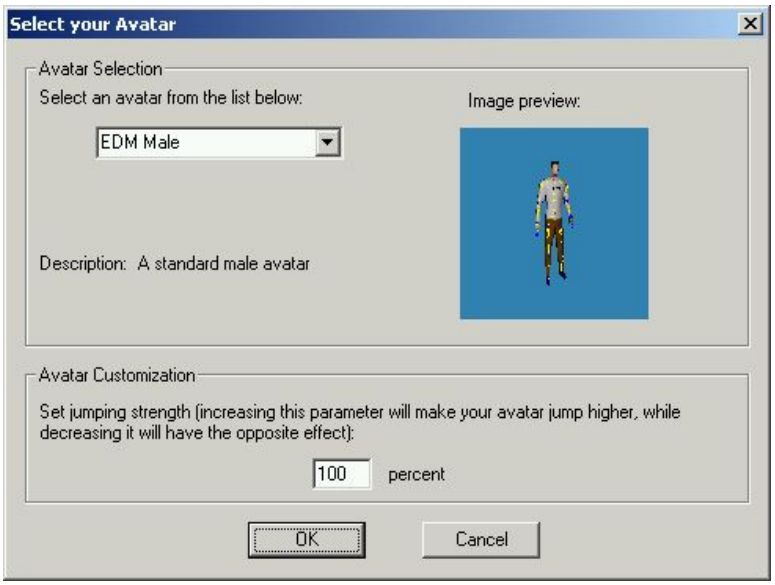
6.4 Combineren van animaties

Het met de hand aanmaken van animaties is een zeer tijdrovende bezigheid. Daarom werd er code voorzien waarmee verschillende enkelvoudige animaties gecombineerd

kunnen worden tot een samengestelde animatie (zoals bijvoorbeeld wandelen en springen tegelijk). Hierbij kan er via percentages gespecificeerd worden hoe groot de invloed van elke samenstellende animatie moet zijn op de resulterende gecombineerde animatie. Dit wil zeggen dat de ontwikkelde applicatie op een zeer flexibele en dynamische manier samengestelde animaties on the fly kan genereren. Het grote voordeel hiervan is dat enkel enkelvoudige animaties met de hand gecreëerd moeten worden. Indien er bijvoorbeeld zowel een wandel als een spring animatie voor een bepaalde avatar voorzien is, kan deze avatar dankzij deze aanpak meteen ook wandelen en springen tegelijk.

Door de introductie van de percentages kunnen er bovendien een aantal interessante effecten gerealiseerd worden. Zo kunnen de gebruikers van de ontwikkelde applicatie bijvoorbeeld via de user interface de capaciteiten van hun avatar personaliseren. Dit wordt geïllustreerd in figuur 6.3. Door het percentage onderaan in deze dialoog box te verhogen, kan een gebruiker zijn avatar hoger laten springen, terwijl het verlagen van dit percentage het omgekeerde effect heeft. Merk op dat dit personaliseren helemaal niet beperkt hoeft te worden tot de spring animatie, maar zonder problemen ook toepasbaar is op andere animaties. Bovendien kunnen de percentages niet enkel door de gebruiker gespecificeerd worden, ze kunnen ook dynamisch door de computer berekend en ingesteld worden. Zo kan de computer er bijvoorbeeld voor zorgen dat de avatar van een bepaalde gebruiker automatisch sterker wordt wanneer deze gebruiker een belangrijke opdracht uitgevoerd heeft in de virtuele omgeving. Het oplossen van een opdracht kan bijvoorbeeld beloond worden door de maximale springhoogte van de avatar te verhogen[2]. Hiervoor moeten er enkel een aantal animatiepercentages aangepast worden. Daarnaast maken de animatiepercentages het ook mogelijk dat samengestelde animaties dynamisch afhangen van omgevingsfactoren. Zo kan er bijvoorbeeld rekening gehouden worden met de huidige gezondheidstoestand van de avatar wanneer een mank animatie en een wandel animatie gecombineerd worden met elkaar. Naarmate de gezondheidstoestand van de avatar daalt (doordat hij bijvoorbeeld van een te grote hoogte naar beneden gevallen is), kan het percentage van de mank animatie bijvoorbeeld stelselmatig toenemen, terwijl het percentage van de wandel animatie evenredig vermindert. Dit zou als gevolg hebben dat de avatar gewoon wandelt wanneer hij nog volledig gezond is, maar steeds meer begint te manken naarmate zijn gezondheidstoestand afneemt.

