**Vrije Universiteit Brussel**

**Faculteit Wetenschappen**

**Departement Informatica**

**Academiejaar 2001-2002**

# Onderzoek naar het ontwikkelen van grafische gebruikersinterfaces met behulp van moderne webtechnologieën

**David Van Damme**

**Promotor: Prof. Dr. Olga De Troyer**

**Verhandeling met het oog op het behalen van de graad van Licentiaat in de Toegepaste Informatica**

# Samenvatting

Deze verhandeling bekijkt het ontwikkelen van grafische gebruikersinterfaces in de context van het Internet en de nieuwste ontwikkelingen op dat gebied, namelijk webapplicaties en *web services*. De thesis van deze verhandeling is dat moderne webtechnologieën in staat zijn om volledig functionele, interactieve grafische gebruikersinterfaces te creëren, zoals we die kennen van traditionele software.

Er zijn grof bekeken twee soorten websites. De eerste soort zijn de informatieve websites, opgesteld als een brochure, een handleiding of iets gelijkaardigs, waarbij het leveren van informatie voorop staat. De tweede categorie zijn de webapplicaties, het gaat hier om websites waar iemand naartoe surft om iets te doen; bvb. elektronische post lezen, SMS berichten versturen, informatie opzoeken, databases beheren… Klassieke webtechnologieën als de *Hypertext Markup* Language (HTML), *Cascading Style* Sheets (CSS) en JavaScript schieten voor deze laatste categorie vaak tekort om op eenvoudige manier een aantrekkelijke interactieve gebruikersinterface te construeren.

We willen namelijk een interface voor onze webapplicatie die lijkt op wat we gewend zijn van de klassieke (*off-line*) applicaties. Daardoor moeten we niet steeds opnieuw met een applicatie leren werken.

Nieuwe webtechnologieën als Curl en de laatste versies van Flash en Shockwave van Macromedia proberen de voordelen van het werken in een webbrowser te combineren met de functionaliteiten van een programmeertaal. Ze beschikken ook over speciale ondersteuning voor de constructie van gebruikersinterfaces. We zouden deze technologieën kunnen gebruiken voor het implementeren van gebruikersinterfaces voor applicaties.

# Dankwoord

In de eerste plaats zou ik mijn promotor, Prof. Dr. Olga De Troyer willen bedanken voor haar richtlijnen, suggesties en verbeteringen bij het schrijven van deze verhandeling.

Ook wil ik mijn ouders Anke Van der Vorst en Erwin Van Damme, mijn broer Steven Van Damme en mijn vriendin Annelies Waer bedanken voor het nalezen van de tekst, voor de suggesties en voor de morele steun.

# Inhoudsopgave

# Inleiding

Het *World Wide Web* is sinds zijn ontstaan in 1992 geëvolueerd van een medium voor wetenschappers naar een globaal informatie- en communicatiekanaal. Een van de belangrijkste toepassingen van het WWW is het verschaffen van informatie via websites. Sinds enkele jaren worden ook de commerciële mogelijkheden ervan gebruikt met *e-commerce* of handel via het Internet.

De *Hypertext Markup Language* of HTML, de belangrijkste taal voor het maken van websites, was oorspronkelijk bedoeld voor het structureren van (wetenschappelijke) documenten. Geleidelijk is er echter een behoefte ontstaan om via het web ook met volledige applicaties te kunnen werken die in het venster van een browser draaien. Deze zogenaamde webapplicaties kunnen nuttig zijn voor bedrijven of particulieren die diensten via het Internet willen aanbieden. Ook voor bedrijven die applicaties via hun Intranet willen gebruiken is deze manier van werken interessant. De gebruiker hoeft namelijk helemaal geen of slechts een beperkte inspanning te leveren voor de installatie en configuratie[1].

Om applicaties via het web te laten werken, hebben we een applicatie *front-end* nodig om met een achterliggende serverapplicatie te communiceren. De mogelijkheden van klassieke webtechnologieën zijn op het gebied van interactieve gebruikersinterfaces echter te beperkt.

In deze verhandeling zullen we onderzoeken of we met de nieuwste webtechnologieën complexe interactieve grafische gebruikersinterfaces kunnen maken. We zullen in de eerste 3 hoofdstukken van deze verhandeling een aantal criteria opstellen waaraan de webtechnologieën, die we in het 4de hoofdstuk zullen voorstellen en bespreken, moeten voldoen. Het 5de en laatste hoofdstuk geeft een beoordeling van de webtechnologieën en we zullen proberen een complexe interactieve interface te implementeren met een van de voorgestelde technologieën.

[1] Men spreekt in dit opzicht ook wel van *Zero Client Administration.*

# Hoofdstuk 1

# Client/server modellen

Aan het begin van het WWW-tijdperk had men vooral oog voor het informatieve: wetenschappelijke documenten publiceren en informatie verspreiden. Nu het Internet ook voor meer geavanceerde doeleinden gebruikt wordt, schieten technologieën vaak tekort en is er behoefte aan nieuwe met meer mogelijkheden. Een nieuwe trend op het web is bijvoorbeeld het leveren van webdiensten of *web services*. Het gaat hier in feite om een applicatie die verdeeld of gedistribueerd is over meerdere computers en via een netwerk bereikbaar is.

In dit hoofdstuk zullen we de verschillende architecturen van gedistribueerde applicaties kort aanhalen. Vervolgens zullen we enkele criteria opstellen waaraan webtechnologieën moeten voldoen om te kunnen functioneren in zo'n gedistribueerde applicatie.

## 1.1 Het drie-lagen-model

In de discipline van de softwareontwikkeling spreekt men dikwijls over de drie lagen van een applicatie. Zo probeert men de applicatie op te delen in kleinere stukken om de implementatie ervan te vereenvoudigen.

In de logische laag wordt in feite de kern van de applicatie behandeld. De gegevenslaag verzorgt de opslag en het lezen van informatie die de applicatie nodig heeft[1]. Ten slotte is er de presentatielaag, waar we ons in dit werk het meest zullen op concentreren en waarin de visuele voorstelling en de interactie met de gebruiker gebeurt. Een overzicht van de drie lagen staat in **Tabel 1** (zie ook [Berson 1996]).

**Tabel 1: Applicatielagen**

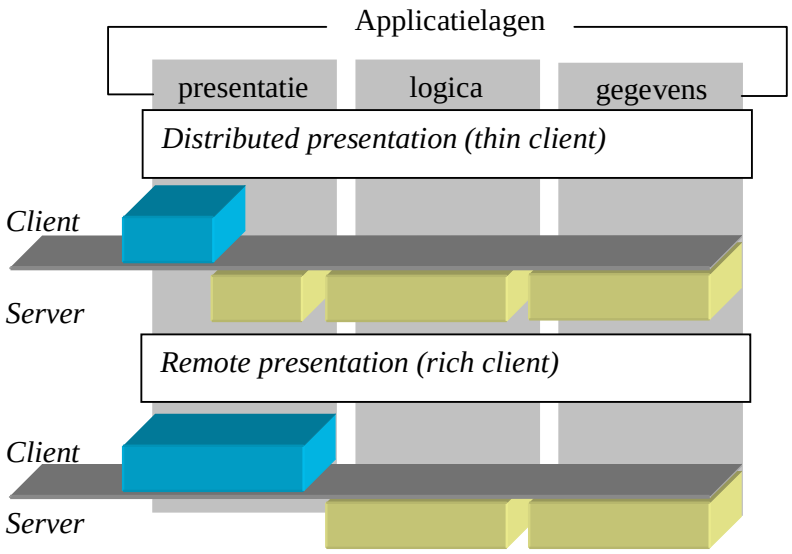
| | **Benaming** | **Alternatieve benamingen** |
|---|---|---|
| 1 | Logische laag | Logic layer, Business layer |
| 2 | Gegevenslaag | Data layer, Database layer + DBMS[1] layer |
| 3 | Presentatielaag | Presentation layer |

In een gedistribueerde applicatie bevinden deze lagen zich vaak ook op een verschillende plaats. De presentatielaag is dan meestal bij de gebruiker (*client*), de logishe en de gegevenslaag bevinden zich op één of meer servers.

Als de drie lagen zich fysiek op drie verschillende apparaten bevinden, spreken we van een *3-tier distributed application* architectuur; analoog, indien de presentatielaag zich bij de *client* bevindt en de andere twee lagen op eenzelfde machine staan, noemen we dit een *2-tier distributed application* architectuur.

Elk van de afzonderlijke lagen kan op zich echter ook nog opgesplitst worden over verschillende apparaten. **Figuur 1** toont een illustratie van twee *2-tier* architecturen. De bovenste voorstelling geeft een *2-tier* architectuur met gedistribueerde presentatielaag, die zich deels bij de *client* en deels op een server bevindt. De logische en de gegevenslaag bevinden zich op de server. De onderste voorstelling stelt een *2-tier* architectuur voor, waarbij de presentatielaag zich volledig bij de *client* bevindt.

[1] Soms wordt de gegevenslaag verder opgesplitst in een logische databasemanipulatielaag en een DBMS of *Database Management System* laag.




Bron: [Berson 1996]

**Figuur 1: *2- tier* architecturen**

## 1.2 Websites als interface voor een applicatie

Steeds meer voornamelijk commerciële websites bieden bepaalde diensten aan via het Internet, zoals bankverrichtingen, *webmail*, verkoop van boeken en CD's… Deze diensten of programma's noemen we "webapplicaties", omdat ze via het WWW[1] werken.

Webapplicaties zijn gedistribueerde applicaties en ze kunnen net als alle andere applicaties opgedeeld worden in drie lagen. De presentatielaag wordt hier met zogenaamde webtechnologieën gerealiseerd. De logische laag en de gegevenslaag worden meestal door *serversoftware* verwezenlijkt, waar we hier verder niet over zullen uitweiden.

[1] *World Wide Web*

Webapplicaties werken meestal met zogenaamde *thin clients*, waarbij de *2-tier* architectuur met gedistribueerde presentatielaag gehanteerd wordt. Er komen echter steeds meer *rich clients* voor, waarbij de volledige presentatielaag zich bij de *client* bevindt.

### 1.2.1 "Thin Clients"

In een webapplicatie met *thin clients* speelt de browser de rol van het gedeelte van de gedistribueerde presentatielaag dat zich bij de *client* bevindt.

Een groot deel van de verwerking binnen de presentatielaag vindt dan ook plaats op een server en niet op de *client*. De server genereert namelijk het HTML-document dat naar de *client* wordt gestuurd. Het enige wat de browser van de *client* hoeft te doen is de indeling van het HTML document vertalen naar een grafische representatie op het scherm van de gebruiker, daarom spreken we van een *thin client*.

We kunnen dit duidelijk zien als we een handeling uitvoeren met een webapplicatie waarvan de gebruikersinterface voornamelijk op HTML is gebaseerd. Als een gebruiker bijvoorbeeld op een link klikt, wordt een aanvraag of *request* naar de webserver gestuurd. Die verwerkt de aanvraag en antwoordt met een nieuwe webpagina, die in het geval van webapplicaties vaak dynamisch wordt samengesteld. De server wordt daardoor extra belast met het genereren van althans een deel van de gebruikersinterface.

Hoewel deze benadering uitermate geschikt is voor de op de brochuremetafoor gebaseerde websites, stellen er zich problemen wanneer we ze voor een webapplicatie willen toepassen.

Wanneer een gebruiker een actie onderneemt, resulteert dit meestal in slechts een kleine verandering van de gebruikersinterface. Op zich is dat geen probleem, ware het niet dat de gebruiker daarvoor relatief lang moet wachten en zijn scherm in afwachting van een antwoord van de server volledig blanco wordt. Hoe complexer de interface, hoe langer de gebruiker moet wachten tot de webserver de pagina heeft samengesteld en de browsersoftware de pagina heeft geformatteerd.

De oorspronkelijke bedoeling van HTML, en de standaardisering ervan door W3C, was dat het een universele standaard zou zijn die er op elke computer hetzelfde uitzag. Door de concurrentiestrijd tussen de grote softwarefabrikanten van webbrowsers (met name Microsoft en Netscape) is er vaak van die standaard afgeweken. Toch kan een goede webmaster ervoor zorgen dat zijn website overal leesbaar is en er ongeveer hetzelfde uitziet. Dat is meteen ook het grootste voordeel van de *thin client* webapplicatie: ze (kunnen) werken in elke browser.

Moderne browsers, die noodzakelijk zijn voor het raadplegen van een website, zijn gratis te downloaden en beantwoorden goed tot zeer goed aan de huidige HTML-standaard.

### 1.2.2 "Rich Clients"

Door de opkomst van *e-commerce* en de snelle evolutie van het web, worden de limieten van nieuwe technologieën al snel bereikt.

Het klassieke model van een webpagina voldoet niet aan de eisen die een webapplicatie stelt. Bedrijven willen hun klanten namelijk dezelfde gebruiksvriendelijkheid bieden als een bureaubladapplicatie. We kunnen echter stellen dat de mogelijkheden van een *thin client*, zoals hierboven beschreven, daarvoor te beperkt zijn.

Onder de term *rich client* verstaan we gedistribueerde applicaties waarbij de presentatielaag zich volledig bij de *client* bevindt. Het kunnen zowel losstaande uitvoerbare programma's zijn als *applets* en dergelijke die in een browser draaien.

*Rich clients* kunnen gebruiksvriendelijker zijn, doordat ze de gebruiker meer interactiemogelijkheden bieden en een betere, aantrekkelijkere visuele presentatie van de software mogelijk maken.

Doordat bij *rich clients* de gebruikersinterface zich volledig bij de *client* bevindt en niet gedistribueerd is, zoals bij *thin clients*, wordt ook de hoeveelheid communicatie beperkt. Bepaalde interface-specifieke bewerkingen, zoals het openen van een menu, het gedeeltelijk vernieuwen van scherminhoud, het tijdelijk onbruikbaar maken van een knop of het controleren van de invoer van de gebruiker, vereisen geen communicatie met de server.

Het voornaamste nadeel van een *rich client* applicatie is het ontbreken van een standaardplatform[1] en de daarmee samenhangende installatie- en configuratieproblemen.

[1] De .NET architectuur van Microsoft stelt misschien een oplossing voor?

Bij *thin clients* is het bezit van een recente webbrowser, die op de meeste systemen vandaag de dag standaard aanwezig is, voldoende. Indien we de gebruiker meer willen bieden onder de vorm van een *rich client*, moet deze wel de moeite nemen om extra software of plug-ins te installeren en/of te configureren.

Een belangrijk criterium bij het beoordelen van een *rich client* applicatie is dus de hoeveelheid en de complexiteit van de bijkomende administratie die nodig is om ze te doen werken.

