Standaardisatie voor NVE toepassingen

Michele FUMAROLA

promotor :
Prof. dr. Wim LAMOTTE

co-promotor :
De heer Maarten WIJNANTS

Universiteit Maastricht

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica afstudeerrichting multimedia

universiteit hasselt

Voorwoord

In mijn voorwoord wil ik enkele mensen mijn dank betuigen voor hun geboden hulp en steun bij het tot stand komen van deze thesis. Op de eerste plaats wil ik mijn promotor, prof. dr. Wim Lamotte bedanken om mij de mogelijkheid te geven aan deze thesis te werken. Verder wil ik mijn begeleiders, Pieter Jorissen en Maarten Wijnants, bedanken voor hun advies, richtlijnen en het zorgvuldig nalezen van deze thesis. Dit resultaat zou niet mogelijk zijn geweest zonder hun hulp. Ik zou ook graag dr. Marius Preda willen bedanken voor het beschikbaar stellen van de MP4SDK en zijn uitleg hieromtrent. Verder wil Wesley De Neve bedanken voor zijn uitleg omtrent MPEG 21 DIA en het me mogelijk te maken de hierbij horende reference software te gebruiken.

Ik wil hierbij ook mijn ouders bedanken die me de mogelijkheid hebben gegeven deze opleiding te volgen en me het belang van de studies hebben doen inzien. Mijn zus zou ik ook willen bedanken voor haar hulp en steun. Uiteindelijk wil ik mijn vrienden bedanken, die voor de nodige afleiding hebben gezorgd tijdens het jaar.

Abstract

Networked Virtual Environments (NVE) kennen een steeds grotere populariteit. De virtuele omgevingen worden grafisch steeds aantrekkelijker en ondersteunen een groot aantal gebruikers. Sinds enkele jaren spreekt men zelfs van massive multiplayer online games om dit groot aantal gebruikers te benadrukken. Hierbij is het tegenwoordig ook noodzakelijk dat de gebruikers niet beperkt worden door bepaalde systeemeisen en zelfs niet door een bepaalde systeemarchitectuur. Zo is er steeds meer vraag naar NVE's die niet enkel met een normale desktopcomputer bezocht kunnen worden, maar ook met moderne PDA's.

Het spreekt voor zich dat achter dergelijke NVE's een complexe architectuur schuilt. Om een beter gestructureerd ontwerp te bekomen en om de ontwikkeling van een NVE te vereenvoudigen en te versnellen, kan men terugvallen op verschillende standaarden. Er zijn echter een zeer groot aantal standaarden en niet iedere standaard is even nuttig binnen de context van een NVE. Het doel van deze thesis is dan ook enkele van deze standaarden te bespreken die bruikbaar zijn binnen een NVE.

In deze thesis worden er verschillende soorten standaarden gepresenteerd. We beginnen met het bestuderen van standaarden ter beschrijving van 3D-scènes en gaan verder met recent ontwikkelde audio- en videostandaarden. We zullen uiteindelijk MPEG 21 bestuderen en ons verder concentreren op het specifiek deel van deze standaard die het mogelijk maakt om schaalbaarheid te bereiken voor verschillende type binaire bestanden.

Aan de hand van een implementatie zullen we zien dat de besproken standaarden de ontwikkeling van een NVE niet alleen vergemakkelijken maar ook mogelijkheden bieden om een NVE aan te passen aan de noden van de gebruikers. Enkele resultaten zullen verder aantonen dat het gebruik van deze standaarden het beoogd doel bereiken.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van codefragmenten

Hoofdstuk 1

Inleiding

Genetwerkte virtuele omgevingen komen in verschillende vormen en maten voor. Ze worden als hulpmiddel gebruikt om taken uit te voeren tussen gebruikers die zich op grotere afstanden van mekaar bevinden, om simulaties uit te voeren voor het leger of, wel eens vaker, als puur recreatief middel. Ongeacht hun toepassingsgebied moeten de ontwerpers van een genetwerkte virtuele omgeving een groot aantal obstakels overbruggen om hun doel te behalen.

Steeds vaker worden vaak voorkomende technieken en architecturen gestandaardiseerd zodat deze op een betere manier kunnen worden gebruikt in bepaalde toepassingen. Ook voor genetwerkte virtuele omgevingen kunnen bepaalde standaarden van belang zijn. Dit omdat een standaard al een solide basis vormt en de goede werking ervan voldoende is bewezen. Verder kan een ontwikkelaar rekenen op reeds bestaande implementaties van een groot aantal standaarden.

Het doel van deze thesis is net een aantal van deze standaarden bespreken die structuur kunnen brengen in het ontwerp van een NVE en mogelijkheden bieden die de gebruikerservaring verbeteren. De standaarden die in deze thesis zullen worden besproken, zorgen voor beschrijvingen van een scènes met het oog op het efficiënt opslaan van deze beschrijvingen zodat deze sneller verstuurd kunnen worden over een netwerk. We zullen ook standaarden bespreken die audio- en videomogelijkheden bieden. Omdat schaalbaarheid steeds belangrijker wordt in het ontwerp van genetwerkte virtuele omgevingen, zullen we ook de MPEG 21-standaard bekijken die net dit mogelijk kan maken.

De inhoud van deze thesis bestaat uit zeven hoofdstukken waarvan het eerste deze inleiding is. In het tweede hoofdstuk zullen we NVE's meer in detail bespreken. Het derde hoofdstuk bespreekt de VRML, X3D en BIFS-standaard: standaarden die het mogelijk maken om 3D-werelden te beschrijven. We gaan met hoofdstuk vier verder en bespreken de verschillende audio- en videostandaarden die de laatste jaren zijn ontwikkeld. Hoofdstuk vijf zal de MPEG 21-standaard beschrijven die, zoals eerder gezegd, onder andere schaalbaarheid mogelijk maakt. Hierna zal een hoofdstuk volgen dat de implementatie bespreekt horende bij deze thesis. Voor deze implementatie werd er gekozen om enkele van de besproken standaarden te gebruiken in de context van een NVE. Uiteindelijk zal hierop een hoofdstuk volgen met een algemene conclusie over de gehele thesis.

Hoofdstuk 2

Genetwerkte Virtuele Omgevingen

2.1 Inleiding

Een genetwerkte virtuele omgeving (networked virtual environment oftewel NVE) wordt in de literatuur als volgt gedefinieerd [1]:

> A Networked Virtual Environment is a software system in which multiple users interact with each other in real-time, even though those users may be located around the world.

Zoals bovenstaande definitie aangeeft, maakt een NVE het mogelijk om verschillende gebruikers te verzamelen in één gemeenschappelijke omgeving en dit zonder geografische beperkingen. Hierbij kunnen de gebruikers de aanwezigheid waarnemen van andere gebruikers in deze omgeving. Alhoewel dit voornamelijk op een visuele manier gebeurt, wordt dit in een groot aantal NVE's ook mogelijk gemaakt op een auditieve manier zodat gebruikers elkaar kunnen horen. Communicatie is dus een fundamenteel aspect van een NVE.

Een NVE bestaat uit verschillende componenten die ieder verantwoordelijk zijn voor een essentieel deel van een NVE. Moderne NVE's hebben vrijwel allemaal een grafische component die zorgt voor de visualisatie van de virtuele omgeving. Hiertoe behoort ook het animeren van de verschillende objecten, zoals avatars, binnen deze omgeving. Ook maakt men veelvuldig gebruik van geluid om het gevoel van immersie binnen de virtuele omgeving te verhogen. Het belangrijkste aspect is de netwerkcomponent die in de volgende secties meer gedetailleerd beschreven zal worden. In sectie 2.2 zullen we de verschillende netwerkaspecten bespreken en, aangezien er talrijke beperkende

factoren zijn, bespreken we ook optimalisatietechnieken in sectie 2.3. Uiteindelijk zal in sectie 2.4 een bespreking volgen van enkele bestaande NVE's, waarbij we zullen merken dat er talrijke NVE's bestaan en dit in verschillende gebieden.

2.2 Netwerk

2.2.1 Algemeen

Het netwerkaspect is duidelijk een basiselement van een NVE en hieraan moet voldoende aandacht worden geschonken bij de ontwikkeling van een dergelijke omgeving. Hierbij moet er worden bepaald welke netwerktopologie er gebruikt gaat worden, wat vaak gekoppeld gaat met een bepaalde wijze van distributie. Uiteindelijk moet er rekening worden gehouden met aspecten zoals *lag* en *jitter*: twee termen die steeds terugkomen zodra men met netwerken te maken heeft. Hier betekent lag de tijd nodig om een bepaald pakket over te brengen en te verwerken, en jitter duidt op de variatie in lag-waarde.

In de volgende paragrafen zullen de belangrijkste onderdelen aan bod komen. We zullen beginnen met de protocollen, vervolgens de verschillende wijzen van distributie bespreken en de verschillende topologieën.

2.2.2 Netwerkprotocollen

Het netwerkprotocol [2] zorgt voor het overbrengen van de data nodig in een NVE. De belangrijkste protocollen die men kan gebruiken in een NVE zijn ongetwijfeld het Transmission Control Protocol (TCP) en het User Datagram Protocol (UDP).

TCP is ontwikkeld om voor een betrouwbare verbinding te zorgen. Het protocol verzekert dat de verstuurde pakketten daadwerkelijk aankomen en dit op de juiste volgorde. Dit zorgt ervoor dat de gebruikte headers relatief groot zijn omdat ze extra informatie zoals volgnummer en checksum bevatten waarmee de data kan worden gecontroleerd op fouten. De pakketten kunnen ook een bepaalde vertraging oplopen omdat de juiste volgorde van aankomst gegarandeerd moet worden.

UDP is een onbetrouwbaar protocol. Er wordt geen controle uitgevoerd op aankomst en volgorde van de verstuurde pakketten. Gevolg hiervan is

wel dat de pakketten een relatief kleine header hebben, daar ze geen extra informatie nodig hebben, en geen last hebben van vertragingen als gevolg van het protocol zelf.

In een NVE is het echter niet noodzakelijk dat men strikt een keuze maakt tussen deze twee protocollen: door de voor- en nadelen van ieder protocol kan men met het gebruik van beide in dezelfde applicatie een optimale oplossing bekomen. Zo kan men minder belangrijke pakketten versturen met behulp van UDP terwijl men essentiële pakketten verstuurd met TCP omdat men zeker wil zijn dat deze aankomen.

2.2.3 Distributie

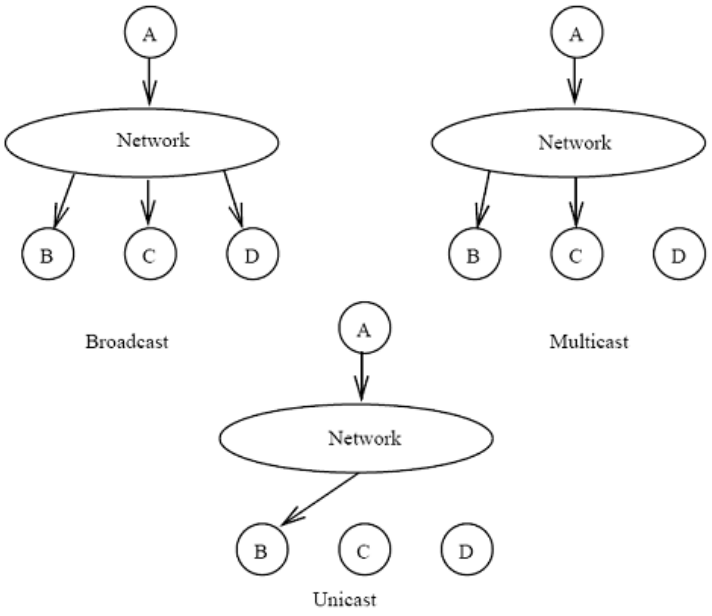
Distributie slaat op de wijze waarop men data naar verschillende gebruikers kan sturen. Dit kan men bekomen op drie manieren [3]:

- broadcast: er is slechts één communicatiekanaal waarop iedereen is verbonden. Eens men een pakket heeft ontvangen, zal men aan de hand van de pakketheader bepalen of deze voor de huidige gebruiker bestemd is of niet.
- multicast: er worden groepen gevormd waarop gebruikers zich abonneren. Een pakket zal hierna gestuurd worden naar iedere gebruiker binnen deze groep.
- unicast: er wordt een directe verbinding opgesteld tussen twee gebruikers waarover de gewenste pakketten worden verzonden.

De distributiemethoden hier besproken worden schematisch weergegeven in figuur 2.1.

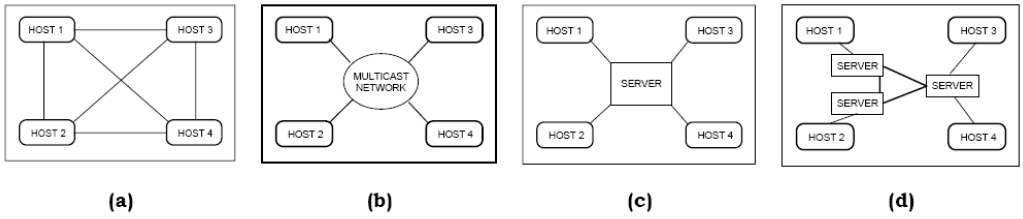
2.2.4 Topologie

Een topologie [4] verzorgt de verbindingen tussen de gebruikers op een dergelijke manier dat iedere gebruiker steeds een update krijgt van welke veranderingen er hebben plaatsgevonden in de virtuele omgeving. Bij de keuze van een topologie is het belangrijk om te bepalen voor hoeveel gebruikers ondersteuning moet worden geboden en welke updates belangrijk zijn voor deze gebruikers.



Figuur 2.1: Distributiemogelijkheden [3]

In figuur 2.2 ziet men de verschillende topologieën mogelijk in een NVE. Hier stelt figuur (a) de unicast-variant voor van de peer-to-peer topologie, figuur (b) de multicast-variant van de vorige topologie, figuur (c) de client-server topologie en uiteindelijk figuur (d) de multi-server topologie.



Figuur 2.2: Verschillende topologieën mogelijk in een NVE [4]

De unicast-variant van de peer-to-peer topologie verbindt iedere gebruiker onderling met behulp van een unicast-connectie (zijnde TCP of UDP) waarbij iedere gebruiker dus steeds een update-bericht zal sturen naar de overige gebruikers. Als de NVE dus N gebruikers telt, zal iedere gebruiker

bij iedere update N-1 berichten moeten sturen over de N-1 verbindingen. De multicast-variant van de peer-to-peer topologie verbetert deze topologie door het mogelijk te maken dat iedere gebruiker slechts één bericht moet sturen voor iedere update.

De client-server topologie maakt gebruik van één centrale server waarmee iedere gebruiker is verbonden. Een gebruiker heeft dus slechts één verbinding met de server en iedere update zal dus naar deze server gestuurd worden. Aangezien de server over verbindingen beschikt met iedere gebruiker in de virtuele omgeving, zullen de ontvangen updates verspreid worden over al de gebruikers. Een uitbreiding hierop is de multi-server topologie waarbij men gebruikt maakt van meerdere servers die met elkaar verbonden zijn. Op deze manier wordt de belasting, die er zou zijn voor één server, verdeeld over verschillende servers. Doordat deze servers onderling verbonden zijn, zullen ook de updates die iedere server aankrijgt, verdeeld worden over de overige servers. Deze zullen op hun beurt de updates verdelen over hun gebruikers.

2.3 Optimalisaties

Bij een groot aantal gebruikers in een NVE is het vaak een moeilijke taak om deze te beheren. Het doorlopend versturen van alle updates naar iedere gebruiker, kan het netwerk zwaar belasten als het een grote virtuele omgeving is met een groot aantal gebruikers. Hierdoor is het nodig om technieken en structuren te ontwikkelen die ervoor zorgen dat het netwerkgebruik drastisch verlaagt. We zullen hier een aantal technieken bespreken die vaak gebruikt worden in bestaande NVE's en hun goede werking al voldoende hebben bewezen. De bespreking zal beginnen met "Dead Reckoning", gevolgd door "Area of Interest Management" en eindigen met "Compressie en Aggregatie van Pakketten".

2.3.1 Dead reckoning

Dead reckoning [5] is een techniek die het mogelijk maakt om minder frequent update-pakketten te sturen van een bepaald object. Dit betekent dat de update-informatie waarover men niet beschikt op een relatief betrouwbare manier gaat worden ingeschat. Het dead reckoning-algoritme bestaat uit twee componenten: de component verantwoordelijk voor de voorspelling en de component verantwoordelijk voor de convergentie.

De voorspelling van een bepaalde update gebeurt aan de hand van polynomen en hun afleidingen. In deze context spreekt men van snelheid en versnelling. De polynoom geeft initieel enkel de positie weer van het object. Als men de eerste afgeleide neemt, bekomt men de snelheid terwijl de tweede afgeleide de versnelling geeft. Door nu gebruik te maken van de snelheid en versnelling, kan men voorspellen waar het object zich zal bevinden na een bepaald tijdsinterval. Deze afgeleiden zijn het meest voorkomend bij het gebruik van dead reckoning. De reden hiervoor is dat de afgeleide van een graad groter dan twee, meer onnauwkeurigheden gaan vertonen en het berekenen hiervan langer kan duren.

Eens er, na een bepaald interval, een updatebericht wordt ontvangen, kan het zijn dat de voorspelling enkele fouten bevat. Om dit op te vangen, wordt er gebruik gemaakt van convergentietechnieken om de positie van een object te verbeteren. Men kan eenvoudigweg de positie in één keer veranderen naar de nieuwe positie, maar dit zorgt voor een plotselinge overgang, wat onrealistisch is en dus een negatieve invloed heeft op de immersie. Om dit te verbeteren kan men gebruik maken van polynomen die vertrekkende vanuit de huidige positie, gaan naar een nieuwe positie die berekend is aan de hand van het updatebericht. Deze polynomen kunnen 1ste-orde polynomen zijn waardoor de convergentie gebeurt in een rechte lijn ofwel 2de-orde polynomen waardoor convergentie gebeurt in een meer natuurlijke curve.

Het bepalen van een fout in predictie kan op verschillende manieren gebeuren. Een mogelijkheid is om hetzelfde algoritme die de nieuwe updates berekent, op de zender van de updates ook uit te voeren. Op deze manier kan de zender bepalen wanneer er een te grote afwijking plaatsvindt en deze dus verbeteren door een updatepakket te versturen. Een andere mogelijkheid is om de zender op vaste tijdsstippen updatepakketen te laten versturen zodat de ontvanger hiervan, indien nodig, de verbeteringen kan uitvoeren.

2.3.2 Area of Interest Management

Het is niet altijd nodig om al de updatesberichten naar iedere gebruiker van een NVE te versturen. Als een bepaalde gebruiker niet zichtbaar is voor de huidige gebruiker, is het in veel gevallen nutteloos dat de huidige gebruiker updateberichten hiervan ontvangt en verwerkt. Gebruikers kunnen elkaar niet zien als zij zich in verschillende ruimten bevinden of als ze te ver verwijderd zijn van elkaar. Een algoritme dat nagaat welke gebruikers elkaar wel en niet kunnen zien, maakt gebruikt van Area of Interest (AOI) [5]: de

zone horende bij een gebruiker waarin deze is geïnteresseerd.

De AOI van een gebruiker kan aangegeven worden door een aura. Als aura's van twee gebruikers elkaar intersecteren, zullen deze gebruikers er belang in hebben om updateberichten van mekaar te ontvangen. Een betere indeling, ook wel spatial model of interaction genoemd, gebeurt aan de hand van focus en nimbus: focus geeft de perceptie aan van de huidige gebruiker en nimbus geeft aan door welke gebruikers de huidige gebruiker wordt waargenomen. Het gevolg hiervan is dat er pas updateberichten worden uitgewisseld zodra de focus van één gebruiker intersecteert met de nimbus van een andere gebruiker.

Een derde mogelijkheid is om gebruik te maken van zoning [1]. Hierbij wordt de virtuele omgeving in cellen ingedeeld en kan er op een relatief eenvoudige manier worden bepaald welke gebruikers updateberichten van elkaar moeten ontvangen. Dit wordt namelijk bepaald aan de hand van de cellen: als twee gebruikers zich in dezelfde cel bevinden, zullen ze updateberichten uitwisselen.

2.3.3 Compressie en Aggregatie van Pakketten

Optimalisaties zijn ook mogelijk op de pakketten die verstuurd worden over het netwerk naar de verschillende gebruikers. Hierbij hoort compressie en aggregatie van pakketten. [5]

Met behulp van compressie-algoritmen kan men de omvang van een pakket verkleinen waardoor het pakket sneller over een netwerk verstuurd kan worden. Compressie-algoritmen kan men in twee categorieën groeperen: enerzijds lossless of lossy, anderzijds internal of external compressie.

Lossless compressie betekent dat niks verloren gaat uit het origineel pakket en deze in zijn originele staat kan worden teruggebracht. Afhankelijk van het compressie-algoritme, kan deze vorm van compressie de omvang van een pakket sterk reduceren en dit vaak tot de helft van de originele grootte. Lossy compressie betekent dat men bepaalde informatie verkiest om te verwijderen, meestal minder relevante informatie. Op deze manier zal na reconstructie het pakket niet volledig overeenkomen met het origineel pakket, maar het verschil zal niet snel opgemerkt worden. Uiteindelijk valt op te merken dat lossy compressie veel betere compressie toelaat dan lossless compressie.

Internal of external compressie slaat op algoritmen die wel of geen gebruik maken van informatie uit overige pakketten. Internal compressie maakt geen gebruik van informatie uit overige pakketten en een dergelijk pakket is dus onafhankelijk gecomprimeerd van overige pakketten. External compressie maakt wel gebruik van overige pakketten waardoor er slechts het verschil moet worden beschreven met voorgaande pakketten en dus minder informatie moet worden doorgestuurd. Hierbij denkt men aan het doorsturen van posities waardoor men één pakket kan sturen met de volledige positiewaarden en waarbij de volgende pakketten enkel de veranderingen bevatten.

Aggregatie van pakketten betekent dat men, indien mogelijk, meerdere pakketten gaat samenvoegen en deze als één pakket zal doorsturen. Op deze manier bespaart men de overhead die komt kijken als men een pakket verstuurt: namelijk de header van een pakket, zijnde 28 bytes voor een UDP-pakket en 40 bytes voor een TCP-pakket. Algoritmen die bepalen hoe deze aggregatie moet gebeuren baseren zich op time-outs of het behalen van een quorum. Op time-outs gebaseerde algoritmen aggregeren ieder pakket dat binnen een bepaald tijdsinterval moet worden verstuurd. Op het einde van het interval zal men een groep pakketten bekomen die men dan in één keer gaat versturen. Algoritmen die gebruik maken van een quorum, wachten totdat er een bepaald aantal pakketten gegroepeerd zijn en versturen deze pas als het vooraf bepaalde aantal bereikt wordt. Omdat beide aanpakken kunnen leiden tot een niet gewenste vertraging bij het versturen van pakketten, kan men een combinatie van deze aggregatie-algoritmen gebruiken waardoor de nadelen deels wegvallen.

2.4 Bestaande NVE's

2.4.1 Academische NVE's

In de academische wereld werden er verschillende NVE's ontwikkeld waarin men nieuwe technieken kan terugvinden die nuttig zijn voor NVE's. In de volgende paragrafen worden er verschillende populaire NVE's besproken, maar door het groot aantal bestaande academische NVE's, is deze opsomming niet volledig.

Naval Postgraduate School Networked Vehicle Simulator (NPSNET) [6] is de eerste NVE die het gebruik over de Multicast Backbone mogelijk maakte. Er zijn verscheidene objecten in de virtuele omgeving van NPSNET: menselijke avatars en verschillende voertuigen voor land, lucht en zee. Communicatie

gebeurt op basis van het Distributed Interactive Simulation-protocol (DIS). Dit protocol is ontwikkeld voor gebruik in gesimuleerde veldslagen waarbij de positie en andere informatie van de verschillende gebruikers worden doorgestuurd.

Scalable Platform for Large Interactive Networked Environments (Spline) [6] is ontwikkeld door Mitsubishi Electric Research Laboratory en is een platform ter ontwikkeling van NVE's. In Spline wordt alles voorgesteld als een object en voor ieder object wordt bijgehouden waar deze zich bevindt, hoe deze eruit ziet en andere nuttige informatie. Deze objecten kunnen enkel gewijzigd worden door gebruikers die hiervoor de nodige rechten hebben. Rechten kunnen hierna overgedragen worden aan andere gebruikers. Iedere actieve applicatie beschikt over een kopie van de data waaruit de wereld is opgesteld en de veranderingen worden met behulp van berichten overgedragen naar andere applicaties. Om een groot aantal gebruikers aan te kunnen wordt er geen eis gesteld dat de werelden identiek gelijk zijn over de verschillende applicaties en wordt er gewerkt met locales [7]. Een locale komt overeen met een bepaald gebied van een virtuele omgeving. Voor ieder van deze gebieden wordt er gebruik gemaakt van een lokaal coördinatensysteem dat tevens het belangrijkst punt vormt van het gebruik van locales. Hierbij vormt men een virtuele omgeving door ieder van deze locales te linken en voor iedere link een lineaire transformatie te bepalen waarmee men kan omschakelen naar een ander coördinatensysteem. Hiermee houdt men voor iedere gebruiker enkel rekening met wat er gebeurt binnen zijn eigen locale.