[2]Dergelijke beloningssystemen worden onder andere in MMORPGs gebruikt.

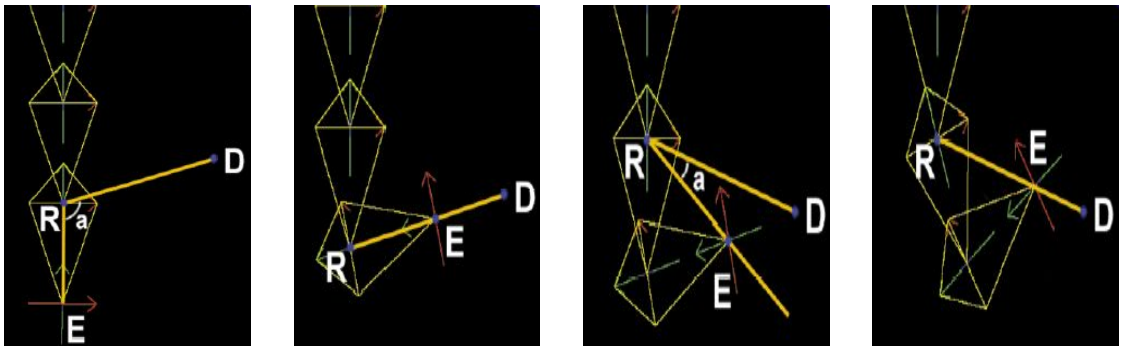


Figuur 6.3: Gebruikers kunnen via de user interface de capaciteiten van hun avatar personaliseren

6.5 Inverse kinematics

De avatars van de ontwikkelde applicatie kunnen niet alleen voorgedefinieerde bewegingen uitvoeren, maar kunnen ook rechtstreeks via inverse kinematics geanimeerd worden. Hiervoor werd er een beroep gedaan op de *Cyclic-Coordinate Descent* methode, een numerieke inverse kinematics methode die gebaseerd is op optimaliseringstechnieken ([45], [48]). Zoals reeds vermeld in sectie 4.5, trachten dergelijke algoritmen de gewenste positie en oriëntatie van de end-effector zo dicht mogelijk te benaderen door een bepaalde error functie te minimaliseren via iteratieve berekeningen. Hoewel numerieke methoden trager zijn dan closed form methoden, werd er toch voor de eerste benadering gekozen omdat deze in staat zijn om willekeurig complexe inverse kinematics problemen op te lossen, terwijl closed form benaderingen enkel eenvoudige inverse kinematics problemen met een beperkt aantal degrees of freedom kunnen oplossen.

De werking van de Cyclic-Coordinate Descent (CCD) methode wordt geïllustreerd in figuur 6.4. De CCD methode doorloopt de gewrichtenketting waarvan de end-effector gepositioneerd moet worden steeds van het losse naar het vaste uiteinde, en past de hoek van de verschillende gewrichten in de ketting één voor één in de goede richting aan. Het algoritme bepaalt steeds twee vectoren voor het gewricht dat momenteel beschouwd wordt, namelijk een vector die vertrekt bij de positie van



Figuur 6.4: De Cyclic-Coordinate Descent methode past de hoek van ieder gewricht in de gewrichtenketting afzonderlijk aan ([45])