## 1.3 Verschil tussen web en grafische gebruikersinterfaces

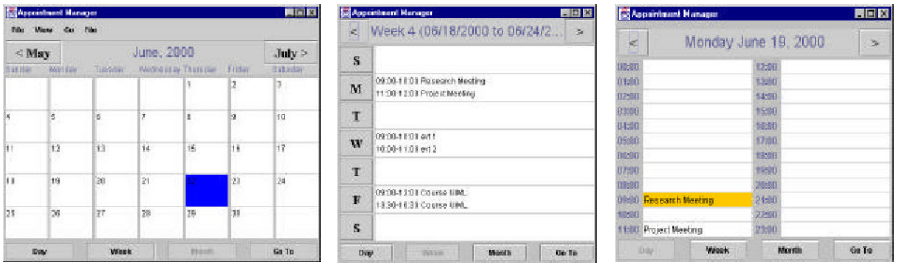
Het ontwerpen van grafische gebruikersinterfaces voor het web is verschillend van het ontwerpen ervan voor traditionele software. Voornamelijk omdat er op het web zoveel verschillende hard- en softwareplatformen aanwezig zijn en de ontwerper vaak niet de mogelijkheid heeft om voor elk platform een afzonderlijk design te maken.

### 1.3.1 Diversiteit aan toestellen

Tegenwoordig kan men niet alleen met een traditionele computer op het Internet surfen en webapplicaties gebruiken, maar ook via een TV, een hand-held computer, een GSM, de boordcomputer van een wagen enz. Al deze verschillende toestellen hebben een verschil in schermoppervlakte dat tot een factor 100 kan oplopen en de beschikbare bandbreedte voor een GSM met WAP1 mogelijkheid is 1000 maal kleiner dan een T-3 connectie (zie [Alertbox 1997]).

**Figuur 4** geeft het maand-, week- en dagoverzicht van een kalenderapplicatie voor het Java-platform weer. **Figuur 3** en **Figuur 2** tonen hetzelfde, maar dan voor het PalmOS en het WML[2] platform repectievelijk.

[1] WAP: Wireless Application Protocol is de methode waarmee men via een GSM op het Internet kan surfen.
[2] WML: Wireless Markup Language: de tegenhanger van HTML voor draadloos surfen via bijvoorbeeld een GSM



**Figuur 4: Maand-, week- en dagoverzicht in Java**

**Figuur 3: Maand-, week- en dagoverzicht voor het PalmOS platform**


**Figuur 2: Maand-, week-, dag- en samengesteld overzicht voor het WML platform**

Door deze diversiteit vallen natuurlijk een hele boel zekerheden voor de designer weg. Voor traditionele software was het WYSIWYG-principe[1] nog toepasbaar, maar nu ziet een ontwerp er op elk apparaat anders uit. Die verscheidenheid moeten we echter niet zien als een gebrek, maar als iets extra's, want een optimale gebruikerservaring vereist een aanpassing aan de karakteristieken van het apparaat van de eindgebruiker.

[1] *What You See Is What You Get*

Het ontwerpen van een abstracte gebruikersinterface, die dan telkens op een andere manier geïnstantieerd wordt, afhankelijk van het platform, is een ingewikkelde zaak, zeker als we ook de interactie met de gebruiker moeten abstraheren.

Voor *thin clients* bestaan er al hulpmiddelen om betekenis en representatie te scheiden, zoals de combinatie van HTML en *Cascading Style Sheets* (zie [CSS]). *Rich client* technologieën hebben echter vaak een verschillende implementatie per platform en er zijn geen standaarden voor.

### 1.3.2 De gebruiker bepaalt de weg

Bij traditionele grafische gebruikersinterfaces is het mogelijk om de gebruiker te verhinderen bepaalde acties te ondernemen door bijvoorbeeld knoppen tijdelijk onbruikbaar te maken, of een dialoogvenster te tonen waarop de gebruiker eerst moet antwoorden alvorens hij of zij verder iets anders kan doen.

Op het Internet bepaalt de surfer echter wat er gebeurt en waar hij naartoe gaat. Zo kan iemand, via een zoekmachine bijvoorbeeld, onmiddellijk midden in een site terecht komen, zonder daarbij langs de startpagina te passeren. Indien de ontwerper van de site dit niet voorzien had, zou het wel eens kunnen dat de gebruiker bepaalde functionaliteit, zoals een navigatiemenu, nooit te zien krijgt.

Iemand die op het Internet vertoeft kan ook plotseling beslissen om niet meer verder te werken met een bepaalde site door op de “Terug” knop te drukken of zijn browser af te sluiten.

*Rich clients* bieden vaak de mogelijkheden die ook bij traditionele gebruikersinterfaces voorhanden zijn om de navigatie van de gebruiker te controleren. Ze hebben echter het nadeel dat ze de gebruiker soms niet de indruk geven dat hij of zij zich op het Internet bevindt; zo ís er bijvoorbeeld helemaal geen browser, of werken bepaalde onderdelen van de browser niet in combinatie met de *rich client* technologie (zoals de “Terug” knop bvb.).

### 1.3.3 Gebrek aan richtlijnen op het web

In traditioneel ontwerp van grafische gebruikersinterfaces is er meestal een stijlgids voorhanden waarop de designer zich kan baseren (zie ook **2.2.1** op pagina 18). Tot op heden is er geen equivalent voor de stijlgids voor het maken van websites of webapplicaties.

## 1.4 Conclusie

Door de opsplitsing van applicaties in drie lagen is een mogelijkheid tot communicatie van een bepaalde technologie met de “buitenwereld” noodzakelijk. Zo moet de presentatielaag op een of andere manier de logische laag kunnen aanspreken.

De voornaamste voordelen van een *thin client* zoals die voorkomt in een webapplicatie met een op HTML gebaseerde gebruikersinterface, zijn de platformonafhankelijkheid en de beperkte administratieoverhead voor de gebruiker.

Een *rich client* biedt mogelijkheden voor een veel gebruiksvriendelijkere ervaring, maar de keerzijde van de medaille zijn de extra installatie- en configuratiekosten.

**Tabel 2: Samenvatting criteria uit Hoofdstuk 1**

| | |
|---|---|
| C1 | Communicatiemogelijkheid met andere applicaties via een gebruikelijke standaard. |
| C2 | Platformonafhankelijkheid, of ondersteuning door zoveel mogelijk verschillende platformen. |
| C3 | Mogelijkheden van een *rich client* m.b.t. interactie en grafische expressie. |

De criteria voor een webtechnologie die we uit dit hoofdstuk halen zijn samengevat in **Tabel 2**.

Ten eerste willen we een technologie die op zoveel mogelijk platformen kan worden gebruikt, teneinde de ontwikkelingskosten te drukken en het doelpubliek zo ruim mogelijk te houden. Een tweede criterium is dat er een mogelijkheid moet zijn om de presentatielaag (de gebruikersinterface), die met de webtechnologie werd ontwikkeld, te verbinden met een achterliggende applicatie. Hoe meer verschillende standaarden en protocols de webtechnologie ondersteunt, hoe beter. Ten slotte kunnen webtechnologieën met de mogelijkheden van een *rich client* zorgen voor meer tevredenheid bij de gebruikers van de applicaties die ermee ontwikkeld werden.

# Hoofdstuk 2

# Moderne grafische gebruikersinterfaces

Gebruikersinterfaces zijn in de loop der jaren snel geëvolueerd. Oorspronkelijk waren ze commandogebaseerd, tegenwoordig maken ze vrijwel allemaal gebruik van de grafische mogelijkheden van een computer. De uiteindelijke bedoeling van een interface is het werken met de computer eenvoudiger en/of sneller te maken.

In dit hoofdstuk beschrijven we de karakteristieken van een moderne grafische gebruikersinterface en de principes die gehanteerd worden om dergelijke interfaces te ontwikkelen.

## 2.1 Karakteristieken van een moderne grafische gebruikersinterface

De meeste al dan niet commerciële software die vandaag de dag door nagenoeg iedereen die met een computer werkt gebruikt wordt, heeft een grafische gebruikerinterface. Programma's als Word, Netscape Communicator, Eudora, Winamp en StarOffice zijn daar slechts enkele voorbeelden van.

Men spreekt soms ook van een WIMP-interface[1] met vensters, icoontjes, menu's en aanwijzers. Toonaangevend op dit gebied zijn de interfaces van de besturingsystemen van Microsoft, Apple Macintosh en de "grafische schillen" rond Linux, zoals KDE en Gnome.

Kenmerkend voor de interfaces van software die onder bovengenoemde systemen draait, zijn het gebruik van tekst in verschillende opmaak en grootte, de weergave in verschillende kleuren en het voorstellen van objecten door iconen of kleine grafische representaties.

Om de gebruiker te tonen wat er gebeurt, wordt er ook vaak gebruik gemaakt van geluiden en/of bewegende beelden in de vorm van animatie.

De gebruiker communiceert of werkt met de applicatie via invoerapparaten zoals een toetsenbord, een muis of een aanraakscherm.

Om de ontwikkeling van een gebruikersinterface, die de hoger genoemde karakteristieken heeft, vlot te laten verlopen, heeft de ontwikkelaar ervan vaak veel steun aan een grafische ontwikkelingsomgeving. Op die manier heeft hij of zij een beter idee van het uitzicht van het afgewerkte product. Het gebruik van een grafische omgeving voor de ontwikkeling van een grafische gebruikersinterface kan dus zeer nuttig en tijdsbesparend zijn.

## 2.2 Principes voor de ontwikkeling van grafische gebruikersinterfaces

### 2.2.1 De stijlgids

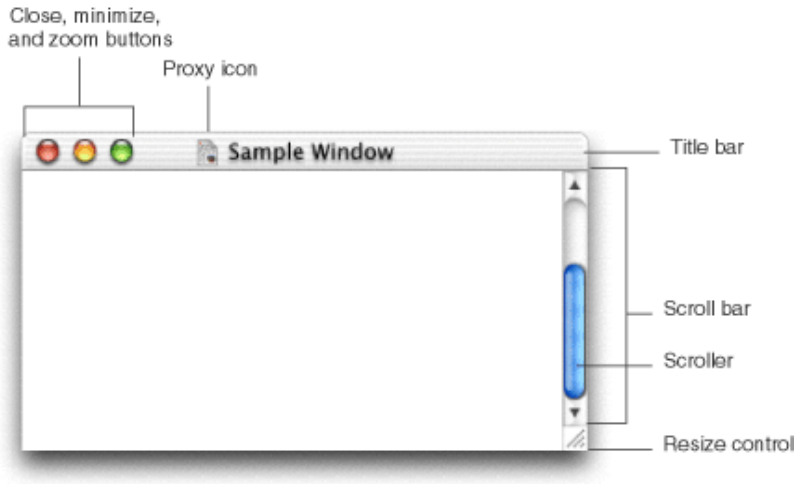
Gebruikersinterfaces worden het best ontworpen volgens een welbepaalde stijlgids. Een dergelijk stijlgids kan specifiek gedefinieerd worden voor een bepaalde applicatie of groep van applicaties, maar vaak wordt er teruggegrepen naar een welbekende en veel gebruikte stijlgids. Voorbeeld hiervan is de Microsoft Windows stijlgids.

[1] WIMP = Windows Icons Menus Pointers

Oorspronkelijk was deze stijlgids bedoeld voor de ontwikkelaars van Microsoft zelf, maar tegenwoordig wordt hij gehanteerd door iedereen die een applicatie bouwt voor het Windows-platform.

Deze stijlgids beschrijft hoe een gebruikersinterface er dient uit te zien en bevat een aantal richtlijnen met betrekking tot de organisatie van menu's, de positie en het gedrag van knoppen, de lay-out van dialoogvensters enz.

Een ander voorbeeld van een bekende stijlgids is deze voor de *Aqua* interface van MacOS X [Aqua Stijlgids]. **Figuur 5** toont het ontwerp van een standaard venster in MacOS X, zoals het er volgens de stijlgids uit moet zien.




**Figuur 5: Fragment uit de stijlgids van MacOS X: Onderdelen van een standaard *window* of venster**

Door het gebruik van een stijlgids voorkomt men inconsistenties in de *look* en het gedrag van de gebruikersinterface binnen hetzelfde platform. Een ontwikkelaar die software maakt voor een bepaald platform hoeft niet na te denken over het uitzicht van zijn programma, hij dient gewoon de richtlijnen van de stijlgids te volgen. De gebruikers zullen hierdoor ook minder problemen ondervinden om met nieuwe software te leren werken.

### 2.2.2 De bibliotheken

Om de implementatie van een grafische gebruikersinterface te vereenvoudigen worden bibliotheken of *libraries* meegeleverd met programmeeromgevingen of besturingssystemen die routines bevatten om de GUI op te bouwen; bijvoorbeeld routines om een venster te openen/sluiten, om een dialoogvenster te openen/sluiten, om een druk op een toets of een muistoets op te vangen… Al deze bibliotheken worden verzameld in een zogenaamde *Application Programming Interface* of API.

De Microsoft API voor de Windows gebruikersinterface bevindt zich bijvoorbeeld in het bestand "User32.dll" van het besturingssysteem en is geschreven voor C/C++, maar kan ook benaderd worden vanuit Visual Basic (zie [MSDN] voor meer informatie). De Swing API zorgt ervoor dat een Java-programmeur op een eenvoudige manier gebruikersinterfaces kan bouwen.

We kunnen de API onderdelen vanuit een objectgeoriënteerd standpunt zien als een verzameling herbruikbare standaard-componenten. Ten eerste verlichten deze componenten het werk van de programmeur door het hergebruik van programmacode en ten tweede dwingen ze een interface conform met de stijlgids af.

### 2.2.3 Programmeren op basis van gebeurtenissen

Om een interactieve gebruikersinterface te maken moet deze kunnen reageren op gebeurtenissen of *events*. Een gebeurtenis wordt meestal veroorzaakt door externe factoren, bijvoorbeeld het aanklikken van een knop door de gebruiker of het ontvangen van een boodschap over het netwerk.

In tegenstelling tot traditionele programma's, "loopt" een op gebeurtenissen gebaseerd programma niet eenvoudigweg van begin tot einde, maar wordt een bepaalde procedure of functie (de *event-handler*) geactiveerd door een gebeurtenis. Doordat er geen begin en einde aan het programma zijn en omdat gebeurtenissen niet noodzakelijk altijd in de zelfde volgorde plaatsvinden, is het veel moeilijker om een dergelijk programma te begrijpen; je kunt het niet "lezen" van begin tot einde als het ware.

Het concept van gebeurtenissen moet ondersteund worden door de programmeertaal of door de omgeving waarin de gebruikersinterface wordt ontwikkeld om de implementatie van een moderne gebruikersinterface enigszins haalbaar te maken. Enkele voorbeelden uit een *tutorial* over Java Swing (zie [Swing Tutorial]) zijn weergegeven in **Tabel 3**.

**Tabel 3: Gebeurtenissen voor het Java Swing platform**

| **Act that results in the event** | **Listener type** |
|---|---|
| User clicks a button, presses Return while typing in a text field, or chooses a menu item | `ActionListener` |
| User closes a frame (main window) | `WindowListener` |
| User presses a mouse button while the cursor is over a component | `MouseListener` |
| User moves the mouse over a component | `MouseMotionListener` |
| Component becomes visible | `ComponentListener` |
| Component gets the keyboard focus | `FocusListener` |
| Table or list selection changes | `ListSelectionListener` |

### 2.2.4 Gebruiksvriendelijkheid

Met de gebruiksvriendelijkheid of *usability* van een systeem bedoelen we het gemak waarmee een systeem te gebruiken is, zijn veiligheid, effectiviteit en efficiëntie en ook de houding en tevredenheid van de gebruiker tegenover het systeem (naar [Preece et al. 1994]).