The Model, Architecture and System for Spatial Interaction in Virtual Environments (MASSIVE) [6] is een NVE ontwikkeld aan de University of Nottingham. Van deze NVE werden drie versies ontwikkeld met ieder belangrijke toevoegingen. MASSIVE 1 maakt gebruik van aura en awareness om communicatie te beperken tot gebruikers die geïnteresseerd zijn in bepaalde data. Men baseert zich op een grootorde voor awareness: 0.0 betekent dat er niets is en 1.0 is het maximum waarbij men de hoogste kwaliteit kan halen op vlak van geluid en video maar ook het hoogste detailniveau voor de voorstelling van de virtuele omgeving. MASSIVE 2 voert een abstractie toe aan dit concept door voor iedere kamer een afzonderlijk multicast-kanaal op te zetten: op deze manier zullen gebruikers enkel data toegestuurd krijgen van gebruikers die zich in dezelfde omgeving bevinden. Uiteindelijk biedt MASSIVE 3 [8] de mogelijkheid om gebruik te maken van locales, die we eerder zijn tegengekomen bij Spline.

De Distributed Interactive Virtual Environment (DIVE) [6] is een NVE-platform ontwikkeld door de Swedish Institute of Computer Science. Het is een volledig gedistribueerd platform waarbij ieder proces op een node actief is die met elkaar verbonden zijn via een LAN of WAN. Ieder proces houdt ook een kopie bij van de wereld waartoe objecten behoren die aangemaakt, gewijzigd en verwijderd kunnen worden. Deze objecten kunnen steeds door hoogstens één gebruiker tegelijk worden aangepast. De gebruikte netwerktopologie is een peer-to-peer architectuur waarbij men steeds berichten stuurt bij wijzigingen.

Virtual Park (VPARK) [9] is een NVE waarbij het de bedoeling is om gebruik te maken van attracties zoals men zou tegenkomen in een pretpark of kermis. VPARK bestaat uit twee componenten: W-VLNET Networked Virtual Environment System en Attraction Builder System. Het eerste heeft de taak om de NVE-aspecten af te handelen terwijl het tweede dient om attracties te ontwikkelen. Om schaalbaarheid van de netwerktopologie te bekomen werd er gebruik gemaakt van de unicast client-server architectuur in combinatie met een multiple server-architectuur. De verschillende servers communiceren met elkaar met behulp van multicast-verbindingen. Het gebruikte protocol voor de communicatie is W-VLNET Network Protocol wat gebaseerd is op het UDP-protocol. De toevoegingen die hierbij horen zijn een mogelijkheid tot identificatie van de inhoud van een pakket, identificatie van de bron, time-stamps voor de pakketten en detecteren van verloren pakketten. Uiteindelijk wordt met behulp van dit protocol ook feedback gegeven aan de server.

2.4.2 Commerciële NVE's

NVE's worden niet enkel gebruikt voor academische doeleinden maar worden steeds populairder als computerspellen en virtuele gemeenschappen waarvan de gebruikers, tegen betaling, gebruik kunnen maken. In veel gevallen zal men niet enkel de mogelijkheid krijgen om met overige gebruikers te communiceren, maar ook om opdrachten samen uit te voeren of spellen te spelen. Verschillende voorbeelden van dergelijke NVE's zullen in de volgende paragrafen worden besproken.

NVE's komen voor in verschillende vormen. Een groot deel spellen waar een netwerkcomponent aanwezig is, kan men scharen onder de noemer NVE. Zo bestaan er actiespellen, de zogenaamde *first person shooters*, waarbij er vaak tot 32 spelers aanwezig zijn in een gemeenschappelijke omgeving. Er

is een enorm aantal spellen die aan voorgaande beschrijving voldoet waarbij bekende voorbeelden de verschillende Quake- en Unrealversies [10][11] zijn. Een opkomende trend zijn de *massively multiplayer online role-playing game* (MMORPG) [12] waarbij een groot aantal gebruikers samenkomen in een (fantasie-) wereld en die een rol aannemen van een bepaald karakter. Bekende voorbeelden hiervan zijn World of Warcraft [13] en EverQuest [14].

Maar NVE's komen ook vaak voor als virtuele gemeenschappen waarbij de nadruk eerder ligt op het sociaal aspect en minder op het vervullen van een bepaalde taak. Second Life [15] is een virtuele gemeenschap die werd gelanceerd in 2003 en tegenwoordig ongeveer 100.000 leden kent. Het biedt een enorm uitgebreide wereld met een gebied geschat op 20.000 virtuele aren. Het doel van de gemeenschap is even gevarieerd, gaande van het spelen van spellen tot het communiceren en dagdagelijkse activiteiten uitvoeren met andere leden.

Om de enorm complexe wereld van Second Life te beheren, werden er nieuwe technieken ontwikkeld [16]. De wereld wordt ingedeeld in vakken die dus een bijhorende grid vormen. Voor ieder element van de grid, dus een bepaald gebied van de wereld, is er een verantwoordelijke server die zorgt voor de nodige physics-berekeningen en scriptsuitvoeringen. De verschillende servers die instaan voor ieder gridelement, zijn ook onderling verbonden waarbij er directe connecties bestaan tussen buren. Op deze manier kan een gebruiker door de wereld navigeren en zullen zijn scripts, objecten en overige bezittingen op een relatief eenvoudige manier worden gekopieerd.

Een ander voorbeeld van een populaire virtuele gemeenschap is There [17]. Net zoals Second Life biedt There een uitgestrekte wereld waarin men met andere gebruikers kan communiceren en activiteiten ondernemen. Men kan bijvoorbeeld naar muziek luisteren en door de wereld navigeren met een voertuig. Dergelijke NVE's zijn niet beperkt tot de twee genoemde voorbeelden. Andere voorbeelden zijn CyberTown [18] en Active Worlds [19].

Uiteindelijk bestaat er nog een opensource alternatief dat wel gratis is en dus niet commerciëel van aard genaamd Solipsis [20]. Deze NVE is volkomen gebaseerd op een peer-to-peer architectuur waarbij er enkel gebruik wordt gemaakt van de clientmachines. De structuur bestaat uit twee basiscomponenten: de *node* en de *navigator*. Een node zorgt voor de implementatie van het gebruikte protocol en de connectie met andere gebruikers. Een navigator verzorgt een bruikbare interface voor de node: hiermee kan de gebruiker

de verschillende activiteiten ondernemen. De navigator zorgt tevens voor een pluginsysteem waarmee de gebruiker specifieke taken kan uitvoeren zoals videoconferencing en filesharing.

2.5 Conclusie

Bij het ontwikkelen van een NVE moet er altijd rekening gehouden worden met het consistent houden van de wereld voor de verschillende gebruikers alsook met het feit dat men moet werken met een netwerk wat vaak een beperkende factor kan zijn. De hier gepresenteerde technieken trachten steeds verbeteringen te brengen aan de architectuur en werking van een NVE. Het resultaat hiervan is dat de moderne NVE's een enorm aantal gebruikers kunnen samenbrengen in één grote virtuele omgeving. De steeds stijgende populariteit van NVE's zorgt voor een snelle ontwikkeling in dit domein. Dit vertaalt zich in steeds meer applicaties met een gemeenschappelijk doel: het samenbrengen van gebruikers.

In dit hoofdstuk hebben we de verschillende protocollen, distributiewijzen en topologieën besproken die in een NVE worden gebruikt. Hiervan hebben we steeds besproken welke voordelen worden geboden en hoe deze worden gebruikt. Hierna hebben we verschillende optimalisatietechnieken bekeken die als doel hebben het netwerktraffiek te verminderen. Tot slot hebben we bestaande NVE's besproken die voorkomen in de academische en commerciële wereld.

Hoofdstuk 3

Standaarden ter beschrijving van 3D-scènes

3.1 Inleiding

Een virtuele wereld kan bestaan uit een groot aantal verschillende objecten. Om een bepaalde samenhang te vormen voor deze scènes, werden er verschillende standaarden ontwikkelen die hierbij uitkomst kunnen bieden. Veelal werd een scène tekstueel beschreven en opgeslagen: hiervan zijn VRML en de opvolger X3D een voorbeeld. In de MPEG4-standaard wordt er echter vertrekkende vanuit VRML BiFS ontwikkeld: de beschrijvingen van de scènes worden niet langer tekstueel opgeslagen, maar binair, wat een aanzienlijke besparing vormt qua opslagruimte ten opzichte van de andere twee varianten. Deze besparing komt natuurlijk ook ten goede als een dergelijke scène wordt verstuurd over een netwerk, wat uiteindelijk belangrijk is voor een NVE. Voor een groot aantal NVE's wordt de beschrijving van de omgeving en de verschillende objecten immers pas naar de client verstuurd op het moment dat hij inlogt. Het gebruik van een standaard ter beschrijving van deze omgevingen, kan dus van groot belang zijn.

Een dergelijke standaard heeft nog bijkomende voordelen. Met behulp van bestaande SDK's kan men dezelfde standaard in verschillende applicaties gebruiken en is het niet nodig dat een ontwikkelaar steeds opnieuw een formaat ontwikkeld waarmee men een scène kan beschrijven. Naargelang de populariteit van een standaard, zullen er ook modelleerpakketten worden ontwikkeld die het makkelijker maken om een scène te beschrijven.

In dit hoofdstuk zullen de drie varianten worden beschreven waarbij van elke variant ook de voordelen zullen worden gegeven die deze drie ten opzichte van elkaar hebben. Er zal aanzienlijk meer aandacht geschonken worden aan BiFS van MPEG 4 daar deze standaard een groot aantal interessante mogelijkheden bevat en omdat deze steeds meer in gebruik wordt genomen voor verschillende applicaties.

3.2 VRML

Virtual Reality Modeling Language (VRML) [21] stamt van 1994, waarbij de versie van 1997 de laatste versie van de standaard is en VRML97 wordt genoemd. VRML is een tekstueel gebaseerd formaat waarmee men modellen en werelden kan construeren in een 3D-ruimte. VRML kan als de grondlegger worden bekeken van de markuptalen voor het beschrijven van 3D-werelden. Verschillende talen en standaarden, zoals X3D en BiFS die later aan bod zullen komen, baseren zich allemaal op VRML.

Objecten en werelden kunnen bestaan uit tekst, punten, lijnen en polygonen. Voor de objecten kan men verder materiaaleigenschappen definiëren zodat deze op een correcte wijze worden weergegeven onder bepaalde belichtingen. Uiteindelijk kan men ook textures gebruiken op de gecreëerde objecten.

Met behulp van groepen van objecten kan men een boom opstellen die dan op een gestructureerde manier de scène beschrijft. Objecten kunnen, na een eerste definitie, meerdere keren worden gebruikt en verder worden getransleerd en geroteerd. Verder kan men ook voor ieder object zogenaamde events bepalen waarna een voorgedefinieerde actie wordt uitgevoerd. Deze actie kan in Java of JavaScript worden bepaald. Uiteindelijk kan men ook geluid toevoegen aan de scène en animaties bepalen voor objecten door ze bijvoorbeeld een bepaald pad te laten volgen.

3.3 X3D

Extensible 3D (X3D) [22] kan gezien worden als de opvolger van VRML. Een beduidende uitbreiding van het aantal beschikbare nodes en een overschakeling naar XML, zorgen echter dat de verschillen duidelijk zichtbaar zijn.

Zoals eerder vermeld, werkt X3D verder op VRML en dit weerspiegelt zich in de structuur van de standaard. X3D biedt hoofdzakelijk volgende mogelijkheden [23]:

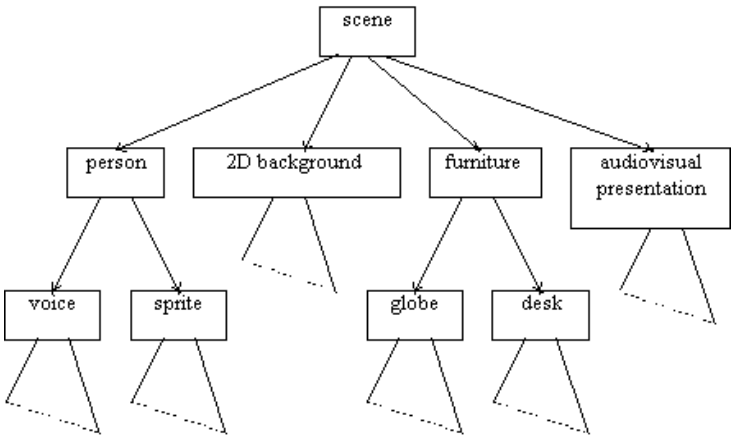
- 3D graphics: zoals polygonen, elementaire objecten, belichting, etc
- 2D graphics: tekst, 2D-figuren, etc
- animatie: met behulp van timers en interpolators de 2D en 3D-objecten animeren
- omgevingsgeluid
- interactie met objecten en navigatie
- de scène dynamisch aanpassen met behulp van scripting
- de verschillende data nodig om een scène op te bouwen, kan via een netwerk worden opgehaald
- fysische simulatie: animatie van een mens, gebruik van geospatiale datasets, etc

Aangezien X3D een groot aantal nodes beschikbaar stelt, wordt er gebruik gemaakt van profiles om enkel subsets van deze nodes te gebruiken. Zo biedt de core profile enkel een klein aantal nodes aan die tevens de basisnodes vormen van de X3D-standaard. Verder zijn er profiles zoals de interchange, interactive en navigation profiles, die steeds die nodes aanbieden beschreven in de naam van de profile.

3.4 BiFS

3.4.1 Algemeen

Binary Format for Scenes (BiFS) wordt beschreven in Part 10 [24] van de MPEG-4-standaard. Net zoals VRML, beschrijft BiFS een scène aan de hand van een hiërarchische structuur. Deze structuur is weergegeven in figuur 3.1. Hierbij moet er rekening gehouden worden met de attributen van de elementen en zelfs met het feit dat de elementen niet statisch zijn: ze kunnen op bepaalde manieren aangepast worden. Hoe dit gebeurt zal later aan bod komen.



Figuur 3.1: Hiërarchische structuur van BiFS [24]

Hoewel BiFS gebaseerd is op VRML, biedt BiFS buiten een uitbreiding van mogelijke knopen [25], enkele bijzondere voordelen. Ten eerste houdt, zoals de naam doet vermoeden, BiFS de scène niet langer bij in puur tekstueel formaat, maar in binaire vorm. Dit heeft natuurlijk het voordeel dat de beschrijving van een scène in grootte afneemt ten opzichte van de tekstuele vorm. Ten tweede is het ook niet langer nodig dat de gehele beschrijving lokaal aanwezig is: het is dus mogelijk om een scène te streamen. Tenslotte kan men in een scène een mix bekomen van 2D en 3D-data. Bij dit laatste wordt er rekening gehouden dat men alle mediastromen afzonderlijke encodeert zodat men voor iedere soort media een efficiënte encodering bekomt. Daar dit gebeurt binnen de MPEG 4-standaard, waar er specifieke encoderingen bestaan voor video en geluid, bestaat een dergelijke mogelijkheden niet voor VRML of X3D.

3.4.2 Structuur

Zoals eerder vermeld, worden scènes bijgehouden in binaire vorm. Het creëren van de beschrijvingen kan echter handmatig gebeuren waarna men deze omzet naar de uiteindelijke vorm. In de volgende paragrafen zullen de belangrijkste elementen [24] uitgelegd worden waarvan men gebruik kan maken om een scène op te stellen.

De nodes die kunnen gebruikt worden in een BiFS-beschrijving, zijn de volgende:

- grouping nodes: verzorgen de structuur van de beschrijving. Er zijn

vier nodes die een volledige scène kunnen omvatten. Onafhankelijk van het aantal dimensies, 2D of 3D, kan men kiezen voor *Group* of *OrderedGroup*. Als men echter kiest voor een vaste dimensie, BiFS laat ook toe om 2D en 3D door elkaar te gebruiken, kan men *Layer2D* en *Layer3D* voor respectievelijk 2D en 3D gebruiken. Verder zijn er ook nodes waarmee men de structuur kan aangeven van een auditieve scène (dit zal later in deze tekst in detail worden besproken).

- children nodes: horen in grouping nodes en zorgen voor de multimediale elementen. Hier horen bijvoorbeeld de nodes die instaan voor de belichting.

- bindable children: bijzondere children nodes waarvan op een gegeven moment steeds slechts één node actief kan zijn. Hierbij horen bijvoorbeeld een viewpoint-node (men kan de scène slechts bekijken vanuit één oogpunt) en de background-node.

- interpolator nodes: nodes die data bijhouden nodig voor interpolatie. Men kan verschillende type interpolaties bekomen, zoals interpolatie van posities en oriëntaties.

- sensor nodes: verzorgen interactiviteit in een scène. Men kan een actie koppelen aan deze nodes, waarbij deze actie wordt uitgevoerd zodra deze type nodes actief worden.

- geometry nodes: de verzameling nodes waarmee men elementaire vormen kan aanmaken zoals cilinder, vlak, balk, enzovoort.

- FBA nodes: nodes die dienen ter ondersteuning van Face and Body Animation. Dit zal later in de tekst uitvoerig worden besproken.

3.4.3 Dynamische scène met behulp van ROUTE's

Een dynamische scène kan worden bekomen met behulp van ROUTE-nodes. De werking hiervan is integraal overgenomen van de VRML-specificatie en zal hier kort worden uitgelegd.

ROUTE-nodes zorgen voor een connectie tussen twee elementen binnen een scène waarbij de waarde van hun velden kan worden overgenomen. In combinatie met interpolatie-nodes kan men animaties bekomen van een bepaald element. Zo kan men met een interpolatie-node die posities interpoleert, de positie veranderen van een node en deze dus binnen een scène transleren.

Een voorbeeld van het gebruik van een dergelijke ROUTE-node wordt weergegeven in codefragment 3.1. Hier zien we dat *MyTime* voor opeenvolgende waarden zorgt zodat de positie *MyHeadline* geïnterpoleerd wordt tussen x = 1 en x = -1.

```
Group {
  DEF position Transform {
    translation 1 0 0
      children [
        Shape {
          geometry DEF MyHeadline Text {
            string "new␣headline␣here"
          }
        }
      ]
  }
  DEF PosInterp PositionInterpolator {
    Key [ 0 1 ]
    KeyValue [1 0 0, -1 0 0]
  }
  DEF MyTime TimeSensor {
    loop TRUE
    stopTime -1
  }
  ROUTE MyTime.fraction_changed TO PosInterp.key
  ROUTE PosInterp.keyValue TO position.translation
}
```

Codefragment 3.1: Gebruik van een ROUTE-node [24]

3.4.4 Access Units

Een scène beschreven met behulp van BiFS is niet statisch: een avatar kan men verplaatsen, men kan objecten toevoegen, verwijderen, etc. Om dit mogelijk te maken is er een relatief eenvoudig protocol gedefinieerd waarmee men deze wijzigingen kan doorvoeren. Dit protocol wordt ook gebruikt om een scène gradueel op te bouwen aan de hand van de binnenkomende data. Op deze manier moet men niet de hele scène lokaal hebben opgeslagen om hiermee te kunnen werken. Om dit mogelijk te maken, maakt men gebruik van access units: updateberichten waarmee men een scène kan aanpassen. Er zijn twee type access units: command frames en animation frames. De eerste zorgen voor een wijziging van bepaalde nodes terwijl de tweede meer geschikt is voor animaties.

Met behulp van command frames kan men volgende acties doorvoeren:

- volledige scène vervangen met een nieuwe

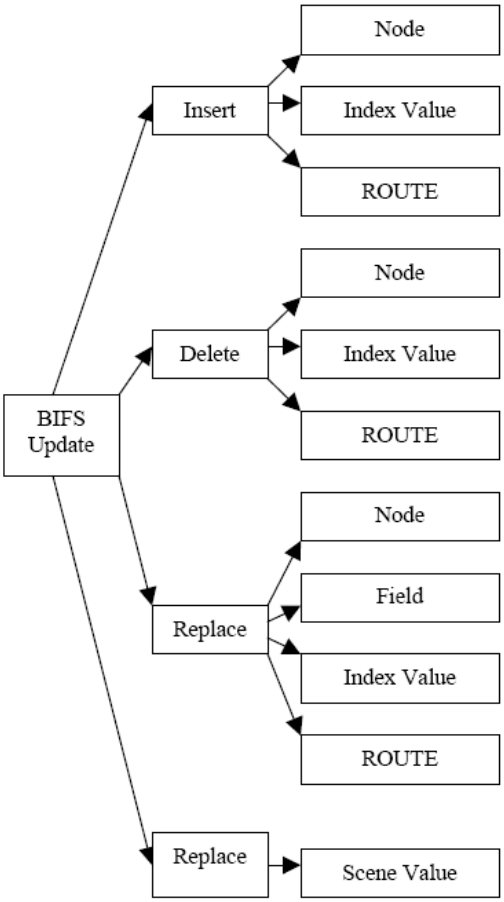
- een node toevoegen
- een node verwijderen
- een attribuut van een node wijzigen

Ieder commando binnen het protocol kan invloed hebben op een verschillend domein van nodes en attributen. In figuur 3.2 worden de bovenstaande punten weergegeven met de bijhorende mogelijkheden. Hierin wordt duidelijk aangegeven dat een node altijd kan worden toegevoegd, verwijderd en vervangen, alsook de index values ervan. Deze index values komen voor binnen bepaalde nodes waar lijsten worden bijgehouden: een concreet voorbeeld is de node *IndexedFaceSet* die een lijst bijhoudt van coördinaten. Om hierbij aan te geven welke coördinaten in welke volgorde een polygoon vormen, moeten de indices van de coordinaten in de correcte volgorde worden meegegeven. Uiteindelijk kunnen ook de velden van bepaalde nodes worden vervangen alsook de hele scène.

Animation frames zorgen ervoor dat men animaties kan controleren. Iedere animation frame zorgt voor een deel van de animatie en kan afzonderlijk verstuurd worden. Ze zijn beter geschikt om animaties te bekomen omdat ze rekening houden met het continue karakter van animaties, in tegenstelling tot de command frames waarbij men steeds afzonderlijke veranderingen kan doorvoeren. Doordat een animatie steeds verderwerkt op reeds gekende data, kan men bepaalde data weglaten (men moet bijvoorbeeld niet continu de naam van de node opslaan). Op deze manier bekomt men een efficiëntere encodering wat weer een rol speelt bij het verzenden van de animatie over een netwerk. De animaties kunnen op twee manieren worden verzonden: in predictive of intra mode. In het eerste geval zal men enkel de verschillen tussen twee opeenvolgende frames sturen terwijl men de volledige waarde zal sturen als men in intra mode werkt.

3.4.5 Tekstformaat

Om het aanmaken en wijzigen van een beschrijving te vergemakkelijken, zijn er verschillende mogelijkheden om een beschrijving tekstueel op te stellen. Deze beschrijving zal dan met behulp van een compiler omgezet worden naar de binaire beschrijving waardoor men dus een volwaardige BiFS-beschrijving bekomt. Hoofdzakelijk zijn er twee tekstuele beschrijvingen: XMT en BT [27]. Het eerste is een op XML gebaseerd formaat terwijl het tweede sterke gelijkenissen vertoont met het formaat dat men gebruikt in een VRML-beschrijving.



Figuur 3.2: BiFS-commands [26]

3.4.6 AudioBiFS

AudioBiFS [28] [29] is een onderdeel van BiFS dat het geluid verzorgt in een virtuele omgeving. AudioBiFS maakt het mogelijk om met behulp van verschillende geluidsfragmenten uiteindelijk één bewerkt geheel te bekomen. Dit is vergelijkbaar met BiFS, waarin vooral het visuele van belang is en men dus een scène opbouwt uit verschillende objecten. Het uiteindelijke geluidsstuk kan men opbouwen uit streaming audio, 3D spatialisatie en routines om het geluid te bewerken. Met behulp van BiFS, waarbij men een beschrijving heeft van de scène, kan men het geluid aanpassen zodat men rekening houdt met de positie van geluidsbronnen en het effect van de omgeving op de geluidsweergave.

AudioBiFS bestaat uit twee versies: de eerste beschrijft een aantal nodes waarmee men een zogenaamde audioscène kan opbouwen, de tweede versie voegt daar geavanceerde auralisatie, mogelijkheid tot koppeling met Java-klassen en een beter bestandsformaat aan toe.