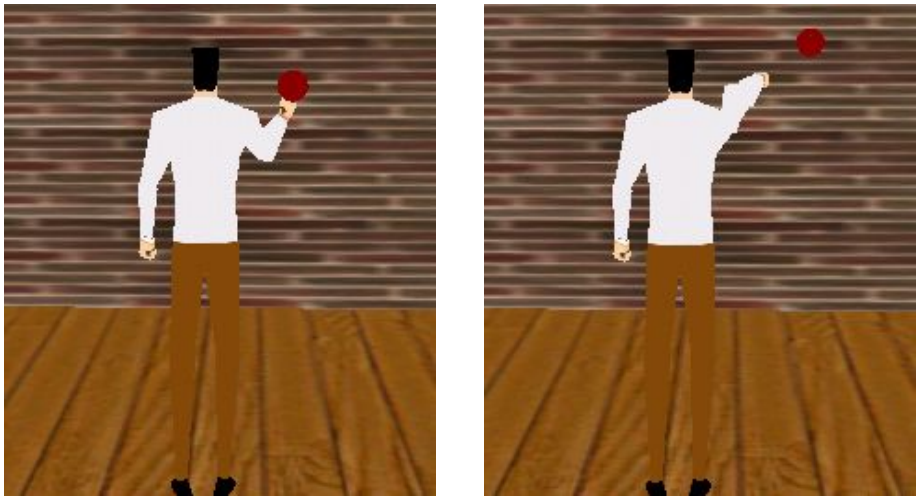
dit gewricht en eindigt bij de *huidige* positie van de end-effector (vector RE), en een vector die vertrekt bij de positie van het gewricht en eindigt bij de *gewenste* positie van de end-effector (vector RD). Het gewricht wordt vervolgens zo geroteerd dat de vector RE samenvalt met de vector RD (zie figuur 6.4(b)). Hiervoor wordt zowel het scalair als het vector product van de vectoren RE en RD berekend. Dit leert ons immers respectievelijk de hoek a tussen deze twee vectoren en de benodigde draairichting. Na het daadwerkelijk roteren van het gewricht, beschouwt het algoritme het volgende gewricht in de gewrichtenketting, en wordt de procedure herhaald (zie figuur 6.4(c) en 6.4(d)). Wanneer het vaste uiteinde van de gewrichtenketting bereikt wordt, begint het algoritme opnieuw met het meest distale gewricht in de ketting (het losse uiteinde). Dit proces wordt herhaald totdat de error functie voldoende geminimaliseerd is[3], of totdat een vooropgesteld maximaal aantal iteraties bereikt wordt[4]. Het is duidelijk dat er slechts een beperkt aantal berekeningen per iteratie uitgevoerd moeten worden, wat wil zeggen dat deze methode bruikbaar is voor interactieve applicaties. Het nadeel van de CCD methode is echter dat ze distale gewrichten sneller zal roteren dan gewrichten die zich dichter bij het vaste uiteinde van de gewrichtenketting bevinden. Het is niet moeilijk om in te zien dat dit in bepaalde gevallen kan resulteren in visueel onrealistische oplossingen voor het beschouwde inverse kinematics systeem.

Figuur 6.5 toont hoe inverse kinematics in de ontwikkelde applicatie aangewend wordt om de rechterhand van een avatar naar een willekeurige positie in de virtuele

[3]De error functie is in dit geval gelijk aan de afstand tussen de huidige en de gewenste positie van de end-effector.

[4]Deze test zorgt ervoor dat het algoritme niet in een oneindige lus terechtkomt wanneer de gewenste end-effector positie niet bereikbaar is.

ruimte te laten wijzen. De gewenste positie van de end-effector wordt voorgesteld door een rode bol, en kan op een interactieve manier door de gebruiker ingesteld worden via het toetsenbord. Indien de rechterhand van de avatar de gewenste positie kan bereiken, worden de gewrichten van de rechterarm zo geroteerd dat de hand ook daadwerkelijk op deze positie terechtkomt (zie figuur 6.5(a)). Indien de gewenste positie echter niet bereikbaar is, wordt er getracht deze positie zo dicht mogelijk te benaderen (zie figuur 6.5(b)).



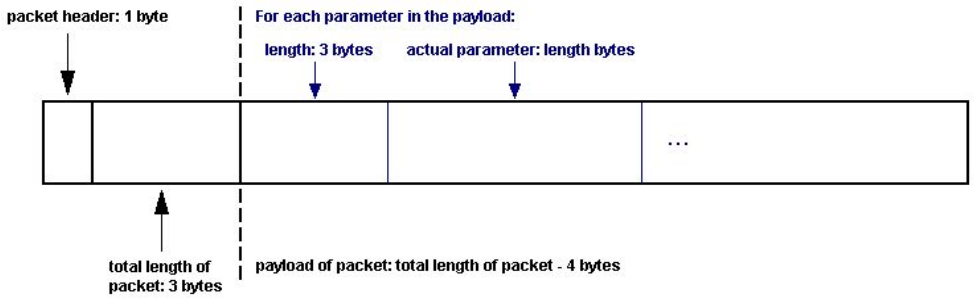
Figuur 6.5: De avatars van de ontwikkelde applicatie kunnen naar gelijk welke positie in de virtuele omgeving wijzen