Er is al heel wat onderzoek verricht naar de gebruiksvriendelijkheid van klassieke gebruikersinterfaces. Als gevolg daarvan is het werken van de interface van de meeste commerciële programma's bevredigend tot goed te noemen.

Gebruikersinterfaces voor webapplicaties laten op het gebied van gebruiksvriendelijkheid vaak te wensen over. Dit is het gevolg van het feit dat de meeste websites en webapplicaties ontworpen worden door personen die daar niet in geschoold zijn. Op zijn best zijn het webtechnologen of grafische vormgevers die weinig of geen kennis hebben van de principes van het ontwerpen van gebruikersinterfaces (*UI Design*). Bovendien is het ook lang niet duidelijk of men de principes voor gebruiksvriendelijkheid van klassieke gebruikersinterfaces zonder meer kan toepassen op webinterfaces.

De meeste computergebruikers zijn gewend aan de manier van werken met de klassieke bureaubladapplicaties in de context van de moderne besturingssystemen zoals Windows, MacOS, Linux…, maar die kunnen niet volledig worden nagebouwd met behulp van klassieke webtechnologieën zoals HTML, CSS[1] en JavaScript.

De communicatie via het Internet, zelfs via een breedbandverbinding, is enkele grootteordes trager dan de communicatie binnen een computer of een werkstation. Daardoor is ook de responstijd op een actie van de gebruiker, zelfs voor heel eenvoudige taken, soms vrij lang, wat de gebruiksvriendelijkheid van de webapplicatie niet ten goede komt. Zo veel mogelijk "nutteloze" communicatie vermijden kan de ervaring voor de gebruikers dus veel aangenamer maken.

## 2.3 Conclusie

Moderne grafische gebruikersinterfaces maken veelvuldig gebruik van tekstopmaak, kleuren, animatie en geluid om de interface aantrekkelijker en gebruiksvriendelijker te maken. Ondersteuning vanuit de ontwikkelingsomgeving en de programmeertaal om met deze elementen te werken, vereenvoudigt het werk van de programmeur.

Om consistente en bruikbare applicaties te kunnen ontwikkelen is ondersteuning voor het (her)gebruik van bibliotheken of standaardcomponenten onontbeerlijk.

De interactie met een gebruiker vereist dat de interface kan reageren op gebeurtenissen veroorzaakt door de gebruiker. Om het programmeren op basis van gebeurtenissen mogelijk te maken, is de ondersteuning ervoor vanuit de ontwikkelingsomgeving of programmeertaal vereist.

De criteria voor een webtechnologie die we uit het tweede hoofdstuk halen zijn samengevat in **Tabel 4** op de volgende pagina.

[1] CSS: *Cascading Style Sheets* (zie ook [CSS])

**Tabel 4: Samenvatting criteria uit hoofdstuk 2**

| | |
|---|---|
| C4 | Ondersteuning voor het gebruik van bibliotheken of standaardcomponenten. |
| C5 | Ondersteuning voor het programmeren op basis van gebeurtenissen. |
| C6 | Ontwikkelingsomgeving en programmeertaal met ondersteuning voor het maken van grafische elementen (tekstopmaak, iconen…). |

# Hoofdstuk 3

# Informatievisualisatie

Vandaag de dag is er een enorme hoeveelheid data beschikbaar uit diverse informatiesystemen, in databases of databanken, op het Internet enz. Het ritme van onze maatschappij verplicht ons vaak informatie zo snel en efficiënt mogelijk te verwerken. In tegenstelling tot computers slaat een mens zijn “gegevens” niet op in tabellen en bestanden, maar in beelden en structuren. Is het gezegde niet: “één plaatje zegt meer dan duizend woorden”?

Een visuele voorstelling van gegevens kan ons vaak helpen bij het interpreteren van de immense datastroom die we te verwerken krijgen. We bedoelen hier met visuele voorstelling geen “wetenschappelijke visualisatie” van gegevens, maar wel de visualisatie van informatie.

In het eerste geval proberen we een fysiek verschijnsel visueel, zo realistisch mogelijk, voor te stellen. Zoals het debiet van een rivier op een bepaalde plaats, of de hoeveelheid neerslag in een bergachtig gebied.

Met de visualisatie van informatie bedoelen we het visueel voorstellen van abstracte informatie, zoals voetbaluitslagen of de fluctuatie van wisselkoersen.

Hoewel er een overlapping is tussen de visualisering van een fysiek verschijnsel en de visualisering van informatie, concentreren we ons in deze verhandeling op het voorstellen van abstracte gegevens, die dus geen directe tegenhanger hebben in de “echte” wereld.

We zullen in dit hoofdstuk verder uitleggen wat informatievisualisatie is en waarvoor ze gebruikt wordt. In het kader van deze thesis zullen we ook onderzoeken aan welke criteria een webtechnologie moet voldoen om het visualiseren van informatie te vereenvoudigen.

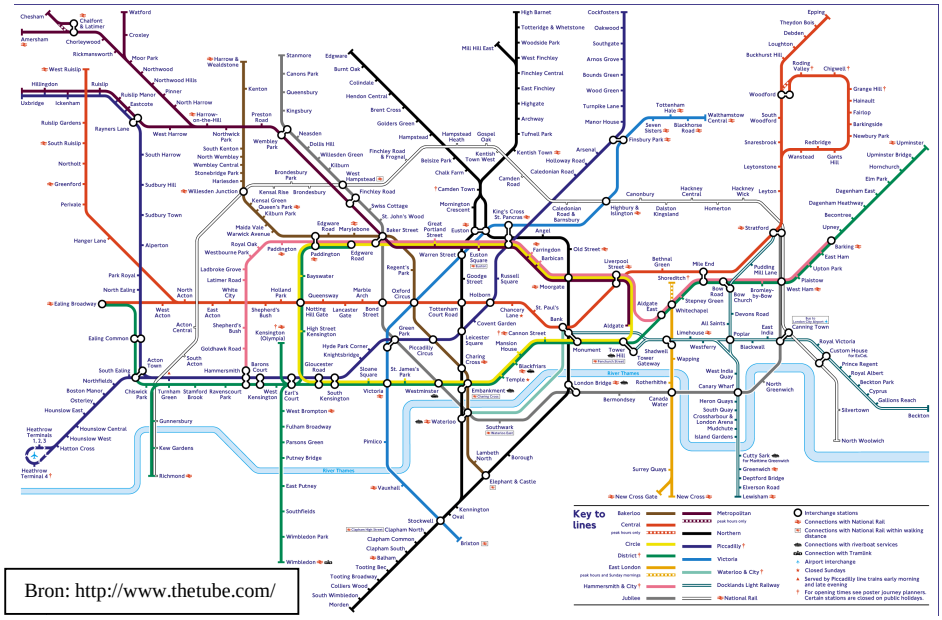
## 3.1 Wat is informatievisualisatie?

### 3.1.1 Definitie

Informatievisualisatie is het proces waarbij we ons een mentaal model van gegevens vormen, waardoor we er (meer) inzicht in krijgen (naar [Spence 2001]).

### 3.1.2 Ontstaan

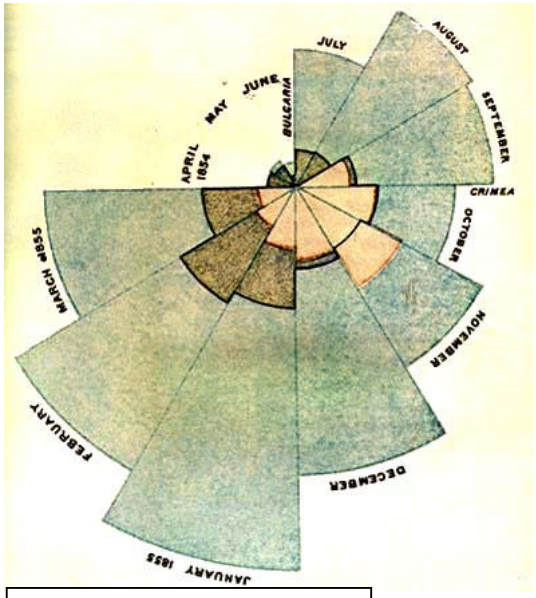
Visualisaties werden oorspronkelijk – toen de computer nog niet bestond of nog niet krachtig genoeg was – door mensen opgesteld. Een mooi voorbeeld van informatievisualisatie is het plan dat soms gebruikt wordt om het spoorwegnet of de metro voor te stellen. **Figuur 6** toont een plan van de Londense metro.




**Figuur 6: Plan van de Londense metro (*London Underground*)**

In tegenstelling tot een gewoon plan, dat de geografische omgeving zo realistisch mogelijk probeert voor te stellen, is dit de visualisatie van gegevens met betrekking tot de verschillende lijnen en haltes van een metronet.

Overbodige informatie – voor de metrogebruiker althans – wordt hier weggelaten. Iemand die de metro wil nemen is namelijk niet geïnteresseerd in de fysieke ligging van de metrotunnels, maar wil gewoon weten welke lijn hij of zij moet nemen om zich van de ene halte naar de andere te verplaatsen. Dit plan is dus geen exacte representatie van de werkelijkheid, maar een voorstelling van informatie, teneinde de gebruiker te helpen bij het vormen van een mentaal model. Zo kan iemand bijvoorbeeld onthouden in welke kleur een bepaalde lijn wordt voorgesteld om ze snel op de kaart terug te vinden of, aan de hand van bolletjes die met elkaar verbonden zijn, zien in welke haltes er overgestapt kan worden op een andere lijn.




Bron: http://www-gap.dcs.st-and.ac.uk/

**Figuur 7: Een "Roos van Nightingale" uit 1855 beeldt het aantal sterfgevallen per ziekenhuis per maand af**

De kaart van de Londense metro werd uitgevonden in 1931 door Harry Beck, dus lang voor er sprake was van visualisaties met behulp van de computer dus.

Er bestaan zelfs nog oudere visualisaties zoals de rozen van Florence Nightingale (**Figuur 7**) uit 1855, die het aantal doden per maand in een Brits ziekenhuis weergeven, en de kaart van dokter John Snow uit 1845 (**Figuur 8**), die de plaats van een overlijden door cholera en de ligging van waterpompen afbeeldt.


**Figuur 8: Kaart van dokter John Snow uit 1845 die de plaats van cholera sterfgevallen en de locatie van waterpompen weergeeft.**

Alle hoger genoemde visualisaties hebben tot doel informatie en vooral inzicht in die informatie over te brengen. De mogelijkheden om dat doel te bereiken kunnen zeer uiteenlopend zijn en zijn vaak het resultaat van de creativiteit van hun schepper. Wat belangrijk is, is dat degene die de visualisatie bekijkt snel een inzicht – een soort “Aha!”-effect – krijgt.

## 3.2 Visualisatietechnieken

Er bestaat een hele waaier van mogelijke visuele voorstellingen van abstracte informatie. Ze hier allemaal bespreken zou ons te ver leiden, maar er zijn er een aantal visualisatietechnieken die meer gebruikt worden dan andere en die bijzonder geschikt zijn om de organisatie van grote hoeveelheden gegevens weer te geven.

Deze visualisatietechnieken worden al in bepaalde computer-applicaties gebruikt als gebruikersinterface. We bespreken deze door de computer gegenereerde visualisaties hier kort. Voor een uitgebreidere bespreking: zie [Spence 2001] en [De Greef 2001].

### 3.2.1 Standaard visualisatie technieken

Een aantal "standaard" visualisatietechnieken wordt al veelvuldig gebruikt in gebruikersinterfaces, zoals de boomstructuur om de organisatie van folders in een besturingssysteem weer te geven.

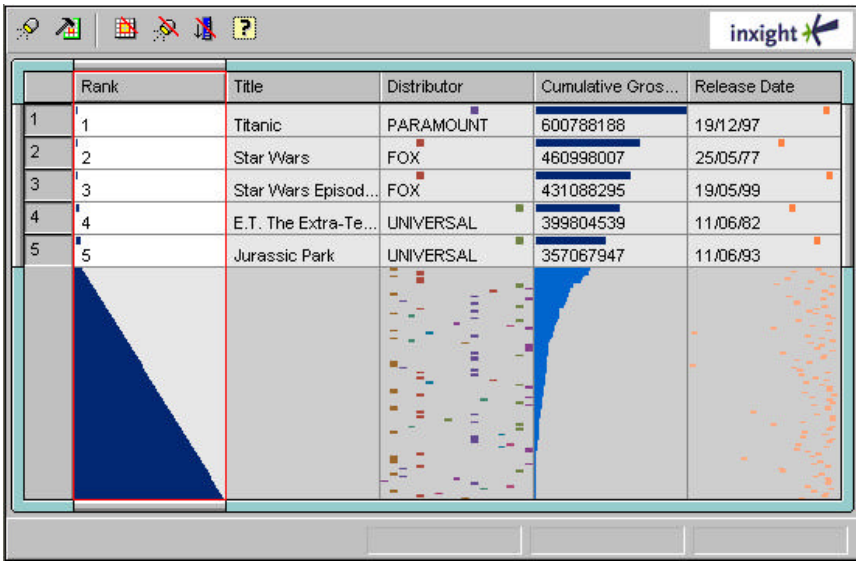
Het probleem bij het afbeelden van een hiërarchie, door middel van een gewone boomstructuur in een plat vlak, is dat de plaats, die nodig is voor het afbeelden van de afstammelingen (*children*) van een bepaalde knoop, steeds beperkter wordt naarmate we afdalen in de hiërarchie of naarmate de hoeveelheid afstammelingen van dezelfde knoop toeneemt.

### 3.2.2 Geavanceerde visualisatietechnieken

Hieronder bespreken we drie geavanceerde visualisatietechnieken: de *table lens*, *cone trees* en de hyperbolische browser.

#### ***Table lens***

Met de *table lens* kunnen we de verschillende cellen van een tabel met meerdere rijen en kolommen voorstellen als histogrammen. Deze techniek is vooral nuttig voor tabellen met veel cijfermateriaal. Door de tabel te sorteren op een bepaalde kolom kunnen we in één oogopslag aan de histogrammen bepaalde informatie waarnemen.