De nodes beschreven in AudioBiFS versie 1 zijn de volgende:

- AudioSource: connectiepunt waar geluidsbestand in de scène wordt geladen. Met *startTime* en *stopTime* kan men het geluid starten en stoppen. Verder kan men details instellen zoals toonhoogte, afspeelsnelheid, aantal te gebruiken kanalen en de fase. Deze node zorgt dus enkel voor het inladen en bepaalt niet waar de geluidsbron zich bevindt in de scène.
- AudioMix: laat toe om het aantal kanalen dat gebruikt wordt te veranderen. Hierbij kan de input komen vanuit één bron of verschillende bronnen. Hiermee kan men de matrix opvragen die gebruikt wordt om te mixen en het aantal inputs en kanalen te bepalen.
- AudioSwitch: dit is een vereenvoudigde versie van AudioMix waarbij het aantal inputkanalen groter is dan het aantal outputkanalen. Hierbij is de berekening sneller en kan men een selectie doen van de kanalen die men als output wilt gebruiken.
- AudioDelay: laat toe een bepaalde vertraging toe te voegen aan een geluidsbron; dit kan bijvoorbeeld gebruikt worden voor synchronisatie.
- AudioFX: node die toelaat om zelf gedefinieerde routines te gebruiken om het geluid te bewerken. De routines zijn niet gestandaardiseerd, maar kunnen aangemaakt worden met behulp van SAOL. Dit is de taal die ontwikkeld werd met als enig doel geluidsverwerking mogelijk te maken. Deze routines kunnen dynamisch gedownload worden.
- AudioClip: hiermee kan men een fragment nemen uit een geluidsbron zoals aangegeven door *AudioSource* en dit fragment kan men op aanvraag meermaals afspelen. Het fragment wordt opgeslagen in het intern geheugen. Men kan ook hier instellingen gebruiken zoals het aantal kanalen, starttijd, etc.
- Sound (Sound2D): een geluidsstroom die meegegeven wordt aan deze node, zal bewerkt worden opdat deze aangepast wordt aan de positie in de scène. *Sound2D* is de variant die gebruikt kan worden voor 2D-ruimtes. Deze node zorgt dus specifiek voor het positioneren van een geluidsbron in de scène.

- Group (Group2D, Transform, Transform2D): nodes die dienen om *Sound*-nodes te groeperen. Hierbij zal alles gesommeerd worden en in het geval van een *Transform*-node wordt de weergegeven positie aangepast.

- ListeningPoint: de positie voor het waarnemen van het geluid komt normaliter overeen met deze voor het waarnemen van de visuele scène, als men echter een andere positie wenst aan te geven om het geluid waar te nemen, dan kan men gebruik maken van deze node.

- TermCap: deze node bevat omgevingsvariabelen waarmee rekening moet gehouden worden bij de weergave van het geluid. Hierbij horen onder andere het aantal luidsprekers of de reële omgeving waarmee men rekening kan houden. Het laatste is vooral belangrijk als de gebruiker zich bijvoorbeeld in een luidruchtige omgeving bevindt waarbij men het geluid best luider kan afspelen.

Om auralisatie te bereiken, worden er met AudioBiFS Versie 2 een aantal nodes toegevoegd. Auralisatie houdt in dat het geluid gepropageerd wordt door de scène, geabsorbeerd wordt en gereflecteerd door muren en voorwerpen. De nodes om dit mogelijk te maken, zijn de volgende:

- AcousticScene: deze node kan gebruikt worden om een scène te definiëren waarin auralisatie mogelijk is. Verschillende attributen kunnen worden voorgesteld zoals *reverbtime* waarmee de reverberatie wordt ingesteld terwijl *useAirabs* en *useAttenuation* de attenuatie aangeven.

- AcousticMaterial: om in een *AcousticScene* objecten te definiëren met de juiste resonantie en absorptie, kan men deze node gebruiken.

- DirectiveSound: deze node laat toe om de richting te bepalen met behulp van de frequentie van de geluidsbron: door deze te laten variëren kan men de indruk geven dat het geluid van een bepaalde bron afkomstig is. Hiermee kan men aangeven hoe sterk het geluid verzwakt in functie van de afstand tot de bron en kan men de snelheid aangeven met *speedOfSound* om een Doppler-effect te bereiken.

AudioBiFS beschikt dus over de nodige middelen om een volwaardige auditieve scène op te bouwen. Een voorbeeld van een beschrijving van een dergelijke scène wordt weergegeven in codefragment 3.2 terwijl de bijhorende SOAL-code om geluidsverwerking mogelijk te maken, wordt weergegeven

in codefragment 3.3. In codefragment 3.2 ziet men dat men een scène beschrijft waarin men gebruik maakt van de AudioMix-node om een monogeluidsbestand om te zetten naar stereo-geluid en deze uiteindelijk te gebruiken in de scène. Men geeft expliciet aan geen omgevingseffecten toe te voegen aan het geluid door de attribuut spatialize waarde 0 te geven. Om de omzetting te maken van mono naar stereo gebruikt men het codefragment weergegeven in codefragment 3.3. Hierin wordt een algoritme gegeven om deze omzetting mogelijk te maken.

```
Group {
  children {
    Sound {
      spatialize 0
      source [
        AudioMix [
          numChan 2
          phaseGroup [1 1]
          matrix [0.8 0.4 1.0 0.1 0.4 0.8 0.1 1.0]
          children [
            AudioSource {
              url "music"
              numChan 2
              phaseGroup [1 1]
              },
            AudioFX {
              orch "..."
              numChan 2
              children [
                AudioSource {
                  url "speech"
                  numChan 1
                  }
] } ] } ] } ] } ] }
```

Codefragment 3.2: Auditieve scène beschreven met behulp van AudioBiFS [28]

```
global {
  outchannels 2;
  send(schoeder; 1; input_bus);
}

instr schoreder(rt) {
  asig ap1, ap2;
  asig c1, c2, c3, c4;
  asig outL, outR;

  ap1 = allpass(input[0], 0.0017, 0.7);
  ap2 = allpass(ap1, 0.005, 0.7);
  c1 = comb(ap2, 0.030, combgain(0.030, rt));
  c2 = comb(ap2, 0.0343, combgain(0.0343, rt));
```

```
  c3 = comb(ap2, 0.393, combgain(0.0393, rt));
  c4 = comb(ap2, 0.045, combgain(0.045, rt));

  outL = (c1 + c2 + c3 + c4)/4;
  outR = (c1 - c2 + c3 - c4)/4;

  output(input + outL, input + outR);
}

opcode combgain(xsig t, xsig rt) {
  return (exp(log(10)* -3 * t / rt));
}
```

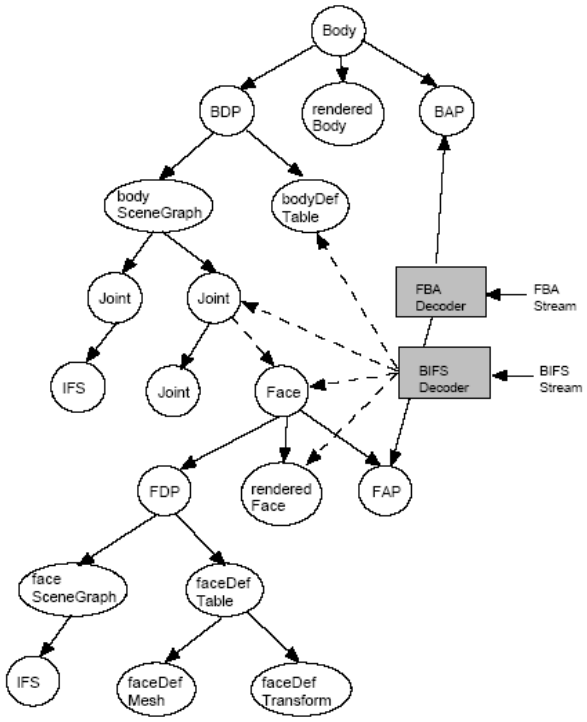
Codefragment 3.3: Geluidsverwerking mbv SAOL [28]

3.4.7 Face and Body Animation

Face and Body Animation (FBA) [30] [31] biedt ondersteuning voor het modelleren en animeren van gezichten en lichamen met behulp van MPEG 4. De modellen die zo gemodelleerd worden, kunnen vervolgens worden geïntegreerd in een BiFS-omgeving. Het ontwikkelen gebeurt aan de hand van parameters: er bestaan parameters voor de vorm van het model, texture en animatie van gezicht en lichaam. De parameters kan men groeperen in vier groepen: Face Definition Parameters (FDP), Face Animation Parameters (FAP), Body Definition Parameters (BDP) en tenslotte Body Animation Parameters (BAP).

FBA wordt opgesplitst in twee delen verdeeld over de MPEG 4-specificatie: het *system*-gedeelte beschrijft de voorstelling en codering van de geometrie en de mogelijkheden om gezicht en lichaam te vervormen, terwijl het *visual*-gedeelte de animatie bepaalt.

FDP en BDP (beschreven in het *system*-gedeelte) vormen een gedetailleerde beschrijving van gezicht en lichaam: gaande van richting van voeten en handen voor het lichaam tot positie en oriëntatie van ogen en tong voor het gezicht. Voor de integratie van FBA binnen BiFS is er een verzameling nodes ontwikkeld: Body, BDP, BAP, BodyDefTable, Face, FDP, FAP, FaceDefTable, FaceDefMesh, FaceDefTransform. De onderlinge relatie tussen deze nodes is weergegeven in figuur 3.3. Hier zien we twee subbomen: deze voor het lichaam waarbij *body* de hoofd-node is en deze voor het gezicht met *Face* als parent. De node *rendered Body* en *rendered Face* zijn verantwoordelijk voor het renderen: hoe dit gebeurt is niet bepaald in de standaard.



Figuur 3.3: FBA-nodes en hun onderlinge relatie [30]

FAP's houden bij wat de wijzigingen zijn ten opzichte van de neutrale positie. De neutrale positie geldt voor ieder wijzigbaar onderdeel zoals een mond, die initieel gesloten is, oogleden, etc. Hetzelfde geldt voor BAP's: hierbij controleert men het skelet van het model.

FaceDefTables en BodyDefTables geven de regels aan om respectievelijk het gezicht en het lichaam te animeren. Voor de animatie van het gezicht zal men twee soorten animatie onderscheiden: om het gezicht te animeren waarbij enkel een translatie, rotatie of schalering plaatsvindt, maakt men gebruik van een *Transform*-node, als men echter kiest voor een werkelijke deformatie van het gezicht, zoals kan gebeuren bij het lachen, moet men gebruik maken van de *IndexedFaceSet*-node. Hierbij wordt via de *FaceDefMesh*-node berekend, welke vertices veranderd worden van het gezicht. BodyDefTables vormt een extensie op FaceDefTables waarbij het volledig lichaam kan worden geanimeerd.

Een interessante mogelijkheid van FBA en specifiek de animatie van gezichten, is *Text-To-Speech*. Hierbij kan een externe text-to-speech synthesizer gebruikt worden die FAP-waarden kan genereren die kunnen gebruikt worden om een animatie te bekomen. Hierbij is het belangrijk te bepalen of de samenwerking met de text-to-speech synthesizer synchroon of asynchroon werkt. Bij een asynchrone werking gebeurt het verwerken zoals net beschreven, bij een synchrone zal de werkwijze anders zijn. Hierbij zal de externe synthesizer niet enkel de FAP-waarden moeten doorgeven, maar ook timing-informatie voor de animatie.

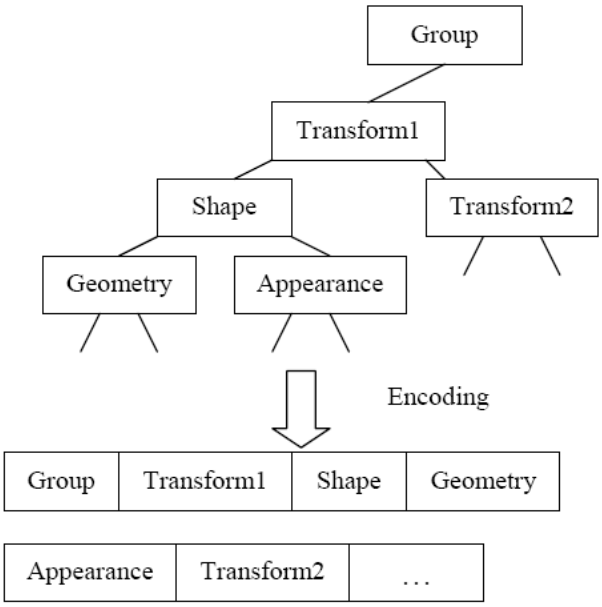
De uiteindelijk codering van FBA gebeurt op een dergelijke manier dat men een bitrate kan halen van 1 kbps voor een model. BDP en FDP worden op verschillende manier gecodeerd en doorgestuurd dan BAP en FAP. BDP en FDP worden in dezelfde stream gecodeerd als BiFS. Deze worden slechts één keer doorgestuurd in tegenstelling tot BAP en FAP. Aangezien BiFS ook speciale streams kent om animatie mogelijk te maken, zoals eerder besproken in de tekst, kunnen deze ook gebruikt worden voor de animatieparameters van FBA.

3.4.8 Streaming van statische scènes en animaties

Een groot voordeel van BiFS is dat men niet over de volledige beschrijving moet beschikken van de scène voordat hierin kan worden genavigeerd [32]. Daar scènes beschreven met behulp van BiFS streaming toelaten, kunnen de scènes al worden gerenderd op het moment dat genoeg gegevens beschikbaar zijn. Op deze manier kan er ook al worden geïnterageerd met de directe omgeving die ingeladen is, terwijl bijvoorbeeld de rest van de virtuele stad nog moet worden ontvangen en opgeslagen.

De mogelijkheid om te streamen is een direct gevolg van de gekozen encodering. Zoals men kan zien in figuur 3.4 wordt de boom in depth-first volgorde doorlopen en geëncodeerd. Een statische scène kan op deze manier dus relatief eenvoudig doorgestuurd worden. In de volgende paragrafen zullen er echter technieken worden gegeven om animaties door te sturen.

Animaties kunnen voorkomen in twee situaties: voorgedefinieerde animaties en animaties die niet voorspeld kunnen worden, bijvoorbeeld ten gevolge van een actie van de gebruiker. In het eerste geval kan men werken met *streaming interpolator frames* terwijl men in het tweede geval kan werken met *access units*.



Figuur 3.4: Encodering van BiFS [32]

Voorgedefinieerde animaties maken gebruik van interpolators om de key frames aan te geven van de animatie. De variabelen die deze animatie bijhouden kunnen met de scène worden geëncodeerd en samen doorgestuurd worden. Deze animaties gebeuren aan de hand van *ROUTE*-nodes die een connectie maken tussen timesensors (tijdstip waarop een animatie moet starten, eindigen, etc), interpolators en objecten. Door de specificatie van de BiFS-encodering kan hier echter een probleem ontstaan: de ROUTE-nodes moeten op het einde van de stream geëncodeerd worden wat zou betekenen dat de animatie pas zou kunnen beginnen zodra de scène is ontvangen. Door de ROUTE-nodes echter door te sturen via een afzonderlijke stream, kunnen deze worden ingevoegd en is er zodus geen afhankelijkheid meer van de rest van de scène. Bij de voorgaande mogelijkheden werd er altijd vanuit gegaan dat de interpolators ook meegeëncodeerd werden met de scènebeschrijvingen. Hierbij vormt zich echter weer een probleem dat bij een groot aantal interpolators, en dit kan bij een gedetailleerde animatie snel oplopen, dit het laden van de scène kan ophouden. Ook hier kan er voor worden gekozen om de waarden van de interpolators in een afzonderlijke stream door te sturen en die lokaal in te voegen. Bij dit laatste zal de animatie echter als een geheel doorgestuurd worden en ontvangen, waarbij dus niets gedaan kan worden met onvolledige data. Om dit euvel te verhelpen, kan men de waarden waaruit

de animatie bestaat, als afzonderlijke gehelen beschouwen en deze dan ook afzonderlijk verwerken. Op die manier kan een gedeelte van een animatie worden getoond terwijl deze nog niet volledig lokaal is opgeslagen.

Animaties die niet kunnen worden voorspeld, zoals acties van een gebruiker, kunnen worden bekomen met behulp van access units. In BiFS vertaalt zich dit in het gebruik van *Field Replacement Command*-nodes. Met behulp van deze nodes kunnen attributen veranderd worden van objecten in een scène. Dit zal bijvoorbeeld betekenen dat een avatar binnen een scène kan bewegen door de translatie- en rotatie-attributen hiervan te wijzigen.

3.5 Voorbeelden

In deze sectie zullen enkele voorbeeldscènes besproken worden die met behulp van BiFS kunnen worden beschreven. We zullen twee eenvoudige voorbeelden geven: één voorbeeld dat een scène definieert in een 2D-ruimte en één dat dit doet in een 3D-ruimte.

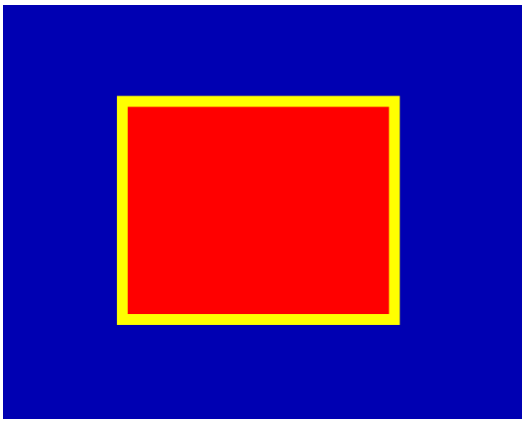
Figuur 3.5 toont een voorbeeld van een 2D-scène. Hierin wordt een achtergrondkleur gedefiniëerd en een eenvoudige rechthoek. We zien dat van deze rechthoek de grootte wordt aangegeven, samen met de kleur. Men bepaalt hierbij ook de omranding door aan tegeven welke dikte deze moet hebben en welke kleur. De beschrijving van deze scène wordt weergegeven in codefragment 3.4.

```
OrderedGroup {
   children [
      Background2D {
         backColor 0 0 0.7
      }
      Shape {
         appearance Appearance {
            material Material2D {
               emissiveColor 1 0 0
               filled TRUE
               lineProps LineProperties {
                  lineColor 1 1 0
                  width 2
               }
            }
         }

         geometry Rectangle {
            size 50 40
         }
```

```
        }
    ]
}
```

Codefragment 3.4: 2D-scène in BiFS [33]



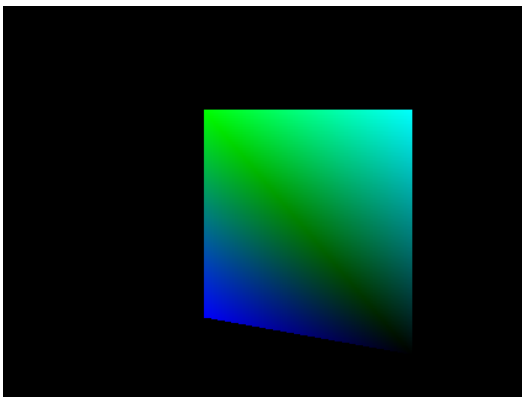
Figuur 3.5: Voorbeeld van een 2D-scène in BiFS [33]

Figuur 3.6 geeft een voorbeeld weer van een eenvoudige 3D-scène bestaande uit een lichtbron en een polygoon die wordt gepositioneerd met behulp van een translatie. De scène hier afgebeeld wordt bekomen met behulp van de beschrijving in codefragment 3.5. Als hoofdknoop wordt hier gekozen van een Group-node en deze heeft de rest van de scène als childnodes. Men ziet dat men een lichtbron definiëert met bepaalde eigenschappen zoals kleur, intensiteit en positie. Verder definiëert men ook een Shape die voor een veelhoek zorgt met de gegeven coördinaten en kleur. Deze Shape-node wordt gepositioneerd aan de hand van een translatie, dit gebeurt door de Shape-node als child te definiëren van de Transform-node.

```
Group {
   children [
      PointLight {
         on TRUE
         intensity 0.8
         color 0.0 0.0 1.0
         location 0.0 0.0 0.0
      }
      Transform {
         translation 10.0 0.0 0.0
         children {
            Shape {
```

```
                geometry IndexedFaceSet {
                    coord Coordinate {
                        point [ 1 -1 2
                                1 1 2
                               -1 1 2
                               -1 -1 2]
                    }
                    coordIndex [3 2 1 0]
                    color Color {
                        color [ 0 0 255
                                0 255 0
                                0 255 255
                                0 0 0]
                    }
                    colorIndex [3 2 1 0]
                }
            }
        }
    ]
}
```

Codefragment 3.5: Voorbeeld van een 3D-scène



Figuur 3.6: Voorbeeld van een 3D-scène in BiFS

3.6 Software Development Kits

3.6.1 VRML en X3D

VRML is de oudste standaard in de reeks die in de vorige secties werden besproken. Hierdoor, en door de grote populariteit, bestaan er een enorm aantal SDK's ontwikkeld voor deze standaard. Een volledige opsomming

hiervan is niet mogelijk, maar tot de reeks bekende SDK's behoren ongetwijfeld Coin3D [34], OpenVRML [35] en CyberX3D [36].

X3D kent, net zoals VRML, ook al een groot aantal SDK's. De eerder vermelde CyberX3D bevat, zoals de naam al doet vermoeden, ondersteuning voor deze standaard. Maar er zijn ook andere SDK's zoals X3DToolkit [37] en Avalon [38]. De opgesomde SDK's hebben allemaal de mogelijkheid om X3D-bestanden in te lezen en te visualiseren aan de hand van OpenGL.

3.6.2 MPEG-4

Implementaties die toelaten om BiFS toe te passen zijn dun gezaaid omdat dit een recent ontwikkelde standaard is. Buiten de reference software van MPEG 4, die enkel bedoeld is als demonstratie van de standaard en niet als productiecode, zijn er twee voorbeelden van SDK's die BiFS ondersteunen, namelijk GPAC [41] en MP4SDK [39],[40]. Zij zullen in deze sectie worden besproken.

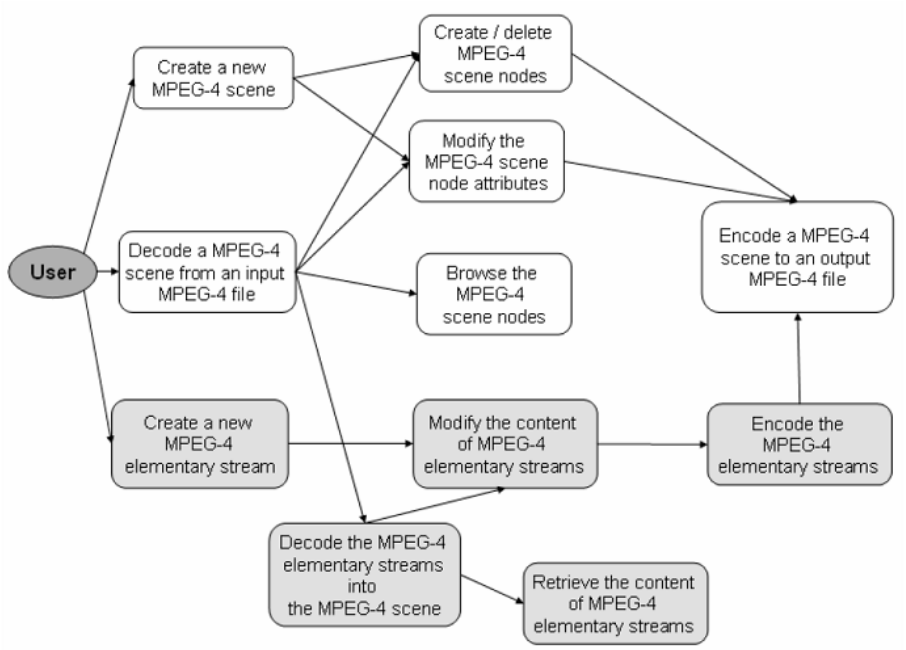
GPAC is een open source project dat als doel heeft de volledige MPEG 4-specificatie te implementeren. Hierdoor is het een relatief groot project dat buiten de ondersteuning voor BiFS, een groot aantal andere functionaliteiten bevat. Dit beslaat ook de encodering en decodering van video- en audiostromen, waaronder H.264/AVC. GPAC is ontwikkeld voor verschillende platformen waaronder Windows, Linux en PocketPC.

De ontwikkeling van MP4SDK valt onder het Artemis Project Unit en is op moment van schrijven nog steeds in ontwikkeling. Uitgangspunt van deze SDK is dat de implementatie van applicaties die gebruik maken van MPEG 4 vergt dat de ontwikkelaars de specificatie van de standaard in te groot detail moet kennen. MP4SDK probeert hier dus een oplossing te bieden door de details van de standaard te verbergen opdat een ontwikkelaar snel aan de slag kan. Het origineel project gebruikte als kern de reference software (een verzameling softwarebibliotheken die worden ontwikkeld om de mogelijkheden van een standaard aan te tonen) maar door instabiliteit in deze software is er een overstap gemaakt naar GPAC waarvan dus de kern wordt gebruikt. MP4SDK zal in de toekomst een meer volledige ondersteuning van de standaard bieden, waarvan Facial and Body Animation ook deel is.

MPEG-4 SDK biedt directe toegang tot de BiFS-nodes en de elementaire streams. De hoofdfunctionaliteiten zijn:

- Encoding en decoding van BiFS en elementaire streams
- bekijken en bewerken van de attributen horende bij de nodes
- creëren en verwijderen van nodes

Een schematisch overzicht van deze architectuur wordt weergegeven in figuur 3.7.



Figuur 3.7: Architectuur van MPSDK

Omdat er geen enkele applicatie denkbaar is die de gehele BiFS-specificatie kan gebruiken, wordt er gebruik gemaakt van profiles. Deze zorgen ervoor dat er een bepaalde subset kan worden genomen, zodat enkel die nodes overblijven die werkelijk worden gebruikt in de beoogde applicatie.