6.6 Netwerk communicatie

Zoals reeds vermeld in de inleiding, maakt de ontwikkelde applicatie gebruik van een zuivere client-server netwerk architectuur. Dit wil zeggen dat de verschillende gebruikers niet rechtstreeks met elkaar kunnen communiceren, maar verplicht zijn hun berichten via een centrale server te routeren. Om de communicatie tussen de clients en de server te verzorgen, wordt er gebruik gemaakt van TCP unicasting. Wanneer een gebruiker connecteert met het systeem, wordt er automatisch een TCP verbinding opgezet tussen deze client en de server. Eens deze verbinding is aangemaakt, wordt vervolgens alle informatie die uitgewisseld moet worden tussen de client en de server over deze verbinding verstuurd. Wanneer een gebruiker bijvoorbeeld bewogen heeft in de virtuele omgeving, maakt hij gebruik van de opgezette verbinding om

de server op de hoogte te brengen van zijn nieuwe positie en oriëntatie. Anderzijds gebruikt de server de opgezette TCP verbindingen om clients te informeren over de acties die andere gebruikers uitgevoerd hebben. Wanneer een gebruiker de connectie met het systeem verbreekt, wordt zijn TCP verbinding met de server terug vrijgegeven. Het gebruik van zowel TCP unicasting als een client-server architectuur wordt gerechtvaardigd door het feit dat het niet meteen de bedoeling was een large-scale NVE te implementeren die veel gelijktijdige gebruikers ondersteunt.

De pakketten die uitgewisseld worden tussen de clients en de server hebben allen eenzelfde formaat. Een pakket bestaat steeds uit drie onderdelen, namelijk een header, een lengte-veld en een payload (zie figuur 6.6). De header geeft aan welke informatie er in de payload van het pakket aanwezig is. In het lengte-veld wordt de totale lengte van het pakket opgeslagen. De payload van het pakket tenslotte bevat de eigenlijke data, waarbij de header van het pakket bepaalt wat deze data voorstelt. Zo zal bijvoorbeeld in het geval van de header HD_REQUEST_CONNECTION de payload van het pakket één string bevatten, namelijk de nickname van een gebruiker die wenst te connecteren met het systeem. Merk op dat een parameter in de payload steeds voorafgegaan wordt door zijn lengte in bytes. Indien de payload van een pakket meerdere parameters bevat, kan de ontvanger met behulp van deze lengtes de verschillende parameters correct uit de payload extraheren.



Figuur 6.6: De pakketten die uitgewisseld worden tussen de clients en de server hebben allen eenzelfde formaat

De belangrijkste beslissing die op het gebied van netwerk communicatie genomen moest worden, betreft de manier waarop gebruikers geïnformeerd worden over de houding van de avatars van de andere gebruikers. Wanneer de avatar van een bepaalde gebruiker een animatie uitvoert, moeten de andere gebruikers daar immers op de één of andere manier van op de hoogte worden gebracht. Zoals reeds vermeld in sectie 3.5 zal een NVE ontwerper hierbij steeds een afweging moeten maken tussen processorbelasting enerzijds en gegenereerde netwerktrafiek ander-

zijds. Zowel processortijd als netwerk bandbreedte zijn immers kostbare resources. Hoewel breedband Internetaansluitingen stilaan ingeburgerd raken, beschikken computers voor thuisgebruik momenteel in verhouding toch over meer processorkracht dan bandbreedte. Daarom werd er getracht de netwerkbelasting zo laag mogelijk te houden door gebruikers enkel high-level beschrijvingen te laten uitwisselen van de huidige toestand van hun avatar. Wanneer de avatar van een bepaalde gebruiker een animatie begint af te spelen, wordt er een pakket naar de andere gebruikers verstuurd dat de naam van de gestarte animatie bevat. Na het ontvangen van een dergelijk updatebericht, moeten remote hosts zelf zorgen voor de animatie van de avatar in kwestie. Wanneer een avatar stopt met het afspelen van een animatie, wordt dit eveneens via een gelijkaardig updatebericht meegedeeld aan de andere gebruikers. Het is duidelijk dat deze updateberichten zeer klein zijn (hoogstens een tiental bytes), en dat dergelijke berichten normaal gezien niet al te vaak verstuurd zullen moeten worden. Het is echter eveneens niet moeilijk om in te zien dat deze aanpak een grote computationele last op de clients plaatst. Ook wanneer de avatar via inverse kinematics geanimeerd wordt, wordt er getracht de gegenereerde netwerktrafiek zo laag mogelijk te houden, en wordt enkel de gewenste positie van de end-effector naar de andere gebruikers doorgestuurd.