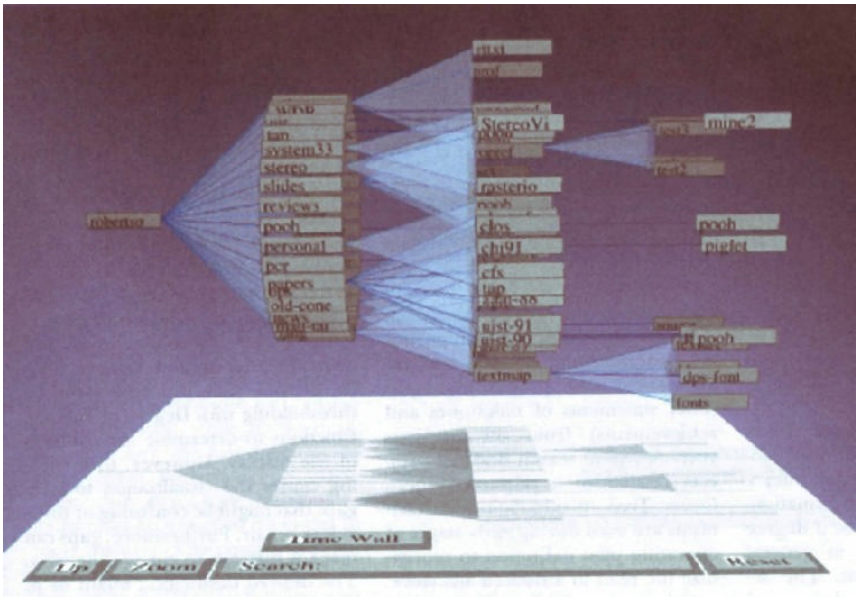

**Figuur 9: Table lens visualisatie: Top 100 van de meest opbrengende films aller tijden gesorteerd op opbrengst (de top 5 is uitvergroot)**

**Figuur 9** toont de top 100 aller tijden van de films die het meeste geld opbrachten in de bioscoop. We zien bijvoorbeeld onmiddellijk dat er geen verband bestaat tussen de opbrengst van een film (voorlaatste kolom) en de leeftijd ervan (laatste kolom).

### *Cone-Trees*

De *Cone-Tree* is een kegelvormige boomstructuur die wordt gebruikt voor het weergeven van grote hiërarchische structuren in drie dimensies. Het voordeel van die derde dimensie is dat er veel meer plaats is voor het weergeven van de afstammelingen van een bepaalde knoop, zoals **Figuur 10** illustreert.

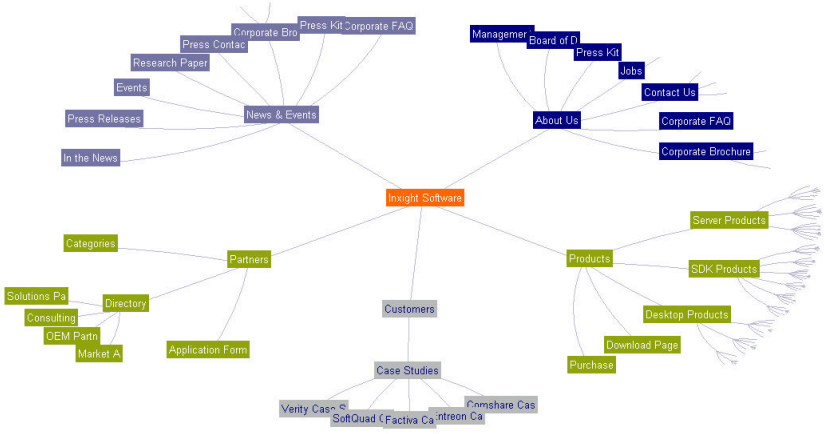
Bij het manipuleren van een C*one-Tree* kunnen de afstammelingen van een bepaalde knoop rondgedraaid worden, om op die manier alle informatie te kunnen waarnemen.




**Figuur 10: Voorbeeld van een Cone-Tree**

## ***Hyperbolische boom***

Een hyperbolische boom beeldt een hiërarchische structuur af op een hyperbolisch vlak (zie [Hyperbolische Meetkunde]). Dat vlak heeft een aantal gunstige eigenschappen, waardoor de plaats voor het afbeelden van knopen beter benut wordt dan het geval is bij gewone boomstructuren.




**Figuur 11: Voorbeeld van een hyperbolische browser Inxight Star Tree™**

**Figuur 11** toont de hyperbolische browser, Star Tree™ genaamd, van Inxight, een dochteronderneming van Xerox die gespecialiseerd is in het ontwikkelen van door de computer gegenereerde visualisaties en daarop gebaseerde gebruikersinterfaces (zie [Inxight]).

De hyperbolische boom is een goed voorbeel van een *Focus+Context*[1] visualisatie, waarbij het midden van de boom het focuspunt is. Elke knoop van de hyperbolische browser kan met de muis versleept worden naar het centrum, waardoor hij meer in de aandacht komt.

Het animeren van de verschillende toestanden tussen het aanklikken, verslepen en loslaten van een knoop, kan voor meer duidelijkheid en een beter begrip van de werking van een hyperbolische boom bij de gebruiker zorgen.

## 3.3 Conclusie

De reden dat we hier enkele visualisatietechnieken kort bespraken, is dat ze ons in de volgende hoofdstukken in staat zullen stellen om de verschillende webtechnologieën beter te evalueren.

Het toepassen van een visualisatie als gebruikersinterface voor een applicatie is namelijk een vrij nieuw en complex proces.

Binnen het kader van deze verhandeling willen we vooral weten of moderne webtechnologieën ons in staat stellen om op een eenvoudige manier informatie te visualiseren. Het implementeren van klassieke gebruikersinterfaces is relatief eenvoudig en er zijn veel hulpmiddelen zoals de stijlgids en diverse voorbeelden, waarop men zich kan baseren.

Geavanceerde visualisatietechnieken bevinden zich, hoewel ze al op diverse plaatsen opduiken, nog in een experimenteel stadium. De implementatie ervan vraagt niet alleen meer creativiteit van de programmeur, maar ook van de technologie en de ontwikkelingsomgeving waarmee we een geavanceerde visualisatie willen maken.

[1] Met een *Focus+Context* visualisatie bedoelen we dat een bepaald deel van de visualisatie waarin we geïnteresseerd zijn groter of duidelijker wordt afgebeeld (de focus). De overige informatie wordt daarbij kleiner of onduidelijker weergegeven (de context), zodat we het overzicht niet verliezen.

Ondersteuning voor het tekenen of importeren van de verschillende interfaceonderdelen in de ontwikkelingsomgeving, kan het werk van de programmeur verlichten. Ten eerste kan hij of zij dan onmiddellijk zien hoe een bepaald grafisch element eruit ziet, zonder daarvoor code te moeten compileren of interpreteren. Ten tweede kan het ontwerpen van een grafisch element eventueel door iemand anders met meer kennis terzake worden uitgevoerd.

De ondersteuning voor animatie van de interface kan bijdragen tot een geslaagde visualisatie en meer gebruiksvriendelijkheid.

We besluiten uit dit hoofdstuk dat de webtechnologieën waarmee we geavanceerde gebruikersinterfaces willen ontwikkelen, moeten voldoen aan de criteria uit **Tabel 5**.

**Tabel 5: Samenvatting criteria uit hoofdstuk 3**

| | |
|---|---|
| C7 | Ondersteuning vanuit de ontwikkelingsomgeving voor het zelf tekenen of importeren van de grafische elementen. |
| C8 | Ondersteuning voor het eenvoudig implementeren van animatie van grafische elementen. |

# Hoofdstuk 4

# Overzicht van enkele webtechnologieën

In dit hoofdstuk bespreken we drie webtechnologieën die momenteel gebruikt worden om dynamische websites of een echte webapplicatie te maken. De drie technologieën zijn Curl, Flash en Shockwave.

Curl is nog een vrij onbekende technologie op het moment van dit schrijven, maar ze kan rekenen op de steun van enkele gerenommeerde “Internetgoeroes”. Flash van Macromedia is geëvolueerd van een animatiepakket voor het web tot een volwaardige webtechnologie en Shockwave, dat afkomstig is van dezelfde producent, zou het grotere broertje zijn van Flash als het aankomt op het creëren van *rich media*.

We zullen voor elke technologie kort zijn ontstaansgeschiedenis aanhalen en de werking van deze technologieën beschrijven. Voor meer informatie, *tutorials*, voorbeelden, bibliotheken en dergelijke kunnen voor Curl, Flash en Shockwave respectievelijk de volgende websites geraadpleegd worden: [Curl], [Flash] en [Shockwave].

# 4.1 Curl

## 4.1.1 Wat is Curl?

Curl is een programmeertaal ontworpen voor de ontwikkeling van interactieve Internetapplicaties. Curl combineert het gemak van opmaaktalen zoals HTML met de functionaliteit van een objectgeoriënteerde programmeertaal. Op die manier willen de uitvinders van Curl de karakteristieken en mogelijkheden van opmaaktalen (zoals HTML), scripttalen en objectgeoriënteerde softwareontwikkeling in één omgeving verenigen.

De *Curl Content Language* werd voor het eerst gedefinieerd en ontwikkeld door Tim Berners-Lee, wijlen Michael Dertouzos en andere leden van het *Massachussets Institute of Technology, Lab of Computer Science*. Berners-Lee en Dertouzos startten *Curl Corporation* op, om een commerciële implementatie van de taal te lanceren. Hoewel de taalspecificatie nu nog niet vrijgegeven is, heeft het bedrijf beloofd delen van de technologie vrij beschikbaar te maken.

Curl programma's worden gemaakt in de ontwikkelingsomgeving, "Surge Lab IDE[1]" genaamd, die *run-time* bibliotheken levert, een dynamische compiler, en een *renderer*.

Curl applicaties worden geleverd onder de vorm van broncode, die dan gecompileerd wordt aan de *client*-zijde door de *Surge plug-in*, een *run-time* omgeving onder de vorm van een browser *plug-in*. Het idee hierachter is dat de hoeveelheid gegevens die over het Internet verstuurd moet worden drastisch verlaagt. De broncode van een programma is doorgaans vele malen kleiner dan de gecompileerde code.

Hiermee is ook het grootste verschil tussen Curl en klassieke webtechnologieën aangehaald: de code wordt niet op de server, maar op de *client* uitgevoerd en zelfs gecompileerd.

Curl is momenteel enkel beschikbaar voor het Windows-platform, ondersteuning voor Mac OS X en Linux (waarvan al een previewversie bestaat) wordt verwacht.

[1] IDE = *Integrated Development Environment*

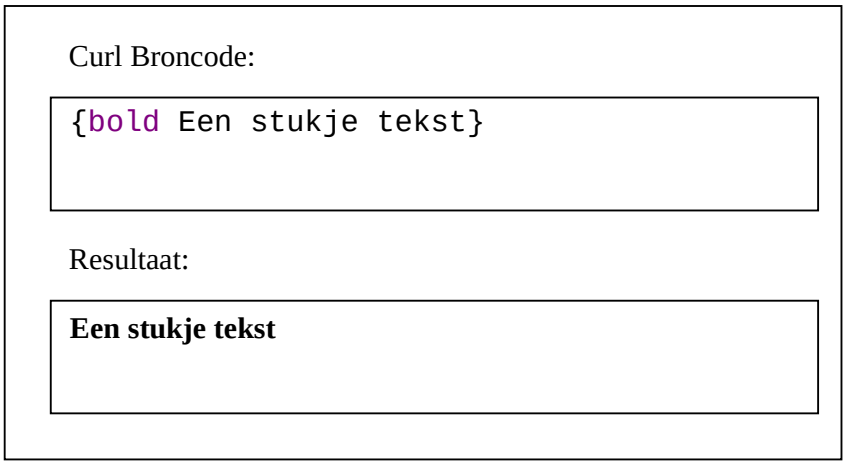
### 4.1.2 Hoe werkt Curl?

Curl technologie is tegelijk een programmeertaal en een taal voor de opmaak en specificatie van inhoud.

Je kunt er tekst mee formatteren, lay-out mee bepalen en er objectgeoriënteerde programma's mee schrijven. Dit alles wordt in één formaat afgeleverd, geïnterpreteerd en visueel getoond door middel van de Curl *engine* die in de plug-in verwerkt is.

De standaard output van een Curl programma is een tekstdocument, waardoor Curl eenvoudig gebruikt kan worden om een webpagina te genereren.

De taal is qua syntax vergelijkbaar met tekstopmaak talen als $T_EX$ en HTML en de programmeertaal Lisp. Gebruikelijk worden tekstattributen en andere functies toegevoegd tussen accolades. De naam van de functie is het eerste element gevolgd door de parameters; voor een voorbeeld: zie **Figuur 12.**

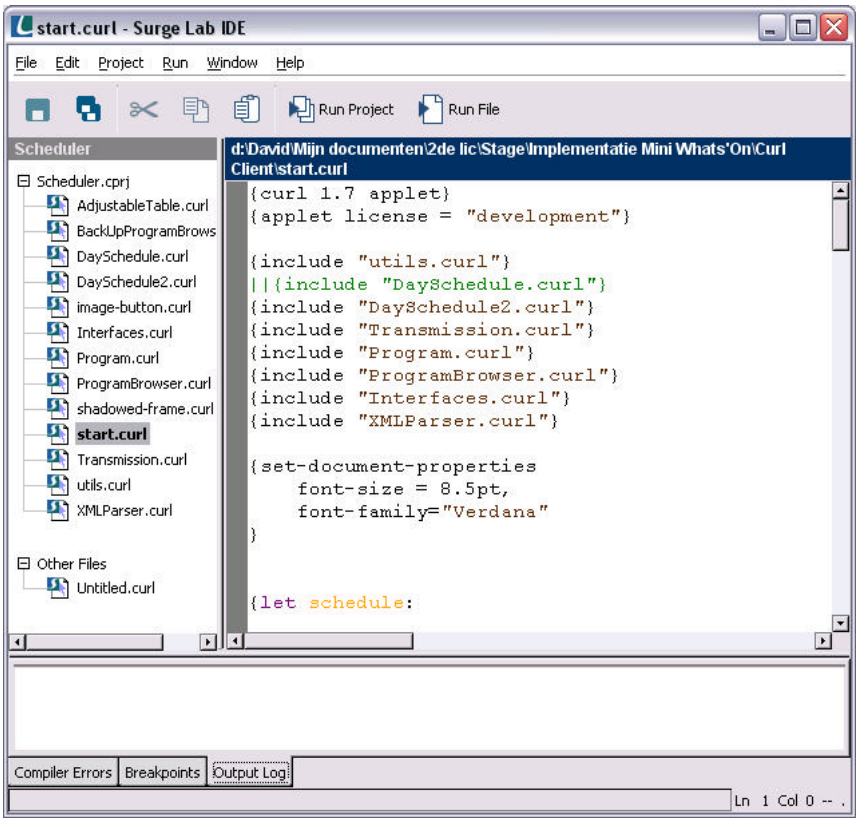



**Figuur 12: Curl syntax voorbeeld**

De taal achter Curl is een sterk getypeerde, objectgeoriënteerde programmeertaal (zoals Java en C++). Aangezien Curl over de mogelijkheden van een complete programmeertaal beschikt, kunnen er gebruikersinterfaces mee gebouwd worden, net zoals dat met een taal als Java of C++ kan, nl. met bibliotheken of *libraries*.