Voor *Face and Body Animation* bestaan er ook verschillende SDK's. Veelal zijn deze van commerciële aard zoals Visage SDK [42]. Deze SDK biedt een volledige implementatie van de FBA-specificatie. Het is onder andere mogelijk om lipsynchronisatie mogelijk te maken zoals deze besproken wordt in de standaard, met behulp van de Face Animation Parameters. Een greep uit de mogelijkheden van deze SDK zijn de volgende:

- animatie van gezichten met behulp van SAPI-5 spraaksynthese met mogelijkheid om spraak afkomstig uit een invoerapparaat zoals een microfoon, real-time om te zetten in animatie van een virtueel gezicht
- ontwikkelen voor plugins bruikbaar voor externe 3D-applicaties
- volledige ondersteuning voor de MPEG 4 FBA-specificatie

3.7 Gebruik binnen een NVE

Zoals er in de inleiding van dit hoofdstuk werd geschreven, zijn bovenstaande standaarden nuttig in het ontwerp van een virtuele omgeving. BiFS biedt bepaalde aspecten die zorgen voor een efficiënt transport van scène en updates, waardoor deze het meest interessante is om te gebruiken in een NVE. We zullen echter zien dat X3D ook bepaalde aspecten biedt die het uiterst bruikbaar maken in de genoemde context.

De oorspronkelijke taal om werelden te beschrijven is VRML. Hiermee werden er in het verleden ook toepassingen ontwikkeld die van belang zijn voor NVE's. Humanoid Animation WG en Living Worlds WG [6] zijn voorbeelden hiervan. Humanoid Animation WG maakt gebruik van VRML voor de voorstelling van menselijke avatars in een NVE. Hieruit werd een verzameling softwarebibliotheken bekomen waarmee deze avatars uitwisselbaar zijn en dergelijke avatars kunnen worden ontworpen.

X3D heeft al in een aantal NVE's toepassing gevonden [43]. Hiervan enkele voorbeelden:

- Blaxxun [44]: deze NVE heeft een server-client-architectuur en is ontwikkeld in Java. Het biedt de gebruikers de mogelijkheid om samen te komen in een virtual classroom, om te vergaderen, etc.
- Octaga [45]: dit is een commercieel pakket dat bestaat uit verschillende applicaties die gebruik maken van X3D. Hierbij biedt Octaga Professional in combinatie met Octaga server mogelijkheden om een genetwerkte virtuele omgeving op te zetten. Octaga biedt verder ook toepassingen die gebruik maken van de MPEG 4-standaard.

EVE [43] is een genetwerkte virtuele omgeving die gebruik maakt van VRML. Voor deze NVE werd er een uitbreiding geschreven zodat de virtuele omgeving niet langer wordt beschreven in VRML maar in X3D. De voordelen

die men hiermee bekomt zijn onder andere een hogere mate van flexibiliteit, door het groot aantal X3D-nodes, en een consistente omgeving voor iedere gebruiker.

Uit de bespreking van BiFS in de vorige secties, heeft men kunnen afleiden dat BiFS, of meer bepaald de MPEG 4-standaard, een uitgebreide standaard is. De belangrijkste voordelen ten opzichte van VRML en X3D zijn het binair formaat en de mogelijkheid tot streaming. *COllaborative System framework based on MPEG-4 Objects and Streams* (COSMOS) [32] is een genetwerkte virtuele omgeving die gebruik maakt van BiFS. Hierin wordt het duidelijk dat de streaming van de scène als voordeel heeft dat men niet moet wachten totdat deze volledig gedownload is, maar dat deze geleidelijk opgebouwd wordt zodra men al bruikbare pakketten lokaal heeft kunnen opslaan. Dit laatste maakt het dan ook mogelijk om te interageren en navigeren zodra de eerste delen van de scène worden weergegeven. Hierbij is het binair formaat een niet te onderschatten voordeel van deze standaard. Doordat deze veel beperkter is in omvang, zal het versturen veel sneller gaan over een netwerk waardoor men ook met een zwakkere netwerkverbinding gebruik kan maken van een bepaalde NVE.

BiFS biedt echter meer voordelen: de uitgebreide ondersteuning voor het modelleren en animeren van het menselijk lichaam en gezicht, en de uitgebreide ondersteuning voor audio. Het is al langer duidelijk dat avatars van groot belang zijn in het ontwerp van een NVE. Men probeert steeds vaker een natuurgetrouwe representatie te geven hiervan om de gebruikerservaring te verbeteren. De uitgebreide ondersteuning hiervoor in MPEG 4 kan duidelijk bijdragen aan deze verbetering. AudioBiFS zorgt er voor dat de omgeving ook op auditief vlak correct kan worden waargenomen. Uiteindelijk is het ook bewezen [46] dat men met behulp van MPEG 4, de datadoorvoer voor animaties aanzienlijk kan verminderen, wat dus weer eens een voordeel is voor het gebruik over een netwerk. Dit is natuurlijk van groot belang bij een genetwerkte virtuele omgeving, waarbij men steeds tracht het datavolume laag te houden, maar waarbij men hieruit zoveel mogelijk informatie wil bekomen.

In sectie 2.4 werd al gesproken over VPARK. Deze NVE maakt gebruikt van enkele van de hier besproken standaarden. Het beschrijven van de wereld gebeurt aan de hand van VRML, terwijl de avatars worden geanimeerd met behulp van de Face and Body Animation horende tot de MPEG 4-standaard.

3.8 Conclusie

In dit hoofdstuk werden er drie standaarden besproken om virtuele werelden te beschrijven. Alsook werd het belang van dergelijke standaarden beschreven voor genetwerkte virtuele omgevingen.

Hoewel dergelijke standaarden nog weinig worden gebruikt in NVE's, zal dit zeker veranderen in de toekomst. Ontwikkeling van gebruiksvriendelijke SDK's is noodzakelijk opdat men dergelijke standaarden meer zou gebruiken. Dit is ook de motivatie voor de ontwikkeling van MP4SDK [39]:

> MPEG-4 is a powerful multimedia standard in terms of media representation, scene composition and user interactivity. However, building an MPEG-4 application demands an in-depth knowledge of MPEG-4 specifications which currently limits the world wide deployment of the standard.

Men heeft gezien dat deze standaarden ondersteuning bieden voor een groot aantal aspecten nodig voor de ontwikkeling van een genetwerkte virtuele omgeving. Dit is onder andere de beschrijving van werelden, animatie van menselijke avatars en gebruik van 3D-geluid. Dit zorgt dan ook dat dergelijke standaarden uiterst bruikbaar kunnen zijn binnen deze context.

Hoofdstuk 4

Audio- en Videostandaarden

4.1 Inleiding

In genetwerkte virtuele omgevingen speelt de overdracht van audiovisuele data steeds vaker een belangrijke rol. Het mogelijk maken dat deelnemers elkaar kunnen horen of zelfs zien, voegt een extra dimensie toe aan de gebruikerservaring. Het gebruik van audiovisuele data kan een netwerk echter zwaar belasten. Daarom zijn compressietechnieken zeer belangrijk als men deze wilt gebruiken in een NVE.

Door de jaren heen werden er verschillende compressiemethoden ontwikkeld voor audiovisuele data. Een aantal hiervan hebben, en kennen nog steeds, een enorme populariteit, terwijl anderen in vergetelheid zijn geraakt. Slechts een greep uit deze compressiemethoden worden als een standaard, of gedeelte hiervan, beschreven: deze zullen hier aan bod komen.

We zullen dit hoofdstuk beginnen met het bespreken standaarden voor audio-compressie. Hierna zullen verdergaan met video-compressie en eindigen met een conclusie.

4.2 Audio

4.2.1 MPEG 1

Binnen de MPEG 1-standaard [47] worden er drie layers beschreven voor audiocompressie, waarvan MPEG 1 Audio Layer 3 (MP3) ongetwijfeld het meest bekende is. De specificaties van deze drie layers zijn terug te vinden in tabel 4.1.

	Audio sampling rate (kHz)	Compressed bit-rate (kbits/sec)	Channels
MPEG 1 Layer 1	32, 44.1, 48	32 - 448	1-2
MPEG 1 Layer 2	32, 44.1, 48	32 - 384	1-2
MPEG 1 Layer 3	32, 44.1, 48	32 - 320	1-2

Tabel 4.1: MPEG 1 audiolayers

MPEG 1 Audio Layer 1 en 2 verdelen de inputstream in 32 subbands met de daarbij horende verdeling in frequenties. Van deze subbands worden er 12 samples genomen die worden genormaliseerd. Hierna gebeurt de quantisatie en het toepassen van een psycho-acoustisch model. Het verschil tussen layer 1 en 2 vertoont zich in dit laatste daar men voor layer 1 512 punten gebruikt voor de fast Fourier-transformatie (FFT), tegenover 1024 voor layer 2. De FFT gebeurt in parallel met de subband-decompositie. Vervolgens zal de bit-allocation unit vaststellen wat de resolutie voor de quantisatie moet zijn om uiteindelijk tot het gecomprimeerd audio-bestand te komen. Buiten de verschillen voor de FFT, verwijdert layer 2 overbodige data en gebruikt een quantisatie-tabel met een hogere precisie.

MPEG 1 Audio Layer 3 werkt verder op de vorige twee layers maar heeft enkele belangrijke verschillen. Het biedt onder andere een hogere resolutie voor de frequentie door een 18-punts *modified discrete cosine transform (MDCT)* te gebruiken. Hierbij gebeurt de transformatie ook dynamisch afhankelijk van karakteristieken van het signaal en gebeurt er een non-uniforme quantisatie. Uiteindelijk wordt er bij de bit allocatie een zogenaamde bit reservoir gebruikt waarbij de overblijvende bits van samples die minder dan het gemiddeld aantal bits nodig hebben, kunnen doorgegeven worden aan samples die er meer nodig hebben.

De layers die hier werden vermeld, kunnen allemaal mono, stereo, dual channel en joint stereo modes leveren. In het geval van joint stereo kan men gelijkenis tussen de twee kanalen uitbuiten om verdere compressie te behalen.

4.2.2 MPEG 2

De MPEG 2-standaard [47] beschrijft twee delen voor audiocompressie: BC en AAC. Beide ondersteunen 5.1 kanalen en lagere frequenties ten opzichte van MPEG 1, met name 16, 22.05 en 24kHz.

Een belangrijk aspect van MPEG 2 BC is dat het backwards compatible (vandaar de afkorting BC) is met de audio-standaarden beschreven in MPEG 1. Hierdoor beperkt deze zich tot een uitbreiding van de velden voor de sampling rate en bit rate. Verder biedt het een nieuw psycho-acoustisch model.

Grotere wijzigingen worden gebracht door MPEG 2 Advanced Audio Coding (AAC). In tegenstelling tot MPEG 2 BC is deze niet backwards compatible is met vorige standaarden. De belangrijkste verschillen zijn een sample frequentie die gaat van 8 kHz tot 96 kHz, mogelijkheid tot 48 kanalen en een betere en efficiëntere codering wat resulteert in een halvering van de bitrate zonder verlies aan kwaliteit. Dit wordt mogelijk gemaakt door:

- gebruik van een gewijzigde filter bank, namelijk een variabele 256- tot 2048-punts MDCT
- Temporal Noise Shaping: een controle over de zogenaamde quantisatie noise
- Predictie
- Quantisatie: hierbij kan men de resolutie preciser bepalen
- Bit-stream formaat: een flexibeler formaat

4.2.3 MPEG 4

De MPEG 4-standaard tracht het heel domein van audiocompressie te omvatten, gaande van spraakcodering tot codering van synthetisch geluid. Hiervoor is er een uitbreiding van MPEG 2 AAC voorzien waarbij er ondersteuning is van een bitrate gaande van 2 tot 64 kb/s. Er bestaan drie type coderingen:

- Parametric coding: deze wordt gebruikt voor spraakcodering waarbij een bitrate van 2 tot 6 kb/s wordt gehaald
- Code-excited linear predictive coding: deze behaalt een goede kwaliteit voor audio met een samplerate van 8 en 16kHz waarbij men een bitrate gebruikt van 6 en 24kb/s
- time/frequency coding: geschikt voor 8 kHz-audiostromen waarbij men een bitrate gebruikt van minimaal 16 kb/s

4.2.4 Toepassingen

De audio-encoderingen die een enorm populair zijn, zijn MPEG 1 Layer 3 (later MPEG 2 Layer 3) en MPEG 2 AAC. MPEG 1 Layer 3 werd, zoals hierboven besproken, eerst ontwikkeld als onderdeel van MPEG 1 en later verbeterd met de komst van MPEG 2. Door de opslag van digitale audio toegankelijk te maken voor het groot publiek, hebben deze standaarden gezorgd voor een kleine revolutie in de wereld van digitale audio. MPEG 2 AAC kent vooral een groot succes doordat deze gebruikt wordt als audiocompressie voor de populaire Apple iPod.

Met MPEG 4 wordt er echter geprobeerd om de audiostandaarden voor een veel breder domein te gebruiken. Het is onder andere bedoeld voor spraak en synthetisch geluid. Het biedt uiteindelijk ook de mogelijkheid om schaalbaarheid te bekomen, iets wat we ook in andere standaarden zien terugkomen.

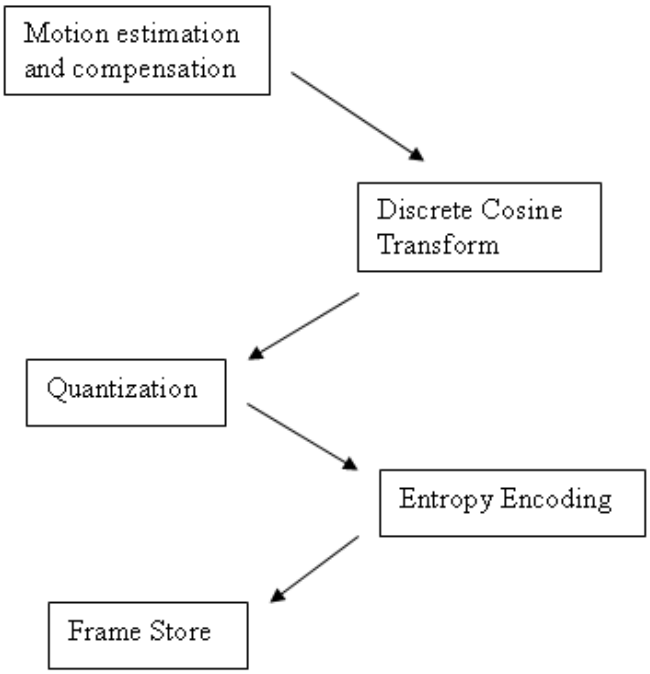
4.3 Video

4.3.1 Algemeen

Een groot aantal videostandaarden gebruiken als basis hetzelfde encoderingsproces. Dit proces zal men voor de verschillende standaarden vaak verfijnen, maar het wordt nooit drastisch veranderd. Het proces doorloopt verschillende stappen die we kunnen zien in figuur 4.1.

Met behulp van motion estimation and compensation zal de te versturen data in eerste instantie al verminderd worden door de verschillen te berekenen van een bepaald frame met het vorige in plaats van steeds een volledig nieuw frame te sturen. Vervolgens zal men met macroblocks, van bijvoorbeeld 16x16, berekenen waar deze zich naartoe verplaatsen opdat men enkel deze verplaatsingen moet encoderen. Hiermee zal Discrete Cosine Transform (DCT) zorgen voor een transformatie waarmee er, vertrekkende vanuit de originele pixelwaarde, overgegaan wordt naar een kleiner aantal coëfficiënten. Met deze kan men vervolgens het origineel beeld terug bekomen.

Een groot aantal coëfficiënten bekomen uit de DCT-stap zullen dicht bij 0 liggen. Met behulp van de quantization-stap zal men bepalen welke coëfficiënten dicht genoeg bij 0 liggen om de decimale waarde te verwijderen. Dit is van belang om de volgende stap efficiënter te doen verlopen. Entropy



Figuur 4.1: Schematisch overzicht van het encoderingsproces voor videostromen

encoding zal er namelijk voor zorgen dat veel voorkomende coëfficiënten met korte binaire codes worden voorgesteld ten koste van langere binaire codes voor minder vaak voorkomende coëfficiënten. Hierbij ziet men dus ook het belang van de quantisatie-stap. Uiteindelijk zorgt de Frame Store voor een opslag om het huidig frame op te slaan, zodat men het kan gebruiken voor de predictie van het volgend frame. Door het frame hier te decoderen met behulp van een decoder, bekomt men hetzelfde frame als dat men later op bijvoorbeeld de bestemming zal bekomen. Hiermee kan men op een correcte manier het volgend frame encoderen.

4.3.2 H.263v1, H.263v2 en H.263v3

H.263 is een videocodering ontwikkeld door International Telecommunications Union (ITU) die voornamelijk wordt gebruikt voor videoconferencing. Hierbij denkt men vooral aan video met lage bitrates, vanaf 20-30 kbps.

Encodering gebeurt op de manier zoals in sectie 4.3.1 beschreven en hierbij worden er vier bijkomende modes gedefiniëerd:

- Unrestricted Motion Vector Mode: in de eerste stap van het encoderingsproces, bepaalt men vectoren die aangeven welke de verplaatsingen zijn van macroblocks. Deze zijn echter in de originele mode beperkt tot vectoren die verplaatsingen aangeven binnen het gegeven frame. Door deze restrictie weg te laten, is het mogelijk om een efficiëntere codering te krijgen en heeft men een grotere flexibiliteit aan de randen van een frame.

- Syntax-Based Arithmetic Coding Mode: door dit te gebruiken in plaats van de variable-length coding, kan men een winst behalen van 5%.

- Advanced Prediction Mode: deze mode laat vier motions vectors per macroblock toe waarbij deze, zoals in de unrestricted motion vector mode, buiten het huidige frame kunnen wijzen.

- PB-Frames Mode: hierbij bestaan frames uit P- en B-layers waarbij men voor de P-layer voor een voorwaartse predictie berekent aan de hand van de vorige P-layer en de B-layer een bidirectionele predictie is van de vorige en volgende P-layer.

H.263v2 biedt meer flexibiliteit waarbij frame grootte en vorm samen met de klokfrequentie niet meer beperkt zijn zoals in H.263v1. Schaalbaarheid krijgt hier ook meer aandacht, wat belangrijk is in foutgevoelige transportkanalen. H.263v2 introduceert ook het gebruik van slices om de predictie te begrenzen binnen een kleiner gebied. Uiteindelijk biedt H.263v3 vier uitbreidingen: het gebruik van meerdere referentie-afbeeldingen op macroblock-niveau, het optimaliseren van het coderen met behulp van slices, toevoegen van extra informatie met betrekking tot de codering en de definitie van verschillende profielen en niveaus. Dit heeft dus een betere compressie tot gevolg. De verschillende profielen en niveaus zorgen voor een grotere flexibiliteit voor de gebruiker die op deze manier een compressie kan bekomen die beter voldoet aan zijn noden.

4.3.3 MPEG Videostandaarden

Binnen het MPEG-consortium zijn er ook verschillende videostandaarden ontwikkeld. Deze kennen een enorm succes en worden gebruikt in verschillende domeinen: MPEG 1 werd vaak gebruikt voor VideoCD's, terwijl MPEG 2 voornamelijk bekend is als formaat voor DVD Video's. Binnen de MPEG

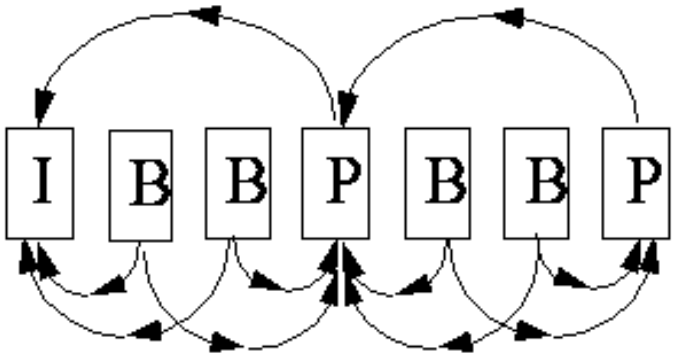
4-standaard zijn er voornamelijk twee videostandaarden: MPEG 4 Simple Profile en MPEG 4 AVC. Doordat deze laatste niet enkel tot MPEG behoort, zal die in het volgend, afzonderlijk onderdeel besproken worden. In de volgende paragrafen zullen de overige standaarden worden besproken.

MPEG 1 [48] is de eerste uit de reeks MPEG-standaarden. Omdat deze standaard werd ontwikkeld met TV-kwaliteit als doel, is de beoogde bitrate 1.5MB/s of minder met een resolutie van 352x288 en een framerate van 24 tot 30 frames per seconde. De frames worden opgeslagen in YUV-formaat waarbij de luminantiekanalen niet worden aangepast terwijl de chrominantiekanalen worden gecomprimeerd met ratio 2:1. MPEG 1 kent drie type frames:

- I-frame: deze frames worden integraal overgenomen en geëncodeerd op een gelijkaardige wijze als een figuur in JPEG-compressie
- P-frame: deze frames zijn afhankelijk van voorgaande referentieframes. Deze referentieframes kunnen voorgaande I-frames zijn maar ook P-frames. Uiteindelijk zal enkel het verschil opgeslagen worden met deze referentieframes. Dit verschil wordt bekomen door macroblocken van 16 bij 16 pixels te nemen van het beeld en hiervan de verplaatsingsvectoren op te slaan.
- B-frame: in tegenstelling tot P-frames, die enkel rekening houden met voorgaande referentieframes, houden B-frames ook rekening met referentieframes die zich in de videostroom na het huidige frame bevinden.

Een voorbeeld van een typische ordening van frames kan men zien in figuur 4.2. Hier zien we ook pijlen die aanduiden hoe de onderlinge frames van elkaar afhankelijk zijn.

In tegenstelling tot MPEG 1, dat voornamelijk werd gebruikt voor videoopslag van lage kwaliteit, beoogt MPEG 2 [49] een breder toepassingsdomein door een grote variëteit aan bitrates aan te bieden. MPEG 2 wordt gebruikt voor de compressie van video van hoge kwaliteit in onder andere broadcasttelevisie en voor DVD Video's, maar ook voor het streamen van video. Door dit breed toepassingsdomein is de MPEG 2-standaard zeer uitgebreid. Aangezien geen enkele toepassing al de mogelijkheden van deze standaard nodig heeft, wordt er gebruik gemaakt van *profiles* en *levels*.



Figuur 4.2: Typische sequentie in een MPEG 1-videostroom

Net zoals men gezien heeft in MPEG 1, zal men ook voor de videocompressie van MPEG 2 gebruik maken van verschillende type frames: de I-, P- en B-frames. De betekenis van deze frames blijft hier ook dezelfde. Het gebruik hiervan is echter bepaald door het gekozen profile. Hoofdzakelijk zijn er de volgende profiles:

- simple profile: dit profile maakt geen gebruik van B-frames. De decoder heeft slechts een kleine buffer nodig waardoor het uitermate geschikt is voor videoconferencing en dus realtime video.

- main profile: dit is de meest gebruikte profile en maakt gebruik van B-frames. Een decoder voor dit profile is verder ook geschikt voor MPEG 1-video.

- SNR profile: dit profile kan gebruikt worden om schaalbaarheid te bekomen in een videostroom. De verschillende lagen van de Descrete Cosine Transform maken gebruik van *signal to noise ratio scalability* om schaalbaarheid te bekomen. Het gebruik hiervan zal zorgen dat men eerst een basislaag zal doorsturen en deze steeds verder verfijnen met extra lagen totdat men de gewenste kwaliteit bekomt.

- spatial profile: extra lagen die zorgen voor meer detail, worden hier onder een lagere resolutie geëncodeerd. Bij het gebruik van de decoder, zal eerst de resolutie van deze extra lagen aangepast worden aan de gewenste resolutie en dan pas toegepast op de basislaag.

- high profile: deze profile maakt gebruik van een 4:2:2-encodering en combineert de SNR en spatial profile.

MPEG 2 kan videostromen encoderen met een resolutie gaande van 352x288 tot 1920x1152. Hierbij hoort een bitrate gaande van 4 Mbit/s tot 80 Mbit/s.

De laatste in de reeks videostandaarden behorende tot MPEG, is MPEG 4. MPEG 4 beschrijft hoofdzakelijk twee video-encoderingen: deze beschreven in het visual-gedeelte van de standaard en Advanced Video Coding. De laatste werd in samenwerking gemaakt met ITU en zal in sectie 4.3.3 worden beschreven. De encoderingen horende bij de visual-gedeelte (part 2 van de MPEG-4 standaard) zullen hier kort worden behandeld.

Net zoals MPEG 2, worden er in MPEG 4 Visual [50] [51] gebruik gemaakt van profiles en levels. Er zijn een groot aantal profiles en levels maar twee hiervan zijn de meest voorkomende:

- simple profile (SP): geschikt voor snelle uitvoering en gebruik in foutgevoelige omgevingen. Dit zorgt dat dit profiel ideaal is voor streaming video op mobiele toestellen, over draadloze netwerken en dergelijke meer. Deze is sterk vergelijkbaar met H.263 van het ITU, die eerder werd besproken.

- advanced simple profile (ASP): dit is het profiel dat zorgt voor een betere kwaliteit dan de vorige. Resultaat hiervan is het gebruik van een hogere bitrate die kan oplopen tot 8 Mbit/s.