6.7 Mogelijke uitbreidingen

Het is belangrijk om op te merken dat de voorgestelde implementatie zijn beperkingen heeft en op verschillende manieren uitgebreid kan worden. Ten eerste zou men tijdens het combineren van animaties rekening kunnen houden met prioriteiten door animaties enkel deelverzamelingen van gewrichten te laten beïnvloeden. Het is bijvoorbeeld niet moeilijk om in te zien dat bij een wandel beweging de animatie van de benen primeert, terwijl voor een wuif animatie de gewrichten van de armen de belangrijkste gewrichten zijn. Deze informatie kan zonder problemen opgenomen worden in de ondersteunende tekstfiles die gesproken werden in sectie 6.3.2. Dit zou als gevolg hebben dat de beweging van de armen van een avatar die wandelt en wuift tegelijk volledig bepaald wordt door de keyframes van de wuif animatie, terwijl de beweging van de benen volledig bepaald wordt door de keyframes van de wandel animatie. Het is duidelijk dat dit een zeer realistisch resultaat zou opleveren. Deze aanpak kan overigens zonder problemen gecombineerd worden met de gekozen percentages-benadering die besproken werd in sectie 6.4.

Ten tweede zouden er constraints geïntroduceerd kunnen worden in het geïmplementeerde inverse kinematics algoritme. De huidige versie van het algoritme tracht

de gewenste end-effector positie zo snel mogelijk te bereiken, en zal in bepaalde gevallen de gewrichten van het model nogal bruusk roteren. Dit kan resulteren in onrealistische houdingen (zie bijvoorbeeld figuur 6.5(b)). Hier kan een mouw aan gepast worden door damping te introduceren in het algoritme, en door beperkingen te plaatsen op de degrees of freedom van de verschillende gewrichten van het model. Damping zorgt ervoor dat de hoek van een gewricht nooit met meer dan een vooropgesteld maximum ineens aangepast kan worden. Beperkingen op de degrees of freedom van het model kunnen dan weer beletten dat de avatar fysisch onmogelijke houdingen aanneemt. Het is duidelijk dat de introductie van deze constraints het visueel realisme van de houdingen die door het geïmplementeerde inverse kinematics algoritme geproduceerd worden drastisch kan verhogen.

Ten derde tenslotte zou de ontwikkelde applicatie op verschillende manieren interessanter gemaakt kunnen worden voor gebruikers. Gebruikers kunnen momenteel immers enkel de aangeboden virtuele omgeving verkennen, en kunnen bijvoorbeeld niet via de applicatie communiceren met elkaar. Bovendien bevat de virtuele omgeving ook geen gemeenschappelijke voorwerpen waar gebruikers mee kunnen interageren (op uitzondering van de avatars van de geconnecteerde gebruikers uiteraard). Het is overigens niet verwonderlijk dat de ontwikkelde applicatie zo eenvoudig gehouden werd. Het doel van de implementatie bestond immers enkel uit het demonstreren van het gebruik van skeletal animation in een networked virtual environment, en het bestuderen van de invloed hiervan op de gegenereerde netwerktrafiek.

Hoofdstuk 7

Conclusie

Eén van de hoofdbetrachtingen van NVE ontwerpers bestaat eruit gebruikers de illusie te geven dat ze zich daadwerkelijk in de aangeboden virtuele omgeving bevinden. Tot op heden hebben NVE ontwerpers zich hierbij voornamelijk geconcentreerd op het visuele communicatiekanaal. Er worden dan ook vaak kosten noch moeite gespaard om de aangeboden virtuele omgeving visueel zo realistisch mogelijk te laten ogen. Maar opdat dergelijke virtuele omgevingen hun geloofwaardigheid zouden behouden, moeten objecten in deze omgevingen eveneens op een realistische manier bewegen. Dit geldt in het bijzonder voor de virtuele belichamingen van geconnecteerde gebruikers. Belangrijk hierbij is dat de animaties voor deze belichamingen (en andere objecten in het algemeen) bovendien in real-time berekend moeten worden. Interactiviteit is immers één van de kenmerkende eigenschappen van een NVE systeem.

Dankzij de opkomst van goedkope grafische hardware heeft real-time character animation de afgelopen jaren een indrukwekkende vooruitgang gekend. Krachtige animatiebenaderingen wiens gebruik enkele jaren geleden nog beperkt was tot offline animatie, worden momenteel volop aangewend in real-time applicaties. Een voorbeeld van een dergelijke animatiebenadering is skeletal animation. Skeletal animation stelt een animator in staat de beweging van een complex model te controleren via een eenvoudige structuur, namelijk zijn onderliggend skelet.