Er zijn een aantal bibliotheken aanwezig in de distributie van de ontwikkelingsomgeving voor Curl, die onder andere routines voor gebruikersinterfaces bevatten. Ook ondersteuning voor menu's, knoppen, tekstvelden, enz. is aanwezig.

De ontwikkelingsomgeving Surge Lab IDE lijkt in de meeste opzichten op de traditionele programmeeromgevingen, zoals we die kennen van Java of C++ bijvoorbeeld. Surge Lab IDE beschikt over een *editor*, een *debugger*, een *applet viewer*, een *inspector* om de details van grafische objecten te bekijken en een *release tool* om de verschillende versies van een applicatie te onderscheiden. **Figuur 13** toont een schermafdruk van de *editor*. Er wordt gewerkt met projecten om de verschillende bestanden die bij elkaar horen te organiseren.




**Figuur 13: Curl ontwikkelingsomgeving: Surge Lab IDE *editor***

```
<embed
     src="url_naar_de_component.xxx"
     pluginspage="pagina_plugin.html"
     type="mime/type"
     width="200"
     height="500"
>
```

**Figuur 14: HTML *embed* syntax**

Op het moment van dit schrijven beschikt Curl niet over een grafische omgeving om gebruikersinterfaces te maken, hoewel daar volgens de website [Curl] wel plannen voor zijn.

Een Curl *applet* kan in een HTML pagina geïntegreerd worden via de *embed* syntax van HTML (zie **Figuur 14**) of op zichzelf gebruikt worden wanneer de computer daarvoor is geconfigureerd.

## 4.2 Flash

### 4.2.1 Wat is Flash?

Flash is een pakket dat ontwikkeld wordt door Macromedia en oorspronkelijk bedoeld was om vectorgebaseerde[1] animaties voor het Internet te maken. Door de snelle vooruitgang van het web is ook Flash mee geëvolueerd en momenteel kan het voor meer dan enkel animaties gebruikt worden.

Flash bestaat uit een ontwikkelingsomgeving (ook wel *authoring envvironment* genoemd) en een plug-in voor een webbrowser (ook wel de Flash *player* genoemd). De plug-in voor Flash 6 is slechts 383 Kb groot en eenvoudig te installeren.

[1] Bij vectorgebaseerde grafische voorstellingen worden de wiskundige bewerkingen voor het reconstrueren van een tekening opgeslagen, dit in tegenstelling tot *bitmapped* grafische voorstellingen, waarbij de informatie voor elke pixel van de tekening wordt opgeslagen.

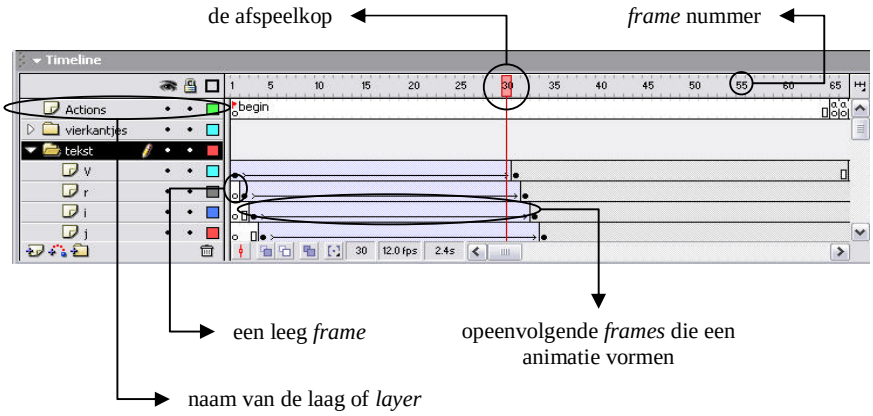
### 4.2.2 Hoe werkt Flash?

Flash bestanden worden meestal gemaakt met de Flash ontwikkelingsomgeving en daar gecompileerd tot SWF bestanden of *Flash* "filmpjes".

Daarna kunnen ze gepubliceerd (*embedded*) worden in een HTML-pagina. Het SWF formaat is een open standaard[1] en er zijn nog andere fabrikanten[2] die het formaat gebruiken.

De Flash plug-in (de *player*) is nodig om de bestanden in een browser te tonen, maar ze kunnen ook tot afzonderlijke uitvoerbare bestanden worden gecompileerd of rechtstreeks met de Flash projector bekeken worden.

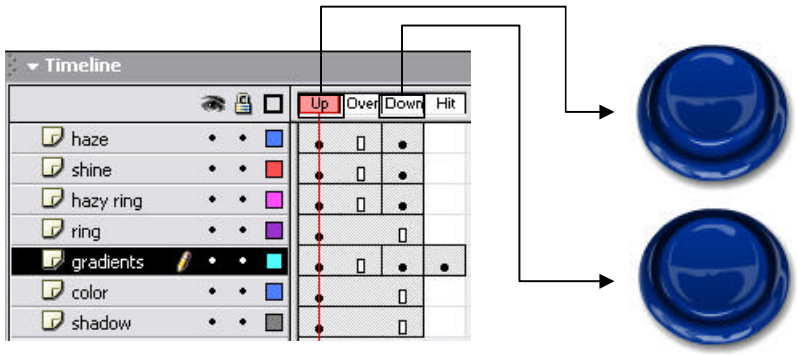
Flash is destijds ontstaan als een pakket voor webanimatie en de ontwikkelingsomgeving is dan ook gebaseerd op een filmstudio-metafoor. De ontwikkelaar moet het idee krijgen dat hij of zij een film of animatie maakt zoals een echte regisseur. De laatste versie van de ontwikkelingsomgeving (Flash MX) is echter in grote mate aan te passen, zodat ze niet hoeft afgestemd te zijn op grafici.




**Figuur 15: Fragment van een Flash MX tijdslijn**

[1] SWF specificatie: zie [SWF]
[2] De pakketten "LiveMotion" van Adobe en "Swift3D" van Electric Rain gebruiken het SWF formaat.

De belangrijkste onderdelen van Flash zijn de tijdslijn, met *frames* en een afspeelkop (zie **Figuur 15**), en de symbolen die in die *frames*




**Figuur 16: Flash Button tijdslijn**

kunnen voorkomen. Symbolen kunnen ofwel MovieClips, Buttons of Graphics zijn. Graphics zijn in feite gewone vectortekeningen, MovieClips zijn filmpjes met een afzonderlijke tijdslijn en Buttons zijn knoppen die reageren op de muisaanwijzer; deze laatste hebben ook een speciale tijdslijn die de verschillende toestanden van een knop bepalen. **Figuur 16** toont de tijdslijn van een Button, met de *Up* en *Down* toestanden; de *Over* toestand wordt bereikt wanneer de muisaanwijzer zich boven het oppervlak, bepaald in de vierde *Hit* toestand, bevindt.

Om het geheel interactief te maken is er de scripttaal ActionScript, die in feite de lijm is die de symbolen en componenten bij elkaar houdt. Zo kunnen we er een gedrag mee aan een Button koppelen of een MovieClip op het scherm plaatsen.

ActionScript is een prototypegebaseerde (dynamisch getypeerde) objectgeoriënteerde programmeertaal en – net als JavaScript – gebaseerd op de ECMA-262 standaard (zie [ECMA] voor de specificatie ervan).

## 4.3 Shockwave

### 4.3.1 Wat is Shockwave?

Shockwave is te vergelijken met Flash, niet in het minst omdat beide pakketten van dezelfde fabrikant (Macromedia) afkomstig zijn. In het algemeen wordt Flash gezien als het kleinere broertje van Shockwave, dat echter gespecialiseerd is voor het Internet. Shockwave wordt voornamelijk gebruikt om visueel aantrekkelijke productdemos, *e-commerce* sites en (*on-line*) spelletjes te maken. Shockwave werkt, net als Flash, via een gratis browser plug-in, de Shockwave *player* genaamd. De plug-in voor Shockwave is met zijn 3190 Kb bijna 10 keer groter dan de Flash plug-in en de installatie ervan is ook iets ingewikkelder.

### 4.3.2 Hoe werkt Shockwave?

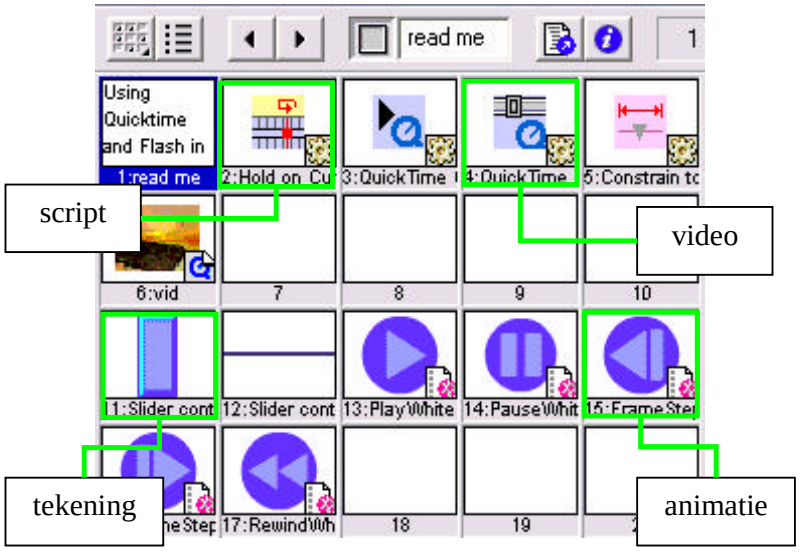
Shockwave applicaties worden ontwikkeld met Director (zie [Shockwave]), een grafische ontwikkelingsomgeving waarmee op een eenvoudige wijze multimediale projecten gecreëerd kunnen worden. De werking van Director lijkt sterk op die van de ontwikkelingsomgeving voor Flash en is eveneens gebaseerd op de filmstudiometafoor, die hier zelfs nog sterker is doorgevoerd.

Zo spreken we in Director niet van een bibliotheek of *library*, zoals dat in Flash het geval is, maar van een *cast* (zie **Figuur 17**) met *members* – de acteurs – die elk hun rol spelen op het podium of de *stage* in het Shockwave filmpje.

Elke *cast member* kan verschillende keren voorkomen op die *stage*, zoals een klasse in objectgeoriënteerde programmeertalen verschillende instanties kan hebben. In de *cast* kunnen verschillende soorten *members* voorkomen, zoals tekeningen, scripts, videomateriaal, animaties…

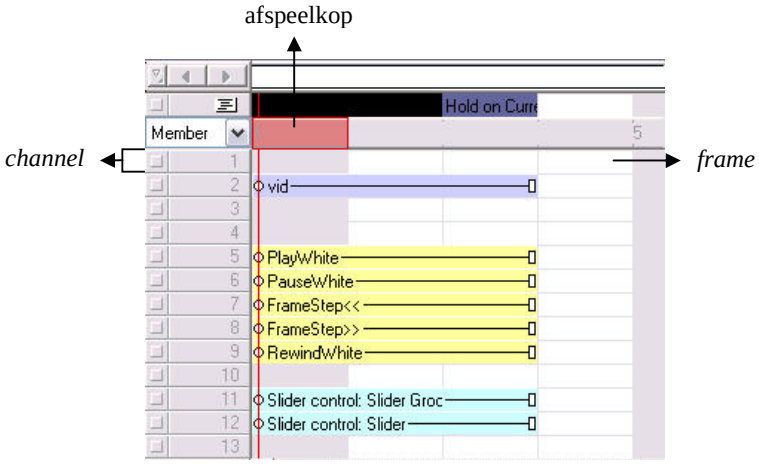
Een instantie van een *cast member* noemen we een *sprite*. De *sprites* zijn de elementen die uiteindelijk in het afgewerkte filmpje op het scherm te zien zijn.

Ook Director was oorspronkelijk een animatiepakket en heeft net als Flash een soort tijdslijn met *frames*, waarop alles geregisseerd wordt. De tijdslijn heet hier echter de *score*, conform aan de filmstudiometafoor (zie **Figuur 18**).




**Figuur 17: De *Internal Cast* met verschillende soorten *Cast Members* in Director 8.5**

Net als Flash heeft Shockwave een achterliggende scripttaal – "Lingo" genaamd – om het geheel interactief en aanpasbaar te maken. Er zijn verschillende soorten *scripts*, nl. *behaviors*, *movie scripts*, *parent scripts* en *scripts* gebonden aan *cast members*.




**Figuur 18: Fragment van de *score* in Director 8.5**

Het grootste verschil tussen Lingo en ActionScript – de scripttaal achter Flash – is dat Lingo geen echte objectgeoriënteerde taal is, hoewel bepaalde principes, zoals overerving of *inheritance*, wel nagebootst kunnen worden.

Er worden een hele boel voorbeelden van scripts *cast* members met Director meegegeven, maar ze kunnen evengoed zelf gemaakt en eventueel gedistribueerd worden als herbruikbare componenten.

## 4.4 Overzicht

Een overzicht van de drie voorgestelde webtechnologieën wordt weergegeven in **Tabel 6**.

**Tabel 6: Overzicht webtechnologieën**

| | **Curl** | **Flash** | **Shockwave** |
|---|---|---|---|
| **Plug-in** | Surge | Flash player | Shockwave player |
| Versie | 1.2.2 | 6 | 8.5.1 |
| Grootte | 12,3 Mb[1] | 383 Kb | 3190 Kb |
| **Ontwikkelings-omgeving** | Surge Lab IDE | Flash MX | Director |
| Versie | 1.7.1 | MX | 8.5.1 |
| Tekenmogelijkheid | Nee | Ja | Ja |
| **Taal** | Curl | ActionScript | Lingo |
| Objectgeoriënteerd | Ja | Ja[2] | Nee |

[1] Bij volledige installatie
[2] Gebaseerd op prototypes, zie [Craig 2000]

# Hoofdstuk 5

# Evaluatie van de webtechnologieën

In dit hoofdstuk testen we een aantal aspecten van de technologieën die we in het vorige hoofdstuk bespraken, om te zien in hoeverre er aan de criteria die we in hoofdstukken 1, 2 en 3 opstelden, wordt voldaan.

Nadat we de criteria hebben getest, zullen we één van de technologieën wat grondiger onderzoeken. We zullen dit doen door een poging te ondernemen om een hyperbolische boom, zoals beschreven in **3.2.2** op pagina 28, te maken in Flash MX.

## 5.1 Criteria

Om te beginnen zullen we per technologie de 8 criteria, zoals weergegeven in **Tabel 7**, toetsen.

We zullen ook de gegevens die we in het vorige hoofdstuk verzamelden kritisch benaderen.

**Tabel 7: Criteria voor de beoordeling van de webtechnologieën**

| | |
|---|---|
| C1 | Communicatiemogelijkheid met andere applicaties via een gebruikelijke standaard. |
| C2 | Platformonafhankelijkheid, of ondersteuning door zoveel mogelijk verschillende platformen. |
| C3 | Mogelijkheden van een *rich client* m.b.t. interactie en grafische expressie. |
| C4 | Ondersteuning voor het gebruik van bibliotheken of standaardcomponenten. |
| C5 | Ondersteuning voor het programmeren op basis van gebeurtenissen. |
| C6 | Ontwikkelingsomgeving en programmeertaal met ondersteuning voor het maken van grafische elementen (tekstopmaak, iconen…). |
| C7 | Ondersteuning vanuit de ontwikkelingsomgeving voor het zelf tekenen of importeren van de grafische elementen. |
| C8 | Ondersteuning voor het eenvoudig implementeren van animatie van grafische elementen. |

## 5.2 Curl

### 5.2.1 C1: Communicatiemogelijkheden

Een van de ontstaansredenen van Curl was het mogelijk maken van webdiensten of *web services* (zie [Curl Webservice]) om van het Internet een rijker medium te maken. Curl is dan ook voorzien van een uitgebreid arsenaal communicatiemogelijkheden.

Curl ondersteunt het versturen van data via de klassieke HTTP *post* en *get* methodes (zie [HTTP]), waarbij een lijst van variabelen met daaraan gekoppelde waarden in tekstvorm kan worden doorgestuurd.

Het werken met XML wordt goed ondersteund en Curl beschikt zelfs over een implementatie van de “SOAP” of *Simple Object Access Protocol* standaard (zie [SOAP]). Dit communicatieprotocol dat gebaseerd is op XML, moet voor een platformonafhankelijke communicatie tussen verschillende webdiensten en *clients* zorgen.

We kunnen gerust stellen dat Curl voldoet aan het criterium voor communicatie met andere applicaties en diensten, omdat de belangrijkste de facto standaard (SOAP) wordt ondersteund.

### 5.2.2 C2: Platformonafhankelijkheid

Momenteel is de Surge plug-in voor Curl, net als de ontwikkelingsomgeving Surge Lab IDE enkel beschikbaar voor het Microsoft Windows-platform. De website kondigt ook versies voor Linux en het MacOS-platform aan; er is zelfs een *preview* versie voor Linux beschikbaar op de website.

De Surge plug-in bestaat in twee versies: één voor Netscape en één voor Internet Explorer browsers.

### 5.2.3 C3: Mogelijkheden van een “rich client”?

Curl beschikt over een uitgebreid gamma grafische mogelijkheden, zowel in 2 als in 3 dimensies. Bovendien is er een API voor het construeren van een gebruikersinterface in Curl.

Een van de hoofddoelen van Curl is het brengen van een rijkere gebruikerservaring op het Internet (zie [Rosenberg 2001]) en Curl biedt dan ook de mogelijkheden van een *rich client* op het gebied van gebruikersinterfaceontwikkeling.

### 5.2.4 C4: Hergebruik van code

Net als in de meeste traditionele programmeertalen, kunnen er in Curl bibliotheken geladen worden. Er zijn op het Internet al een hele reeks bibliotheken en voorbeelden van Curl beschikbaar die zo kunnen worden gebruikt. Zo zijn er bijvoorbeeld een aantal componenten beschikbaar voor het construeren van gebruikersinterfaces.

### 5.2.5 C5: Programmeren op basis van gebeurtenissen

Curl beschikt over een uitgebreid model voor het afhandelen van gebeurtenissen, dat uitvoerig in de handleiding wordt beschreven. Curl werd namelijk ontwikkeld met het oog op het bouwen van grafische gebruikersinterfaces en daarvoor is het programmeren op basis van gebeurtenissen noodzakelijk.

### 5.2.6 C6: Ondersteuning voor grafische elementen

Het maken van grafische elementen in Curl is relatief eenvoudig en de programmeertaal beschikt over een uitgebreide API voor het maken van allerlei standaardelementen voor grafische gebruikersinterfaces en paginalay-out.

### 5.2.7 C7: Zelf tekenen van grafische elementen

Op dit ogenblik bestaat er nog geen grafische omgeving in Curl om zelf onderdelen te tekenen. De website kondigt deze uitbreiding evenwel aan voor toekomstige versies van de ontwikkelingsomgeving.

### 5.2.8 C8: Ondersteuning voor animatie

Hoewel er op de website van Curl beweerd wordt dat de technologie geschikt is voor het creëren van animaties, zijn de getoonde voorbeelden bezwaarlijk complex te noemen. Het gaat voornamelijk over het bewegen of roteren van tweedimensionale grafische elementen of de manipulatie van driedimensionale modellen.

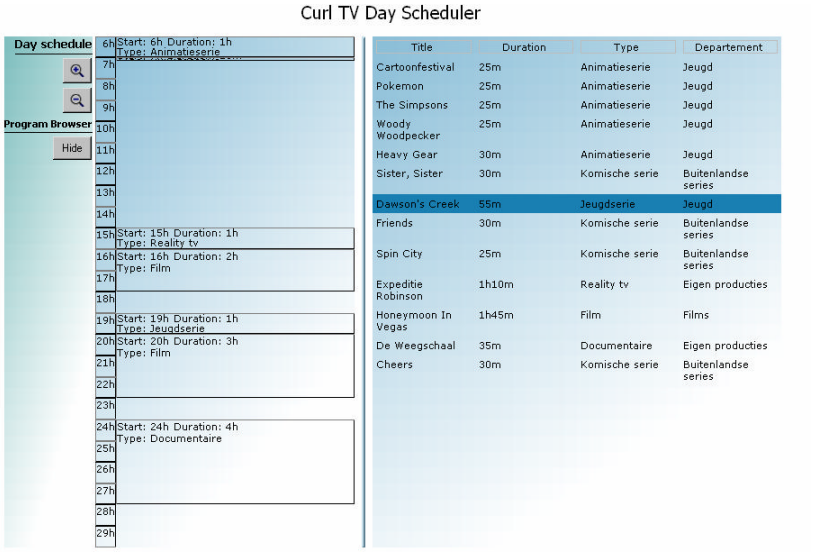
De animatiemogelijkheden in Curl zijn dan ook van een veel beperkter niveau dan die van de twee andere webtechnologieën.

### 5.2.9 Besluit

Curl staat, hoewel de ontwikkeling ervan reeds in 1995 begon, nog in de kinderschoenen en de ondersteuning voor verschillende platforms is nog vrij zwak.

Indien de beloftes van de uitvinders van Curl nageleefd worden en er versies van de plug-in en de ontwikkelingsomgeving voor Linux en MacOS komen, zou de populariteit wel eens snel kunnen toenemen. Het probleem is namelijk dat iemand die momenteel Curl applets ontwikkelt daarmee enkel gebruikers van het Windows-platform kan bereiken. Bovendien worden ontwikkelaars die op het Linux of MacOS-platform werken, verplicht over te schakelen op het Windows-platform indien ze in Curl willen programmeren.

Curl blinkt echter uit doordat het een rasechte objectgeoriënteerde programmeertaal is (in tegenstelling tot beide andere webtechnologieën die in feite geëvolueerde animatiesoftware zijn) en is daardoor ook zeer populair bij webapplicatieontwikkelaars.




**Figuur 19: Voorbeeld van een Curl applicatie voor het plannen van televisie-uitzendingen**

Indien er, zoals aangekondigd op de Curl website, een grafische ontwikkelingsomgeving voor het samenstellen van een grafische gebruikersinterface zou komen, kan Curl wel eens de belangrijkste webtechnologie worden.

**Figuur 19** toont een afbeelding van een applicatie geschreven in Curl, die de planning van de uitzendingen voor een televisiestation kan regelen (voor meer informatie: zie [Stageverslag]).

## 5.3 Flash

Het belangrijkste strategische voordeel van Flash – en de reden voor de populariteit ervan – is volgens Macromedia de omvang van de plug-in, die voor Flash 5 slechts 219 Kb en voor Flash 6 maar 383 Kb groot is. Bovendien is deze software vrij eenvoudig te installeren en is er verder geen configuratie voor nodig.

### 5.3.1 C1: Communicatiemogelijkheden

Flash ondersteunt – net als Curl – de standaard HTTP *post* en *get* methodes en beschikt over de mogelijkheid om XML documenten te lezen, aan te maken en te versturen.

Bovendien kan Flash met *sockets* werken, zodat de communicatie ook vanuit de server kan geïnitieerd worden.

Ondersteuning voor SOAP is er niet in Flash zelf, maar er bestaan wel SOAP implementaties voor Flash (zie o.a. [FlashKit]).

Flash beschikt over voldoende mogelijkheden om te communiceren met andere applicaties en diensten.

### 5.3.2 C2: Platformonafhankelijkheid

De ontwikkelingsomgeving voor Flash is enkel op het Windows- en het MacOS-platform beschikbaar. De plug-in is daarentegen in veel verschillende versies te verkrijgen (o.a. voor Windows, MacOS. Linux, Sun Solaris, OS/2, Pocket PC…).

Op de Windows- en MacOS-platformen is de plug-in al goed ingeburgerd, maar andere platformen blijven achterop, doordat de nieuwste versies er pas later voor verschijnen.

Volgens Macromedia bezitten 96% van de surfers een of andere versie van de Flash plug-in voor hun browser (zie [Flash]).

De meeste grafische componenten (zie ook **5.3.4** op pagina 49) zijn zo gemaakt dat ze op een eenvoudige manier in een ander kleedje gestopt kunnen worden, te vergelijken met de *skins* van sommige softwarepakketten.

Het implementeren van een grafische gebruikersinterface conform de stijlgids van een bepaald platform is dan ook niet moeilijk indien de ontwikkelaar over voldoende tekentalent beschikt.

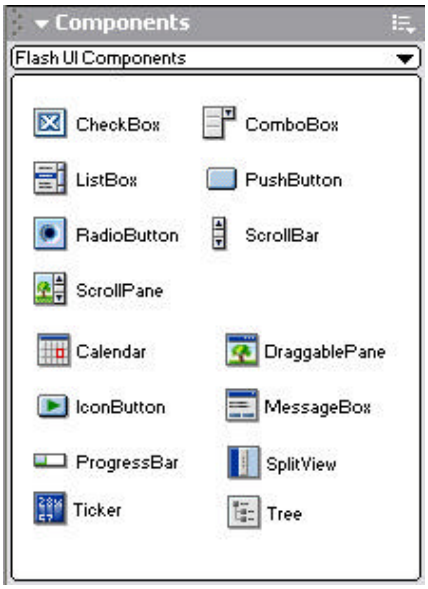
### 5.3.3 C3: Mogelijkheden van een “rich client”?

Hoewel Flash ontstaan is als een animatiepakket, heeft Macromedia de mogelijkheden van hun product fors uitgebreid om met de technologie een volwaardige *rich client* te kunnen ontwikkelen (zie [Allaire 2002]). De uitgebreide mogelijkheden van Flash 6/MX op het gebied van gebruikersinterfaceontwikkeling zijn daar het bewijs van.

### 5.3.4 C4: Hergebruik van code

Flash ondersteunt herbruikbare componenten (*Smart Clips* genaamd in Flash 5), wat de ontwikkeling van gebruikersinterfaces enorm kan versnellen. Macromedia levert standaard al een aantal basis-componenten voor gebruikersinterfaces mee, zoals een *dropdown menu*, *radio buttons*, *scrollbars*… (zie **Figuur 20**).

De componenten in Flash vormen een eenvoudige manier om niet alleen programmacode, maar ook grafische elementen of een complexe combinatie van beide opnieuw te gebruiken.




**Figuur 20: Componenten voor gebruikersinterfaces in Flash MX (bewerkt)**

Er is een enorm aanbod van Flash programma's en componenten op het Internet en Macromedia stelt ook regelmatig zelf bibliotheken van componenten beschikbaar op de website (zie [Flash]).

### 5.3.5 C5: Programmeren op basis van gebeurtenissen

Oorspronkelijk werden gebeurtenissen in Flash gekoppeld aan bepaalde grafische elementen, zoals knoppen of filmpjes en waren de mogelijkheden voor de programmeur eerder beperkt.

Sinds versie 5 van het pakket is ook de achterliggende scripttaal, ActionScript, ingrijpend veranderd en in Flash 6 heeft de ontwikkelaar meer vrijheid en kan hij of zij zelf gebeurtenissen definiëren en afhandelen.

### 5.3.6 C6: Ondersteuning voor grafische elementen

Vanwege de oorsprong van Flash als animatiepakket is de ondersteuning voor grafische elementen zeer uitgebreid.

Tekst kan in alle formaten, kleuren en posities op het scherm geplaatst worden. De meeste grafische elementen – al dan niet zelf getekend – kunnen op verschillende manieren aangepast en gepositioneerd worden, zowel vanuit de ontwikkelingsomgeving als vanuit de achterliggende taal ActionScript.

### 5.3.7 C7: Zelf tekenen van grafische elementen

De Flash ontwikkelingsomgeving beschikt over een groot palet aan grafisch tekengerei voor het maken van vectorgebaseerde tekeningen en kan bovendien een aantal elementen importeren, zoals bitmaptekeningen, filmmateriaal enz.

### 5.3.8 C8: Ondersteuning voor animatie

Uiteraard is de ondersteuning voor animatie in Flash zeer uitgebreid. De mogelijkheden vanuit de ontwikkelingsomgeving zijn te talrijk en te ingewikkeld om hier uit te leggen. Ze zijn vooral gericht op de animatiespecialist.

Ook vanuit ActionScript kunnen grafische elementen geanimeerd worden, wat zeer goed van pas komt bij het ontwikkelen van grafische gebruikersinterfaces.

### 5.3.9 Besluit

Met Flash kunnen, ondanks de oorsprong van het pakket in de animatiewereld, geavanceerde interactieve gebruikersinterfaces ontwikkeld worden.

Flash heeft soms echter een kwalijke reputatie bij de Internetgebruikers en *usability*-experts (zie [Alertbox 2000]) vanwege de vele website-intro's die ermee gemaakt werden en worden.


**Figuur 21: "Rich Text Editor" webapplicatie**

De recentste versie van Flash biedt echter de mogelijkheid om complexe interfaces en *rich clients* voor het web te maken, die het merendeel van de Internetgebruikers kunnen bekijken. Hoe de technologie in werkelijkheid wordt aangewend, valt natuurlijk moeilijk te voorspellen en is geen criterium voor de beoordeling ervan.

**Figuur 21** beeldt een eenvoudige tekstverwerker af, ontwikkeld in Flash MX en volledig functioneel (zie ook [Schoneveld RTE]), waarmee de mogelijkheden van Flash voor het creëren van een klassieke interface aangetoond worden.

De Flash documenten zijn relatief klein te houden, doordat ze gebruik maken van vectorgebaseerde tekeningen en gecomprimeerd worden bij publicatie. Daardoor worden downloadtijden ingekort, wat de gebruiksvriendelijkheid ten goede komt.

De Flash ontwikkelingsomgeving is eenvoudig in het gebruik: je kunt allerlei zaken zelf tekenen met geavanceerde tekengereedschappen en ze op een tijdlijn in zogenaamde *frames* plaatsen. Deze manier van werken vraagt een aanpassing ten opzichte van de traditionele, op tekst gebaseerde manier van programmeren, maar kan het ontwikkelingsproces van een grafische interface versnellen.

## 5.4 Shockwave

Shockwave is een zeer veelzijdig en uitgebreid softwarepakket dat veel gelijkenissen vertoont met het kleinere broertje: Flash. De doelgroep voor beide producten verschilt echter een beetje. Shockwave is meer op de professionele ontwikkelaar van rijke multimediale inhoud gericht, terwijl Flash eenvoudiger in het gebruik is en meer voor het web gemaakt werd.

De Shockwave plug-in kan eventueel uitgebreid worden door zogenaamde *Xtras*. Deze uitbreidingen kunnen ontwikkeld worden voor verschillende Macromedia pakketten en er is momenteel een rijk aanbod aan zowel commerciële als niet-commerciële Xtras op het Internet (zie [Shockwave]). Die *Xtras* hebben echter ook nadelen, doordat de gebruiker verplicht wordt om ze te downloaden en te installeren.

### 5.4.1 C1: Communicatiemogelijkheden

Net als de twee voorgaande webtechnologieën ondersteunt Shockwave de HTTP *post* en *get* methodes voor het versturen van gegevens over het Internet.

Het gebruik van XML in Shockwave is ondermeer mogelijk via een *Xtra*, maar er bestaan ook implementaties van XML-*parsers* in pure Lingo, waardoor *clients* niet verplicht worden de *Xtra* te downloaden.

### 5.4.2 C2: Platformonafhankelijkheid

Zowel de Shockwave plug-in als de ontwikkelingsomgeving Director zijn enkel voor het Windows- en het MacOS-platform beschikbaar. Niettegenstaande deze beperking is de verspreiding van de Shockwave plug-in vrij omvangrijk

Director en Shockwave dateren nog van voor het WWW-tijdperk en Shockwave geldt tot op de dag van vandaag zowat als de standaard wat betreft multimediaproducties.

### 5.4.3 C3: Mogelijkheden van een “rich client”?

Met Shockwave kunnen zeer uitgebreide *client* applicaties geconstrueerd worden. De technologie beschikt over uitgebreide grafische mogelijkheden om een interactieve, rijke gebruikersinterface te maken.

### 5.4.4 C4: Hergebruik van code

Het is in Director mogelijk om code te gebruiken die door derden werd geschreven. Ook via de *Xtras* kunnen componenten toegevoegd worden aan Shockwave.

### 5.4.5 C5: Programmeren op basis van gebeurtenissen

Shockwave bezit een gebeurtenissenmodel dat sterk gelijkt op dat van Flash. Er kunnen ook zelf zogenaamde gedragingen of *behaviours* ontwikkeld worden met Shockwave die uitgevoerd worden wanneer een bepaalde *trigger* geactiveerd wordt door een gebeurtenis.

### 5.4.6 C6: Ondersteuning voor grafische elementen

Net als bij Flash zijn de grafische mogelijkheden van Shockwave zeer talrijk. Met manipuleren van grafische componenten, zoals tekst, knoppen, filmpjes en dergelijke is zeer eenvoudig.

### 5.4.7 C7: Zelf tekenen van grafische elementen

Shockwave is ongetwijfeld het uitgebreidste van de drie webtechnologieën als het aankomt op teken- en importmogelijkheden. Het pakket kan overweg met een groot aantal formaten, dat gangbaar is in de wereld van de multimedia, zoals 3D modellen, diverse grafische bestanden, geluidsfragmenten en bewegende beelden in de vorm van video- of animatiebestanden.

Director beschikt ook over tekenmogelijkheden voor het creëren van zowel vectorgebaseerde als bitmap tekeningen en ook het tekenen vanuit de taal Lingo is mogelijk.

### 5.4.8 C8: Ondersteuning voor animatie

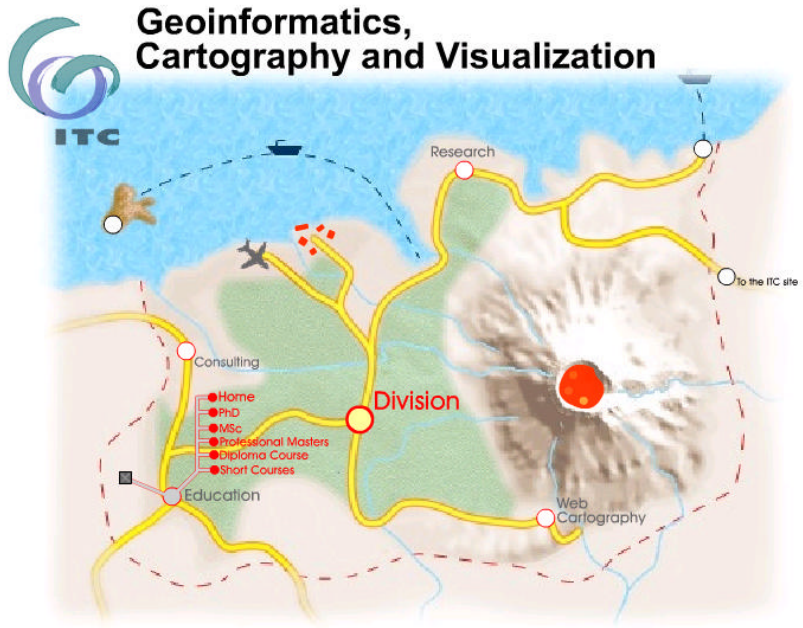
Shockwave is in de eerste plaats software voor het aanmaken van opvallende en rijke multimedia-inhoud. De animatiemogelijkheden zijn dan ook zeer uitgebreid, zowel in 2 als in 3 dimensies. Het roteren van en inzoomen op driedimensionale objecten is voor Shockwave dagelijkse kost. Er zijn ook diverse hulpmiddelen voor het animeren van tweedimensionale grafische elementen en er worden zelfs tekenfilms, videoclips en spelletjes met het pakket ontwikkeld (zie [Shockwave]).

### 5.4.9 Besluit

Met Director en Shockwave kunnen zeker *rich client* applicaties ontwikkeld worden met complexe gebruikersinterfaces die eventueel zelfs in drie dimensies kunnen worden voorgesteld. De mogelijkheden van het pakket zijn enorm uitgebreid, wat soms wat overweldigend kan zijn.

Lingo, de taal achter Shockwave, biedt veel mogelijkheden en is gemakkelijk uit te breiden via de *Xtras*. Het nadeel van deze programmeertaal ten opzichte van de talen waarover de andere webtechnologieën beschikken, is echter dat ze niet volledig objectgeoriënteerd is. Ook het feit dat de taal in geen enkel opzicht lijkt op een andere populaire programmeertaal, zoal Java of C++, kan het aanleren van de technologie bemoeilijken.

**Figuur 22** toont een interactieve visualisatie van een landkaart die ontwikkeld werd in Shockwave.




**Figuur 22: Voorbeeld van een gebruikerinterface in Shockwave die een interactieve kaart visualiseert**

Shockwave is als webtechnologie echter niet geschikt voor applicaties die maar over een beperkte bandbreedte kunnen beschikken. Hoewel de bestandsgrootte door compressie al gevoelig werd verkleind in vergelijking met oudere versies, blijven ze groot ten opzichte bijvoorbeeld de Flash bestanden. Shockwave wordt in de praktijk dan ook voornamelijk gebruikt vanwege zijn 3D capaciteiten.

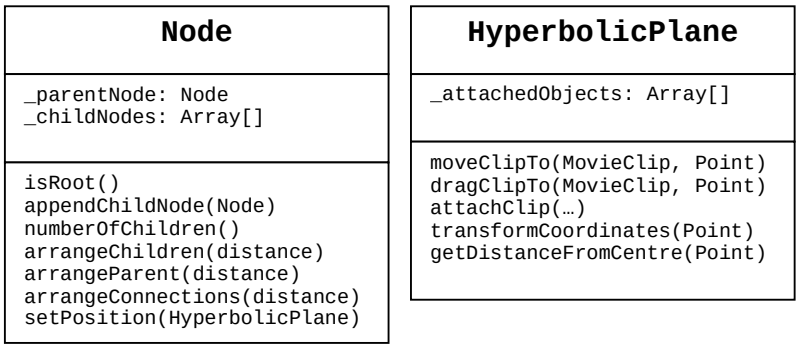
## 5.5 Ontwikkeling van een hyperbolische browser in Flash

In dit onderdeel zullen we proberen om een hyperbolische browser, die we in **3.2.2** al bespraken, te implementeren met een webtechnologie. Op deze manier willen we te weten komen of de webtechnologieën ons in staat stellen om complexe interactieve gebruikersinterfaces te maken, die niet zo alledaags zijn. Visualisaties, zoals de hyperbolische browser, komen namelijk nog niet zo vaak voor als interface en ze vereisen een zekere mate van flexibiliteit van de omgeving waarin ze ontwikkeld worden. Vooral op grafisch gebied zijn visualisaties over het algemeen erg veeleisend.

We zullen de hyperbolische browser construeren met de nieuwste versie van Macromedia Flash (Flash MX en Flash *player* 6). Deze webtechnologie benadert de ontwikkeling van gebruikersinterfaces namelijk vanuit een totaal ander standpunt dan wat we van traditionele programmeertalen en ontwikkelingsomgevingen gewoon zijn. Curl kan tot op zekere hoogte vergeleken worden met Java applets, maar dan met meer mogelijkheden, en Shockwave is enigszins te vergelijken met Flash, maar is niet speciaal voor het web bedoeld en is moeilijker om aan te leren.

### 5.5.1 Problemen

Door een patent van Inxight, een dochteronderneming Xerox, op het algoritme om de knopen van een hyperbolische boom te positioneren, zullen we ons beperken tot het verplaatsen van de knopen. In een echte hyperbolische browser moeten de hoeken tussen verschillende knopen ook aangepast worden bij het manipuleren van een knoop, wat we hier dus niet zullen doen.




**Figuur 23: Klassen voor de implementatie van de hyperbolische browser in Flash**

Verder zullen we ons toespitsen op de functionaliteit die voor de gebruikersinterface van belang is. De hyperbolische browser moet dus gemanipuleerd kunnen worden, maar verder heeft de browser geen functie.

### 5.5.2 Klassen

Net als Java, Smalltalk en C++ bijvoorbeeld is ActionScript – de programmeertaal achter Flash – een objectgeoriënteerde programmeertaal. We kunnen de ontwerptechnieken, die we voor objectgeoriënteerde systemen gebruiken dan ook overnemen.

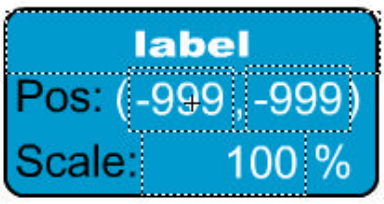
Als we de hyperbolische browser, zoals afgebeeld in **Figuur 11** op pagina 30, bekijken, dan onderscheiden we de volgende onderdelen: de knopen, en het vlak waarop de browser wordt afgebeeld[1].

We kunnen bijgevolg de indeling in klassen[2] maken, zoals weergegeven in **Figuur 23**. Enkel de belangrijkste methodes en attributen van de klassen worden hier weergegeven. De volledige broncode kan afgehaald worden van [Thesis Site].

De *Node* klasse stelt de verschillende knopen van de uiteindelijke hyperbolische boom voor en de *HyperbolicPlane* klasse stelt het vlak voor waarop de boom geprojecteerd wordt.

[1] Eventueel zouden we de verbindingslijnen tussen knopen ook als afzonderlijke entiteiten kunnen beschouwen.
[2] Technisch gezien bestaan er geen klassen in een op prototypes gebaseerde programmeertaal zoals ActionScript, maar we gebruiken de term toch om verwarring te vermijden

### *De knopen*




**Figuur 24: Representatie van een knoop of *Node***

Het belangrijkste onderdeel van een boomstructuur zijn de knopen of *nodes*, ook wel bladeren of *leafs* genoemd. Ze stellen meestal een stukje informatie voor dat zich op een bepaalde plaats in de hiërarchische structuur, die door de boom wordt voorgesteld, bevindt.

In ons geval worden de knopen voorgesteld door rechthoekige afbeeldingen (zie **Figuur 24**) en bevatten ze hun naam, informatie over hun huidige positie in het hyperbolische vlak en hun grootte.

Elke knoop is de afstammeling van een andere knoop (de *parentNode*), tenzij het de *root* of wortel is waar de rest van de hiërarchie onder staat. Een knoop kan ook afstammelingen hebben (de *childNodes*).

Qua functionaliteit beschikt elke knoop over de mogelijkheid om de afstammeling te worden van een andere knoop, om zelf andere afstammelingen te krijgen en om alle knopen waarmee er verbindingen zijn opnieuw te positioneren (de *arrangeChildren*, *arrangeParent* en *arrangeConnections* functies). Een knoop kan ook zijn positie in het vlak wijzigen (*setPosition*).

### *Het hyperbolische vlak*

Met de klasse *HyperbolicPlane* stellen we het vlak voor waarin de hyperbolische boom geprojecteerd wordt (voor meer informatie: zie [Hyperbolische Meetkunde]). Dat vlak wordt op het scherm voorgesteld door een schijf en het beschikt over een aantal functies om de objecten die erop worden voorgesteld te positioneren (*attachClip*, *moveClipTo* en *dragClipTo*[1]). Bovendien houdt het vlak een lijst[2] bij van alle objecten die erop worden weergegeven (*attachedObjects*).

[1] Het woord *clip* verwijst naar de *MovieClips* die in Flash de belangrijkste objecten zijn.
[2] In programmeertermen een *Array*.

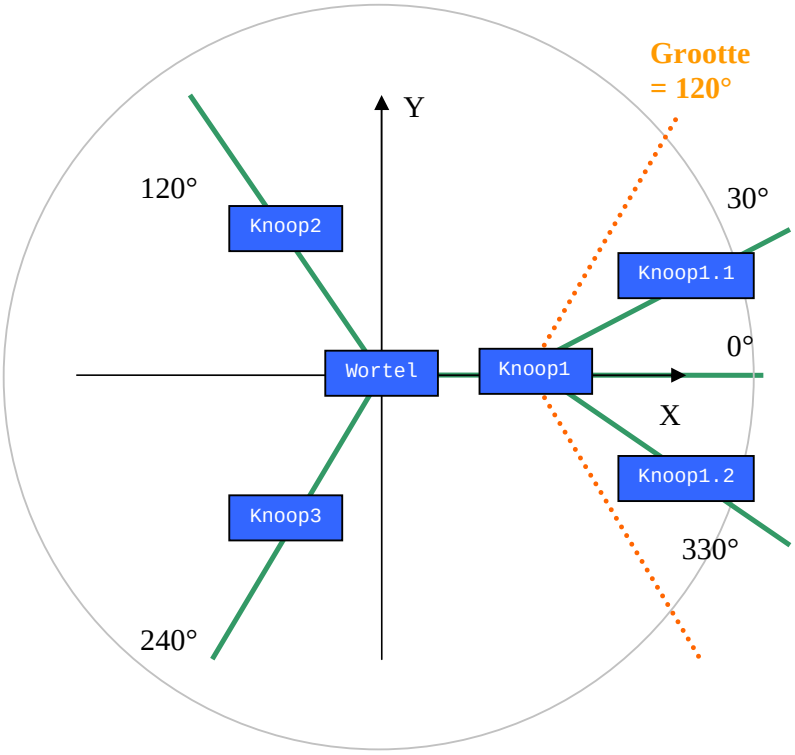
### 5.5.3 Algoritmes

Het belangrijkste algoritme voor onze hyperbolische browser bestaat erin de verschillende knopen te herpositioneren wanneer de gebruiker een van de knopen met de muis verplaatst.