ASP, dat hierboven werd aangehaald, werkt verder op SP en heeft dan ook meer mogelijkheden. Het ondersteunt onder andere interlaced video, B-frames en motion quantization: deze zijn grotendeels overgenomen van de MPEG 2-standaard. Het introduceert ook technieken als het gebruik van global motion compensation, wat erop neerkomt dat men met behulp van enkele parameters kan aangeven welke bewegingen worden gemaakt door de camera. Hierbij denkt men aan het horizontaal en verticaal bewegen van de camera, voor- en achteruit bewegen, etc. Door deze bewegingen te modelleren kan men tot een betere compressie komen. Hiernaast wordt er ook gebruik gemaakt van quarter motion compensation, wat betekent dat er macroblokken worden gebruikt van 4x4 pixels.

MPEG 4, en in het bijzonder ASP, kent tegenwoordig een grote populariteit. Er zijn tal van implementaties van deze standaard. Microsoft heeft ondersteuning in Windows Media voor een licht aangepaste versie van

de standaard, Apple ondersteunt de standaard volledig in de Quicktime-implementatie en verder zijn er de bekende DivX en Xvid die een opensource implementatie bieden.

4.3.4 H.264/AVC

Algemeen

H.264/AVC [52] is ontstaan dankzij een samenwerking tussen ITU en MPEG. Hiervan is ook de dubbele benaming afkomstig. Binnen de MPEG-standaarden behoort deze video-encodering tot de MPEG 4-standaard.

H.264/AVC is op moment van schrijven de meest recente videostandaard. Het doel van H.264/AVC is enerzijds een meer efficiënte compressie ten opzichte van de vorige standaarden en anderzijds een formaat te creëren dat beter geschikt is om te gebruiken over een netwerk. Bij dit laatste maakt men onderscheid tussen real-time video zoals een videofoon en opgeslagen video maar geschikt voor streaming over een netwerk. De aandachtspunten die men voorop stelt, komen tot uiting in het design door middel van een Video Coding Layer en een Network Abstraction Layer.

Ten opzichte van voorgaande videocoderingstandaarden zijn er op verschillende vlakken een groot aantal verbeteringen aangebracht. Het predictie-algoritme werd verbeterd door onder andere het gebruik van een gewogen predictie en een variabele grootte voor de blocks waarmee men het beeld opsplitst. De codering zelf werd ook aangepast door bijvoorbeeld een verbetering van de entropy coding, de zogenaamde arithmetic entropy coding. Uiteindelijk is er ook gedacht aan robuustheid en dus het opvangen van fouten. Bij dit laatste dus vooral fouten die optreden bij het versturen van een videostroom over een netwerk.

De Network Abstraction Layer (NAL) zorgt ervoor dat de gecomprimeerde video opgesplitst wordt in NAL units. Dankzij deze NAL units bekomt men een architectuur die geschikt is voor pakket- en bitstreamgebaseerde transmissie. Een welbepaalde groep NAL units vormt een access unit die exact één frame bevatten van de videostroom. Hierbij voegt men ook *supplemental enhancement information* (SEI) toe, die timinginformatie kan bevatten.

Zoals ook bij een aantal voorgaande videocoderingen het geval is, zal de Video Coding Layer (VCL) van H.264/AVC ook gebruik maken van macroblocks. Met behulp van deze macroblocks vormt men slices waarmee men

slicegroups en uiteindelijk een frame opbouwt. De slices kunnen volledig afzonderlijk gecodeerd worden, wat betekent dat ze de informatie van de overige slices in een frame niet nodig hebben. Iedere slice is van de volgende drie typen: I-slice, P-slice en B-slice. De betekenis van deze slices is vergelijkbaar met I-, P- en B-frames die voorkomen in de videocodering horend bij de MPEG 2-standaard. Verder zijn er nog twee nieuwe typen: Switching P (SP)- en Switching I (SI)-slices. Met behulp van SP-slices kan men een slice reconstrueren ook al gebruikt men verschillende referentieframes, terwijl men met SI-slices identieke reconstructies kan maken horende bij de overeenkomstige SP-slices.

MPEG 4 SVC

MPEG 4 Scalable Video Coding [53] is een extensie op MPEG 4 AVC met als doel een betrouwbare overdracht van videostromen over een heterogeen netwerk mogelijk te maken, waarbij men geen weet heeft over systeem- en netwerkmogelijkheden. SVC laat dus schaalbaarheid toe op bitstreamniveau en dit voor de volgende drie aspecten:

- schaalbaarheid op spatiaal niveau
- schaalbaarheid op temporeel niveau
- schaalbaarheid in kwaliteit

Een SVC-videostroom bestaat uit verschillende lagen. De basislaag is een AVC-videostroom die voor de laagste kwaliteit zorgt. Deze laag zorgt er ook voor dat een SVC-videostroom ook gedecodeerd kan worden door AVC-decoders. Op deze AVC-laag bouwt men verder met verschillende lagen die steeds voor een betere kwaliteit van de videostroom zorgen.

De verschillende lagen zorgen steeds voor een verbetering op één van de drie aspecten die schaalbaarheid mogelijk maken, zoals we eerder hebben gezien. SVC maakt het mogelijk om deze lagen te combineren waardoor men voor één videostroom een groot aantal kwaliteitsniveau kan bekomen. Een layer dat instaat voor een hogere resolutie, en dus schaalbaarheid verzorgt op spatiaal niveau, kan gecombineerd worden met verschillende soorten layers die instaan voor het schaleren op temporeel niveau. Op deze manier kan men bijvoorbeeld een videostroom bekomen met een hogere resolutie maar met een lagere framerate.

4.4 Conclusie

In dit hoofdstuk werden een aantal audio- en videostandaarden besproken die als hulpmiddel kunnen dienen in NVE's om de gebruikerservaring te verbeteren. Door gebruikers met behulp van een webcam en microfoon de mogelijkheid te geven elkaar te kunnen zien en horen, kan men het persoonlijk karakter van een avatar verbeteren.

Het probleem dat zich echter voordoet bij het gebruik van audiovisuele data over een netwerk, is dat men te maken heeft met te grote datavolumes. Dit is dan ook de reden waarom men kiest voor een compressie van deze data. De standaarden hierboven beschreven zorgen steeds voor een zo goed mogelijke compressie voor een bepaalde doel. Door de jaren heen ziet men ook een evolutie in de compressiemethoden, die voor een steeds hogere kwaliteit een steeds kleiner datavolume nodig hebben.

Hoofdstuk 5

MPEG 21

5.1 Inleiding

Het MPEG-21 Multimedia Framework wordt als volgt gedefiniëerd [54]:

> MPEG-21 aims at defining a normative open framework for multimedia delivery and consumption for use by all the players in the delivery and consumption chain. This open framework will provide content creators, producers, distributor and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer providing them access to a large variety of content in an interoperable manner.
>
> The MPEG-21 Multimedia Framework is based on two essential concepts: the definition of a fundamental unit of distribution and transaction (the Digital Item) and the concept of Users interacting with Digital Items. The Digital Items can be considered the "what" of the Multimedia Framework (e.g. a video collection, a music album) and the Users can be considered the "who" of the Multimedia Framework.

Het doel van de MPEG 21-standaard is dus het vereenvoudigen van de toegang tot verschillende media (zijnde video, geluid, etc) zodat men niet meer afhankelijk is van netwerk- en systeemvereisten. Hierbij bekomt men een geheel dat het mogelijk maakt om creatie, distributie, transport, manipulatie en gebruik van digitale media te vergemakkelijken.

De volgende onderdelen van dit hoofdstuk zullen de verschillende aspecten beschrijven van de MPEG 21-standaard. We zullen beginnen met een korte

beschrijving van ieder aspect van de standaard. Hierna zullen we verdergaan met een meer gedetailleerde beschrijving van Part 7 van de standaard, daar deze technieken beschrijft die interessant kunnen zijn voor het gebruik binnen een NVE. We zullen dit hoofdstuk beëindigen met een conclusie.

5.2 MPEG 21 Parts

Part 1 van MPEG 21 geeft een algemene beschrijving van de standaard en het mogelijk gebruik hiervan. Hierbij verstaat men het groeperen van verschillende technologiën die in het expertisedomein van de MPEG-groep passen. De integratie van deze technologiën vormt dan een framework. Concreet betekent dit dat men gaan werken met zogenaamde "Digital Items" (DI). Deze zijn gestructureerde digitale objecten die gedefiniëerd, geïdentificeerd en een metadata hebben binnen de MPEG 21-standaard. Voorbeelden hiervan kunnen de volgende zijn:

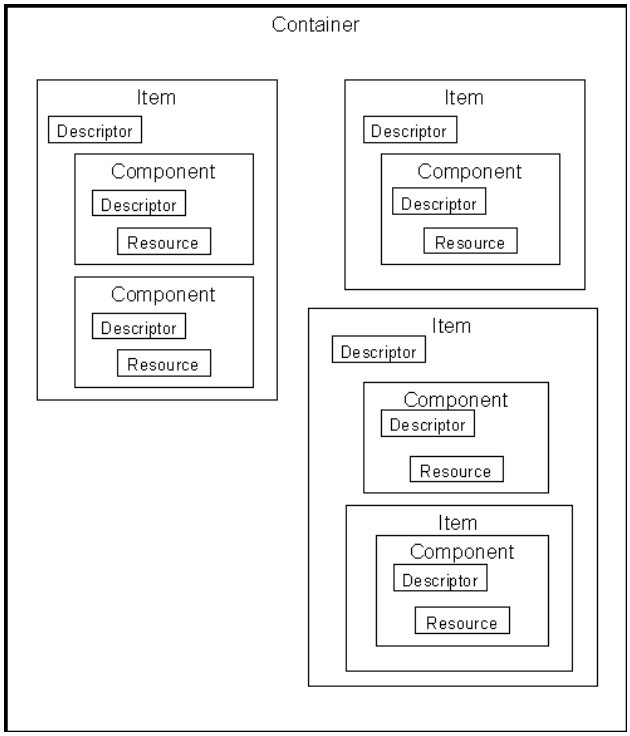
- muziek
- foto's
- video
- animation graphics
- muziektekst
- ...

Part 2 van de MPEG 21-standaard beschrijft de "Digital Item Declaration" (DID). Deze dient als een geabstraheerd model voor het definiëren van een DI. DID bestaat uit drie delen:

- Model: definiëren van abstracte termen en concepten nodig bij het modelleren van DI's
- Representation: de syntax en semantiek van iedere DID-element zoals beschreven in de XML
- Schema: XML-schema voor de gebruikte grammatica

DID maakt gebruik van verschillende onderdelen die een hiërarchische structuur vormen. Deze onderdelen worden met behulp van de Digital Item Declaration Language gedeclareerd [55]. Een *container* bevat andere *containers* en

items en vormt een logisch geheel. De *descriptor* kan meer informatie geven over een *container*. De *items* die bevat worden door *containers*, groeperen andere *items* en/of *components*. Een *item* kan een *choice* bevatten. Dit is een beschrijving van gerelateerde *selections* waarbij men exact één selecteert of een variabel aantal. De hier genoemde *selections* bestaan uit *conditions* binnen een *item*. Deze *conditions* zorgen ervoor dat de status van de *selection* drie waarden, ook wel *predicate* genaamd, kan aannemen: true, false of undecided. Verder spreekt men over een compilate als een *item* sub-items bevat en over een entiteit als deze geen sub-items bevat. *Components* die bevat worden door *items* verbinden een *resource* met zijn *descriptors*. Een *resource* is iedere vorm van digitale media waaronder video en geluid. Een schematisch overzicht van de belangrijkste onderdelen wordt in figuur 5.1 weergegeven.



Figuur 5.1: Schematisch overzicht van de belangrijkste onderdelen van DID [54]

Digital Item Identification (DII), het identificeren van een DI, wordt beschreven in part 3 van de MPEG 21-standaard. DII biedt de nodige hulp-

middelen om een DI te identificeren. Deze identificatie gebeurt binnen de DID-specificatie door gebruik te maken van het *statement*-element dat op zijn beurt door een *descriptor* wordt omvat.

Intellectual Property Management and Protection (IPMP) wordt beschreven in part 4 van de standaard. Deze dient om een bepaalde controle te hebben over de digital items. Concreet houdt het in dat men de gebruiks- en toegangsrechten, beschreven met behulp van REL (zie volgende paragraaf), correct toepast.

Part 5 van de MPEG 21-standaard beschrijft de "Rights Expression Language" (REL). Hiermee wordt het mogelijk om gebruiks- en toegangsrechten te definiëren op digital items. Het REL-model bestaat uit vier onderdelen:

- Principal: dit is de rechthebbende partij waarvan er exact één is. Deze zal zich op een veilige en robuuste manier moeten identificeren.
- Right: dit bevat de handelingen die de *principal* kan uitvoeren op de *resource*
- Resource: dit is het object waarop de *principal* een *right* heeft
- Condition: dit omschrijft de omstandigheden die moeten gelden opdat een *principal* een *right* kan hebben

REL maakt gebruik van de "Rights Data Dictionary" (RDD) om te beschikken over een verzameling duidelijke, consistente, gestructureerde, geïntegreerde en uniek gedefiniëerde termen. RDD wordt beschreven in part 6 van de MPEG 21-standaard.

Part 7 van de MPEG 21-standaard beschrijft het onderdeel van de standaard die schaalbaarheid mogelijk maakt van digital items. Daar deze, voor toepassingen binnen een NVE, interessanter is dan de overige secties van de standaard, zal deze meer gedetailleerd besproken worden in de volgende sectie.

De overige twee parts van de MPEG-standaard zijn "Reference Software" en "File Format". De achtste part, zijnde "Reference Software", bestaat uit een aantal systeemgerelateerde specificaties. Mogelijk zal deze ook een binaire respresentatie van de DID en een specifiek MPEG 21-bestandsformaat omvatten. Het negende en tevens laatste part van de MPEG 21-standaard

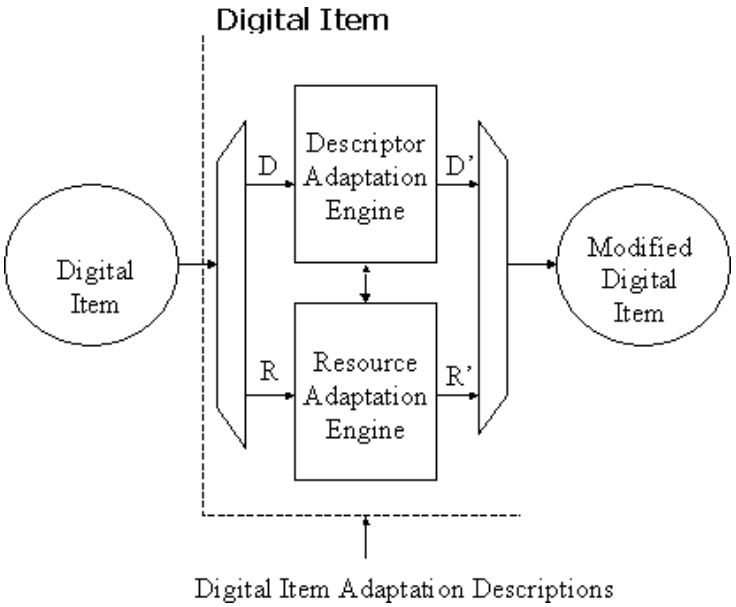
beschrijft uiteindelijk een bestandsformaat die binaire en textuele informatie kan bevatten en compatibiliteit biedt met MPEG 4-software.

5.3 MPEG 21 Part 7: Digital Item Adaptation

5.3.1 Inleiding

"Digital Item Adaptation" (DIA) [56], [57], [58], [59] bepaalt een manier om DI's op een dergelijke manier te transformeren opdat die toegankelijk zijn via diverse netwerk- en systeemarchitecturen. Bij de laatstgenoemden is het vooral belangrijk op te merken dat men hier in het bijzonder denkt aan netwerken met een lage doorvoersnelheid en systemen met een kleiner geheugen en rekencapaciteit.

Om dit doel te bereiken wordt er gebruik gemaakt van twee processen: een *resource adaptation engine* en een *description adaptation engine*. Hieruit ontstaat een DI dat geschikt is om optimaal te werken op de gekozen architectuur en volgens de wensen van de gebruiker. Een schematische weergave van deze twee processen wordt in figuur 5.2 weergegeven.



Figuur 5.2: Schematisch overzicht van DIA [54]

De twee laatstgenoemde processen worden niet bepaald door de MPEG 21-standaard. Deze zijn implementatie-afhankelijk en zodoende vinden ze toepassing in een breed domein, gaande van 3D-modellen tot video en geluid.

DIA heeft de volgende architectuur: [56]:

- Usage Environment Description Tools
  - User Characteristics: voorkeuren van de gebruiker
  - Terminal Capabilities: systeemspecificaties zoals (de)codeermogelijkheden, hardware, schermresolutie, enzoverder
  - Network Characteristics: netwerkspecificaties zoals snelheid, foutmarge, etc
  - Natural Environment Characteristics: eigenschappen zoals locatie, tijd, omgevingslicht en geluid, enzoverder
- Resource Adaptability
  - Bitstream Syntax Description: XML-document dat de bitstream op een hoger niveau beschrijft. Dit document wordt bij de transformatie gebruikt om de oorspronkelijke mediastream aan te passen.
  - Terminal and Network QoS: dit zorgt voor het afwegen van terminal- en netwerkbeperkingen, zodat het uiteindelijk resultaat zo goed mogelijk voldoet aan de gevraagde eigenschappen.
  - Metadata Adaptability: dit dient als leidraad om de complexiteit aan te passen om het DI te verwerken
- Digital Item Declaration Adaptation Tools
  - DIA Configuration (DIAC)
  - Session Mobility: hulpmiddel om de huidige toestand van de gebruiker te behouden eens deze van toestel verandert
  - DID Configuration Preferences (DIACP)
  - DIA Description Messages (DIADM)

Usage Environment Description Tools [60] hebben dus als doel een beschrijving te vormen voor de gebruiksomgeving zoals mogelijkheden van de terminal, netwerkeigenschappen, eigenschappen van de gebruiker en van de natuurlijke omgeving. Deze beschrijving gebeurt in een XML-document waarvan de XML Schema, ter validatie van het XML-bestand, wordt bepaald door de MPEG 21-standaard. Een voorbeeld van een dergelijk document wordt weergegeven in codefragment 5.1. Hierin worden verschillende eigenschappen beschreven zoals de levensduur van de batterij van een bepaald toestel, opslagcapaciteit en input/output-mogelijkheden.

```
<DIA>
   <Description xsi:type="UsageEnvironmentType">
      <UsageEnvironmentProperty
          xsi:type="TerminalsType">
          <TerminalCapability
              xsi:type="PowerCharacteristicsType"
              batteryTimeRemaining="4200" />
          <TerminalCapability xsi:type="StoragesType">
              <Storage xsi:type="StorageType">
                  <StorageCharacteristic
                      xsi:type="StorageCharacteristicsType"
                      inputTransferRate="8" size="1200"
                      writable="true" />
              </Storage>
          </TerminalCapability>
          <TerminalCapability xsi:type="DataIOsType">
              <DataIO xsi:type="DataIOType">
                  <DataIOCharacteristic
                      xsi:type="DataIOCharacteristicsType"
                      busWidth="128" />
              </DataIO>
          </TerminalCapability>
      </UsageEnvironmentProperty>
   </Description>
</DIA>
```

Codefragment 5.1: XML-document bepaald door de Usage Environment Description Tools[60]

Digital Item Declaration Adaptation Tools beschrijven hoe een digital item kan aangepast worden in zijn geheel. Hierbij zorgt Session Mobility ervoor dat de gebruiker gebruik kan blijven maken van het digital item terwijl hij verandert van toestel. DIAC geeft aan op welke manier de de adaptatie moet gebeuren en DIADM zorgt voor de nodige updates van de beschrijvingen in de digital items. DIDCP geeft uiteindelijk aan hoe de configuratie van de digital item moet gebeuren om de voorkeuren van de gebruiker tegemoet te komen.

Digital Item Resource Adaptation Tools gaan de eigenlijke resources, oftewel data, aanpassen.

5.3.2 Bitstream Syntax Description

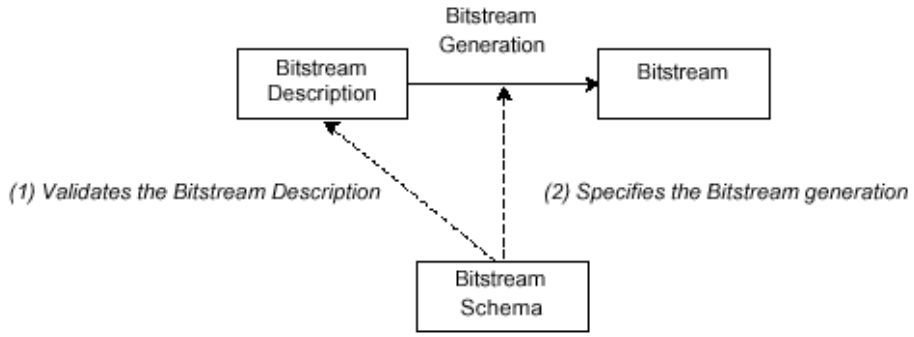
Algemeen

Resource adaptability, hierboven beschreven, heeft een belangrijk onderdeel dat op moment van schrijven het onderwerp is van menig onderzoek, namelijk Bitstream Syntax Description. Een BSD is een XML-document dat binaire data op een hoger niveau beschrijft [61]. Dit betekent niet dat het de binaire data vervangt, maar wel dat het op een gestructureerde manier voorstelt door aan te geven wat de verschillende delen van de binaire data voorstellen. De beschrijving groepeert een bereik aan bytes en zal hiervan aangeven wat deze voorstellen: zo zou men bijvoorbeeld een videostroom kunnen opdelen in de afzonderlijke frames en deze frames beschrijven aan de hand van de verschillende kleurcomponenten. Het detailniveau dat men in de beschrijving wil, is volledig afhankelijk van de implementatie, zodat men hier een grote controle over heeft. Om een BSD te bekomen, maakt men gebruik van een Bitstream (BS) Schema: dit zal later in meer detail worden besproken.

Met behulp van de BS Schema bekomt men een XML-bestand dat een beschrijving vormt van het binair bestand. Deze kan hierna worden getransformeerd met behulp van de *description adaptation engine*. De *description adaptation engine* kan een XSLT-document zijn, wat in de praktijk ook meestal het geval is, maar daar het niet vastgelegd is in de MPEG 21-standaard, kan dit variëren. Uiteindelijk bekomt men een XML-bestand die de inhoud beschrijft van een binair bestand dat men zou willen bekomen. Deze is echter nog onbestaand en zal ontwikkeld worden met behulp van de volgende stap waarbij men de binaire data aanpast.

Na de transformatie van het XML-bestand zal men vervolgens hiermee de binaire data aanpassen. Doordat in het XML-bestand, oftewel aangepaste BSD, nu aangegeven is welke delen van het originele bestand men wil aanpassen, zal de transformatie van dit laatste eenvoudig verlopen. Men zal uit het originele binair bestand enkel die delen nemen en deze gebruiken zoals beschreven in het nieuw XML-bestand: hierbij zal de data aangepast en gekopiëerd worden naar het nieuw bestand dat uiteindelijk het resultaat vormt.

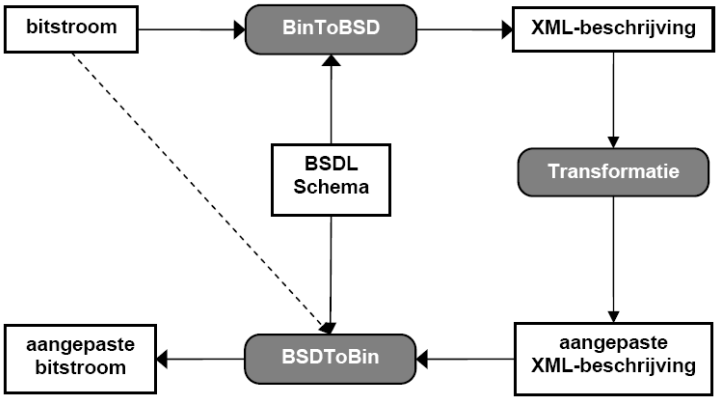
Bitstream Syntax Description Language (BSDL) is de taal die ontwikkeld werd om BSD's op te stellen. Deze taal werd ontwikkeld bovenop XML Schema. Naast de functionaliteiten die XML Schema biedt, namelijk het valideren van inhoud en structuur van een XML-bestand, werd ook gedacht aan een beschrijving die aangeeft hoe de transformatie van binair bestand naar BSD en omgekeerd, moet gebeuren. Het XML Schema dat men bekomt met behulp van BSDL wordt Bitstream (BS) Schema genoemd. Dit wordt verduidelijkt in figuur 5.3



Figuur 5.3: Het genereren van binaire data mbv BDSL [62]

De concrete werking van het bovenstaand proces gebeurt aan de hand van een zogenaamde *BinToBSD* en een *BSDToBin parser*. De BinToBSD-parser zal met behulp van de hierboven beschreven BSD, een XML-bestand genereren dat de binaire data beschrijft. BSDToBin zal het omgekeerde werk verrichten, waarbij men met behulp van het getransformeerde XML-bestand de aangepaste binaire data genereert. In een concrete implementatie van de BSDToBin zal deze ook de nodige programmatuur bevatten om een XML-bestand zelf te transformeren. Meer specifiek zal dit neerkomen op het integreren van bijvoorbeeld een SAX-parser die XSLT-transformaties mogelijk kan maken. Dit betekent dat de BSDToBin-parser als input een XML-bestand, een XSLT-bestand en de binaire data zal krijgen. Het XSLT-bestand is hier een mogelijke implementatie van de description adaptation engine. Het verloop van dit proces wordt grafische weergegeven in figuur 5.4.