Het doel van het implementatiegedeelte van deze thesis bestond uit het demonstreren van het gebruik van skeletal animation in een networked virtual environment. Deze implementatie heeft aan het licht gebracht dat een NVE ontwerper op het gebied van avatar animatie steeds een afweging zal moeten maken tussen processorbelasting enerzijds en gegenereerde netwerktrafiek anderzijds. Het aantal berekeningen dat remote clients moeten uitvoeren om een bepaalde avatar te ani-

meren, blijkt immers omgekeerd evenredig te zijn met de hoeveelheid animatiedata die doorgestuurd werd via het onderliggende computernetwerk. Dit wil zeggen dat een NVE ontwerper steeds moet trachten te achterhalen welke resource het meest kostbaar is (processortijd of netwerk bandbreedte), en hier vervolgens rekening mee moet houden wanneer hij de NVE daadwerkelijk implementeert.

Bovendien heeft het bestuderen van bestaande NVE systemen aangetoond dat de avatars in het merendeel van de huidige NVEs enkel voorgeprogrammeerde acties kunnen uitvoeren, en dat dynamische interactie tussen avatar en virtuele omgeving nog maar zeer zelden mogelijk is. De implementatie die in het kader van deze thesis uitgevoerd werd, maakte echter duidelijk dat het on the fly genereren van animaties via bijvoorbeeld inverse kinematics enorm interessante perspectieven biedt. Een dergelijke strategie zou bijvoorbeeld aangewend kunnen worden om avatars op een realistische manier objecten te laten oprapen in de virtuele omgeving. Dynamisch gecreëerde animaties introduceren echter ook een aantal problemen op het gebied van netwerk communicatie. Hoe kunnen dergelijke animaties bijvoorbeeld meegedeeld worden aan de andere gebruikers van de NVE? Indien ervoor geopteerd wordt enkel een high-level beschrijving van de animatie door te sturen over het computernetwerk, bestaat de mogelijkheid dat de gereconstrueerde animatie op verschillende remote hosts niet identiek is. Indien daarentegen een volledige beschrijving van de dynamisch gegenereerde animatie doorgestuurd wordt, bestaat de kans dat het netwerk te zwaar belast wordt. De toekomst zal moeten uitwijzen hoe dergelijke problemen het best geadresseerd kunnen worden.

Bibliografie

[1] Steve Benford, Chris Greenhalgh, Tom Rodden, James Pycock; *To what extent is cyberspace really a space? Collaborative Virtual Environments*; Communications of the ACM Volume 44, Issue 7, July 2001

[2] Pavel Curtis, David A. Nichols; *MUDs Grow Up: Social Virtual Reality in the Real World*; Proceedings of the 1994 IEEE Computer Conference (1994) 193-200

[3] Discworld MUD; http://discworld.imaginary.com:5678/

[4] Maja Matijasevic; *A Review of Networked Multi-User Virtual Environments*

[5] Tolga K. Capin, Maja Jovovic, Joaquim Esmerado, Amaury Aubel, Daniel Thalmann; *Tolga K. Capin, Maja Jovovic, Joaquim Esmerado, Amaury Aubel, Daniel Thalmann*; CA 1998: 41-48

[6] Anarchy Online; http://www.anarchy-online.com/ - Funcom

[7] Dark Age of Camelot; http://www.darkageofcamelot.com/ - Mythic Entertainment

[8] World of Warcraft; http://www.blizzard.com/wow/ - Blizzard Entertainment

[9] Star Wars Galaxies; http://starwarsgalaxies.station.sony.com/ - Sony & LucasArts Entertainment

[10] The Active Worlds Website, Home of the 3D Chat Virtual Reality Building Platform; http://www.activeworlds.com/ - Activeworlds Inc.

[11] Frederick P. Brooks, Jr.; *What's Real About Virtual Reality?*; IEEE Computer Graphics and Applications, volume 19, p. 16-27, 1999

[12] Yam San Chee; *Networked Virtual Environments for Collaborative Learning*; In Proceedings of ICCE/SchoolNet 2001Ninth International Conference on Computers in Education , Seoul, S. Korea, pp. 311. ICCE/SchoolNet 2001; http://www.comp.nus.edu.sg/labs/learning/cvisions.htm

[13] Sandeep Singhal, Michael Zyda; *Networked Virtual Environments: Design and Implementation*; Addison-Wesley Pub Co, 1999

[14] Andrew S. Tanenbaum; *Computer Networks - Third edition*; Prentice Hall International Editions, 1996

[15] Michael R. Macedonia, Michael J. Zyda; *A Taxonomy for Networked Virtual Environments*; IEEE Multimedia, 4(1):48–56, January - March 1997

[16] Denis Lukianov; *Advanced WinSock Multiplayer Game Programming: Multicasting*; http://www.gamedev.net/reference/articles/article1587.asp

[17] Thomas A. Funkhouser; *Network Topologies for Scalable Multi-User Virtual Environments*; IEEE VRAIS '96, San Jose, CA, April, 1996

[18] Thomas A. Funkhouser; *Network Services for Multi-User Virtual Environments*; IEEE Network Realities '95, Boston, MA, October, 1995

[19] Miguel Antunes, Antnio Rito Silva, Jorge Martins; *An Abstraction for Awareness Management in Collaborative Virtual Environments*; Proceedings of the ACM symposium on Virtual reality software and technology (November 2001)

[20] John W. Barrus, Richard C. Waters, David B. Anderson; *Locales and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments*; IEEE Computer Graphics and Applications, Vol. 16, No. 6, November 1996, pp. 50-57

[21] James Purbrick, Chris Greenhalgh; *Extending Locales: Awareness Management in MASSIVE-3*; Proceedings of Virtual Reality, Feb.2000

[22] Steve Benford, Lennart Fahlen; *A Spatial Model of Interaction in Large Virtual Environments*; Proceedings of the Third European Conference on CSCW (CSCW93), Milano, Italy, 1993, pp. 109-124

[23] Steve Benford, Chris Greenhalgh; *Introducing Third Party Objects into the Spatial Model of Interaction*; European Conference on Computer Supported Cooperative Work, Lancaster, UK, September, 1997, pp. 189-204.

[24] Steve Benford, Chris Greenhalgh, David Lloyd; *Crowded Collaborative Virtual Environments*; Proc. CHI97, Atlanta, Georgia, US, March 22-27, 1997

[25] Sandeep K. Singhal, David R. Cheriton; *Using Projection Aggregations to Support Scalability in Distributed Simulation*; In Proceedings of the 16 th International Conference on Distributed Systems (ICDCS), pp. 196-206, IEEE Computer Society, May 1996

[26] Wing Ho Leung, Tsuhan Chen; *Creating a Multiuser 3-D Virtual Environment*; IEEE Signal Processing Magazine, vol. 18, pp.9-16, May 2001

[27] Juan S. Casanueva, Edwin H. Blake; *Presence and Co-Presence in Collaborative Virtual Environments*;

[28] Juan S. Casanueva, Edwin H. Blake; *The Effects of Avatars on Co-presence in a Collaborative Virtual Environment*; Technical Report CS01-02-00, Department of Computer Science, University of Cape Town, South Africa, 2001

[29] Daniel Thalmann; *The Role of Virtual Humans in Virtual Environment Technology and Interfaces*; In Proceedings of Joint EC-NSF Advanced Research Workshop, Bonas, France, 1999

[30] S. Benford, J. Bowers, L. Fahlen, C. Greenhalgh, D Snowdon; *User Embodiment in Collaborative Virtual Environments*; In Proceedings of CHI95 New York, pages 242-249, 1995

[31] Tolga K. Capin, Igor S. Pandzic, Nadia Magnenat Thalmann, Daniel Thalmann; *Integration of Avatars and Autonomous Virtual Humans in Networked Virtual Environments*; Proc. ISCIS 98, Amsterdam, Netherlands, pp. 326 333, IOS Press, 1998

[32] Mel Slater, Martin Usoh; *Body Centred Interaction in Immersive Virtual Environments*; in N. Magnenat Thalmann and D. Thalmann (eds.), editors, Artificial Life and Virtual Reality, pages 125-148. John Wiley and Sons, 1994

[33] Tolga K. Capin, Igor S. Pandzic, Nadia Magnenat Thalmann, Daniel Thalmann; *Realistic Avatars and Autonomous Virtual Humans in VLNET Networked Virtual Environments*; Virtual Worlds in the Internet (R.Earnshaw and J. Vince, eds) IEEE Computer Society Press, pp.157-174, 1998

[34] Anthony Guye-Vuillme, Tolga K. Capin, Igor Sunday Pandzic, Nadia Magnenat Thalmann, Daniel Thalmann; *Nonverbal Communication Interface for Collaborative Virtual Environments*; Virtual Reality J., vol. 4, pp. 49-59, 1999. McBREEN AND JACK: EVALUATING HUMANOID SYNTHETIC AGENTS 405

[35] M. Fabri, D.J. Moore, D.J. Hobbs; *Expressive Agents: Non-verbal Communication in Collaborative Virtual Environments*; in Proceedings of Autonomous Agents and Multi-Agent Systems (Embodied Conversational Agents), July 2002, Bologna, Italy

[36] Dennis Burford, Edwin Blake; *Real-Time Facial Animation for Avatars in Collaborative Virtual Environments*; In South African Telecommunications Networks and Applications Conference '99, pages 178-183, 1999

[37] Tolga K. Capin, Maja J. Jovovic, Joaquim Esmerado, Amaury Aubel, Daniel Thalmann; *Efficient Network Transmission of Virtual Human Bodies*; Proc. Computer Animation '98, Philadelphia, IEEE Computer Society Press, pp. 41-48, 1998

[38] Tolga K. Capin, Joaquim Esmerado, Daniel Thalmann,; *A Dead-Reckoning Technique for Streaming Virtual Human Animation*; IEEE Transactions on Circuits and Systems for Video Technology, pp. 411-414, April 1999

[39] Nadia Magnenat-Thalmann, Chris Joslin; *The Evolution of Virtual Humans in NVE Systems*; ICAT2000, pp. 29, Oct. 2000

[40] Eike F. Anderson; *Real-Time Character Animation for Computer Games*; National Centre for Computer Animation, Bournemouth University, 2001

[41] Prem Kalra, Nadia Magnenat-Thalmann, Laurent Moccozet, Gael Sannier, Amaury Aubel, Daniel Thalmann; *Real-time animation of realistic virtual humans*; IEEE Computer Graphics and Applications, pages 42-56, Sept. 1998

[42] Thomas W. Sederberg, Scott R. Parry; *Free-Form Deformation of Solid Geometric Models*; ACM SIGGRAPH Computer Graphics, pp.151-160, August, 1986

[43] Thanh Giang, Robert Mooney, Christopher Peters, Carol O'Sullivan; *Real-Time Character Animation Techniques*; Technical Report TCD-CS-2000-06, February 2000

[44] Jeff Lander; *Graphic Content: Oh My God, I Inverted Kine!*; Game Developer, September 1998

[45] Jeff Lancer; *Graphic Content: Making Kine More Flexible*; Game Developer, November 1998

[46] Michael Putz, Klaus Hufnagl; *Character Animation for Real-time Applications*; Institute of Computer Graphics, Graz University of Technology, Austria; http://www.cg.tuwien.ac.at/studentwork/CESCG/CESCG-2002/MPutzKHufnagl/

[47] Jeff Lander; *On Creating Cool Real-Time 3D*; Gamasutra, October 1997; http://www.gamasutra.com/features/visual_arts/101797/rt3d_01.htm

[48] Chris Welman; *Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation*; Master's thesis, School of Computer Science, Simon Fraser University, 1989

[49] Richard C. Waters, David B. Anderson, John W. Barrus, David C. Brogan, Michael A. Casey, Stephan G. McKeown, Tohei Nitta, Ilene B. Sterns, William S. Yerazunis; *Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability*; Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4, pp. 461-480, August 1997

[50] David B. Anderson, John W. Barrus, John H. Howard, Charles Rich, Chia Shen, Richard C. Waters; *Building Multi-User Interactive Multimedia Environments at MERL*; IEEE Multimedia, Vol. 2, No. 4, pp. 77-82, Winter 1995

[51] Michael R. Macedonia, Michael J. Zyda, David R. Pratt, Donald P. Brutzman, Paul T. Barham; *Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments*; Proceedings of IEEE Virtual Reality Annual International Symposium, pp. 2-10, 1995

[52] Thomas A. Funkhouser; *RING: A Client-Server System for Multi-User Virtual Environments*; Proceedings of the 1995 symposium on Interactive 3D graphics, p.85-92, April 09-12, 1995, Monterey, California, United States

[53] The Dive Homepage; http://www.sics.se/dive/

[54] Emmanuel Frécon, Marten Stenius; *DIVE, A scaleable network architecture for distributed virtual environments*; Distributed Systems Engineering Journal, Vol. 5, No. 3, pp. 91-100, 1998

[55] Brett Porter; *NeHe Productions, Lesson 31: Model Rendering*; http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=31

[56] Brett Porter; *Real SOON Now Productions: Skeletal Animation Tutorial*; http://rsn.gamedev.net/tutorials/ms3danim.asp

[57] The MilkShape 3D homepage; http://www.swissquake.ch/chumbalum-soft/ - chUmbaLum sOft