Zoals reeds hoger vermeld werd, kunnen we geen gebruik maken van het algoritme van de Star Tree™ van Inxight (zie [Inxight]) en daarom zullen we de knopen enkel bij het creëren van de boom positioneren. Bij het verplaatsen van een knoop zullen we enkel de afstanden tussen de knopen en de grootte van elke knoop wijzigen.

De *transformCoordinates* methode van de *HyperbolicPlane* klasse is dus een plaatshouder voor het algoritme om de coördinaten van een knoop om te rekenen van het gewone (Euclidische) vlak naar het hyperbolische.

***Positionering van de knopen***




**Figuur 25: Positionering van de knopen van de hyperbolische browser emulatie**

De wortel wordt in de oorsprong van het hyperbolische vlak geplaatst en heeft een hoek van 360° ter beschikking om zijn afstammeling te plaatsten. Indien de wortel bijvoorbeeld 3 afstammelingen zou hebben, beschikken die op hun beurt elk over een hoek van 120° om hun eigen afstammelingen te plaatsen en zo verder.

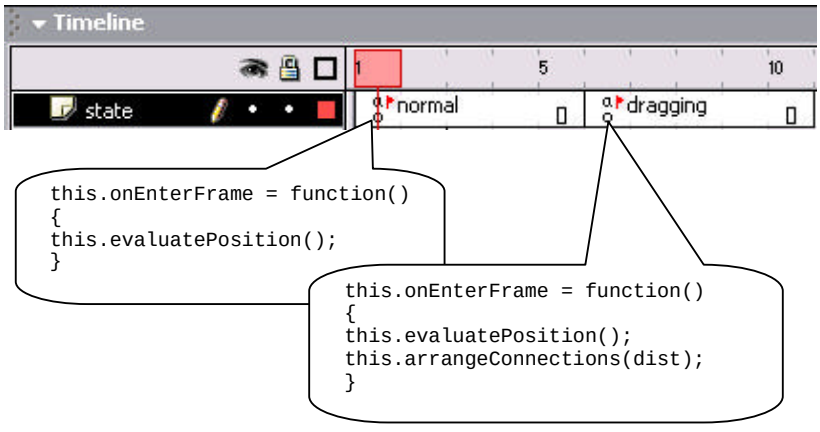
Elke knoop, behalve de wortel, heeft dus ook een bepaalde hoek ten opzichte van de knoop waarvan hij afstamt. Zo zal de eerste afstammeling van een wortel, met in totaal drie rechtstreekse afstammelingen, onder een hoek van 0° ten opzichte van de wortel geplaatst worden. De tweede afstammeling zal onder een hoek van 120° en de derde onder een hoek van 240° geplaatst worden. **Figuur 25** toont een schets van dit algoritme.

De afstand tussen een knoop en zijn afstammelingen wordt geschaald, omgekeerd evenredig met de afstand van die knoop tot de oorsprong. Op de rand van de cirkel wordt de afstand tussen een knoop en zijn afstammelingen dus oneindig klein.

Dit algoritme maakt geen optimaal gebruik van de beschikbare ruimte in het hyperbolische vlak, in tegenstelling tot het algoritme van Inxight.

### ***Verplaatsing van een knoop***

Een knoop kan zich in één van de twee volgende toestanden bevinden. Ofwel wordt hij verplaatst doordat de gebruiker de knoop met de muis heeft vastgenomen en de muis beweegt, ofwel bevindt de knoop zich in de normale toestand.




**Figuur 26: De twee toestanden van een knoop**

Een van de voordelen van Flash is dat het modelleren van toestanden zeer eenvoudig is. **Figuur 26** toont hoe de twee toestanden in feite twee[1] verschillende *frames* op de tijdslijn zijn. Wanneer de gebruiker de linker muisknop ingedrukt houdt boven een knoop of wanneer de muisknop wordt losgelaten boven een knoop, zal deze van toestand veranderen.

In Flash wordt er een *enterFrame* gebeurtenis gegenereerd telkens het beeld vernieuwd wordt en we kunnen een methode *onEnterFrame* maken die dan iedere keer wordt uitgevoerd.

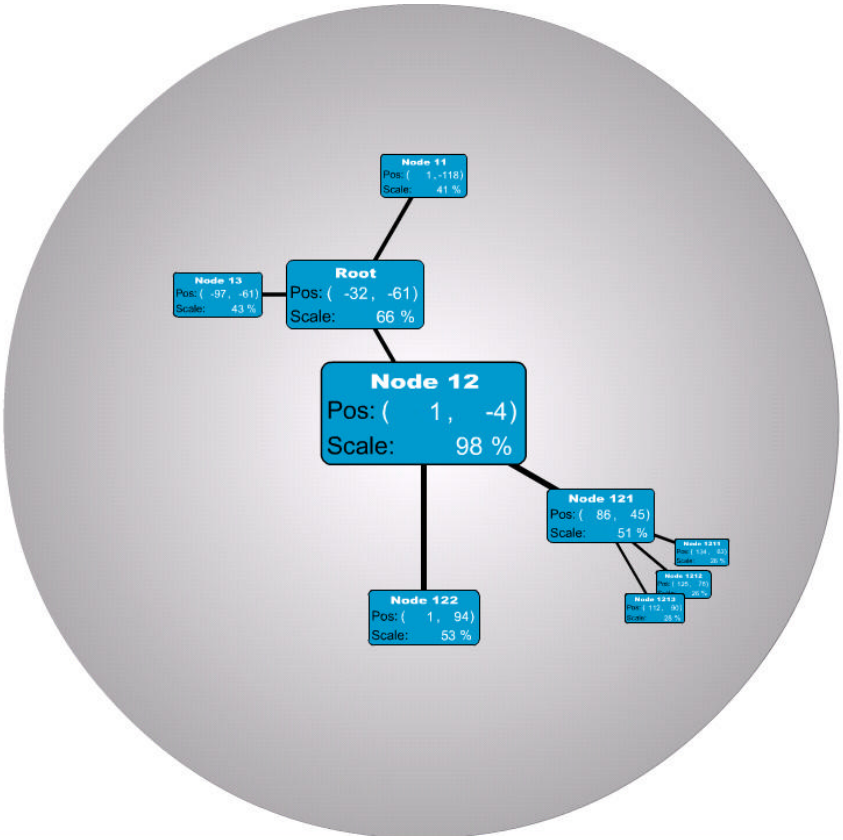
Doordat ActionScript een op prototypes gebaseerde programmeertaal[2] is, kunnen we de inhoud van de *onEnterFrame* methode vervangen, telkens een knoop van toestand verandert.

Het resultaat is dat een knoop voortdurend zijn eigen positie evalueert in de *onEnterFrame* methode en, indien hij verplaatst wordt, in de methode met dezelfde naam, maar waarvan de inhoud werd gewijzigd, de *arrangeConnections* methode oproept, zodat alle andere knopen zich herpositioneren.

### 5.5.4 Besluit

De implementatie van de hyperbolische browser in Flash is slechts gedeeltelijk geslaagd, maar dat is voornamelijk te wijten aan het laattijdig ontdekken van het Inxight patent op het algoritme voor de lay-out van knopen in het hyperbolisch vlak. We kunnen dit dan ook beter een “emulatie”van een hyperbolische browser noemen; **Figuur 27** toont het resultaat.

[1] Om de labels zichtbaar te maken werd elke toestand uitgerokken over 6 frames.
[2] Voor meer uitleg over op prototypes gebaseerde programmeertalen: zie [Craig 2000]




**Figuur 27: Emulatie van de hyperbolische browser in Flash**

De manier van werken in een ontwikkelingsomgeving als Flash MX vraagt een aanpassing van de programmeur die gewend is aan traditionele omgevingen. Flash biedt echter een heleboel mogelijkheden die het werk van de ontwikkelaar verlichten, zoals het zelf tekenen van grafische elementen, de flexibiliteit van een prototype gebaseerde programmeertaal en de mogelijkheden van de tijdslijn.

# Conclusie

De onderzochte webtechnologieën Curl, Flash en Shockwave bieden elk op hun eigen manier de mogelijkheid om complexe interactieve gebruikersinterfaces te maken.

Het ontwikkelen van *rich clients* voor het web gebeurt op dit ogenblik al met bovengenoemde technologieën en zal in de toekomst hoogstwaarschijnlijk alleen maar toenemen.

Curl heeft vooral succes omdat het speciaal voor het maken van *rich clients* werd ontwikkeld. Een programma gemaakt met Curl lijkt op grafisch gebied dan ook dikwijls op een traditionele website.

Flash is met de laatste versie[1] een volwaardig platform voor het ontwikkelen van *rich clients* geworden, maar die trend werd met Flash 5 al ingezet. Flash kan ook op een trouwe aanhang van grafische ontwikkelaars rekenen, wat het produceren van grafische elementen voor gebruikersinterfaces alleen maar ten goede kan komen. Door de mogelijkheden van het gebruik van componenten is het in Flash ook relatief eenvoudig om zonder programmeerervaring een goed functionerende gebruikersinterface te maken.

Shockwave verdiende zijn pluimen al op het gebied van *rich media* ontwikkeling en het kan gebruikt worden voor de ontwikkeling van driedimensionale gebruikersinterfaces, die in de toekomst misschien een belangrijke rol zullen spelen.

[1] Flash MX kwam op de markt op 15-03-2002

# Referenties

## Boeken


[Berson 1996] ***Client/Server Architecture***
door Alex Berson
2de editie 1996 McGraw-Hill
ISBN 0-07-005664-1

[Chappel 1996] ***Understanding ActiveX and OLE***
door David Chappell
1996 Microsoft, Redmond (Wash.)
ISBN 1-57231-216-5

[Chun 2001] ***Professioneel op weg: Flash voor Windows en Macintosh***
door Russel Chun
2001 Pearson Education
ISBN 09-430-0456-1

[Craig 2000] ***The interpretation of Object-Oriented Programming Languages***
door Ian Craig
2000 Springer-Verlag London Limited
ISBN 1-85233-159-3

[Preece et al. 1994] ***Human-Computer Interaction***
door Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. & Carey, T
1994 Addison-Wesley, Wokingham
ISBN 0-201-62769-8

[Spence 2001] ***Information Visualization***
door Robert Spence
2001 ACM Press, Pearson Education Limited
ISBN 0-201-59626-1

## Artikels


[Alertbox 1997] ***The Difference Between Web Design and GUI Design***
door Jakob Nielsen
Alertbox van 1 mei 1997
http://www.useit.com/alertbox/9705a.html

[Alertbox 2000] ***Flash 99% Bad***
door Jakob Nielsen
Alertbox van 29 oktober 2000
http://www.useit.com/alertbox/20001029.html

[Allaire 2002] ***Macromedia Flash MX – A next generation rich client***
door Jeremy Allaire
Maart 2002 Macromedia
http://www.macromedia.com/software/flash/

[Ayler et al. 2001] ***Flash Forward***
door Graeme Aymer en Stuart Dredge
”Flash: Saviour of the universe?”
uit Cre@teOnline 014 July 2001

[BaseOne 2001] ***Peer-to-Peer, Rich Client Architecture***
2001 Base One International Corporation
http://www.boic.com/

[Curl Webservice] ***Enabling Web Services with the Curl Content Language***
White Paper
2001-2002 Curl Corporation
http://www.curl.com/

[De Greef 2001] ***Generating Web Query Interfaces based on Conceptual Schemas***
Chapter 3: Visualization Maps
door Jo De Greef
2001 VUB

[Rosenberg 2001] ***Eliminating the “Submit Button” Web***
door Jothy Rosenberg Ph.D.
2001-2002 Curl Corporation
http://www.curl.com/

[Stageverslag] ***Stageverslag webtechnologieën***
David Van Damme
Gregory Vandenbroucke
Jonathan Van Eeckhoudt
2002 VUB

[UIML 2000] ***Simplifying Construction of Multi-Platform User Interfaces Using UIML***
door Mir Farooq Ali en Marc Abrams
Harmonia Inc. 2000
http://www.harmonia.com/

## Voorbeelden

[Aqua Stijlgids] ***Aqua Human Interface Guidelines***
Apple Macintosh, MacOS X
http://developer.apple.com/techpubs/macosx/Essentials/AquaHIGuidelines/

[Schoneveld RTE] ***Illogicz***
Rich Text Editor
Stuart Schoneveld
http://www.illogicz.com/

[Swing Tutorial] ***The Java Tutorial: Swing Features and Concepts***
Event Handling
Sun Microsystems, Java
http://java.sun.com/docs/books/tutorial/uiswing/overview/event.html

[Thesis Site] ***Thesis website***
Flash Tests
http://student.vub.ac.be/~hw56418/

## Websites

[CSS] ***Cascading Style Sheets***
W3C standaard
http://www.w3.org/Style/CSS/

[Curl] ***Curl***
Curl technologie, Surge plug-in en Surge Lab IDE
http://www.curl.com/

[ECMA] ***Standard ECMA-262***
Specificatie van de ECMAScript taal
http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM

[Flash] ***Macromedia Flash***
Macromedia Flash Player & Authoring Tool
http://www.macromedia.com/software/flash/
http://www.macromedia.com/software/flash/player/

[FlashKit] ***Flash Kit, A Developer Resource***
Informatieve site over Flash met FAQ en Forum
http://www.flashkit.com/

[HTTP] ***Hypertext Transfer Protocol HTTP/1.1***
RFC2616
http://www.rfc.net/rfc2616.html
http://www.w3.org/Protocols/

[Hyperbolische Meetkunde] ***Hyperbolische Meetkunde***
http://www.pandd.demon.nl/complex1/hypm5.htm

[Inxight] ***Inxight Software Inc.***
Star Tree, Table Lens
http://www.inxight.com/

[MSDN] ***Microsoft Developer Network***
MSDN Library
http://msdn.microsoft.com/library

[Quantumwave] ***Quantumwave Interavtive Inc.***
Flash ActionScript en OOP
http://www.quantumwave.com/html/flashcode.html

[Shockwave] ***Macromedia Shockwave***
Macrmedia Shockwave & Director
http://www.macromedia.com/software/shockwaveplayer/
http://www.macromedia.com/software/director/

[SOAP] ***Simple Object Access Protocol***
Voorstel voor een standaard ingediend bij het W3C
http://www.w3.org/TR/SOAP/

[SWF] ***OpenSWF.org***
Informatie over en specificatie van het SWF formaat
http://www.openswf.org/