Het is belangrijk op te merken dat de twee componenten, de BSDToBin en BinToBSD-parsers, volledig onafhankelijk zijn van binaire data. Aangezien men een beschrijving heeft van de binaire data in de vorm van een BSD



Figuur 5.4: Transformatieproces met behulp van BSD(L) [62]

en de transformatie die erop moet worden toegepast, zijn deze twee componenten voor tal van doeleinden bruikbaar. De inhoud van de binaire data kan alle vormen aannemen zoals een videostroom maar ook een foto: met een gepaste BSD, die bekomen wordt met behulp van een specifiek BS Schema, en transformatie kan men één en dezelfde description adaptation engine gebruiken.

Zoals eerder vermeld, vormt BSDL een extensie op XML Schema. Er zijn hierbij twee extensies: Schema for BSDL-1 Extensions en Schema for BSDL-2 Extensions.

Schema for BSDL-1 Extensions zorgt dat men, buiten de reeds aanwezige datatypes in XML Schema, ook een aantal andere belangrijke datatypes kan gebruiken. Deze datatypes zijn nodig om semantische eigenschappen aan te geven die nodig zijn voor de beschrijving van binaire data. Datatypes die geen betekenis hebben binnen de context van binaire aanpassingen, worden weggelaten.

Belangrijke beperkingen die horen bij BSDL-1 Extensions zijn de volgende:

- binaire data die wordt opgenomen in het XML-bestand (bv een aantal bytes die het begin van een frame aangeven) mogen enkel als XML-element voorkomen en niet als attribuut. De reden hiervoor is dat de volgorde van belang is: de validatie van een XML-bestand controleert enkel op volgorde van de afzonderlijke elementen en niet op de volgorde van de attributen.

- mixed content is niet toegestaan, de binaire data moet van één bepaald type zijn, bv audio scheiden van video
- elk element moet een datatype hebben

Schema for BSDL-2 Extensions introduceert nieuwe constructies zoals variabelen en conditionele operatoren. Opdat deze uitbreidingen geldig blijven onder de XML Schema-specificatie, werden deze opgenomen als applicatie specifieke delen door middel van *xml::appinfo*. Attributen werden toegevoegd met behulp van zogenaamde namespaces. De uitbreidingen die mogelijk worden gemaakt door BSDL-2 zijn van belang voor de BinToBSD-parser.

Een belangrijke toevoeging aan de mogelijke datatypes (int, double, long, etc) is het datatype *byteRange*. Dat datatype zorgt ervoor dat men bepaalde delen van de binaire data kan groeperen in een korte beschrijving van twee waarden. De eerste waarde geeft de start aan van het bereik dat men wil aangeven, en de tweede waarde de grootte van het bereik.

Voorbeeld van een BSD en BSDL

Als voorbeeld van BSD kan men een eenvoudige MPEG4-videostroom bekijken die de Visual standaard voor videocodering volgt [62]. Kennis van de videostroom geeft een opsplitsing in I-, B- en P-frames waarmee de opsplitsing zal gebeuren in de beschrijving van het bestand. Iedere frame wordt aangegeven als Video Object Plane oftewel VOP. Met behulp van een geschikte BSD bekomt men een beschrijving zoals getoond in codefragment 5.2. Hierbij kan men zien dat er drie verschillende waarden zijn voor *VOP_coding_type*: 0 verwijst naar een I-beeld, 1 naar een P-beeld en 2 naar een B-beeld. Een belangrijk element van iedere VOP is de payload. De payload beschrijft het interval in de videostroom die behoort tot een bepaalde VOP. De beschrijving bestaat uit steeds twee getallen: het eerste getal geeft de offset aan in bytes vanaf het begin van de videostroom en het tweede getal geeft de grootte aan van het interval horende bij een bepaalde VOP.

```
<VOP>
  <VOP_code>000001B6</VOP_code>
  <VOP_coding_type>0</VOP_coding_type>
  <stuffing>16</stuffing>
  <payload>46 5747</payload>
</VOP>
<VOP>
  <VOP_code>000001B6</VOP_code>
  <VOP_coding_type>1</VOP_coding_type>
  <stuffing>17</stuffing>
```

```
  <payload>5798 1119</payload>
</VOP>
<VOP>
  <VOP_code>000001B6</VOP_code>
  <VOP_coding_type>2</VOP_coding_type>
  <stuffing>16</stuffing>
  <payload>6922 150</payload>
</VOP>
```

Codefragment 5.2: Gedeeltelijke BSD van een MPEG4-stroom. [62]

Eens er een beschrijving van de bitstroom beschikbaar is, kunnen transformaties worden toegepast. Een manier om dit toe te passen is gebruik te maken van XSLT. In dit voorbeeld werd er gekozen om de B-frames weg te laten die voorkomen in een MPEG4-videostroom. Dit aangezien overige frames niet afhankelijk zijn van de B-frames, wat in het geval van I- en P-frames wel zo is. We zien in codefragment 5.3 dat er B-frames worden gedetecteerd (iedere VOP met coding type 2) en hierbij expliciet niets wordt gedaan. Aangezien de overige type frames niet worden vermeld, zullen deze worden overgenomen. Na deze transformatie zal er een BSD worden bekomen waarbij iedere VOP weggelaten wordt die een coding type 2 heeft. Eens deze transformatie is doorvoerd op de originele data met behulp van de BSDToBin-parser, zal een MPEG4-videostroom worden bekomen van minder grote omvang dankzij het weglaten van B-frames.

```
<xsl:template name="tplB_VOP"
    match="m:VOP[m:VOP_coding_type=2]">
    <!-- do nothing -->
</xsl:template>
```

Codefragment 5.3: XSLT-transformatie waarbij men B-frames verwijderd uit een videostroom [62]

5.3.3 Generic Bitstream Syntax Schema

Generic Bitstream Syntax Schema (gBSS) heeft als doel een verbetering te zijn voor BS Schema [61]. Een belangrijk verschil is dat een gBS Schema zelf implementatie-onafhankelijk is. Dit wil dus concreet zeggen dat één en dezelfde gBS Schema kan gebruikt worden voor verschillende doeleinden. Zoals we in deze sectie zullen zien, wordt er onder andere gebruik gemaakt van markers, waardoor een beschrijving niet meer specifiek voor één type encodering van toepassing is. Dezelfde beschrijving kan dan gebruikt worden voor bijvoorbeeld een MPEG 2- en een MPEG 4-videostroom. Dit doel wordt bereikt met de volgende functionaliteiten:

- Belangrijke elementen in het document aanduiden met een *marker*.
- Hiërarchische indeling van de beschrijving van de binaire data, waardoor een efficiënte verwerking mogelijk wordt.
- Flexibele adressering zodat men steeds kan voldoen aan verschillende applicatievereisten en men makkelijk delen van de binaire data kan terugvinden.

gBSS definiëert drie type elementen:

- Header: algemene informatie over het document waaronder het aangeven van de gebruikte elementnamen en informatie nodig om de binaire data te kunnen verwerken
- gBSDUnit: een deel van de binaire data. Deze wordt aangegeven door de locatie van dit deel en niet door de eigenlijke binaire data. De informatie hier gegeven, verwijst naar data die niet aangepast zal worden.
- Parameter: data die aangepast moet worden. Hier geeft men, in tegenstelling tot gBSDUnit, de eigenlijk binaire data.

De laatste twee elementen kunnen gebruikt worden in combinatie met een *marker*. Dit zou het bijvoorbeeld mogelijk maken om, in combinatie met MPEG-7, gewelddadige scènes uit een film te verwijderen.

De verwerking van de data gebeurt op de volgende manier: uit de binaire data wordt initiëel een BSD gegenereerd zoals al eerder werd besproken. Hier wordt dus niet gebruik gemaakt van een specifieke parser die binaire bestanden direct omzet naar een gBSD. De bekomen BSD transformeert men naar een gBSD met behulp van een XSLT-transformatie, of in enkele gevallen zelfs manueel. Bij deze transformatie kan men bijvoorbeeld rekening houden met reeds verzamelde gegevens, bijvoorbeeld het gebruik van MPEG 7 om aan te geven waar de gewelddadige scènes zich bevinden die men wilt verwijderen. Eens er een gBSD is bekomen, zal een tranformatie worden toegepast die de gBSD aanpast zodat het binaire data beschrijft die voldoet aan de gestelde eisen. Een mogelijke manier om deze aanpassing toe te passen, is door gebruik te maken van een XSLT-transformatie. Deze laatste vormt dan de *description adaptation engine* en beslist welke veranderingen er moeten plaatsvinden. De gBSD zal dan worden gebruikt om de oorspronkelijke binaire data aan te passen waardoor men het gewenste resultaat bekomt. Dit gebeurt dan wel met een specifieke gBSDToBin-parser.

Ter illustratie van het gBSD-proces, kan men codefragment 5.4 bestuderen dat een gBSD-schema weergeeft voor een bepaald videofragment. Men kan hier het hiërarchisch karakter van de beschrijving herkennen aan het gebruik van de gBSDUnits. gBSDUnits zijn speciale datatypes binnen een gBSD-schema die het mogelijk maken om delen van de binaire data te groeperen. Al de frames, die zelf als afzonderlijke gBSDUnits worden aangeduid, behorende tot een scène, worden gegroepeerd in een gBSDUnit en hierbij hoort een marker dat als hulpmiddel dient bij de transformatie. In dit geval zien we dat de marker een waarde heeft die aangeeft dat we te maken hebben met een geweldadige scène. Zoals eerder besproken, kan men deze informatie bekomen met behulp van MPEG-7. Als men nu wenst bepaalde scènes te verwijderen uit een videofragment, zal het enkel nodig zijn om te zoeken naar scènes die met deze marker worden aangeduid.

```
<dia:DIA>
  <dia:Description xsi:type="gBSDType"
          addressUnit="byte" addressMode="Absolute"
          bs1:bitstreamURI="content/bsone.cmp">
    <gBSDUnit syntacticalLabel=":M4V:VO"
            start="0" length="4" />
    <gBSDUnit syntacticalLabe=":M4V:VOL"
            start="4" length="14" />
    <gBSDUnit start="18" length="520176"
            marker="ICRAParentalRatingViolenceCS-2">
      <gBSDUnit syntacticalLabel=":M4V:I_VOP"
              start="18" length="13522" />
      <gBSDUnit syntacticalLabel=":M4V:P_VOP"
              start="13540" length="15128" />
      <gBSDUnit syntacticalLabel=":M4V:B_VOP"
              start="28668" length="2734" />
      <gBSDUnit syntacticalLabel=":M4V:B_VOP"
              start="31402" length="2714" />
    </gBSDUnit>
  </dia:Description>
</dia:DIA>
```

Codefragment 5.4: gBSD-schema dat geweldadige scènes uit een film verwijdert[61]

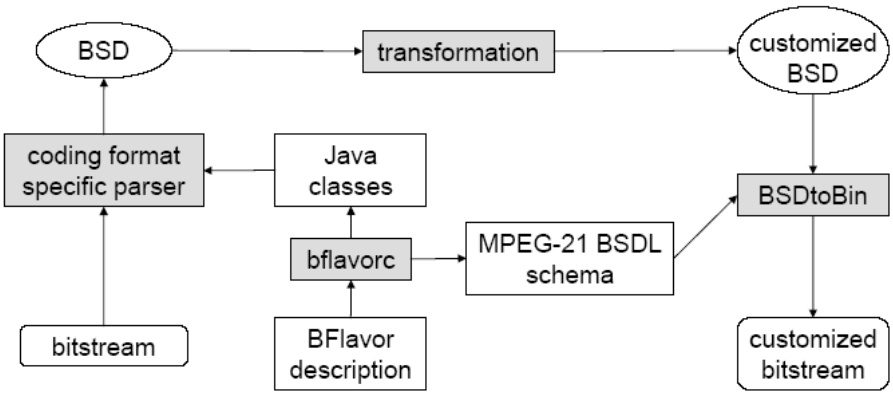
5.3.4 BFlavor

Eerder in dit hoofdstuk werden BSD en gBSD beschreven als talen geschikt voor de beschrijving van bitstreams en behorende tot de MPEG21-standaard. Deze twee hebben echter enkele punten die als nadeel ervaren worden. De parser die ervoor zorgt dat men van een bitstream een beschrijving bekomt heeft een trage uitvoersnelheid en een hoog geheugenconsumptie. Dit laatste

valt te wijten aan het feit dat men de originele bitstroom in het geheugen moet houden om de aanpassingen te kunnen doorvoeren. Verder kan men met de gebruikte taal geen constructies vormen die wel mogelijk zouden zijn in een taal zoals C++ of Java.

Naast de schaalbaarheidmogelijkheden die de MPEG21-standaard biedt, is er nog een mogelijkheid om dit te bekomen. XFlavor werkt op een gelijkaardige manier als BSD. Van de bitstroom wordt er ook een beschrijving opgesteld, deze bevat echter ook de volledige bitstroom. Dit zorgt ervoor dat de beschrijvingen zeer groot worden, uiteraard afhankelijk van het origineel bestand. XFlavor biedt echter een voordeel ten opzichte van BSD: het is ontwikkeld als een extensie voor C++ en Java. Hierdoor moet men niet meer teruggrijpen naar de constructies die in een BSD worden gebruikt.

Vertrekkende vanuit bovenstaande opmerkingen, heeft men BFlavor ontwikkeld [63]. BFlavor is bovenop XFlavor ontwikkeld. Een beschrijving van de bitstroom wordt doorgegeven aan *bflavorc* waarmee een aantal Java-klassen en een BSDL schema mee wordt ontwikkeld. De Java-klassen worden gebruikt voor het ontwikkelen van een specifieke parser voor het coding formaat. Uiteindelijk kan men met de bekomen BSD de gebruikelijke transformaties toepassen die eerder werden beschreven. De stap terug van BSD naar binaire data wordt verkregen door gebruik te maken van de reeds bestaande BSDToBin-parser. Figuur 5.5 verduidelijkt het beschreven proces.



Figuur 5.5: BFlavor-proces [63]

5.3.5 Toepassingen voor video

Het aanpassen van een videostroom kan op verschillende manieren gebeuren. Bij iedere mogelijkheid is het echter belangrijk om aan te duiden wat de invloed hiervan is op de kwaliteit en de complexiteit van dergelijke bewerkingen. De complexiteit van de bewerkingen is belangrijk als men de bewerkingen in real-time wil doen gebeuren. In de volgende paragrafen zullen de verschillende manieren aan bod komen om een videostroom aan te passen en dit voornamelijk voor video-encoderingen horende tot de MPEG-4 standaard. Verder zal ook de aandacht gevestigd worden op een recente ontwikkeling op dit gebied: MPEG-4 Part 10.

Schaalbaarheid kan op verschillende manier mogelijk worden gemaakt. Temporele schaalbaarheid en met name *frame dropping* is een populaire techniek die weinig computationele kracht vergt. Bij het verwijderen van frames zijn de volgende punten van belang [64]:

- resterende kwaliteit nadat men het frame verwijderd heeft
- distributie van de te verwijderen frames over het tijdsdomein
- de framegrootten
- het belang van de te verwijderen frames

Verwijderen op willekeurige of vaste tijdstippen biedt niet altijd het beste resultaat. Bewegingen komen in bepaalde gevallen niet natuurlijk over en verwijderen van specifieke frames kan een grotere besparing opleveren dan het verwijderen van een groep frames. Daarom kan het nuttig zijn bepaalde algoritmes te gebruiken die voorgaande punten afweegt bij het verwijderen van een frame.

Quality Controlled Temporal Video Adaptation (QCTVA) [64] verwijdert uit een videostroom bepaalde B-frames rekening houdend met de resulterende kwaliteit. Er worden enkel B-frames verwijderd omdat ze in een videostroom het enige type frames zijn waarvan de overige frames niet afhankelijk zijn.

Om de kwaliteit van een groep frames te evalueren, kan men gebruik maken van de Peak Signal-to-Noise Ratio (PSNR). De PSNR vergelijkt de kwaliteit van een frame genomen uit de gecodeerde videostroom en het bijhorende

ongecodeerde frame. Hiermee bekomt men één getal dat eenvoudig valt te berekenen met de volgende formule:

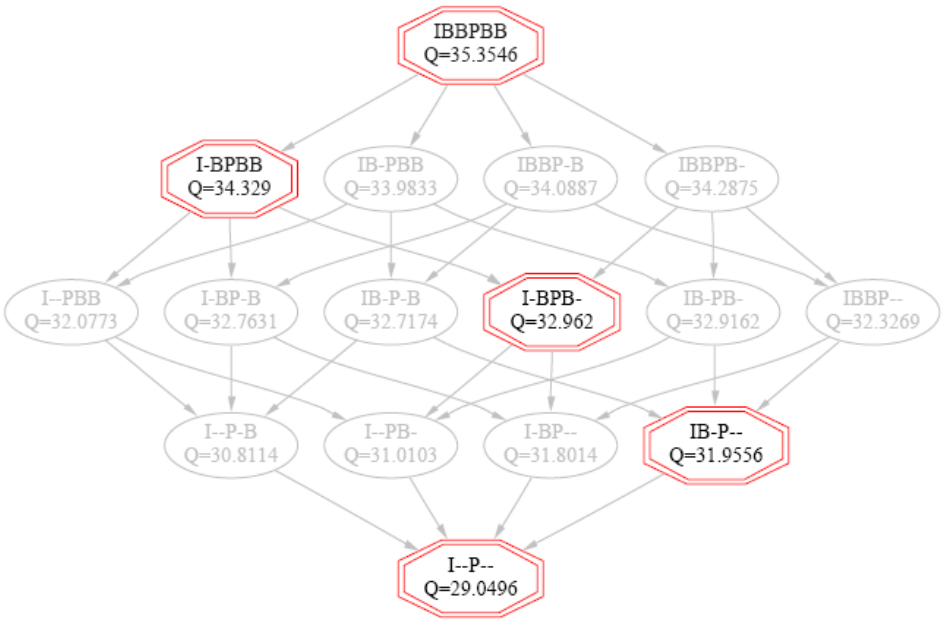
$$psnr(I, L) = 20 \log_{10} \left(\frac{255}{\sqrt{\frac{1}{XY} \sum_{x=1}^{X} \sum_{y=1}^{Y} (I_{x,y} - L_{x,y})^2}} \right) \tag{5.1}$$

waarbij I het originele frame is, L het gecomprimeerde en x en y de coördinaten voorstellen van de pixels. Er is hier dus een verband tussen de berekende PSNR en de kwaliteit: hoe hoger de PSNR, hoe hoger de kwaliteit. De uiteindelijke kwaliteit van een groep frames kan vervolgens met de volgende formule worden berekend:

$$Q_P = \frac{\sum_{i=1}^{n} psnr(F_i, G_i)}{n} \tag{5.2}$$

Met behulp van formules 5.1 en 5.2 kan men nu de kwaliteit berekenen van een bestaande groep frames en deze herberekenen nadat men één of meerdere frames verwijderd heeft. Men kan uit een bestaande groep frames, bijvoorbeeld IBBPBB, verschillende combinaties vinden waarbij men één of meerdere B-frames verwijdert. Deze verschillende combinaties kan men opstellen in een zogenaamde *modification lattice*. Hierbij gaat men per niveau een toenemend aantal B-frames verwijderen, wat dus betekent dat men begint bij al de combinaties waarbij men één B-frame verwijdert en eindigt bij een sequentie waar er geen enkele B-frame meer overschiet. Zoals men ziet in figuur 5.6, kan men op ieder niveau nu de beste sequentie uitkiezen die de beste kwaliteit levert bij een vast aantal verwijderde B-frames. Daar er nu voor ieder niveau van de lattice geweten is welke sequentie de beste kwaliteit biedt, is het doel van het QCTVA-algoritme bereikt.

Schaalbaarheid kan echter ook op een andere manier bekomen worden. Men kan dit bereiken met behulp van H.264/Advanced Video Coding beschreven in MPEG-4 Part 10 [65]. H.264/AVC verdeelt een stroom op in slices die samen een frame vormen. Door middel van deze slices bekomt men sequenties door een stroom heen, vergelijkbaar met een framesequentie in andere videostandaarden. In de context van H.264/AVC spreekt men dan ook van P-, I- en B-slices. Het is echter niet mogelijk om eenvoudigweg B-slices horende tot een subsequentie te verwijderen, omdat deze gebruikt worden om andere slices te reconstrueren (dit in tegenstelling tot B-frames waar andere frames onafhankelijk van zijn). Wat wel mogelijk is, is om volledige subsequenties te verwijderen. Dit is mogelijk omdat de slices binnen een frame totaal onafhankelijk zijn van elkaar en men deze hierdoor makkelijk



Figuur 5.6: De modification lattice geeft voor ieder niveau de beste sequentie aan [64]

kan verwijderen. Informatie over de aanwezigheid van deze subsequenties kan bijgehouden worden in Supplemental Enhancement Information (SEI). Deze SEI-berichten worden dus een cruciaal onderdeel bij de transformatie van dergelijke videostromen.

5.3.6 Toepassingen voor 3D-formaten

Tot nu toe werd de schaalbaarheid van binaire data met behulp van MPEG 21 enkel besproken in de context van videostromen. Dit is echter een onnodige beperking, daar men gBS/BS Schema's en daarbij horende transformaties voor verschillende domeinen kan gebruiken. Een ander domein dat in aanmerking komt, is de binaire voorstelling van 3D-objecten [66]

Een methode bestaat eruit om een model in te delen in clusters zodat vereenvoudiging van een model op een eenvoudige manier kan gebeuren: men verwijdert bepaalde clusters. Daar een model bestaat uit een verzameling vertices V en faces F, kan men voor iedere cluster een deelverzameling nemen van deze twee verzamelingen. De modellen die men bekomt, kan men ordenen

in afnemende complexiteit, zijnde M_n, M_{n-1}, ... , M_1, M_0. Hierbij geldt dat:

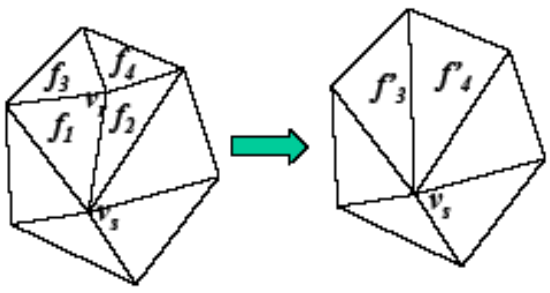
$$M_i = M_0 \cup (\cup_{j=1}^{i} C(j) \cap \cup_{j=1}^{i} N(j)) \tag{5.3}$$

In formule 5.3, stelt C(i) de verzameling vertices en faces voor die uit een model M_i verwijderd worden om een simplificatie te bekomen. N(i) stelt de nieuwe verzameling vertices en faces voor die men bekomt nadat men C(i) verwijderd heeft uit een model M_i. Deze laatste verzameling bekomt men op twee mogelijke manieren: met behulp van half edge-collapsing operators en Quadric Error Metrics.

De half edge-collapsing operator houdt in dat men een kant vervangt door een vertex. In figuur 5.7 ziet men dat men links vier faces heeft, f_1 tot en met f_4 en een kant bestaande uit de vertices v_r en v_s. Door de kant (v_r, v_s) te vervangen door de vertex v_s zal de vertex v_r verdwijnen en blijven er van de vier oorspronkelijke faces, enkel twee faces, f_3' en f_4', over. Hierbij is C(i) dus de verzameling bestaande uit de elementen f_1, f_2, f_3 en f_4 terwijl N(i) bestaat uit f_3' en f_4'. Uit formule 5.3 volgt dat N(i) een deelverzameling is van de unie van de verzamelingen M_0, C(1), ... , C(i-1). Hieruit volgt dat een cluster C(i) kan opgedeeld worden in clusters C(i,j) en C(i,i). De tweede term in C(i,j) duidt op de verzameling N(j) waartoe deze behoort, gebruikmakend van het gevolg van formule 5.3. Formule 5.3 kan dus als volgt worden herschreven:

$$M_i = \cup_{k=0}^{i} C(k,k) \cap \cup_{j=i+1}^{n} C(k,j) \tag{5.4}$$

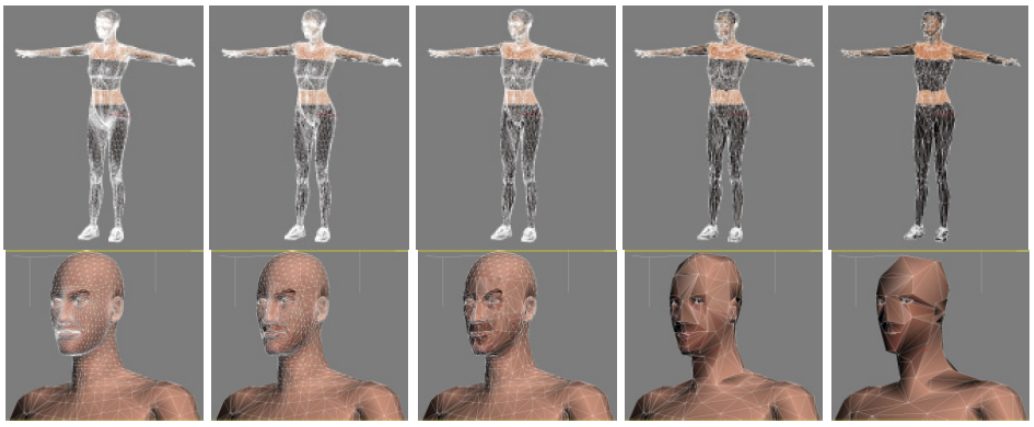
In deze formule komt M_0 niet meer expliciet voor, maar geldt $M_0 = C(0)$.



Figuur 5.7: Half edge-collapsing operator [66]

Door het bovenstaande toe te passen op een model, bekomt men verschillende niveau van detail. Ieder niveau bestaat uit een verzameling clusters C(x, y) waarbij x het niveau aangeeft. Door deze clusters dus te groeperen aan de hand van hun niveau, kan men de vereenvoudiging eenvoudig toepassen. De vereenvoudiging, oftewel de selectie niveau die men bekomt na de transformatie van de BSD, bestaat dus uit het nemen van de clusters horende bij een bepaald detailniveau. Hierbij moet men zich wel bedenken dat men nog geen rekening houdt met andere variabelen horende bij een mesh, zoals kleur en texture coördinaten. Deze zijn echter afhankelijk van de vertices waardoor het proces hetzelfde wordt, op twee details na: meerdere vertices gebruiken dezelfde waarde voor een eigenschap en een vertex waarbij meerdere eigenschappen zijn gedefiniëerd.

Een voorbeeld van het resultaat dat men bekomt, gebruikmakend van de bovenstaande techniek, is weergegeven in figuur 5.8. Men ziet hier duidelijk dat men uit het initiëel model links, steeds ruwere modellen bekomt door het bovenstaand algoritme toe te passen.



Figuur 5.8: Een voorbeeld van schaalbaarheid met 3D-modellen [66]

5.3.7 Toepassingen voor textures

Een toepassing voor textures [67] baseert zich op het principe van view-dependent streaming. Een texture die gebruikt wordt op een model zal in bepaalde omstandigheden slechts gedeeltelijk zichtbaar zijn. De delen die zichtbaar zijn zullen bovendien niet allemaal even gedetailleerd moeten zijn, omdat men kan rekening houden met bijvoorbeeld de afstand tot de camera.

Binnen de MPEG 4-standaard bestaat de zogenaamde Visual Texture Coding. Dit is een schaleerbaar, wavelet-gebaseerd compressie-algoritme voor textures. Het maakt gebruikt van zes modes die ontstaan uit de combinatie van drie quantizatie-modes en twee beeld-scanning modes. Hierbij worden de wavelets hiërarchisch geordend met opeenvolgende quantizatiestappen. Belangrijk om op te merken is dat de data bestaat uit pakketten waarbij ieder pakket een bepaald deel van de texture aan een welbepaalde kwaliteit bevat. Ieder pakket is hierbij ook nog aangeduid met een specifieke bitsequentie.

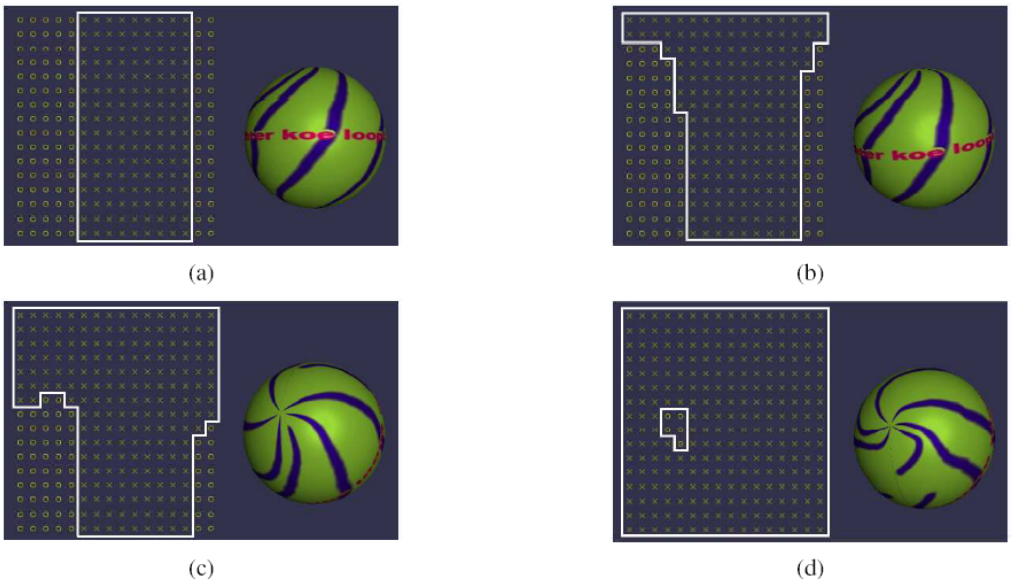
Nadat men een BSD heeft opgesteld, rekening houdend met de markers aanwezig in een VTC-datastream, kan men de transformatie toepassen. Hierbij kan men een selectie maken van rechthoekige delen van de texture waarbij men de kwaliteit zelf kan bepalen. In figuur 5.9 ziet men een bal die men roteert en waarbij deze animatie 50 frames beslaat. Bij iedere frame zal de region of interest veranderen en zal men steeds meer delen van de texture hebben opgeslagen. Aan het einde van de animatie zal men bijna de hele bal hebben gezien waardoor ook bijna de hele texture gedecodeerd is.

In een client-server-architectuur zou betekenen dat clients enkel de voor hun zichtbare delen van de textures zouden streamen. De gestreamde delen zouden bovendien voldoen aan de vooraf gekozen kwaliteit.

5.4 Gebruik binnen een NVE

In moderne NVE's wordt vaak gebruik gemaakt van verschillende soorten media en dit voor gebruikers die beschikken over verschillende apparatuur en netwerkconnecties. Om deze media beschikbaar te stellen aan al deze gebruikers, is het noodzakelijk dat deze aangepast worden aan de noden van de gebruikers. Hiervoor biedt MPEG 21 een krachtig hulpmiddel.

In dit hoofdstuk hebben we gezien hoe men met behulp van MPEG 21 videostromen, 3D-objecten en textures kan aanpassen. In NVE's wordt dit soort data veelvuldig gebruikt: videostromen worden gebruikt opdat gebruikers elkaar kunnen zien, 3D-objecten om de virtuele omgeving te visualiseren en textures om deze objecten een bepaald uitzicht te geven. De mogelijkheid die MPEG 21 biedt om deze data te schaleren, zorgt ervoor dat men kan vertrekken vanuit een vaste groep bestanden die men vervolgens on-the-fly kan aanpassen indien hier vraag naar is. Zo kan een gebruiker die over een zwakke netwerkverbinding beschikt, alsnog videostromen ontvangen doordat



Figuur 5.9: BSDL-toepassing voor textures ((a) frame 1, (b) frame 16, (c) frame 32, (d) frame 50) [67]

deze via MPEG 21 aangepast kunnen worden zodat ze een lagere bitrate gebruiken.

MPEG 21 is tot nu toe echter nooit gebruikt in een NVE. Dit omdat MPEG 21 nog steeds in ontwikkeling is en er nog geen SDK's bestaan voor deze standaard. In de toekomst zal hier ongetwijfeld verandering in komen aangezien de vele mogelijkheden van MPEG 21, en DIA in het bijzonder, nuttig zijn in deze context.

5.5 Conclusie

MPEG 21 biedt een uitgebreid framework voor multimediatoepassingen. DIA, beschreven in de MPEG 21-standaard, concentreert zich op verschillende aspecten ter bevordering van de toegankelijkheid van multimediale data. Dit houdt concreet in dat de data aangepast kan worden aan de noden van de gebruiker.

Een belangrijk onderdeel van de standaard is DIA en de daarbij horende technieken. Zoals besproken in dit hoofdstuk, kan met met behulp van

BSD de data aanpassen met gebruik van XML-beschrijvingen van de data en XSLT-transformaties om deze data aan te passen. Men heeft gezien hoe men hiermee videostromen, 3D-figuren en textures kan aanpassen, en dit steeds met dezelfde basistechnieken.

Het uiteindelijk doel van MPEG 21 DIA is een antwoord te bieden aan de heterogeniteit die men terugvindt bij de gebruikers. Het gebruik van PDA's wordt bijvoorbeeld alsmaar populairder en lang niet alle gebruikers beschikken over dezelfde netwerkmogelijkheden. MPEG 21 DIA zal hier dan een oplossing voor bieden door, vertrekkende van dezelfde data, deze aan te passen voor iedere soort gebruiker.

Hoofdstuk 6

Implementatie

6.1 Inleiding

In de vorige hoofdstukken hebben we een aantal standaarden besproken die kunnen worden gebruikt in een NVE. In dit hoofdstuk zullen we zien hoe dit kan gebeuren en welke voordelen dit biedt. We zullen dit doen aan de hand van twee testimplementaties waarin enkele standaarden worden gebruikt die we eerder in deze thesis zijn tegengekomen.

We zullen beginnen met het bespreken van een eerste testimplementatie waarin het duidelijk wordt hoe MPEG 21 praktisch werkt. Hierna gaan we verder met een toepassing waarin het mogelijk wordt om een virtuele omgeving aan te passen aan de noden van de gebruiker. We zullen zien hoe dit mogelijk is en wat dit als voordeel heeft ten opzichte van een vooraf bepaalde scène. We zullen dit hoofdstuk beëindigen met een conclusie.

6.2 Implementatie 1: MPEG 21 BSDL

Het doel van deze implementatie is een voorbeeld te geven van de mogelijkheden van MPEG 21 BSDL. Gegeven een binair bestand en daarbij horende BS Schema en transformatie, kan met met behulp van deze implementatie het binair bestand transformeren. In het geval van videobestanden of afbeeldingen is het tevens mogelijk om het origineel bestand en getransformeerd bestand te bekijken.

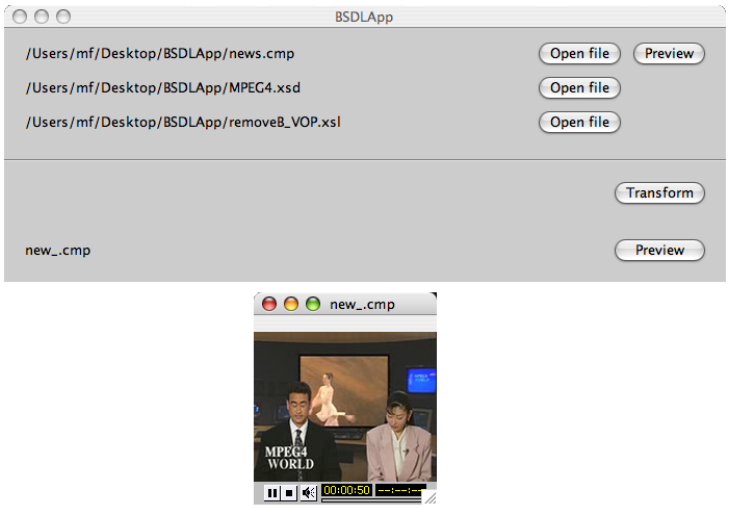
Deze implementatie werd gemaakt op basis van de MPEG 21 Reference Software die werd ontwikkeld om de mogelijkheden van de standaard duidelijk te maken. De reference software van MPEG 21 is een verzameling

Java-klassen waarmee men de verschillende operaties kan uitvoeren horende bij een DIA-implementatie. De software biedt twee afzonderlijke applicaties: de BinToBSD- en BSDToBin-parsers zoals beschreven in sectie 5.3.2. De eerste applicatie zorgt ervoor dat men vanuit een binair bestand en een BS Schema, een XML-beschrijving bekomt van het binair bestand. De gegenereerde XML-beschrijving moet men dan, samen met het origineel binair bestand en de gewenste transformatie, meegeven aan de tweede applicatie om de eigenlijke transformatie uit te voeren. Men kan inzien dat een dergelijke aanpak met intermediaire bestanden voor een vertraging zorgt doordat er continue toegang tot de harde schijf nodig is. Om deze reden zijn er voor de implementatie enkele wijzigingen gemaakt waardoor het proces vlotter kan verlopen.

Om van de twee afzonderlijke implementaties één geheel te maken, werd allereerst de interface van beide implementaties verwijderd. Hierdoor bekomt men enkel de kern van de implementaties waarmee men de transformaties kan uitvoeren. Met deze kern werd er één implementatie ontwikkeld welke als invoer een binair bestand, een BS Schema en een transformatie nodig heeft en hieruit rechtstreeks het uiteindelijk getransformeerd binair bestand genereert. Het is dus niet meer nodig om een XML-parser te gebruiken om de tussentijdse beschrijving op te slaan aangezien deze in het geheugen wordt gehouden en rechtstreeks getransformeerd.

In deze implementatie werd er dus voor gezorgd dat het volledige MPEG 21 DIA proces in één stap gebeurt voor de gebruiker. Het is dus enkel nodig om een binair bestand te kiezen, samen met de bijhorende XML Schema en XSLT-transformatie. In figuur 6.1 zien we de interface van deze implementatie.

Hiermee kan men het voorbeeld bekeken die al eerder in sectie 5.3.2 aan bod kwam [62]. Een MPEG 4-videobestand neemt men als input en door middel van een transformatie bekomt men het videobestand waarbij de B-frames zijn verwijderd. Eens de transformatie is gebeurd, kan men dit bestand bekijken met behulp van de ingebouwde videospeler.



Figuur 6.1: Voorbeeldapplicatie MPEG 21 BSDL

6.3 Implementatie 2: MPEG 4 en 21 binnen een NVE

De doelstelling van deze implementatie is het aantonen van de bruikbaarheid van verschillende standaarden in de context van een NVE. De implementatie vormt een basis die uitgebreid kan worden naar een genetwerkte virtuele omgeving waarin een scène beschreven en getransformeerd kan worden aan de noden van de gebruiker. Zoals we in de volgende secties zullen zien, wordt er rekening gehouden met verschillende eigenschappen van de gebruikers, om een scène te bekomen die op maat gemaakt is voor de clients. Een dergelijke aanpak kan belangrijk zijn in een NVE waarbij de gebruikers niet over dezelfde systemen (bijvoorbeeld PDA en desktopcomputer) en netwerkconnecties (bijvoorbeeld inbelverbinding en lokaal netwerk) beschikken.

De standaarden die in deze implementatie werden gebruikt, zijn MPEG 4 BiFS en MPEG 21 DIA. Zoals we eerder hebben gezien, biedt MPEG 4 BiFS de mogelijkheid om een scène te beschrijven en dit binair op te slaan in een MPEG 4-stream. Om deze scènes op een flexibele wijze te kunnen aanpassen, werd er gekozen voor MPEG 21 DIA wat met behulp van XML-beschrijvingen, transformaties mogelijk maakt.

De implementatie bestaat uit een afzonderlijke server en client. De taak van de server is om de tekstuele beschrijving van een scène als invoer te krij-

gen, samen met de gewenste transformaties. Met behulp van een configuratiebestand, kan er aangegeven worden welke transformatie er moet gebeuren voor specifieke clientsysteemeigenschappen. Eens de connectie opgezet is, zal er dus een binair bestand worden gemaakt en doorgestuurd. Hiermee zal de client de scène visualiseren en kan de gebruiker er verder gebruik van maken. In de volgende secties zullen de client en server uitgebreid aan bod komen.

6.3.1 Client

Algemeen

De client van de implementatie stelt de gebruiker in staat om een virtuele wereld te verkennen. De gebruiker bepaalt welke zone van de virtuele wereld hij wil bekijken en dit wordt meegedeeld aan de server. Verder wordt er ook rekening gehouden met bepaalde systeemeigenschappen van de client. Uiteindelijk zal de virtuele omgeving worden gevisualiseerd aan de hand van OpenGL.

In de volgende secties zullen de verschillende aspecten van de client in detail worden besproken. We zullen zien hoe de systeemeigenschappen worden bepaald en verstuurd, de werking van het netwerkcomponent bespreken en leggen uit hoe de verwerking en visualisatie van de ontvangen scène gebeurt.

Systeemeigenschappen

De server houdt rekening met de systeemeigenschappen van de gebruiker om te bepalen welke transformatie er moet toegepast worden op de virtuele omgeving. Om deze beslissing te nemen, is het noodzakelijk enkele systeemeigenschappen te kennen van de gebruiker. Aangezien de nadruk van deze thesis ligt in het gebruik van standaarden, werd er besloten om ook de systeemeigenschappen te verzenden in een formaat conform de MPEG 21-standaard. Hiervoor werd er, ter validatie van het XML-bestand, een XML Schema-bestand gebruikt horende bij de MPEG 21 Reference Software.

In codefragment 6.1 zien we een XML die bepaalde systeemeigenschappen beschrijft (headers en declaratie van DTD-documenten zijn achterwege gelaten). Voor deze implementatie werd er rekening gehouden met drie aspecten, namelijk CPU, vrije opslagruimte en netwerkmogelijkheden. We zien deze drie terugkomen in het codefragment. De snelheid van de CPU komt terug in de node DeviceBenchmark welke de kloksnelheid van de CPU in megahertz uitdrukt. De vrije opslagruimte wordt bijgehouden in de node Stor-

ageCharacteristic en wordt uitgedrukt in het aantal megabytes. Uiteindelijk zijn er nog de netwerkmogelijkheden in de node AvailableBandwith welke in kilobytes per seconde de doorvoersnelheid aangeeft de netwerkverbinding van de gebruiker.

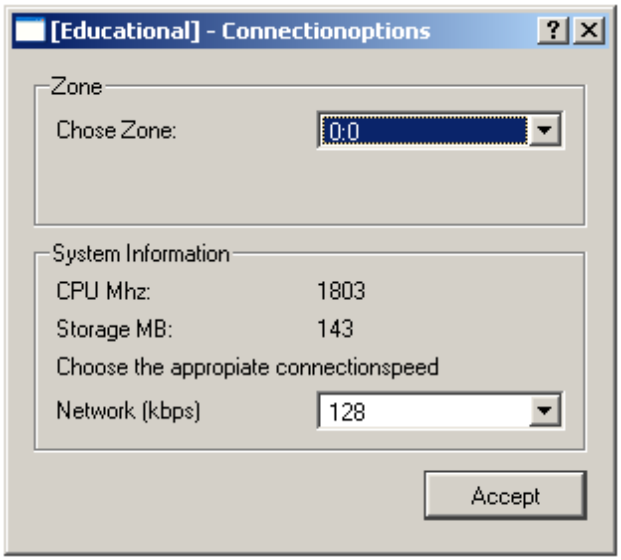
De waarde voor iedere systeemeigenschap wordt automatisch verkregen met behulp van de SysInfo-SDK [68]. Doordat het automatisch detecteren van netwerkmogelijkheden een complexere operatie is, werd er gekozen om dit door de gebruiker te laten ingeven. Eens men over deze waarden beschikt, zullen deze verstuurd worden naar de server. In afbeelding 6.2 wordt de interface afgebeeld die de gebruiker voorgeschoteld krijgt bij het bepalen van de systeemeisen. Hier zien we dat de gebruiker de gewenste zone van de virtuele wereld en de snelheid van zijn netwerkverbinding moet aangeven.

```
<UsageEnvironmentProperty xsi:type="CPUBencmarkType">
   <TerminalCapability xsi:type="BenchmarksType">
      <Benchmark>
         <DeviceBenchmark xsi:type="CPUBenchmarkType"
            baseValue="1803"/>
      </Benchmark>
   </TerminalCapability>
</UsageEnvironmentProperty>
<TerminalCapability xsi:type="StoragesType">
   <Storage xsi:type="StorageType">
      <StorageCharacteristic
         xsi:type="StorageCharacteristicsType"
         size="2000" writable="true"/>
   </Storage>
</TerminalCapability>
<UsageEnvironmentProperty xsi:type="NetworksType">
   <Network>
      <NetworkCharacteristic
         xsi:type="NetworkConditionType">
         <AvailableBandwidth average="512"/>
      </NetworkCharacteristic>
   </Network>
</UsageEnvironmentProperty>
```

Codefragment 6.1: Systeemspecificatie in MPEG 21 DIA

Netwerkaspect

Om de client te gebruiken, moet de gebruiker eerst een connectie maken met de server. Dit gebeurt, zoals gewoonlijk, met behulp van een IP-adres en een poortnummer. De gemaakte netwerkconnectie wordt hierna gebruikt voor verschillende taken. Om deze taken uit te voeren, wordt er gebruik gemaakt van een protocol bestaande uit pakketjes van telkens vier bytes.



Figuur 6.2: Opties bij het connecteren met de server

De connectie tussen server en client wordt met behulp van een TCP-verbinding gemaakt. Zoals we in het tweede hoofdstuk hebben gezien, zorgt TCP voor een betrouwbare verbinding waarbij volledigheid en juiste volgorde van de pakketten wordt verzekerd. Aangezien het protocol dat ontwikkeld werd voor deze applicatie, bestaat uit een vaste volgorde van berichten waarbij volledigheid van deze berichten van belang is, werd het gebruik van TCP verkozen boven UDP.

Zodra de connectie gemaakt is, zal de server het aantal zones versturen waaruit de virtuele omgeving bestaat. Dit is nodig opdat de gebruiker een zone kan kiezen die werkelijk bestaat. Hierna zal de gebruiker zijn systeemspecificatie versturen zoals beschreven in de voorgaande sectie. Als laatste stap zal de client een MPEG 4 BiFS-scène ontvangen en deze visualiseren. Aangezien de server rekening houdt met de systeemeigenschappen van de gebruiker, onvangt de client steeds een scène die geschikt is voor zijn toestel en netwerkverbinding.

Verwerking en Visualisatie

Het verwerken van de ontvangen scène gebeurt met MP4SDK die, zoals we in sectie 3.6 hebben gezien, ondersteuning biedt voor MPEG 4 BiFS. Verder gebeurt de visualisatie met behulp van OpenGL. Dit gebeurt zodra

de gebruiker aangeeft de scène te willen bekijken.

MP4SDK voorziet de mogelijkheid om een binair MP4-bestand in te lezen en hiervan de nodes op te vragen. Doordat deze SDK zich nog in een vroeg ontwikkelstadium bevindt, worden niet alle BiFS-nodes ondersteund. De deelverzameling van nodes die wel ondersteund worden, zijn echter genoeg om een volwaardige virtuele omgeving te beschrijven. Het is bijvoorbeeld mogelijk om voor nodes te kiezen die de belichting verzorgen en nodes geschikt voor veelhoeken waarop transformatienodes kunnen worden gebruikt. Verder kan er ook gebruik gemaakt worden van textures voor de IndexedFaceSet-nodes.

Het valt echter op te merken dat het gebruik van de MP4SDK het systeem volledig in beslag neemt. Dit zorgt ervoor dat verdere operaties, zoals visualisatie van de scène, onmogelijk zijn. Om deze reden werd er gekozen om de nodes die deze SDK ter beschikking stelt volledig te kopiëren naar een interne structuur en verder het gebruik van de SDK stop te zetten.

Daar MP4SDK enkel mogelijkheden biedt tot het inlezen van de scène, moet men terugvallen op een externe softwarebibliotheek om deze te visualiseren. Voor deze implementatie werd er gekozen voor OpenGL aangezien deze het meest geschikt is voor dergelijke taken en flexibel genoeg is voor het visualiseren van een dergelijke 3D-omgeving.

Voor iedere node die MP4SDK ter beschikking stelt, werd er een gepaste visualisatie geïmplementeerd. Hierbij valt op te merken dat de belichting van de scène een alternatieve aanpak vergt. Aangezien de hier gebruikte OpenGL-implementatie maximaal 8 lichtbronnen toelaat, kan dit niet gecombineerd worden met het onbeperkt aantal lichtbronnen die men met behulp van BiFS kan definiëren. Daarom werd er gekozen om enkel de 8 lichtbronnen te selecteren die zich het dichtst bij de huidige positie van de gebruiker bevinden. De belichting wordt hierbij dus dynamisch veranderd aan de positie van de gebruiker.

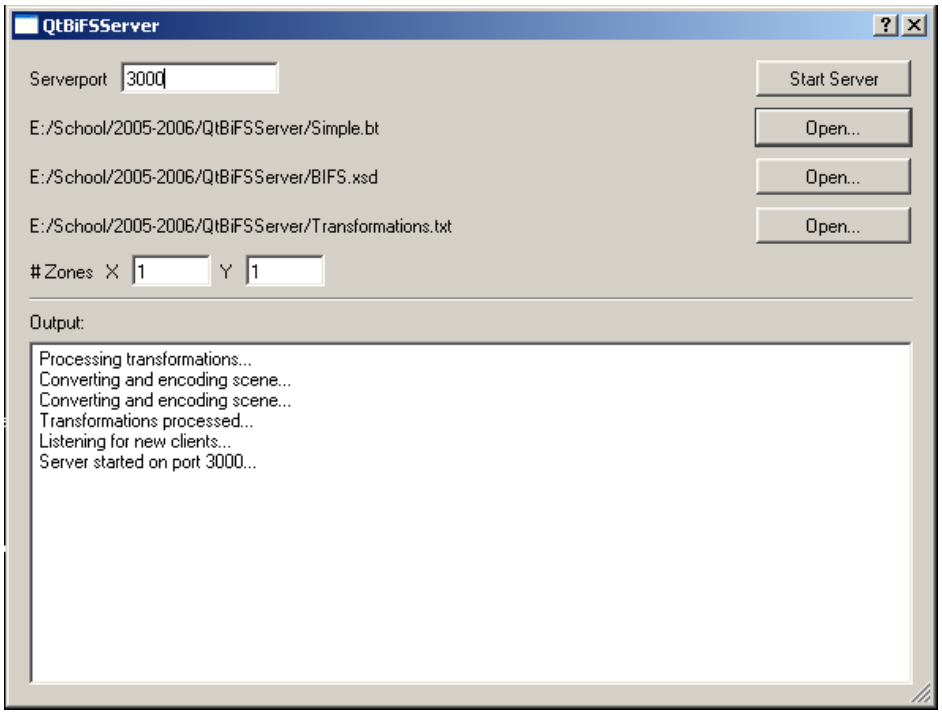
6.3.2 Server

Algemeen

De server bestaat voornamelijk uit drie componenten:

- netwerkcomponent die inkomende verbindingen van gebruikers opvangt en afhandelt
- component die als taak heeft om MPEG 4 BiFS-scènes te transformeren gegeven een BSD-beschrijving en een XSLT-transformatie
- component die als taak heeft om een tekstuele beschrijving van een MPEG 4-scène om te zetten naar het binair equivalent

De interface van de server biedt de mogelijkheid om een BiFS-bestand te kiezen in tekstueel formaat, een BS Schema-bestand en een configuratiebestand. Verder kan er het aantal zones ingegeven worden waarvoor er transformaties zijn en uiteindelijk het poortnummer waarop de server clients kan accepteren. In figuur 6.3 wordt deze interface afgebeeld.



Figuur 6.3: Interface van de server

Netwerkcomponent

De netwerkcomponent accepteert nieuwe verbindingen en zorgt voor de nodige afhandeling. Eens er een connectie wordt gemaakt, zal de netwerkcom-

ponent de overige componenten aanroepen om de gewenste scène te kunnen doorsturen. Men zal dus aan de hand van de ontvangen systeemeigenschappen bepalen welke transformatie voor de huidige client van toepassing is en de getransformeerde scène vervolgens verzenden. Hierna zal de connectie worden verbroken.

Scène transformatie

Gegeven een BiFS-scène wordt deze getransformeerd zodat ze voldoet aan de noden van de gebruiker. De transformatie gebeurt met behulp van de MPEG 21-standaard en meer bepaald DIA. De implementatie is gebaseerd op de reference software van MPEG 21 [62].

Het doel van deze transformatie is om het mogelijk te maken dat men éénzelfde scènebeschrijving kan gebruiken voor gebruikers met verschillende eisen en mogelijkheden. De transformaties gebeuren op basis van zone en systeemeigenschappen van de client. Daar het niet nodig is dat een client de volledige beschrijving van een scène ontvangt, gebruikt men zones om te bepalen in welk deel van de omgeving de gebruiker zich bevindt. Hierdoor zal enkel de beschrijving van dit deel van de omgeving worden verzonden.

Nadat men heeft bepaald in welke zone een gebruiker zich bevindt, wordt ook het gewenste detailniveau van deze zone gebruikt om te bepalen welke transformatie nodig is. Men maakt hiervoor gebruik van de systeemeigenschappen van een bepaalde gebruiker. Deze systeemeigenschappen worden ontvangen als MPEG 21-conform XML-bestand dat reeds werd besproken in sectie 6.3.1. Om te bepalen welke transformaties horen bij een bepaald systeem, wordt er gebruik gemaakt van een configuratiebestand. Dit configuratiebestand bevat een aantal regels die de grenzen bepalen waarvoor een bepaalde transformatie van toepassing is. In codefragment 6.2 wordt een eenvoudig voorbeeld gegeven van een dergelijk bestand. Dit bestand toont slechts twee regels die bepalen of een client de scène laag of hoog gedetailleerd zal ontvangen. We hebben in sectie 6.3.1 gezien dat de transformatie gebeurt aan de hand van de kloksnelheid van de CPU, de beschikbare vrije opslagruimte en de netwerkcapaciteit. In het configuratiebestand zien we dit terugkomen na de initiële twee cijfers die aangeven voor welke zone dit van toepassing is. De eerste regel geeft dus aan dat de transformatie van toepassing is voor een client met een kloksnelheid tussen 100 en 500 Mhz, vrije opslagruimte tussen 10 en 100MB en een doorvoersnelheid van zijn netwerk tussen 10 en 500kbps. Hiervoor zal de transformatie LowDetail.xsl worden

gebruikt. De tweede regel uit het configuratiebestand wordt op analoge manier ontleed.

```
0/0/100/500/10/100/10/500/LowDetail.xsl
0/0/500/3000/100/1000/500/1000/HighDetail.xsl
```

Codefragment 6.2: Configuratiebestand voor de server

De kern van de implementatie besproken in sectie 6.2 wordt hier gebruikt om de transformatie mogelijk te maken. De bekomen implementatie is, zoals eerder vermeld, een Java-implementatie. Daar de uiteindelijk implementatie ontwikkeld werd in C++, is er voor een koppeling gezorgd tussen deze twee componenten. De koppeling werd bekomen met behulp van de Java Native Interface [69] waarbij een opgezette Java Virtual Machine kan gebruikt worden om de nodige operaties uit te voeren.

6.3.3 Resultaten

De hier besproken implementatie kan ervoor zorgen dat men minder data moet versturen naar gebruikers van een NVE. Door enkele voorbeeldscènes te nemen, zal hier besproken worden hoe groot dit voordeel is ten opzichte van tekstueel gebaseerde methoden waarbij men volledige scènes moet doorsturen in plaats van gepaste transformaties.

Voor de tests werd er steeds gebruik gemaakt van relatief eenvoudige scènes. Dit verklaart waarom de bestandsgrootte van iedere scène relatief klein blijft. Meer complexe scènes zullen echter veel meer objecten bevatten en dit zal zich dan ook vertalen in grotere bestanden. Het is dus belangrijk in te zien dat de relatieve verhoudingen tussen de grootten van de scènes belangrijker zijn dan de absolute bestandsgrootten.

BiFS wordt, zoals eerder in deze thesis besproken, in een binair formaat opgeslagen. Het voordeel van dit formaat is dat men aanzienlijk kleinere bestanden bekomt ten opzichte van de originele tekstuele beschrijving. In tabel 6.1 zien we het verschil tussen de tekstuele en binaire voorstelling van enkele scènebeschrijvingen. Hierbij werd er gebruik gemaakt van verschillende testscènes waarbij de inhoud van de scènes genoeg variatie biedt. We zien hier dat er een enorme winst is in het encoderen van de tekstuele beschrijving van de scène.

	Tekstueel	Binair	Verschil
Testscène 1	4130 bytes	1390 bytes	66%
Testscène 2	40600 bytes	10100 bytes	75%
Testscène 3	77200 bytes	18800 bytes	75%

Tabel 6.1: Verschil tussen tekstuele en binaire voorstelling

Scéne	Bestandsgrootte
Originele zone	3020 bytes
Zone 0:0	784 bytes
Zone 0:1	1110 bytes
Zone 1:0	2250 bytes
Zone 1:1	765 bytes

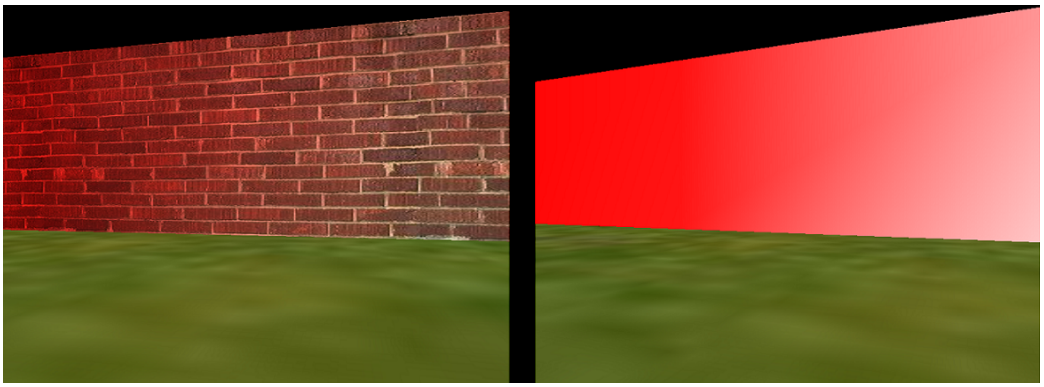
Tabel 6.2: Transformaties voor verschillende zones

We hebben reeds besproken hoe een scène kan worden opgedeeld in zones zodat een gebruiker enkel een deel van de virtuele wereld te zien krijgt van zijn directe omgeving. We zullen aan de hand van enkele tests illustreren wat het voordeel is van het versturen van een specifieke zone in tegenstelling tot de volledige scène.

De scène in tabel 6.2 bevat per zone een verschillend aantal objecten en dit voor vier verschillende zones. Dit verklaart het verschil in grootte tussen de verschillende zones. We zien verder dat de som van de bestandsgrootten van de deelscènes groter is dan de bestandsgrootte van de originele scène. De reden hiervoor is dat iedere zone een vaste MPEG-4 specifieke header heeft die informatie bevat over het MPEG 4-bestand.

Als uiteindelijke test nemen we een transformatie waarin we bepaalde textures verwijderen van bepaalde objecten. Dit kan bijvoorbeeld een texture zijn van een object dat enkel als decoratie dient en dus minder belangrijk is of een object dat herkenbaar blijft zonder texture. Een muur kan bijvoorbeeld een texture hebben waarop bakstenen zijn afgebeeld, maar als men de muur een rode kleur geeft, zal dit nog steeds als hetzelfde object herkenbaar blijven. Dit wordt geïllustreerd in figuur 6.4.

Een transformatie die het mogelijk maakt om textures te verwijderen, zal de totale bestandsgrootte van de scène hoofdzakelijk verminderen doordat



Figuur 6.4: Links de oorspronkelijke scène met een texture voor de muur, rechts na de transformatie die ervoor zorgt dat er geen texture meer wordt gebruikt

	Bestandsgrootte
Met textures	35700 bytes
Zonder textures	3020 bytes

Tabel 6.3: Enkele textures verwijderen uit een scène

deze textures niet meer verstuurd moeten worden. Een kleinere winst wordt gemaakt op het positioneren van deze texture in de scène, maar deze is verwaarloosbaar. In tabel 6.3 zien we duidelijk dat de besparing hoofdzakelijk afkomstig is van het gebrek aan bepaalde textures. De bestandsgrootten hier weergegeven zijn inclusief de bestandsgrootten van de textures.

6.4 Conclusie

Een genetwerkte virtuele omgeving kan enkel bestaan als de gebruiker een bepaalde wereld aangeboden krijgt waarbinnen hij kan interageren. De beschrijving van deze omgeving kan op verschillende manier gebeuren: veelal wordt er gebruik gemaakt van zelf gedefiniëerde formaten maar men kan ook gebruik maken van een standaard. In de implementatie besproken in dit hoofdstuk werd er gebruik gemaakt van de MPEG 4-standaard. Deze keuze werd gemaakt omdat deze standaard tal van voordelen biedt die we al eerder hebben besproken in deze thesis.

Schaalbaarheid wordt steeds belangrijker door de toenemende heterogeniteit van de clients die gebruik willen maken van een bepaalde genetwerkte virtuele omgeving. Dit betekent dat men over een mogelijkheid wil beschikken om bepaalde data aan te passen aan de noden van deze clients. DIA kan hiervoor zorgen en is dan ook hier gebruikt om een scènebeschrijving te transformeren opdat deze bruikbaar zou zijn voor clients met verschillende eigenschappen. We hebben gezien dat een dergelijke transformatie kan gebruikt worden om onbelangrijke details te verwijderen uit een scène. Maar ook, en niet onbelangrijk, om enkel dat gedeelte van de scène door te sturen dat zichtbaar is voor de gebruiker. Door de algemene aanpak van de implementatie is er echter een bepaalde vrijheid in het kiezen van scènes en transformaties, waardoor ook andere mogelijkheden kunnen bedacht buiten de twee voorgenoemde.

De resultaten hebben aangetoond dat het gebruik van MPEG 4 BiFS en de hierop uitgevoerde transformaties een enorme winst kunnen opleveren. Dit heeft als gevolg dat de scénebeschrijving minder inneemt en dus minder belastend is voor het netwerk. Uiteindelijk werd er aangetoond hoe de transformaties kunnen worden gebruikt voor clients met zwakkere systeemeigenschappen, door bijvoorbeeld de textures van objecten te verwijderen uit een scéne.

Hoofdstuk 7

Conclusie

Standaardisatie is een belangrijk proces in iedere tak van de wetenschap. Binnen de informaticawereld zijn er een enorm aantal standaarden opgesteld met ieder een specifiek nut. Het doel van deze thesis was enkele standaarden zoeken en bespreken die nuttig kunnen zijn binnen de context van een networked virtual environment (NVE).

Aan de basis van een NVE ligt de wereld of, zoals de naam al aangeeft, een virtuele omgeving waarin de gebruikers navigeren en interageren. Het is daarom zinvol om voor de creatie van deze omgevingen een standaard te gebruiken die zich reeds heeft bewezen als zijnde werkende en goed gestructureerd. Dit is dan ook de motivatie geweest om standaarden ter beschrijving van 3D-scènes meer in detail te bespreken in deze thesis. We hebben gezien dat deze standaarden een groot aantal mogelijkheden bieden en ervoor zorgen dat een ontwikkelaar hiermee over een stevige basis kan beschikken bij het ontwerpen van een NVE.

Een NVE kan echter uitgebreid worden met audiovisuele media. Een virtuele omgeving zonder geluid is tegenwoordig ondenkbaar en het gebruik van videobronnen binnen een NVE begint ook een opmars te maken. Het is praktisch onmogelijk om audiovisuele data digitaal op te slaan zonder enige vorm van compressie. Een videostroom van enkele minuten zou een enorme omvang aannemen en niet meer bruikbaar zijn over een netwerk. Maar ook hier zijn er verschillende standaarden die een oplossing bieden. Deze standaarden hebben door de jaren heen een sterke evolutie gekend waardoor de kwaliteit steeds beter wordt aan een steeds lagere bitrate. Dit zorgt ervoor dat het steeds aantrekkelijker wordt om audiovisuele data te gebruiken in een NVE.

Steeds meer aandacht wordt besteed aan het schaalbaar maken van de virtuele wereld. Om ondersteuning te kunnen bieden aan een groot aantal gebruikers, moeten de ontwikkelaars van een NVE beseffen dat deze gebruikers over systemen beschikken die onderling sterk kunnen verschillenen. Het onderliggend netwerk is onderhevig aan verschillende factoren wat zorgt voor grote snelheidsverschillen. Zoals we hebben gezien biedt MPEG 21 hiervoor een oplossing door met behulp van XML-beschrijvingen en gepaste transformaties, ervoor te zorgen dat men binaire data kan aanpassen aan de noden en de mogelijkheden van de gebruiker.

Van iedere standaard die we hebben besproken, werd aangegeven hoe deze gebruikt kan worden in een NVE. Door middel van een implementatie hebben we het praktisch aspect hiervan bekeken. Het is duidelijk naar voor gekomen dat deze standaarden een belangrijke rol kunnen spelen bij het ontwerp van een NVE. De resultaten hebben aangetoond dat het transformeren van MPEG 4 BiFS-scènes een groot voordeel bieden. Deze transformaties hebben er niet alleen voor gezorgd dat het netwerk minder wordt belast, maar ook dat de scènes op een relatief eenvoudige manier kunnen worden aangepast aan de systeemmogelijkheden van de gebruikers. Het is dan ook te verwachten dat men in de toekomst steeds meer gebruik gaat maken van deze standaarden en dat toekomstige standaarden steeds meer mogelijkheden gaan bieden die men in een NVE kan benutten.

Bibliografie

[1] Sandeep Singhal and Michael Zyda. *Networked Virtual Environments: Design and Implementation*. 1999.

[2] Andrew Tanenbaum. *Computernetwerken*. 2003.

[3] Michael Macedonia and Michael Zyda. A taxonomy for networked virtual environments. 1997.

[4] Chris Joslin, Igor Pandzic, and Nadia Magnenat Thalmann. Trends in networked collaborative virtual environments. 2003.

[5] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. A review on networking and multiplayer computer games. 2002.

[6] Maja Matijasevic. A review of networked multi-user virtual environments. 1997.

[7] John Barrus, Richard Waters, and David Anderson. Locales and beacons: Efficient and precise support for large multi-user virtual environments. 1996.

[8] James Purbrick and Chris Greenhalgh. Extending locales: Awareness management in massive-3. 2000.

[9] Heywon Seo, Chris Joslin, Uwe Berner, Nadia Magnenat-Thalmann, Maja Jovovic, Joaquim Esmerado, Daniel Thalmann, and Ian Palmer. Vpark - a windows nt software platform for a virtual networked amusement park. 2000.

[10] id Software. Quake. `http://www.idsoftware.com/`.

[11] Epic Games. Unreal. `http://www.epicgames.com/`.

[12] Wikipedia. Mmorpg. `http://en.wikipedia.org/wiki/MMORPG`.

[13] Blizzard Entertainment. World of warcraft. http://www.worldofwarcraft.com.

[14] Verant Interactive. Everquest. http://everquest.station.sony.com.

[15] Linden Lab. Second life. www.secondlife.com.

[16] Philip Rosedale and Cory Ondrejka. Enabling player-created online worlds with grid computing and streaming. 2003.

[17] Makena Technologies. There. http://www.there.com.

[18] IVN/Cybertown Inc. Cybertown. http://www.cybertown.com.

[19] The Activeworlds Corporation. Active worlds. http://activeworlds.com.

[20] Joaquin Keller and Gwendal Simon. Solipsis. http://solipsis.netofpeers.net.

[21] Bob Crispen. Vrml works. http://vrmlworks.crispen.org/.

[22] Web3D Consortium. Extensible 3d: open standards for real-time 3d communication. http://www.web3d.org.

[23] Web3D Consortium. Extensible 3d part 1: Architecture and base components. http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification.

[24] Julien Signes, Yuval Fisher, and Alexandros Eleftheriadis. Mpeg-4's binary format for scene description. http://www.chiariglione.org/MPEG/tutorials/papers/icj-mpeg4-si/05-BIFS_paper/5-BIFS_paper.htm.

[25] Julien Signes. Binary format for scene: Combining mpeg-4 media to build rich multimedia services. 1997.

[26] Mojtaba Hosseini and Nicolas Georganas. Suitability of mpeg4's bifs for development of collaborative virtual environments. 2001.

[27] Rob Koenen. Overview of the mpeg 4 standard. http://www.chiariglione.org/MPEG/standards/mpeg-4/mpeg-4.htm.

[28] Eric Scheirer, Riitta Vaananen, and Jyri Huopaniemi. Audiobifs: The mpeg-4 standard for effects processing. 1999.

[29] Eric Scheirer, Riitta Vaananen, and Jyri Huopaniemi. Audiobifs: Describing audio scenes with the mpeg-4 multimedia standard. 1998.

[30] Tolga Capin, Eric Petajan, and Joern Ostermann. Efficient modeling of virtual humans in mpeg-4. 2000.

[31] Marius Preda and Francoise Preteux. Advanced virtual humanoid animation framework based on the mpeg-4 snhc standard. 2001.

[32] Mojtaba Hosseini and Nicolas Georganas. Mpeg-4 bifs streaming of large virtual environments and their animation on the web. 2001.

[33] Cyril Concolato and Jean Le Feuvre. Mpeg-4 bifs and xmt tutorial. http://gpac.sourceforge.net/tutorial/bifs_intro.htm.

[34] Systems in Motion AS. Coin3d. http://www.coin3d.org.

[35] Braden McDaniel, Christopher St. John, and Chris Morley. Openvrml. http://openvrml.org.

[36] Satoshi Konno. Cyberx3d. http://www.cybergarage.org.

[37] Yannick Le Goc. X3dtoolkit. http://artis.imag.fr/Members/Yannick.Legoc/X3D.

[38] ZGDV and Fraunhofer IGD. Avalon. http://www.zgdv.de/avalon.

[39] Octavian Folea, Marius Preda, and Francoise Preteux. Mpeg-4 sdk: From specifications to real applications. 2005.

[40] Sasko Celakovski, Marius Pread, Slobodan Kalajdziski, Danco Davcev, and Francoise Preteux. Mpeg-4 3d graphics: from specifications to the screen. 2005.

[41] Jean Le Feuvre. Gpac project on advanced content. http://gpac.sourceforge.net/.

[42] Visage Technologies. Visage sdk. http://www.visagetechnologies.com/products_sdk.html.

[43] Ch. Bouras, A. Panagopoulos, and Th. Tsiatsos. Advances in x3d multiuser virtual environments. 2005.

[44] Blaxxun Technologies. Blaxxun. http://www.blaxxun.com.

[45] Octaga. Octaga professional and octaga server. http://www.octaga.com/.

[46] Chris Joslin and Nadia Magnenat-Thalmann. Mpeg-4 animation clustering for networked virtual environments. 2002.

[47] Chi-Min Liu and Wen-Whei Chang. Audio coding standards. 1999.

[48] Siddhartha Devadhar, Cederic Krumbein, and Kim Man Liu. Mpeg background. http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_overview.html.

[49] P.N. Tudor. Mpeg-2 video compression. 1995.

[50] MPEG-4 Industry Forum. Mp4 sp/asp.

[51] Wikipedia. Mpeg-4 part 2. http://en.wikipedia.org/wiki/MPEG-4_Part_2.

[52] Thomas Wiegand, Gary Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h.264/avc video coding standard. 2003.

[53] Edouard Francois, Jerome Vieron, and Guillaume Boisson. Mpeg svc: why a new video coding standard? 2004.

[54] Requirements Group. Mpeg-21 overview v.5, 2002. http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm.

[55] Jeron Bekaert, Patrick Hocstenbach, and Herbert Van de Sompel. Using mpeg-21 didl to represent complex digital objects in the los alamos national laboratory digital library. *D-Lib Magazine*, 9, 2003.

[56] Christian Timmerer. Resource adaptation using xml within the mpeg-21 multimedia framework. Master's thesis, Universitat Klagenfurt, 2003.

[57] Multimedia Description Schemes Subgroup. *Text of ISO/IEC 21000-7 FCD - Part 7: Digital Item Adaptation*, 2003.

[58] Anthony Vetro and Christian Timmerer. Digital item adaptation: Overview of standardization and research activities. 2005.

[59] Jan Bormans, Jean Gelissen, and Andrew Perkis. Mpeg-21: The 21st century multimedia framework. 2003.

[60] ISO/IEC. Mpeg 21 part 7: Digital item adaption fdis 21000-7:2004(e), 2004.

[61] Gabriel Panis, Andreas Hutter, Jorg Heuer, Hermann Hellwagner, Harald Kosch, Christian Timmerer, Sylvain Devillers, and Myriam Amielh. Bitstream syntax description: a tool for multimedia resource adaptation within mpeg-21. *Signal Processing: Image Communication*, 2003.

[62] Rik Van De Walle. Practicum 6: Mpeg-21 bitstream syntax description language, 2005.

[63] Davy Van Deursen and Rik Van de Walle. Bflavor: a new bitstream structure description language. *Sixth FirW PhD Symposium*, 2005.

[64] Klaus Leopold, Hermann Hellwagner, and Michael Kropfberger. Qctva - quality controlled temporal video adaptation. 2003.

[65] Wesley De Neve. On the usage of bitstream structure descriptions for exploiting temporal scalability in h.264/avc. 2005.

[66] HyungSeok Kim, Chris Joslin, Thomas Di Giacomo, Stephane Garchery, and Nadia Magnenat-Thalmann. Adaptation mechanism for three dimensional content within the mpeg-21 framework. *Computer Graphics International 2004*, 2004.

[67] Roberto Osorio, Sylvain Devillers, Eric Delfosse, Myriam Amielh, and Gauthier Lafruit. Bitstream syntax description language for 3d mpeg-4 view-dependent texture streaming. 2002.

[68] Paul Wendt. Sysinfo. `http://www.codeproject.com/system/sysinfo.asp`.

[69] Sun Microsystems. Java native interface. `http://java.sun.com/j2se/1.5.0/docs/guide/jni/`.

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen en uw akkoord te verlenen.

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Standaardisatie voor NVE toepassingen
Richting: **Master in de informatica** Jaar: **2006**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Deze toekenning van het auteursrecht aan de Universiteit Hasselt houdt in dat ik/wij als auteur de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij kan reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

U bevestigt dat de eindverhandeling uw origineel werk is, en dat u het recht heeft om de rechten te verlenen die in deze overeenkomst worden beschreven. U verklaart tevens dat de eindverhandeling, naar uw weten, het auteursrecht van anderen niet overtreedt.

U verklaart tevens dat u voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen hebt verkregen zodat u deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal u als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze licentie

Ik ga akkoord,

Michele FUMAROLA

Datum: