

Hoge kwaliteit video op een lage bandbreedte verbinding

Erik Gast & Stijn de Gouw

06 Februari 2008

1. Inhoudsopgave

1. Inhoudsopgave.....	2
2. Inleiding & probleemstelling.....	3
2.1 Probleemstelling.....	3
2.2 Uitvoering.....	3
3. Introductie.....	5
3.1 Silverlight.....	5
3.2 XAML.....	6
3.3 De Klant.....	8
4. Architectuur.....	9
4.1 Bandbreedte meting.....	9
4.2 Kwaliteitskeuze.....	9
4.3 De progressbar.....	10
4.4 Controleknoppen.....	12
4.5 Fullscreen modus.....	12
4.6 Playlist.....	13
5. Infrastructuur.....	14
5.1 Bestandsstructuur.....	14
5.2 VB.NET vs. Javascript implementatie.....	14
5.3 Functionaliteit Progressbar.....	16
5.4 Functionaliteit videospeler.....	16
5.5 Functionaliteit Playlist.....	17
5.6 XAML Layout.....	18
6. Lessons Learnt.....	21
6.1 Downloader object.....	21
6.2 Bijhouden van bestandsgrootte.....	22
6.3 Caching van testbestand.....	23
6.4 Weinig documentatie.....	24
6.5 Ontbreken van standaardobjecten.....	24
6.6 Silverlight object nog niet geladen.....	24
6.7 Timers en StoryBoards.....	25
7. Conclusie.....	27
8. Referenties.....	28
9. Appendix.....	29
9.1 Videospeler screenshots.....	29
9.2 Bron code.....	31

2. Inleiding & probleemstelling

Tegenwoordig zijn er veel online videospelers te vinden op het internet. Populaire sites zoals Youtube.com [1] streamen on-demand video's van een relatief lage kwaliteit, ook als een gebruiker een snelle internetverbinding heeft. Steeds meer mensen hebben een breedband internet verbinding en dus is het mogelijk om video's van een hogere kwaliteit aan te bieden. Voor ons project bij Labwing [2] was het de bedoeling om trainingsvideo's voor hun applicaties te streamen. Deze trainingsvideo's bestaan uit screenshots en moeten in hoge kwaliteit geëncodeerd worden om de leesbaarheid te behouden. Bij een langzame internet verbinding is het echter niet de bedoeling dat er erg lang gewacht wordt totdat de video afgespeeld kan worden. Hoge kwaliteit video's zijn echter vrij groot en dus moet er een goede afweging gemaakt worden tussen de video kwaliteit en de wachttijd om zo'n video af te spelen. Zo'n afweging is mogelijk door de snelheid van de internetverbinding te meten en te relateren aan de video kwaliteit. Geen van de bestaande videospelers in Silverlight maken deze afweging echter.

2.1 Probleemstelling

Is het mogelijk om een videospeler in een webapplicatie te maken die de kwaliteit van de video's relateert aan de snelheid van de internet verbinding? Deze vraag hebben we proberen te beantwoorden door daadwerkelijk zelf zo'n videospeler te implementeren. Door het probleem op deze manier aan te pakken wordt ook duidelijk of zo'n speler in de praktijk gebruikt kan worden.

2.2 Uitvoering

Het project is in samenwerking met Labwing gemaakt. Samen is bepaald welke functionaliteit vereist was voor de videospeler. Ook is in samenspraak met Labwing bepaald op welk platform onze videospeler ontwikkeld (.NET Framework) wordt en op welk platform de videospeler moet kunnen draaien (Microsoft Windows). Verder is ook het videoformaat en de kwaliteit van de video's besproken. Het belangrijkste punt van Labwing was dat deze video's op een zeer hoge kwaliteit en vloeiend afgespeeld moest kunnen worden. Wij hebben voor Labwing het videoformaat en zijn kwaliteit bepaald en we hebben een programma gekozen die zij kunnen gebruiken om deze video's te maken. We hebben uiteindelijk gekozen voor het programma CamStudio niet alleen omdat dit programma erg intuïtief en goed werkt maar ook omdat Lawing al ervaring heeft met dit programma.

Nadat dit allemaal besproken was zijn we begonnen met het maken van de videospeler. We hebben onze vooruitgang regelmatig aan Labwing doorgegeven zodat

ze op tijd op of aanmerkingen konden geven waar wij vervolgens weer mee aan de slag konden gaan om de videospeler te verbeteren. Ook stelden we onze videospeler beschikbaar via het internet (m.b.v. een eigen server) zodat zij het op elk gewenst moment de vooruitgang konden bekijken. Uiteindelijk hebben we een aantal aanmerkingen van Labwing gekregen en die vervolgens verwerkt in de Videospeler. Tevens is een uitleg over de werking van de source code geschreven zodat de videospeler gemakkelijk geïntegreerd kan worden in de website van Labwing.

3. Introductie

In dit hoofdstuk wordt wat verteld over Microsoft Silverlight en waarom dit platform gekozen is. Verder een korte introductie over XAML, wat door Silverlight gebruikt wordt om de lay-out te definiëren. En als laatste wordt er algemene informatie gegeven over het bedrijf Labwing.

3.1 Silverlight

Silverlight [3] is een nieuw Microsoft [4] cross-browser, cross-platform implementatie van het .NET Framework om interactieve media gerichte web applicaties te maken. Er is een plugin voor nodig om de applicaties te draaien en deze plugin is beschikbaar voor Microsoft Internet Explorer, Mozilla Firefox en Apple Safari. Silverlight is het best te vergelijken met Adobe Flash [5] en het is ook bedoeld als tegenhanger van Adobe Flash.

Op dit moment zijn er twee versies van Silverlight; Silverlight 1.0 en Silverlight 1.1 Alpha. Het grootste verschil tussen de twee versies is dat Silverlight 1.0 alleen Javascript ondersteund en basis functionaliteit heeft. Silverlight 1.1 Alpha daarentegen ondersteund het .NET Framework [6] met zijn talen zoals, VB.NET en C#.

Een belangrijke eigenschap van Silverlight is dat de layout gescheiden is van de runtime code. Hierdoor ontstaat er een flexibiliteit voor zowel de programmeur als de grafische ontwerpers.

Omdat Silverlight vooral media gericht is kwam Silverlight in aanmerking als platform voor onze videoplayer. Een andere keuze was Adobe Flash, dat zichzelf al heeft bewezen als capabel videospeler (bijvoorbeeld Youtube.com). Het belangrijkste verschil tussen de twee is de taal waarin geprogrammeerd wordt. In Adobe Flash zit je vast aan ActionScript, maar in Silverlight had je de keuze uit alle .NET talen en Javascript. Voor ons en Labwing was de taalkeuze erg belangrijk, en omdat VB.NET de taal is die de voorkeur heeft bij Labwing, is de keuze (door Labwing) gemaakt om Silverlight 1.1 Alpha met VB.NET te gebruiken.

Naast de Silverlight videospeler is er ook een Javascript extensie gemaakt om meer functionaliteit en controle vanuit een HTML pagina te hebben. Met deze extensie zal een playlist functionaliteit worden toegevoegd waarbij playlists gemaakt kunnen worden met daarbij typische playlist functionaliteit zoals automatisch naar de volgende video springen en de mogelijkheid tot herhalen van de playlist. De reden dat er gekozen is voor een Javascript extensie in plaats van bijvoorbeeld een PHP of ASP.NET extensie is omdat Javascript makkelijk te integreren is in Web-pagina's en in Silverlight al gebruik wordt gemaakt van Javascript (om de Silverlight plugin te laden).

3.2 XAML

Extensible Application Markup Language (XAML) [7] is een markup taal van Microsoft en is gebaseerd op XML. XAML wordt gebruikt om een user interface (UI) voor het .NET Framework te maken. In een XAML bestand kan je UI elementen definiëren en daarmee de UI definities scheiden van de run-time logic. Doordat XAML gebaseerd is op XML en dus alleen een declaratieve functie heeft, is het erg makkelijk in gebruik en makkelijk om te debuggen. Door het gebruik van gescheiden UI en run-time logic bestanden met behulp van XAML, wordt de code een stuk overzichtelijker en makkelijker in het gebruik. Omdat XAML bestanden eigenlijk XML bestanden zijn met een .XAML extensie, bevat het dus ook alle mogelijkheden en de flexibiliteit van XML. Er kunnen dus objecten met bepaalde eigenschappen (attributen) gedefinieerd worden in een bepaalde hiërarchie. Een object kan dus een ouder en/of een kind (child) hebben en zijn. Bijvoorbeeld een tekst object met als naam textblock kan beschreven worden als

```
<Text naam="textblock"/>
```

of

```
<Text naam="textblock"></Text>
```

Meer informatie betreffende XML is te vinden op [8].

Elk Silverlight XAML bestand begint met de Canvas tags waarin de Silverlight namespace declaratie staat. Deze namespace declaratie bestaat uit een attribuut xmlns (de Silverlight namespace) en een attribuut xmlns:x die de XAML namespace declareert. Alle declaraties van objecten moeten zich bevinden tussen de root Canvas tags (tussen de eerste <Canvas> en laatste </Canvas>). De basis XAML voor Silverlight ziet er dus als volgt uit:

```
<Canvas  
  xmlns="http://schemas.microsoft.com/client/2007"  
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
>
```

Verder kunnen er nog een aantal attributen van het eerste (root) canvas object gezet worden om bijvoorbeeld te achtergrondkleur te bepalen. Door het x:Class attribuut (x:Class="LabwingVideoPlayer.Page;assembly=ClientBin/LabwingVideoPlayer.dll") toe te voegen wordt het LabwingVideoPlayer.dll gebruikt als library. De namespace LabwingVideoPlayer is nu te vinden in LabwingVideoPlayer.dll. Ook kan er aangegeven worden welke functie als eerst wordt uitgevoerd bij het laden van het XAML bestand, dit doe je door de Loaded attribuut in het root canvas object te zetten. Ook is het mogelijk om de achtergrond kleur van de Silverlight plugin te veranderen en dit doe je met het Background attribuut.

```
<Canvas  
  xmlns="http://schemas.microsoft.com/client/2007"
```

```

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Name="parentCanvas"
Loaded="Page_Loaded"
x:Class="LabwingVideoPlayer.Page;assembly=ClientBin/LabwingVideoPlayer.dll"
Background="White"
>
</Canvas>

```

Voor een volledige lijst van canvas attributen, zie [9].

Als het root canvas object aangemaakt is, dan kunnen de objecten die de UI definiëren aangemaakt worden. Deze objecten moeten tussen de Canvas tags geplaatst worden. Aan deze objecten kunnen unieke namen worden toegewezen door het attribuut `x:Name` van een object te zetten. Met deze unieke naam kan je het object vanuit de code benaderen. Je hebt dan toegang tot hun eigenschappen en hun methodes.

Een van de belangrijkste objecten in XAML is het Canvas object. Het Canvas object definieert een gebied waarin kind (child) elementen relatief aan de coördinaten van het canvas object geplaatst worden. Een voorbeeld is de weergave van een simpele tekst:

```

<Canvas
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

  <TextBlock>deze tekst wordt weergegeven</TextBlock>
</Canvas>

```

De weergegeven tekst wordt nu links bovenaan weergegeven (relatief aan het eerste (root) canvas object). Als het attribuut `Canvas.Left = "50"` aan het `<TextBlock>` object wordt toegevoegd, dan zal de tekst 50 pixels naar rechts vanaf de de canvas positie weergegeven worden. En met `Canvas.Top = "50"` zal de tekst 50 pixel naar beneden relatief aan het canvas object weergegeven worden. Voor meer informatie over het Canvas object zie [9].

Een van de voordelen van het canvas object is dat objecten gegroepeerd kunnen worden. Een gegroepeerde groep objecten kun je bijvoorbeeld makkelijk in een keer verplaatsen door alleen de positie van het canvas object te veranderen.

Tussen de canvas tags kunnen andere objecten gedefinieerd worden, zoals `Shapes`. `Shapes` [13] zijn bijvoorbeeld, rechthoeken, ellipsen, lijnen etc. Deze `Shapes` hebben eigenschappen als `Height` (hoogte), `Width` (breedte) en `Fill`. `Height` en `Width` zijn eigenschappen om de grootte te definiëren en `Fill` is om de kleur te definiëren. Deze objecten hebben nog meer eigenschappen die beschreven staan samen met de eigenschappen van andere XAML objecten in [7].

3.3 De Klant

LabWing is een Nederlandse onderneming die in 2000 werd opgericht door een groep laboratoriumprofessionals, elk met vele jaren ervaring in de verschillende vakgebieden die de analytische sector kent: de analysepraktijk op de laboratoriumvloer, softwareontwikkeling, marketing en verkoop, financieel beleid en algemeen management. Dit kernteam is in korte tijd uitgegroeid tot het middelgrote, ambitieuze softwarehuis dat zij vandaag vormen. Het is dankzij het vertrouwen van prominente spelers op de Europese analysemarkt dat wij in hoog tempo onze ambities hebben kunnen verwezenlijken.

Het centrale doel van de onderneming is het faciliteren van de dagelijkse werkzaamheden in laboratoria door middel van vernieuwende, voor het merendeel web-based softwareoplossingen, die zijn uitgedacht, ontworpen en ontwikkeld op basis van een grondige en praktische ervaring met de professionele laboratoriumpraktijk. Een van de voornaamste oogmerken van LabWing is het stroomlijnen en harmoniseren van alle datastromen in het lab tot één doeltreffend en transparant model dat laboratoria in staat stelt hun opdrachtgevers te voorzien van analyseresultaten van topkwaliteit, op een zo snel, zo economisch verantwoorde en zo accuraat mogelijke wijze. [10]

4. Architectuur

4.1 Bandbreedte meting

De bandbreedte wordt gemeten door een testbestand (met de `setBandwidthTestFile()` functie kan worden aangegeven welk bestand dat is) maximaal 1 seconde lang te downloaden. Het bestand zelf wordt met behulp van een `Downloader` [11] object (`down`, gedefinieerd in `Page.xaml.vb`) gedownload. Door de `init()` functie in `videoConf.js` wordt de locatie van het testbestand gezet in `down` en de grootte van het bestand wordt bijgehouden (zie aanroep van `plugin.Content.mediaControl.setBandwidthTestFile()`). Hierna wordt de meting gestart door de aanroep van `plugin.Content.mediaControl.startMeasureBandwidth()`.

In de `startMeasureBandwidth()` functie (in `Page.xaml.vb`) wordt een timer gestart die 1 seconde loopt. Als deze timer afloopt (`bandwidthTimer.Completed`) dan is de seconde om, wordt `stopMeasureBandwidth()` automatisch aangeroepen door de eventhandler [12] van `bandwidthTimer.Completed` en wordt het downloaden door `stopMeasureBandwidth()` gestopt. Er is nog een andere manier waarop `stopMeasureBandwidth()` kan worden aangeroepen: als het downloaden al eerder afgelopen is, bijvoorbeeld omdat het testbestand klein is of de verbinding snel, wordt door de eventhandler van `down.Completed` de `stopMeasureBandwidth()` functie aangeroepen. In `stopMeasureBandwidth()` wordt nu de timer gestopt.

De bandbreedte kan nu worden berekend door de `measureBandwidth()` functie en is gelijk aan het aantal gedownload kilobytes gedeeld door de tijd die de timer heeft gelopen. Het aantal gedownload kilobytes is `down.DownloadProgress()*filesize/1000` waarbij `filesize` de grootte van het testbestand is.

Deze manier van meten bleek een nauwkeurig resultaat te geven: de afwijking van de werkelijke verbindingssnelheid van 10 kB/s was maximaal 0.1 kB/s, een fout van minder dan 1%.

4.2 Kwaliteitskeuze

Nadat de bandbreedtemeting is gedaan (zie de vorige paragraaf voor details daarover) kan de kwaliteit van de video aan de hand de bandbreedte worden bepaald. Er is gekozen voor drie niveau's: lage kwaliteit, gemiddelde kwaliteit en hoge kwaliteit. Iedere video moet in elk van die kwaliteitsniveau's geëncodeerd zijn (één video geeft dus 3 bestanden). De designer van de webpagina met de video's bepaalt vooraf bij welke bandbreedte de video's op de pagina in lage, gemiddelde en hoge kwaliteit moeten worden afgespeeld door de `setLimits(low, med)` functie aan te roepen. Stel

de gemeten bandbreedte is b kB/s, dan wordt de video in lage kwaliteit afgespeeld als $b < low$, in gemiddelde kwaliteit als $low \leq b < med$ en in hoge kwaliteit als $b \geq med$.

Een manier om low en med te bepalen zou kunnen zijn door te kijken op welke bitrate de gemiddelde en hoge kwaliteit video's zijn geëncodeerd. Als de video van gemiddelde kwaliteit bijvoorbeeld op een bitrate van $256 \text{ kbit/s} = 32 \text{ kB/s}$ is geëncodeerd dan is 32 kB/s wellicht een goede waarde voor low . Wij hebben ervoor gekozen de webdesigner zelf handmatig de waarden voor low en med te laten bepalen omdat dit meer flexibiliteit geeft. Ook als de bandbreedte bijvoorbeeld 30 kB/s is kan de designer ervoor kiezen de video van gemiddelde kwaliteit af te laten spelen. Er moet dan weliswaar gebufferd worden maar de tijd dat er gebufferd wordt is verwaarloosbaar (tov een verbinding met bandbreedte van 32 kB/s die de video real-time kan afspelen) terwijl de kwaliteit van de video duidelijk beter is dan bij de lage kwaliteit video.

4.3 De progressbar

De progressbar van de videospeler is een van de belangrijkste componenten. Het is daarom erg belangrijk dat deze goed in elkaar zit en flexibel is. De progressbar moet bijvoorbeeld makkelijk aangepast kunnen worden, als je bijvoorbeeld wilt dat zijn uiterlijk veranderd dan moet dit snel en simpel te doen zijn. Ook moet er een grote controle zijn over zijn functies. Het streven was dus ook om een flexibele progressbar te maken die een apart component is van de speler met een aantal belangrijke functies. De progressbar moet kunnen spoelen naar een bepaalde positie met behulp van het slepen van de slider en het klikken op de progressbar. Er moet ook een indicatie zijn van het percentage van de video dat al gebufferd is.

De progressbar bestaat uit een aantal componenten; een achtergrond, een bufferbar en een slider. Deze componenten zijn allemaal gedefinieerd in een XAML bestand en zijn ook allemaal gemaakt met rectangle componenten (een `Shape` object [13]). Meer over dit XAML bestand en hoe dit bestand aangepast kan worden naar eigen smaak is te vinden in het hoofdstuk 5.

Om de progressbar te laten spoelen het behulp van de slider moet je een aantal dingen weten. Je moet bijvoorbeeld weten of de gebruiker zijn linker muisknop heeft ingedrukt, of hij die cursor ergens naartoe sleept en waar naartoe hij hem sleept. Dit moet je allemaal weten om de sleep functionaliteit van de slider te kunnen implementeren. Er moet natuurlijk ook gecheckt worden of de slider niet buiten de progressbar en het gebufferde stuk geslept wordt. Om dit voor de slider te realiseren zijn drie functies nodig. Een functie die aangeroepen wordt als er op de slider met de linker muisknop geklikt wordt, een functie die aangeroepen wordt als de linker muisknop wordt losgelaten en een functie die aangeroepen wordt als de muis verplaatst wordt. In de eerste functie (linker muisklik) moeten twee dingen gedaan worden, ten eerste moet er een variabele gezet worden waardoor je weet dat er geklikt is op de slider en dat er dus geslept kan worden. Ten tweede moet de

`CaptureMouse()` [14] functie aangeroepen worden. Deze functie zorgt ervoor dat het element dat deze functie aangeroepen heeft muis input krijgt, ook als de muiscursor niet binnen de randen van het object is. Het object blijft muis input krijgen totdat de functie `ReleaseMouseCapture()` [15] is aangeroepen en deze functie wordt dan ook aangeroepen als de gebruiker de linker muisknop loslaat. Verder moet deze functie ook de variabele zetten dat er niet meer gesleept wordt. Nu we weten wanneer er gesleept wordt en wanneer niet, kan er een functie aangeroepen worden als er gesleept wordt. In deze functie wordt de slider met behulp van de positie van de muiscursor op de X-as (horizontale as) gezet. Er wordt ook nog gecheckt of het een legale positie is, dus of het binnen de progressbar valt en binnen het gebufferde gebied. Als dat allemaal correct is wordt de positie van de slider ten opzichte van de progressbar berekend als een percentage, en de lay-out van de progressbar wordt uiteindelijk hertekent (geupdate). Omdat de slider ook verplaatst moet worden als er binnen het gebufferde gebied (bufferbar) geklikt wordt (naar de geklikte locatie), is er ook een muis event aan de bufferbar toegevoegd. Als er dus op de bufferbar met de linker muisknop geklikt wordt, dan verplaatst de slider naar deze positie.

De besproken progressbar bezit zelf niet veel functionaliteit. Het enige dat hij doet is ervoor zorgen dat de slider correct verschoven kan worden, dat zijn positie berekend wordt en dat de bufferbar juist getekend wordt. Omdat de progressbar verder zelf niet veel doet kan hij voor meerdere doelen gebruikt worden. Wij hebben hem gebruikt voor een videospeler, maar hij kan ook bijvoorbeeld makkelijk gebruikt worden als een progressbar in een muziekspeler.

Om de progressbar werkend te krijgen in onze videospeler moet de speler kunnen communiceren met de progressbar om bijvoorbeeld de positie van de slider te veranderen terwijl de video afspeelt, maar ook andersom, als de slider verplaatst wordt dan moet de huidige positie van de video ook verplaatst worden. De communicatie van de speler naar de progressbar loopt via een `Storyboard (timer)` [16] en de communicatie van de progressbar naar de speler wordt verzorgd het behulp van events. Voor de motivatie achter de keuze van `Storyboard` als timer, zie §6.7.

Zoals eerder al is genoemd, wordt de communicatie van de speler naar de progressbar toe geregeld met behulp van een `Storyboard (timer)`. Het `Storyboard` vuurt elke keer als hij afgelopen is een event. Als dit event gevuurd is, dan zorgt de speler ervoor dat de video en de slider van de progressbar weer synchroon lopen. In onze speler wordt de positie van de slider elke seconde geupdate.

De omgekeerde communicatie wordt gedaan met behulp van events die de progressbar vuurt. De progressbar vuurt een event elke keer als de positie van zijn slider is gewijzigd. De speler weet dus door middel van het afhandelen van dit event of de positie van de slider is verandert en dus of de positie van de video gewijzigd moet worden.

4.4 Controleknoppen

De videospeler is door de knoppen op het controlpanel te besturen. De knoppen hebben elk hun eigen functie zoals pauze/play, stop en fullscreen. Doordat de pause, play en stop functies al in het media-element geïmplementeerd waren, is het dus alleen noodzakelijk om het grafische gedrag van de knoppen te implementeren. De play/pauze knop heeft bijvoorbeeld twee functies en dus twee symbolen. Als de video aan het spelen is moet het pauze symbool zichtbaar zijn en als de video gepauzeerd is dan moet het play symbool zichtbaar zijn.

4.5 Fullscreen modus

Elke respectabele videospeler heeft een fullscreen modus en daarom deze natuurlijk ook. De fullscreen modus geeft de gebruiker de mogelijkheid om een video in het volledige scherm te bekijken. Deze modus is te activeren en te deactiveren door op de fullscreen knop op het controlpanel te drukken of door één keer op de afspelende video te klikken.

Als de video in fullscreen afgespeeld wordt, dan moet er voor de gebruiker ook een mogelijkheid zijn om makkelijk de video te kunnen bedienen. Een mogelijkheid om dit te doen is door het controlpanel ook zichtbaar te maken in fullscreen modus, echter dit mag paneel niet een deel van de video blokkeren. Dit probleem is opgelost door het controlpanel in 3 seconden geleidelijk transparant te maken als de gebruiker de muis niet beweegt. Hiervoor was nog een `Storyboard` nodig die over een bepaalde periode de `Opacity` (zichtbaarheid) eigenschap van het controlpanel veranderd. Door de `TargetName` en de `TargetProperty` eigenschap van het `Storyboard` te zetten, geef je het `Storyboard` toegang tot een eigenschap van een object (component). De XAML code voor dat `Storyboard` ziet er dan zo uit:

```
<Storyboard x:Name="controlTimer">
  <DoubleAnimation
    Storyboard.TargetName="videoControls"
    Storyboard.TargetProperty="(Opacity)"
    To="0" Duration="0:0:03" />
</Storyboard>
```

In het voorbeeld is de naam van het component `videoControls` en de eigenschap `Opacity` wordt door het `Storyboard` veranderd. Het `Storyboard` duurt 3 seconden en in die 3 seconden wordt de `Opacity` van `videoControls` veranderd van 1 naar 0. Omdat je het controlpanel alleen maar zichtbaar wilt hebben als je met je muis op het controlpanel staat, wordt het `Storyboard` (`controlTimer`) alleen gestart als je met je muis niet op het paneel staat, als dat wel het geval is, dan wordt de timer gestopt en staat de `Opacity` weer op 1.

Iets anders dat je niet wilt hebben terwijl je in fullscreen modus zit, is dat de muiscursor nog steeds zichtbaar is. Dit probleem is wederom aangepakt met behulp

een `Storyboard`. Het `Storyboard` zorgt ervoor dat na een aantal seconden een functie wordt aangeroepen die de muiscursor onzichtbaar maakt. Maar als de gebruiker zijn muis beweegt dan zal de muiscursor weer zichtbaar worden en de timer wordt weer geactiveerd. Door deze techniek verdwijnt de muiscursor na een aantal seconden totdat de muis weer bewogen wordt.

4.6 Playlist

De playlist is geïmplementeerd in het Javascript `videoConf.js` bestand en werkt als volgt:

1. De video's die in de playlist moeten worden gezet worden bijgehouden in het `training` array. Dit array wordt gevuld door de `setVideos()` functie (aangeropen in `demoPage.html` met de namen van de video's als argument).
2. De eerste video uit het `training` array wordt als `Source` gezet in het Silverlight media-element en begint automatisch af te spelen. De kleur van de video in het video overzicht links van de player verandert. Als `autoNext` aanstaat (aan te zetten door `enableAutoNext()` aan te roepen) wordt een timer gestart die iedere seconde controleert of de video die wordt afgespeeld ten einde is (Dit wordt gedaan in de `init()` en `timeOut()` functies)
3. Als `autoNext` aanstaat en de video is ten einde wordt de volgende video uit het `trainingsarray` afgespeeld (`setNextVideo()`). Door deze functie wordt ook gecontroleerd of de video die ten einde is ook de laatste video uit de afspeellijst was. Als dit zo is en de loop optie staat aan (te activeren door `enableLoop()` aan te roepen) wordt de eerste video afgespeeld. Is de afspeellijst ten einde en de loop optie staat niet aan dan wordt er geen volgende video afgespeeld en gebeurt er verder niets – de afspeellijst is volledig afgespeeld.

5. Infrastructuur

De bestandsstructuur wordt in beeld gebracht, de taalkeuze wordt uitgelegd, er wordt besproken hoe de Silverlight videospeler te gebruiken is en hoe hij aangepast kan worden. Ook wordt er besproken hoe de progressbar component gebruikt kan worden in andere Silverlight projecten.

5.1 Bestandsstructuur

De videoplayer bestaat uit 3 verschillende soorten bestanden; Javascript (.js), Visual Basic .NET (.vb), Extensible Application Markup Language (.xaml) en een HTML bestand (.html). De Visual Basic en de XAML bestanden vormen de Silverlight videospeler en de Javascript bestanden zijn er om de Silverlight plugin in de HTML pagina te laden en om de functionaliteit van de playlist te verzorgen.

Om de videospeler te starten moet er eerst een HTML pagina aangemaakt worden (bij ons demoPage.html). In deze pagina moet Silverlight.js en demoPage.js geïncludeerd worden. Als Silverlight.js is geïncludeerd in de HTML pagina, dan zal er eerst gekeken worden of de juiste Silverlight plugin geïnstalleerd is en of de gebruiker het juiste besturingssysteem heeft. Als dit niet het geval is dan wordt er een knop weergegeven die naar de download locatie van Silverlight wijst.

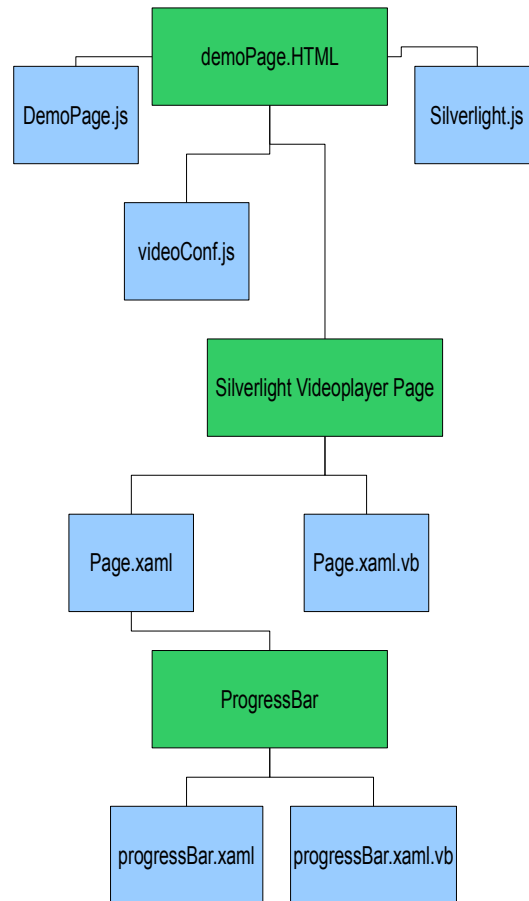
Silverlight.js is een standaard bestand en is automatisch gegenereerd door Visual Studio 2008 Beta 2 [17]. In demoPage.js staat de functie `createSilverlight()`, deze zorgt ervoor dat de Silverlight plugin en de videospeler in de browser geladen wordt. In plaats van direct de functie `createSilverlight()` aan te roepen is het beter om `videoConf.js` te includeren in de HTML pagina en de functie `initSilverlight()` aan te roepen, zodat de playlist functionaliteit gebruikt kan worden.

De relaties van de bestanden tot elkaar is te zien in figuur 1.

5.2 VB.NET vs. Javascript implementatie

De videospeler bestaand uit twee delen, het Silverlight deel dat geschreven is in VB.NET en het playlist deel wat geschreven is in Javascript. De keuze om de playlist in Javascript te schrijven is gemaakt vanwege de makkelijke integratie van Javascript in HTML pagina's. Maar in principe zouden er ook een aantal van deze functies (in plaats van ze te implementeren in Javascript) ook in Silverlight in de videospeler zelf geïmplementeerd kunnen worden en andersom. Je zou bijvoorbeeld de playlist en zijn functies allemaal kunnen implementeren in Silverlight en alleen maar met Javascript naar die functies verwijzen. De reden dat de playlist en zijn functies niet in de videospeler zelf zijn geïmplementeerd maar in een apart Javascript bestand is omdat we de Silverlight videospeler puur een videospeler willen laten zijn. De videospeler moet alleen de basis functionaliteit hebben en dus zo makkelijk mogelijk en op meer

sites dan de Labwing site te gebruiken zijn. Als je de playlist in de videoplayer zelf implementeert, dan zit je eigenlijk al vast aan het gebruiken van een playlist. Nu hoef



Figuur 1: De bestandsstructuur.

je niet perse een playlist te gebruiken om video's af te spelen met de videospeler. Ook zal de complexiteit van de code van de videospeler alleen maar onnodig toenemen als de playlist in de videospeler wordt geïmplementeerd. Het scheiden van de playlist en de videospeler heeft nog een voordeel, de functionaliteit en het gedrag van de playlist kan nu makkelijk aangepast worden zonder dat de videospeler code aangeraakt hoeft te worden waardoor de kans op fouten en een niet werkende videospeler wordt geminimaliseerd.

5.3 Functionaliteit Progressbar

Zoals eerder is genoemd, is de progressbar in de videospeler te gebruiken als spoelmechanisme en om een indicatie te geven hoeveel van de video er is gebufferd. Ook kan de progressbar naar eigen voorkeur geschaald worden en kan bijna de gehele lay-out aangepast worden door middel van het aanpassen van zijn XAML bestand. Hoe dit gedaan moet worden wordt besproken in §5.6.

Om de progressbar te schalen moeten twee eigenschappen (properties) van de progressbar gezet worden; de `Length` en de `Size` eigenschap. De `Length` eigenschap geeft de breedte (horizontaal) in pixels aan en de `Size` eigenschap geeft de hoogte (verticaal) in pixels aan.

Andere functies van de progressbar zijn het ophalen van de huidige positie van de slider en het ophalen van het percentage dat gebufferd is. Dit doe je met de functies `GetSliderPos()` en `GetBuffered()` respectievelijk. Met deze twee functies kan je alle benodigde informatie uit de progressbar halen, en door naar de variabele `sliderIsMoving` (boolean) te kijken, kan je te weten komen of de slider bewogen wordt of niet.

Om informatie over de positie en het gebufferde deel in de progressbar te “zetten” kunnen de functie `SetSlider()` en `SetBufferBar()` aangeroepen worden.

`SetSlider()` is om de positie van de slider te zetten en `SetBufferBar()` is om de bufferbar te zetten. Als een van deze twee functies aangeroepen wordt, dan wordt automatisch de lay-out van de progressbar geupdate (de positie van de slider en bufferbar). Voor meer informatie over de functies van de progressbar zie §9.2.

5.4 Functionaliteit videospeler

De implementatie van de videospeler is te vinden in het `Page.xaml.vb`. In dit bestand wordt een actie die op de layout van de speler (te vinden in `Page.xaml`) wordt uitgevoerd gekoppeld aan functies van het media-element. De `Play()` functie van het media-element wordt bijvoorbeeld uitgevoerd als er op de play-knop gedrukt wordt. Even zo voor de quality, pauze, stop en full-screen knoppen. Tevens worden de componenten in de lay-out (knoppen, progress bars etc.) van de speler juist gepositioneerd bij het wisselen tussen de gewone en fullscreen modus in de `FullscreenChanged()` functie.

Ook de bandbreedtemeting is geïmplementeerd in de Page-klasse. Door `startMeasureBandwidth()` aan te roepen wordt de meting gestart. `stopMeasureBandwidth()` wordt automatisch uitgevoerd (met behulp van een eventhandler) als de meting voltooid is. Door `measureBandwidth()` wordt daarna de bandbreedte berekend. Meer details over de bandbreedtemeting zijn te vinden in §4.1.

5.5 Functionaliteit Playlist

Om de functionaliteit van de videospeler uit te breiden en om de player flexibeler en makkelijker in gebruik te maken is er nog een playlist functionaliteit toegevoegd. Deze playlist is geschreven in Javascript en staat in videoConf.js. Deze playlist zorgt ervoor dat de Silverlight videospeler makkelijk en snel bedient kan worden vanuit bijvoorbeeld een HTML pagina. Zo kan er een lijst met video's (een playlist) opgegeven worden die de videospeler moet afspelen. Ook zijn er een aantal extra functies in het videoConf.js bestand toegevoegd voor extra controle over de videospeler. Dit zijn:

- Het automatisch spelen van de volgende video (`enableAutoNext()`).
- Het automatisch herhalen van de playlist wanneer deze is afgelopen (`enableLoop()`).
- De bandbreedte test uit zetten (`disableBandwidthMeasurement()`).
- De bandbreedte limieten specificeren (`setLimits()`).
- Het bandbreedte testbestand en zijn grootte specificeren (`setBandwidthTestFile()`).
- Automatisch HTML links genereren (de playlist) die naar de video's verwijzen (`createButtons()`).

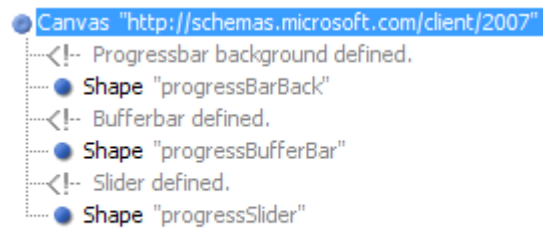
Al deze functies zijn makkelijk vanuit een HTML pagina te aan te roepen. Per functie is een Javascript functieaanroep in de HTML pagina nodig. In het voorbeeld hieronder wordt een playlist gemaakt met drie video's (video1, video2 en video3) die zich bevinden in de map videos. De bandbreedte wordt getest met het bestand 1MED.wmv en zijn bestandsgrootte is 547.036 kB. Met `setLimits()` wordt gespecificeerd wanneer een video van lage (low), medium (med) of hoge (high) kwaliteit gekozen moet worden. Verder wordt ook het herhalen van de playlist en het automatisch afspelen van de volgende video aangezet met behulp van `enableLoop()` en `enableAutoNext()`.

```
<script type="text/javascript">
    setVideos(new Array("videos/video1","videos/video2","videos/video3"));
    setBandwidthTestFile("1MED.wmv", 547.036);
    setLimits(14, 20);
    enableLoop();
    enableAutoNext();
</script>
```

Zoals is te zien, is dit een erg gemakkelijke manier om de videospeler te gebruiken. Je hoeft alleen een paar Javascript functies aan te roepen om de speler te laten werken. Er is geen direct contact met Silverlight nodig.

5.6 XAML Layout

Een belangrijke eigenschap van XAML is dat de lay-out en de runtime code gescheiden zijn. Dit geeft ons de flexibiliteit om de lay-out te veranderen zonder functionaliteit te verliezen. Door het XAML bestand te veranderen kan een compleet andere lay-out voor de progressbar en de speler gecreëerd worden. Om de lay-out te veranderen moet er wel rekening gehouden worden aan een aantal dingen; de namen van de objecten in de XAML bestanden, de events zoals mouseclicks van bepaalde objecten en de hiërarchische structuur van alle objecten moet behouden blijven. De structuur samen met de object namen van de progressbar zijn te zien in figuur 2.



Figuur 2. De progressBar.xaml structuur

De structuur in XAML:

```
<Canvas xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Background="White">

  <!-- Progressbar background defined. -->
  <Shape x:Name="progressBarBack" />

  <!-- Bufferbar defined. -->
  <Shape x:Name="progressBufferBar" MouseLeftButtonDown="bufferBarClick"/>

  <!-- Slider defined. -->
  <Shape x:Name="progressSlider" MouseLeftButtonDown="slider_LeftDown"
  MouseLeftButtonUp="slider_LeftUp" MouseMove="slider_move"/>
</Canvas>
```

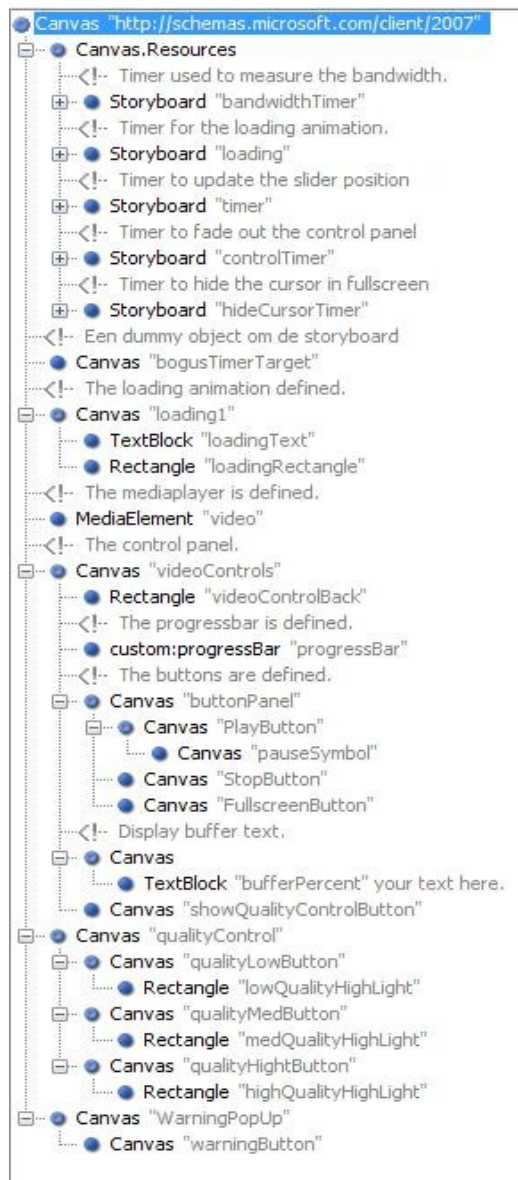
Zoals is te zien, moet de progressbar uit minimaal drie objecten bestaan; een achtergrond (progressBarBack), een bufferbar (progressBufferBar) en een slider (progressSlider). De drie objecten kunnen elke Shape (System.Windows.Shape) zijn zoals Rectangle, Canvas, Ellipse etc. Om de werking van de progressbar te garanderen moet bij het bufferbar object een MouseLeftButtonDown event aanwezig zijn en bij de progressSlider moet er een MouseLeftButtonDown, MouseLeftButtonUp en MouseMove event aanwezig zijn. Om van de slider bijvoorbeeld een ellips te maken in plaats van een rechthoek kun je progressSlider van

```
<Rectangle x:Name="progressSlider" Width="15" Height="32"
MouseLeftButtonDown="slider_LeftDown" MouseLeftButtonUp="slider_LeftUp"
MouseMove="slider_move"/>
```

veranderen in

```
<Ellips x:Name="progressSlider" Width="15" Height="32"
MouseLeftButtonDown="slider_LeftDown" MouseLeftButtonUp="slider_LeftUp"
MouseMove="slider_move"/>
```

Net zoals de lay-out van de progressbar is de lay-out van de speler ook op soortgelijke wijze te veranderen. De structuur is echter wat ingewikkelder en moet om de functionaliteit te behouden dus met zorg verandert worden. Een overzicht van zijn structuur is te zien in figuur 3.



Figuur 3: De Page.xaml structuur.

Hetzelfde geldt voor de videospeler. Als je je aan de structuur (zijn hiërarchie), object namen en events houdt, dan kun je de lay-out naar eigen wensen veranderen zonder de functionaliteit van de videospeler te verliezen.

6. Lessons Learnt

Bij het ontwikkelen van de videospeler zijn we op een aantal problemen gestuit. Zo waren er browser dependent problemen: het gedrag van Internet Explorer en Firefox verschilde bij functieaanroepen in Javascript van zelfgeschreven Silverlight-functies in Visual Basic (§6.2). Een ander browser dependent probleem was dat het testbestand gecacht werd (§6.3). Ook was er zeer weinig documentatie te vinden over het ontwikkelen van Silverlight applicaties in Visual Basic (§6.4). Verder waren er ook weinig standaardobjecten in Visual Basic voor Silverlight. Er is bijvoorbeeld geen MessageBox. Met Javascript is dit wel aanwezig: `alert()`. Meer over het ontbreken van standaardobjecten in §6.5.

Hieronder volgt een overzicht van al de leermomenten en oplossingen van de problemen die we tegen kwamen.

6.1 Downloader object

Er is een object in Silverlight speciaal bedoeld om bestanden te downloaden: het `Downloader` object. Met dit object is het bijvoorbeeld mogelijk om “content on demand” te downloaden, dat wil zeggen dat gegevens pas gedownload worden als ze echt nodig zijn. Dit kan erg handig zijn voor websites met veel grootte plaatjes. Als er op een pagina 100 plaatjes staan dan zou op de traditionele manier alle 100 plaatjes gedownload worden, ook als de bezoeker van de website maar 3 van die plaatjes wil zien. Het `Downloader` object maakt het mogelijk dat de gebruiker enkel de plaatjes hoeft te downloaden die hij wil zien. Tevens is te zien welk percentage van het bestand gedownload is. Zo kan er bijvoorbeeld een progressbar gemaakt worden gebaseerd op de voortgang (de `DownloadProgress()` [18] functie) van het downloaden van het bestand. Vooral bij grote bestanden is die optie belangrijk: de bezoeker van de webpagina hoeft niet meer te denken dat de download bijvoorbeeld is vastgelopen als het lang duurt (want er is voortgang in het downloaden zichtbaar) en kan zelf ook nog een inschatting maken hoe lang het downloaden nog gaat duren.

Ondanks deze mogelijkheden van het `Downloader` object zijn er vooral veel mogelijkheden afwezig. Het `Downloader` object heeft bijvoorbeeld perse een webserver nodig om bestanden te downloaden (dit is ook logisch want er wordt een HTTP GET-commando gestuurd om bestanden te downloaden). Omdat er verbingsproblemen kunnen optreden tussen de webserver en de bezoeker van de website is het lastiger om te debuggen voor de ontwikkelaar van de website dan als er een mogelijkheid was om ook lokaal bestanden te downloaden met de `Downloader`.

Een andere belangrijke mogelijkheid die het `Downloader` object mist is de grootte van het bestand dat gedownload wordt te zien. In theorie zou dit te berekenen zijn door een klein stukje van het bestand te downloaden (bijvoorbeeld 1 seconde lang te

downloaden), dan de lengte van de string in `ResponseText` [19] bekijken (in het `ResponseText` attribuut van het `Downloader` object staan de gedownloade gegevens in een string) en de voortgang van de download op te halen. De grootte van het totale bestand is dan `Lengte(ResponseText) / DownloadProgress()`. In de praktijk is dit echter niet mogelijk omdat `ResponseText` enkel geldig is als het bestand volledig gedownload is (zie ook [19]).

Ook is het niet mogelijk om het downloaden van bestanden te pauzeren. Dit zou juist bij grote bestanden waar het `Downloader` object voor bedoeld is van pas komen. Het zou bijvoorbeeld kunnen dat een gebruiker voor een korte periode al zijn bandbreedte nodig heeft in een ander programma. In die situatie zou een optie om het downloaden te pauzeren een uitkomst zijn. De gebruiker kan dan verder gaan met downloaden als er weer bandbreedte beschikbaar is.

Het downloaden van een bestand vanaf een bepaalde positie is ook niet mogelijk. Vooral bij het bekijken van video's zou zo'n optie handig zijn: als de gebruiker bijvoorbeeld naar het midden van de video spoelt dan zou er met deze optie ook pas vanaf die positie begonnen kunnen worden met downloaden. De gebruiker zou dan ook direct kunnen zien of een video interessant is of niet; er hoeft niet te worden gewacht tot het hele videobestand binnen is om te kijken of er op het eind van de video misschien nog iets interessants komt. Dit bespaart veel dataverkeer.

Een optie om snelheid te limiteren van het bestand dat wordt gedownload ontbreekt ook. Deze optie zou ook handig zijn voor een mediaserver: als een video geëncodeerd is met een bitrate van 256 kbit/s (= 32 kB/s) dan zou de snelheid gelimiteerd kunnen worden op 32 kB/s zonder dat er haperingen in het afspelen van de video optreden ook al kan de cliënt sneller downloaden. Dit voorkomt onnodige belasting van de mediaserver die de video's host.

6.2 Bijhouden van bestandsgrootte

Omdat voor de bandbreedtemeting een `Downloader` object gebruikt wordt en het niet zichtbaar is bij een `Downloader` object hoeveel bytes er gedownload zijn en ook niet hoe groot het bestand is dat gedownload wordt (zie ook §6.1) slaan we de bestandsgrootte van het testbestand in een variabele in Javascript op (het is niet handig om dat in Visual Basic te doen omdat dan de speler opnieuw gecompileerd moet worden als enkel de naam van het testbestand verandert).

De bandbreedte van een verbinding is namelijk het aantal gedownloade bytes van het testbestand gedeeld door de tijd dat het duurde om die bytes te downloaden, en het aantal gedownloade bytes is het percentage dat gedownload is van het bestand (dit is wel op te vragen in het `Downloader` object) vermenigvuldigt met de grootte van het bestand.

Omdat de meting van de bandbreedte door een functie in Visual Basic wordt gedaan en niet in Javascript moet de bestandsgrootte en de naam van het testbestand naar Visual Basic gecommuniceerd worden. Dit leek op het eerste gezicht goed te werken - in Firefox werd de meting correct uitgevoerd - maar bij verdere tests bleek het niet in Internet Explorer te werken. Het probleem bleek te zijn dat als de bestandsgrootte van het testbestand meer dan 32 kB (2^{15} bytes) was (in bytes opgeslagen in een integer) dat Internet Explorer het getal dat de bestandsgrootte bevatte verkeerd interpreteerde. De oplossing was de bestandsgrootte niet meer als een integer op te slaan maar als een double (type dat gebruikt wordt om reële getallen met grotere precisie op te slaan). Hierdoor kunnen bestanden tot 2 GB (2^{31} bytes) gebruikt worden voor de bandbreedtemeting.

6.3 Caching van testbestand

Sommige browsers kunnen (delen van) gedownloade bestanden tijdelijk opslaan zodat als iemand hetzelfde bestand nog een keer wil downloaden, het niet daadwerkelijk ook echt gedownload hoeft te worden: het bestand wordt gecacht.

Dit is normaal gesproken een pluspunt, maar voor de bandbreedtemeting levert het problemen op. Beschouw namelijk een testbestand van 3 MB. Als de helft van het testbestand dat gebruikt wordt voor de meting gecacht is (1500 kB) doordat de gebruiker de pagina al een keer heeft gezien en de gebruiker gaat opnieuw naar de pagina met video's toe wordt de meting opnieuw uitgevoerd, maar omdat de eerste helft van het testbestand is gecacht wordt die helft dus niet opnieuw gedownload. De gebruiker zal nu in de seconde waarin de meting wordt gedaan (het begin van) de laatste helft van het testbestand wel echt downloaden. Als er nu in die seconde 15 kB wordt gedownload dan lijkt het of er $1500 \text{ kB} + 15 \text{ kB} = 1515 \text{ kB}$ is gedownload in 1 seconde wat een snelheid van 1515 kB/s oplevert terwijl de werkelijke snelheid 15 kB/s is. Caching van het testbestand levert dus een te hoge snelheid bij de bandbreedtemeting op.

Dit probleem kan opgelost worden door het testbestand niet te cachen. Het aan- of uitzetten van caching is echter een browserinstelling en de Silverlight videospeler kan zelf geen browserinstellingen veranderen. Uitzetten van caching voor het testbestand bleek dus geen mogelijkheid te zijn (tenzij de bezoeker van de pagina met de videospeler dit zelf handmatig doet uiteraard). Een andere oplossing is om cache headers mee te sturen of de URL naar het testbestand dynamisch te genereren; dynamische content (zoals online databases) wordt namelijk niet gecacht door browsers. Hiervoor is wel ondersteuning nodig van de webserver die het testbestand host.

6.4 Weinig documentatie

Op de Silverlight MSDN [20] is het grootste deel van de Silverlight-code in Javascript geschreven en maar weinig van de voorbeelden zijn ook in Visual Basic geschreven. Dit is ook logisch gezien het feit dat voor Silverlight 1.0 de code enkel in Javascript geschreven kon worden en Visual Basic pas sinds de laatste alpha-versie (1.1) ondersteund wordt. Dit maakt het echter wel lastig om een Silverlight applicatie in Visual Basic te maken. Op [21] zijn inmiddels al wat meer voorbeelden in Visual Basic te vinden maar een goede referentie zoals de MSDN voor Javascript ontbreekt nog. Gezien het feit dat er over het coderen van Silverlight toepassingen in Javascript wel veel documentatie op de MSDN is te vinden zal dit waarschijnlijk in de nabije toekomst als er een definitieve versie van Silverlight 1.1 gereleased is opgelost zijn.

6.5 Ontbreken van standaardobjecten

Wat onmiddellijk opvalt is het ontbreken van veel ingebouwde standaardobjecten in Visual Basic voor Silverlight. Silverlight is ontworpen voor webapplicaties maar er is bijvoorbeeld geen goede timer (nodig voor bijvoorbeeld een progressbar om de positie in een video te weergeven, de voortgang van downloads te laten zien etc). De enige timer die aanwezig is, is de `HtmlTimer` en die is obsoleete en onnauwkeurig en zal verdwijnen in volgende versies van Silverlight.

Ook iets simpels als een messagebox is niet aanwezig. In Javascript is deze overigens wel aanwezig (`alert()`). Een slider, listbox en zelfs een button ontbreekt ook. Dit kan allemaal gemaakt worden uit andere componenten die wel aanwezig zijn maar dit zorgt er wel voor dat een heel simpele webapplicatie met een progressbar, slider en timer toch al zeer veel code kost alleen al vanwege het feit dat deze componenten zelf gecodeerd moeten worden. Al deze extra code leidt ook af van de code waar het werkelijk om gaat.

Wij verwachten wel dat veel componenten zullen worden toegevoegd in de release versie van Silverlight 1.1 (een nieuwe timer is al aangekondigd).

6.6 Silverlight object nog niet geladen

Omdat de juiste video in het Source attribuut van het Silverlight media-element wordt gezet door een functieaanroep (`plugin.Content.mediaControl.setVideo()` in de `setNextVideo()` functie) uit het Javascript `videoConf.js` bestand - dit wordt vanuit Javascript gedaan omdat de playlist in Javascript is geïmplementeerd, zie §5.2 - moet het media-element natuurlijk al wel geladen zijn voordat deze functieaanroep wordt uitgevoerd. Uit tests bleek dat het daadwerkelijk mogelijk was dat de functieaanroep gedaan werd voordat het media-element geladen was.

Dit probleem is opgelost door iedere seconde te controleren of het het media-element al geladen is (dit is het geval als `plugin.content.findName()`; uit de `getMediaElement()` functie in `videoConf.js` niet null is). Pas wanneer het media-element geladen is wordt verdergegaan. Hierdoor is gegarandeerd dat het media-element geladen is voordat het gebruikt wordt.

6.7 Timers en StoryBoards

Zoals er in §6.5 al is genoemd, zijn er een aantal problemen met de timers in Silverlight 1.1 als je VB.NET gebruikt. De timer die er op dit moment is, de `HtmlTimer`, zal in de volgende release van Silverlight niet meer aanwezig zijn. Deze timer is te onnauwkeurig en zal daarom verdwijnen. Helaas is de `HtmlTimer` in Silverlight 1.1 de enige beschikbare timer en dit is dus een groot probleem, zeker omdat we intensief gebruik maken van timers in onze videospeler. Er is echter wel een manier/hack om een timer te krijgen, namelijk `StoryBoards`.

Het oorspronkelijke doel van het `Storyboard` object is om objecten te animeren. Met een `Storyboard` kan je eigenschappen van een object over een bepaalde periode veranderen. Ook kan het `Storyboard` een event vuren als de periode afgelopen is, dit gebruiken we voor onze timer. Wanneer er gebruik gemaakt wordt van `StoryBoards` om timers te vervangen zijn er een aantal vervelende nadelen. Zo heeft een `Storyboard` een nauwkeurigheid van 1 seconde. Als je dus een timer wilt maken van 0.5 seconden is dit niet mogelijk en wordt je dus gedwongen om een timer van 1 seconden te gebruiken. Een ander nadeel is dat als je een `Storyboard` puur als timer wilt gebruiken er een exceptie optreedt bij het starten van het object. Wanneer zo'n exceptie voorkomt zal de timer wel starten, alleen springt hij dan wel uit de huidige functie (waar de timer wordt gestart) en zal hij alle code na de timerstart regel niet uitvoeren. Dit is een hele vervelende eigenschap en veroorzaakt veel problemen. De reden dat dit gebeurt is omdat het `Storyboard` gemaakt is om te animeren en niet om timers te vervangen. Er zijn gelukkig wel twee oplossingen voor dit probleem maar dit zijn allebei niet zulke mooie oplossingen (meer hacks). De eerste oplossing is om een `try catch` om de `Storyboard` start code te zetten. Hij zal dan wel een exceptie geven, maar hij zal dan wel doorgaan met het uitvoeren van de code in de functie en de timer zelf wordt ook gestart. Een andere oplossing is om het `Storyboard` een dummy object als target mee te geven. Je maakt dus een dummy object (bijvoorbeeld een `Canvas` object) en laat het `Storyboard` met `Storyboard.TargetName` en `Storyboard.TargetProperty` naar dat object verwijzen. Met deze methode zal het starten van het `Storyboard` geen exceptie veroorzaken, alleen heb je nu wel een “onnodig” (loos) object.

De reden dat de laatste oplossing werkt is omdat `StoryBoards` gemaakt zijn om te animeren en dus altijd een target object nodig hebben.

Wij gebruiken de dummy object methode omdat het naar ons idee beter is om een loos object aan te maken dan een exceptie te genereren. Een exceptie wordt namelijk niet voor niets veroorzaakt, het geeft aan dat er iets niet goed is ook al lijkt alles te werken.

In Silverlight 1.1 zijn de timers een probleem. Maar wanneer Silverlight 1.0 gebruikt wordt is er wel een timer beschikbaar. Je kunt dan van `setTimer()` gebruik maken zodat de `Storyboard` hack niet gebruikt hoeft te worden. Als je gebruik wilt maken van `Storyboards` moet je wel opletten hoe je ze gebruikt. Er is namelijk een verschil in het gebruik van een `Storyboard` in Silverlight 1.1 ten opzichte van Silverlight 1.0. In versie 1.1 moeten alle `Storyboards` binnen de `Canvas.Resources` tags geplaatst worden. In Silverlight 1.0 was dit niet nodig en kan dus voor verwarring zorgen.

7. Conclusie

Zoals aangetoond is het mogelijk om een videospeler in een webapplicatie te implementeren die de videokwaliteit relateert aan de snelheid van de internetverbinding. Onze speler heeft alle functionaliteit die je van een topvideospeler mag verwachten met daarbij een mogelijkheid voor het optimaal benutten van de aanwezige bandbreedte. Helaas bleek het niet altijd mogelijk de bandbreedte in Silverlight correct te meten omdat het testbestand gecacht kon worden. Een manier om dit probleem te omzeilen is de URL naar het testbestand dynamisch te genereren. Hiervoor is wel ondersteuning van de webserver en Silverlight vereist. Meer problemen zijn te vinden in het Lessons Learnt hoofdstuk.

Er is genoeg ruimte voor eventuele verbeteringen, zowel in Silverlight als in onze videospeler. Zo is er in Silverlight geen controle over het downloaden van video's en het testbestand, er kan bijvoorbeeld niet vanaf een bepaalde positie gedownload worden met het `Downloader` object. Een uitbreiding voor onze speler zou kunnen zijn dat er een mogelijkheid wordt gegeven de kwaliteitslimieten per video aan te passen.

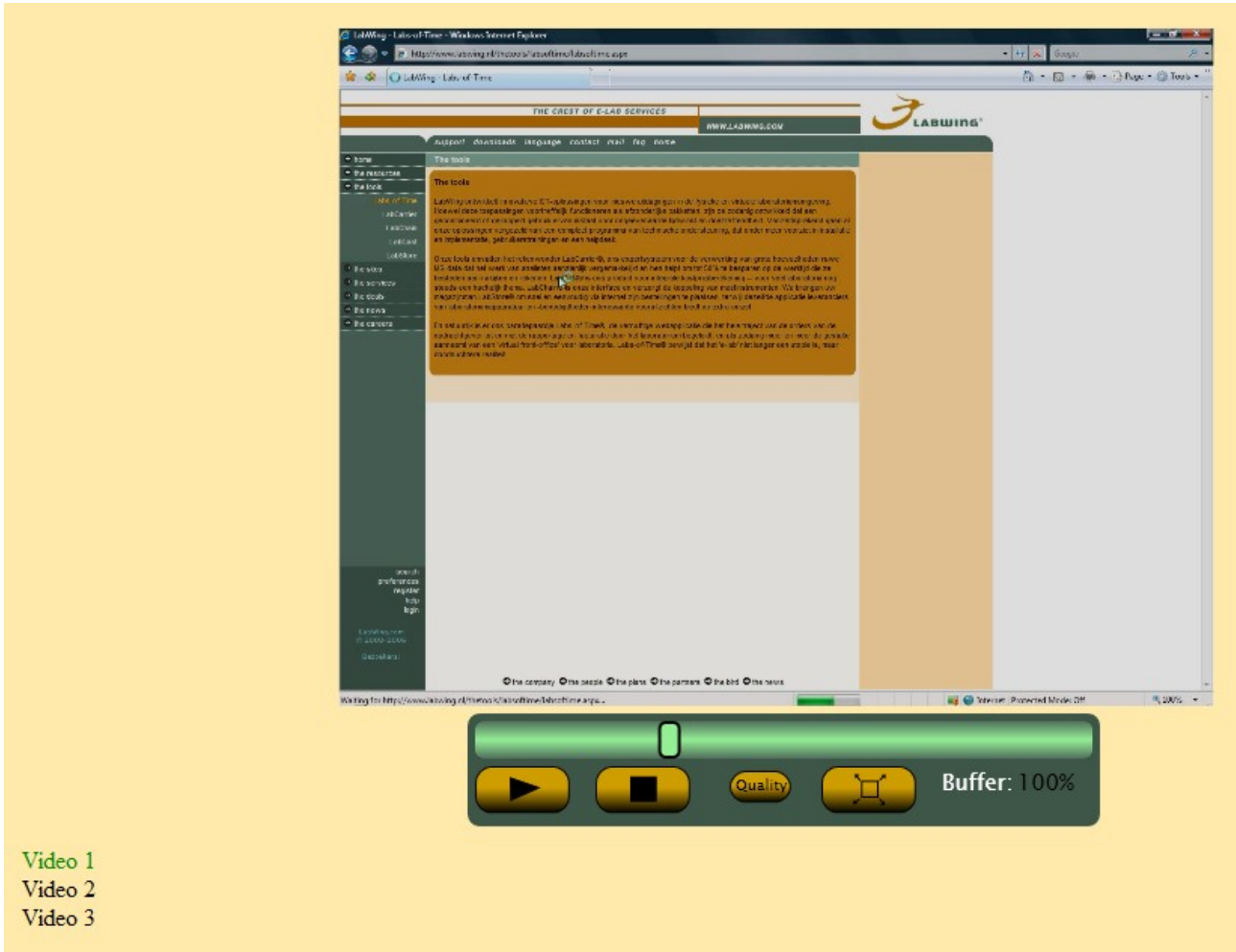
Hoewel de speler is te implementeren in Silverlight zijn er nog wel een aantal problemen waarmee rekening moet worden gehouden. Zo is er erg weinig documentatie en zijn nog minder voorbeelden te vinden. Ook is het duidelijk dat Silverlight nog in zijn beginstadium is. Zo ontbreken veel standaardobjecten en is er weinig basisfunctionaliteit aanwezig, zoals bijvoorbeeld het opvragen van de bestandsgrootte in het `Downloader` object. Positief aan Silverlight is dat de implementatie duidelijk gescheiden is van de lay-out. Ook is erg mooi dat elke .NET-taal gebruikt kan worden om een Silverlight applicatie te ontwikkelen. Echter door de tekortkomingen en omdat Silverlight versie 1.1 nog in alpha-stadium is, is het niet raadzaam Silverlight versie 1.1 te gebruiken in een productie omgeving.

8. Referenties

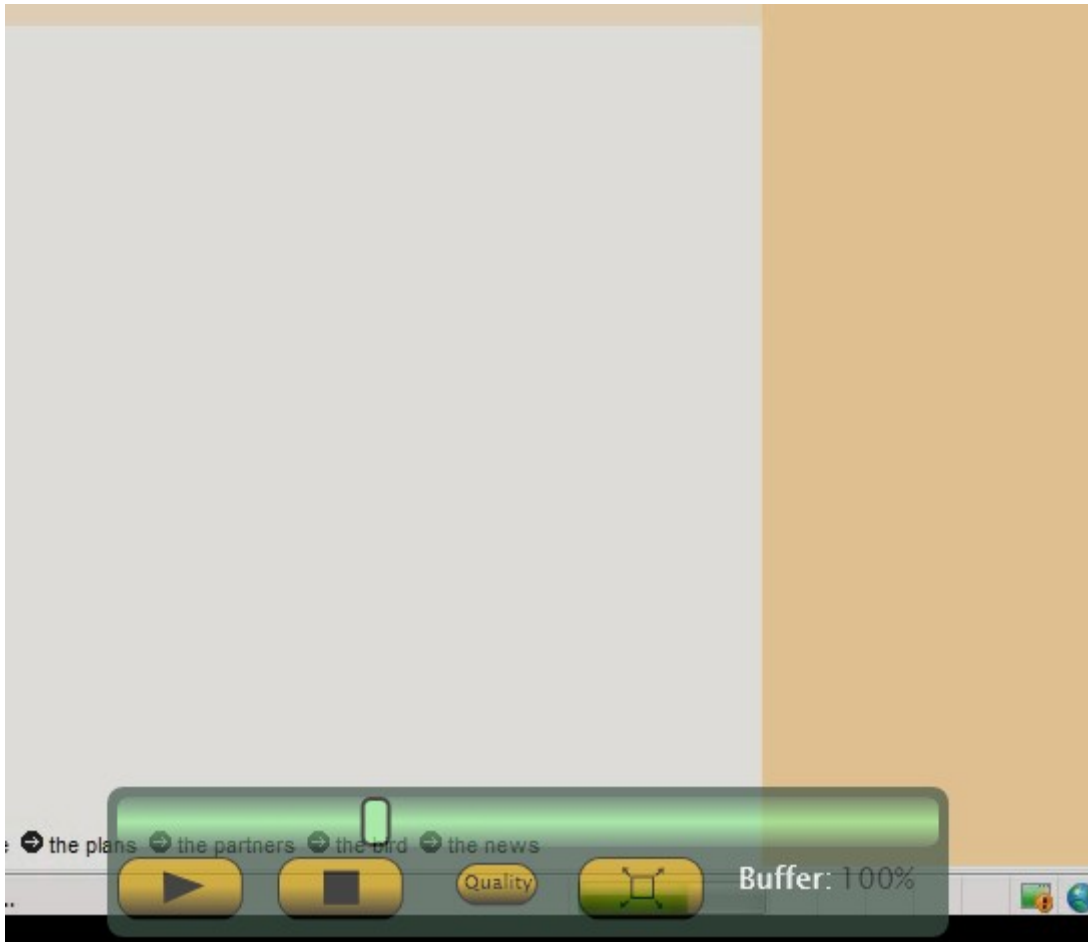
- [1] Youtube, <http://www.youtube.com>
- [2] Labwing, <http://www.labwing.com>
- [3] Silverlight, <http://www.microsoft.com/silverlight>
- [4] Microsoft, <http://www.microsoft.com>
- [5] Adobe Flash, <http://www.adobe.com>
- [6] Microsoft .NET Framework, <http://msdn.microsoft.com/netframework>
- [7] XAML, <http://msdn2.microsoft.com/en-us/library/ms752059.aspx>
- [8] XML, <http://en.wikipedia.org/wiki/XML>
- [9] Canvas, <http://msdn2.microsoft.com/en-us/library/bb979989.aspx>
- [10] Over Labwing, <http://www.labwing.com/theresources/thecompany.aspx>
- [11] Downloader, <http://msdn2.microsoft.com/en-us/library/bb979676.aspx>
- [12] AddHandler, [http://msdn2.microsoft.com/en-us/library/7taxzka\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/7taxzka(VS.80).aspx)
- [13] Shape, <http://msdn2.microsoft.com/en-us/library/system.windows.shapes.shape.aspx>
- [14] captureMouse, <http://msdn2.microsoft.com/en-us/library/bb190645.aspx>
- [15] releaseMouseCapture, <http://msdn2.microsoft.com/en-us/library/bb190645.aspx>
- [16] StoryBoard, <http://msdn2.microsoft.com/en-us/library/bb190645.aspx>
- [17] Visual Studio 2008, [http://msdn2.microsoft.com/nl-nl/vstudio/default\(en-us\).aspx](http://msdn2.microsoft.com/nl-nl/vstudio/default(en-us).aspx)
- [18] DownloadProgress, <http://msdn2.microsoft.com/en-us/library/bb979765.aspx>
- [19] ResponseText, <http://msdn2.microsoft.com/en-us/library/bb979792.aspx>
- [20] Silverlight MSDN, <http://msdn2.microsoft.com/en-us/library/bb404703.aspx>
- [21] Silverlight Documentatie, <http://silverlight.net/>

9. Appendix

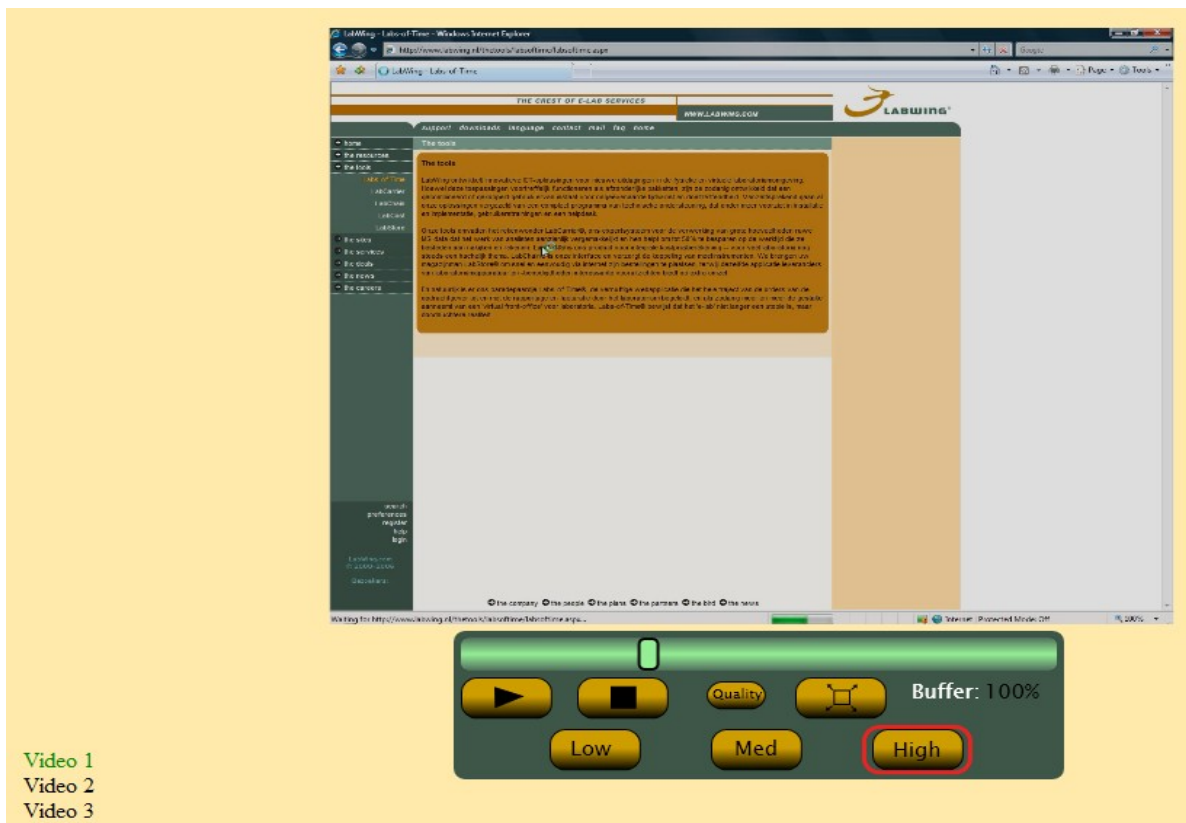
9.1 Videospeler screenshots



Figuur 4: Standaard videospeler.



Figuur 5: Videospeler in fullscreen modus met fade effect.



Figuur 6: Videospeler met kwaliteit keuze.

9.2 Bron code

Page.xaml.vb:

```
Imports System.Windows.Browser
```

```
<Scriptable()> _
```

```
Partial Public Class Page
```

```
    Inherits Canvas
```

```
    Private down As Downloader
```

```
    Private bandwidth As Integer
```

```
    Private beginTime As Date
```

```
    Private endTime As Date
```

```
    Private filesize As Integer
```

```
    Private filename As String
```

```
    Private quality As String
```

```
    Private vName As String
```

```
    Private speed As Double
```

```
    Private lowLimit, medLimit As Double
```

```
Public Sub Page_Loaded(ByVal o As Object, ByVal e As EventArgs)
```

```
    'Allow javascript to access scriptable silverlight functions.
```

```
    Try
```

```
        WebApplication.Current.RegisterScriptableObject("mediaControl", Me)
```

```
    Catch ex As Exception
```

```
        Dim err As String = ex.Message
```

```
    End Try
```

```
InitializeComponent()
```

```
'Used to measure the bandwidth.
```

```
down = New Downloader
```

```
'quality of the video not determined yet.
```

```
quality = "HIGH"
```

```
vName = ""
```

```
'Event handlers defined.
```

```
AddHandler System.Windows.Interop.BrowserHost.FullScreenChange, AddressOf Me.FullscreenChanged
```

```
AddHandler timer.Completed, AddressOf Me.TimerTick
```

```
AddHandler progressBar.sliderHasChanged, AddressOf Me.SliderChanged
```

```
AddHandler video.DownloadProgressChanged, AddressOf Me.DownloadChanged
```

```
AddHandler hideCursorTimer.Completed, AddressOf Me.hideCursor
```

```
AddHandler video.CurrentStateChanged, AddressOf Me.stateChanged
```

```
AddHandler down.Completed, AddressOf Me.stopMeasureBandwidth
```

```
AddHandler bandwidthTimer.Completed, AddressOf Me.stopMeasureBandwidth
```

```
'Starting some timers.
```

```
Try
```

```
    loading.Begin()
```

```
    timer.Begin()
```

```
Catch ex As Exception
```

```
    Dim err As String = ex.Message
```

```
End Try
```

```
End Sub
```

```
'Change interface of the player if the videoplayer state has changed.
```

```

Private Sub stateChanged(ByVal sender As Object, ByVal e As System.EventArgs)
    If video.CurrentState = "Paused" Or video.CurrentState = "Stopped" Then
        pauseSymbol.Opacity = 0
        playSymbol.Opacity = 1
    ElseIf video.CurrentState = "Playing" Then
        pauseSymbol.Opacity = 1
        playSymbol.Opacity = 0
    End If
    If video.CurrentState = "Opening" Or video.CurrentState = "Closed" Then
        loading1.Opacity = "1"
    ElseIf video.CurrentState = "Playing" Then
        loading1.Opacity = "0"
    End If
End Sub

```

'Set the video position according to the slider position (only when fast forward/backward).

```

Private Sub SliderChanged(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim newPosition As TimeSpan
    Dim sliderPos As Double

    timer.Pause()

    sliderPos = progressBar.GetSliderPos()
    newPosition = New TimeSpan(0, 0, (sliderPos / 100) * video.NaturalDuration.TimeSpan.TotalSeconds)
    video.Position = newPosition

    timer.Resume()
End Sub

```

'Set the slider position according to the video position.

```

Private Sub TimerTick(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim videoPosition As Integer
    Dim videoLength As Integer
    Dim percentPlayed As Double

    videoPosition = video.Position.TotalSeconds
    videoLength = video.NaturalDuration.TimeSpan.TotalSeconds

    percentPlayed = (videoPosition / videoLength) * 100

    progressBar.SetSlider(percentPlayed)

    timer.Begin()
End Sub

```

'Update the bufferbar.

```

Private Sub DownloadChanged(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim percentDownloaded As Double

    percentDownloaded = video.DownloadProgress() * 100
    progressBar.SetBufferBar(percentDownloaded)
    zbufferPercent.Text = CInt(percentDownloaded)
    bufferPercent.Text += "%"
End Sub

```

'Resize the control panel when going to/from fullscreen mode.

```

Private Sub FullscreenChanged(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim col As New SolidColorBrush

    If Interop.BrowserHost.IsFullScreen Then
        video.Width = System.Windows.Interop.BrowserHost.ActualWidth
        video.Height = System.Windows.Interop.BrowserHost.ActualHeight
    End If
End Sub

```

```

videoControls.SetValue(Canvas.LeftProperty, (video.Width / 2) - (progressBar.Length / 2))
If qualityControl.Opacity = 1 Then
    videoControls.SetValue(Canvas.TopProperty, video.Height - 120)
Else
    videoControls.SetValue(Canvas.TopProperty, video.Height - 80)
End If

```

```

videoControls.Opacity = 0.7

```

```

loading1.SetValue(Canvas.LeftProperty, (video.Width / 2) - (loading1.Width / 2))
loading1.SetValue(Canvas.TopProperty, (video.Height / 2) - (loading1.Height / 2))

```

```

col.Color = Colors.White
loadingText.Foreground = col

```

```

controlTimer.Begin()
hideCursorTimer.Begin()

```

Else

```

video.Width = 640
video.Height = 450

```

```

videoControls.SetValue(Canvas.LeftProperty, 115)
videoControls.SetValue(Canvas.TopProperty, 455)

```

```

controlTimer.Stop()
videoControls.Opacity = 1.0

```

```

loading1.SetValue(Canvas.LeftProperty, 250)
loading1.SetValue(Canvas.TopProperty, 220)

```

```

col.Color = Colors.Black
loadingText.Foreground = col

```

End If

End Sub

'Plays or pauses the video when clicking on the play/pause button.

```

Private Sub OnClickPlay(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)

```

```

    If video.CurrentState = "Paused" Or video.CurrentState = "Stopped" Then

```

```

        video.Play()

```

```

    ElseIf video.CurrentState = "Playing" Then

```

```

        video.Pause()

```

End If

End Sub

'Stops the video when clicking on the stop button.

```

Private Sub OnClickStop(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)

```

```

    If video.CurrentState = "Paused" Or video.CurrentState = "Playing" Then

```

```

        video.Stop()

```

End If

End Sub

```

Private Sub OnClickShowQualityControl(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs)

```

```

    If qualityControl.Opacity = 0 Then

```

```

        If Interop.BrowserHost.IsFullScreen Then

```

```

            videoControls.SetValue(Canvas.TopProperty, video.Height - 120)

```

End If

```

qualityControl.Opacity = 1

```

```

videoControlBack.Height = 112

```

```

setQualityHighLight()

```

Else

```
    If Interop.BrowserHost.IsFullScreen Then
        videoControls.SetValue(Canvas.TopProperty, video.Height - 80)
    End If
    qualityControl.Opacity = 0
    videoControlBack.Height = 75
End If
```

```
End Sub
```

```
Private Sub setQualityHighLight()
    lowQualityHighLight.Opacity = 0
    medQualityHighLight.Opacity = 0
    highQualityHighLight.Opacity = 0
    If quality = "LOW" Then
        lowQualityHighLight.Opacity = 1
    ElseIf quality = "MED" Then
        medQualityHighLight.Opacity = 1
    Else
        highQualityHighLight.Opacity = 1
    End If
End Sub
```

'Plays the video in fullscreen.

```
Private Sub OnClickFullscreen(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    If Not System.Windows.Interop.BrowserHost.IsFullScreen Then
        System.Windows.Interop.BrowserHost.IsFullScreen = True
    Else
        System.Windows.Interop.BrowserHost.IsFullScreen = False
    End If
End Sub
```

```
Private Sub OnClickWarningPopUp(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    WarningPopUp.Opacity = 0
    setVideo(vName)
End Sub
```

```
Private Sub OnClickSetToLowQuality(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs)
    quality = "LOW"
    setQualityHighLight()
    WarningPopUp.Opacity = 0
    setVideo(vName)
End Sub
```

```
Private Sub OnClickSetToMedQuality(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs)
    quality = "MED"
    setQualityHighLight()
    WarningPopUp.Opacity = 0
    setVideo(vName)
End Sub
```

```
Private Sub OnClickSetToHighQuality(ByVal sender As Object, ByVal e As
System.Windows.Input.MouseEventArgs)
    quality = "HIGH"
    setQualityHighLight()
    WarningPopUp.Opacity = 0
    setVideo(vName)
End Sub
```

```
Private Sub hideCursor(ByVal sender As Object, ByVal e As System.EventArgs)
    video.Cursor = Cursors.None
End Sub
```

'Display control panel in fullscreen when moving the mouse over the control panel.

```
Private Sub mouseOverControl(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    If System.Windows.Interop.BrowserHost.IsFullScreen Then
        controlTimer.Stop()
    End If
End Sub
```

'Fade out the control panel when moving the mouse out of the control panel.

```
Private Sub mouseLeaveControl(ByVal sender As Object, ByVal e As EventArgs)
    If System.Windows.Interop.BrowserHost.IsFullScreen Then
        controlTimer.Begin()
    End If
End Sub
```

'When in fullscreen, display the cursor when mouse moves otherwise hide the cursor.

```
Private Sub mouseMoveOverVideo(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    If Interop.BrowserHost.IsFullScreen Then
        hideCursorTimer.Stop()
        video.Cursor = Cursors.Arrow
        hideCursorTimer.Begin()
    Else
        hideCursorTimer.Stop()
        video.Cursor = Cursors.Arrow
    End If
End Sub
```

'Begin downloading the test file for measuring the bandwidth.

```
<Scriptable(> _
Public Sub startMeasureBandwidth()
    quality = "NONE"
    down.Open("GET", New Uri(filename, UriKind.Relative))
    down.Send()
    beginTime = Now()
    Try
        bandwidthTimer.Begin()
    Catch ex As Exception
        Dim err As String = ex.Message
    End Try
End Sub
```

'Calculate bandwidth.

```
Private Function measureBandwidth(ByVal fsize As Integer)
    Dim timeDiff As TimeSpan
    Dim speed As Double
    timeDiff = endTime - beginTime
    speed = fsize / (timeDiff.TotalSeconds() * 1000)
    Return speed 'In kb
End Function
```

'If bandwidthTimer expired or the down (the downloader) completed then we're ready to compute the bandwidth.

```
Private Sub stopMeasureBandwidth(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim percentDown As Double
    endTime = Now()
    percentDown = down.DownloadProgress()

    If percentDown = 1 Then 'down completed.
        bandwidthTimer.Stop()
        speed = measureBandwidth(filesize)
    Else 'bandwidthTimer expired.
        down.Abort()
        speed = measureBandwidth(filesize * percentDown)
    End If
End Sub
```

```
calcQuality()  
End Sub
```

'Set the video quality based on the measured bandwidth.

```
Private Sub calcQuality()  
    If speed < lowLimit Then  
        quality = "LOW"  
    ElseIf speed < medLimit Then  
        quality = "MED"  
    Else  
        quality = "HIGH"  
    End If
```

```
    If quality = "LOW" Then  
        WarningPopUp.Opacity = 1  
    Else  
        setVideo(vName)  
    End If  
End Sub
```

'Set the file and its filesize (in kB) used to measure the bandwidth.

```
<Scriptable()> _  
Public Sub setBandwidthTestFile(ByVal fname As String, ByVal fsize As Double)  
    filename = fname  
    filesize = fsize * 1000 'Calculate filesize in bytes  
End Sub
```

'Set the limits (kB/s) which the video quality selection is based on.

```
<Scriptable()> _  
Public Sub setLimits(ByVal low As Double, ByVal med As Double)  
    lowLimit = low  
    medLimit = med  
End Sub
```

'Get the current video quality.

```
<Scriptable()> _  
Public Function getQuality() As String  
    Return quality  
End Function
```

'Play video videoname.

```
<Scriptable()> _  
Public Sub setVideo(ByVal videoName As String)  
    vName = videoName  
    If quality <> "NONE" Then  
        videoName = videoName + quality + ".wmv"  
        Try  
            video.SetValue(MediaElement.SourceProperty, videoName)  
        Catch ex As Exception  
            Dim err As String = ex.Message  
        End Try  
    End If  
End Sub  
End Class
```

ProgressBar.xaml.vb:

Public Class progressBar

Inherits Control

Private implementationRoot As FrameworkElement

Private progressBarBack As Shape

Private progressBarBuffer As Shape

Private progressBarSlider As Shape

Private progressBarHeight As Double

Private progressBarWidth As Double

Private progressBarLeft As Double

Private progressBarTop As Double

Private buffered As Double 'Percentage gebufferd.

Private sliderPos As Double 'Percentage op bar.

Public Event sliderHasChanged(ByVal sender As Object, ByVal e As System.EventArgs)

Public sliderIsMoving = False

Public Sub New()

Dim s As System.IO.Stream =

Me.GetType().Assembly.GetManifestResourceStream("LabwingVideoPlayer.progressBar.xaml")
implementationRoot = Me.InitializeFromXaml(New System.IO.StreamReader(s).ReadToEnd())

progressBarBack = implementationRoot.FindName("progressBarBack")

progressBarBuffer = implementationRoot.FindName("progressBufferBar")

progressSlider = implementationRoot.FindName("progressSlider")

progressBarWidth = progressBarBack.Width

progressBarHeight = progressBarBack.Height

progressBarLeft = progressBarBack.GetValue(Canvas.LeftProperty)

progressBarTop = progressBarBack.GetValue(Canvas.TopProperty)

End Sub

'Set the length (width) of the progressbar.

Public Property Length() As Double

Get

Return progressBarWidth

End Get

Set(ByVal value As Double)

progressBarWidth = value

UpdateLayout()

End Set

End Property

'Set the height of the progressbar.

Public Property Size() As Double

Get

Return progressBarHeight

End Get

Set(ByVal value As Double)

progressBarHeight = value

UpdateLayout()

End Set

End Property

'Set the position (in pixels from the left) of the progressbar.

Public Property LeftPos() As Double

Get

Return progressBarLeft

End Get

```
Set(ByVal value As Double)
    progressBarLeft = value
    UpdateLayout()
End Set
End Property
```

'Set the position (in pixels from the top) of the progressbar.

```
Public Property TopPos() As Double
    Get
        Return progressBarTop
    End Get
    Set(ByVal value As Double)
        progressBarTop = value
        UpdateLayout()
    End Set
End Property
```

'Redraws the progressbar.

```
Protected Sub UpdateLayout()
    Dim sliderPosition As Double
```

```
    progressBarBack.SetValue(Canvas.HeightProperty, progressBarHeight)
    progressBarBack.SetValue(Canvas.WidthProperty, progressBarWidth)
    progressBarBuffer.SetValue(Canvas.HeightProperty, progressBarHeight)
    progressSlider.SetValue(Canvas.HeightProperty, progressBarHeight)
```

```
    progressBarBack.SetValue(Canvas.LeftProperty, progressBarLeft)
    progressBarBack.SetValue(Canvas.TopProperty, progressBarTop)
    progressBarBuffer.SetValue(Canvas.LeftProperty, progressBarLeft)
    progressBarBuffer.SetValue(Canvas.TopProperty, progressBarTop)
```

```
    progressBarBuffer.Width = (progressBarWidth / 100) * buffered
    sliderPosition = (((progressBarWidth - progressSlider.Width) / 100) * sliderPos) + progressBarLeft
    progressSlider.SetValue(Canvas.LeftProperty, sliderPosition)
    progressSlider.SetValue(Canvas.TopProperty, progressBarTop)
```

```
End Sub
```

'Updates the bufferbar.

```
Public Sub SetBufferBar(ByVal percent As Double)
```

```
    If percent > 100 Then
        percent = 100
    ElseIf percent < 0 Then
        percent = 0
    End If
```

```
    buffered = percent
```

```
    UpdateLayout()
```

```
End Sub
```

'Updates the slider position.

```
Public Sub SetSlider(ByVal percent As Double)
```

```
    If percent > 100 Then
        percent = 100
    ElseIf percent < 0 Then
        percent = 0
    End If
```

```
    If Not sliderIsMoving Then
        sliderPos = percent
    End If
```

```
    UpdateLayout()
```

End Sub

'When left clicking on the slider grab the slider for moving.

```
Private Sub slider_LeftDown(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    progressSlider.CaptureMouse()
    sliderIsMoving = True
End Sub
```

'When releasing the left mouse button and the slider is moving, then the slider is set to the current position.

```
Private Sub slider_LeftUp(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    progressSlider.ReleaseMouseCapture()
    sliderIsMoving = False
End Sub
```

'Update the slider position when moving the slider.

```
Private Sub slider_move(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    Dim mousePos As Double
    If buffered = 100 Then
        If sliderIsMoving Then
            mousePos = e.GetPosition(Me).X
            progressSlider.SetValue(Canvas.LeftProperty, mousePos)
            If mousePos > (progressBarWidth + progressBarLeft) Then
                sliderPos = buffered
            ElseIf mousePos < progressBarLeft Then
                sliderPos = 0
            Else
                sliderPos = ((mousePos - progressBarLeft) / progressBarWidth) * 100
                If sliderPos > buffered Then
                    sliderPos = buffered
                End If
            End If
        End If
        RaiseEvent sliderHasChanged(Me, System.EventArgs.Empty)
        UpdateLayout()
    End If
End Sub
```

'When clicking on the bufferbar then place the slider to the clicked position.

```
Private Sub bufferBarClick(ByVal sender As Object, ByVal e As System.Windows.Input.MouseEventArgs)
    Dim mousePos As Double

    mousePos = e.GetPosition(Me).X
    progressSlider.SetValue(Canvas.LeftProperty, mousePos)
    If mousePos > (progressBarWidth + progressBarLeft) Then
        sliderPos = buffered
    ElseIf mousePos < progressBarLeft Then
        sliderPos = 0
    Else
        sliderPos = ((mousePos - progressBarLeft) / progressBarWidth) * 100
        If sliderPos > buffered Then
            sliderPos = buffered
        End If
    End If

    RaiseEvent sliderHasChanged(Me, System.EventArgs.Empty)
    UpdateLayout()
End Sub
```

'Get the position of the slider (%).

```
Public Function GetSliderPos() As Double
    Return sliderPos
```

End Function

'Get the amount buffered (%)

Public Function GetBuffered() As Double

Return buffered

End Function

End Class

VideoConf.js:

```
// JavaScript Document
var plugin;
var mediaEl;
var naturalDuration = 0;
var currentVideo = 0;
var mediaTimer;
var training = new Array();
var loop = false;
var autoNext = false;
var lowLimit = 8, medLimit = 14; //Default values.
var filename = "bandwidthTestFile"; //Used for bandwidth measurement.
var filesize = 4797996;
var timer;
var bandwidthMeasure = true;

//Initialise the Silverlight plugin.
function initSilverlight(){
    createSilverlight();
    getMediaElement();
}

//When Silverlight and the mediaelement is loaded, initialise the playlist.
function getMediaElement(){
    plugin = document.getElementById("SilverlightControl");
    mediaEl = plugin.content.findName("video");

    if(!mediaEl){
        mediaTimer = setTimeout("getMediaElement()", 1000);
    }else{
        init();
    }
}

//Initialise the playlist and play the first video.
function init(){
    if (autoNext == true){
        timer = setTimeout("timeOut()", 1000);
    }

    if(bandwidthMeasure){
        plugin.Content.mediaControl.setBandwidthTestFile(filename, filesize);
        plugin.Content.mediaControl.setLimits(lowLimit, medLimit);
        plugin.Content.mediaControl.startMeasureBandwidth();
    }
    plugin.Content.mediaControl.setVideo(training[0]);
    changeColor(0,0); //Indicate the playing video in the playlist by changing the color.
}

//When autoNext is enabled change to next video when the current video is finished.
function timeOut(){
    if(naturalDuration == 0){ //Get the length of the playing video.
        naturalDuration = Math.round(mediaEl.GetValue("NaturalDuration").GetValue("Seconds"));
    }
    var pos = Math.round(mediaEl.GetValue("Position").GetValue("Seconds"));
    if(pos == naturalDuration && pos != 0){ //Video is finished.
        setNextVideo();
    }
    timer = setTimeout("timeOut()", 1000);
}
```

```

//Create the video playlist.
function createButtons(){
  for(i=0;i<training.length;i++){
    document.write("<div id='"+knop"+(i+1)+"' onMouseOver='\"document.body.style.cursor = 'hand';\"
onClick=setVideo(\"'+training[i]+'\">Video "+(i+1)+" </div>")
  }
}

//Change the color of the previous video (lastVideo) in the playlist to black and the current video to green.
function changeColor(lastVideo, curVideo){
  var button = document.getElementById("knop"+(lastVideo+1));
  button.style.color="black";
  button = document.getElementById("knop"+(curVideo+1));
  button.style.color="green";
}

//Fill the playlist with the video's.
function setVideos(videos){
  if(videos.constructor == Array){
    training = videos;
  }else{
    alert("videolist is not an array");
  }
}

//Set the file and its filesize used to measure the bandwidth.
function setBandwidthTestFile(fname, fsize){
  filename = fname;
  filesize = fsize;
}

//Set the limits (kB/s) which the video quality selection is based on. If not set the default file (bandwidthTestFile) is
used.
function setLimits(low, med){
  lowLimit = low;
  medLimit = med;
}

//Disable the bandwidth measurement to determine the video quality.
function disableBandwidthMeasurement(){
  bandwidthMeasure = false;
}

//Loop the playlist.
function enableLoop(){
  loop = true;
}

function enableAutoNext(){
  autoNext = true;
}

```

//Change the current video to the next video in the playlist.

```
function setNextVideo(){
  var lastVideo = currentVideo;
  currentVideo += 1;
  if (loop == false){
    if(currentVideo >= training.length){
      currentVideo -= 1;
      return currentVideo ;
    }
  }else{
    currentVideo = currentVideo%training.length;
  }

  plugin.Content.mediaControl.setVideo(training[currentVideo]);
  naturalDuration = 0;
  changeColor(lastVideo, currentVideo); //Indicate the playing video in the playlist by changing the color.

  return currentVideo;
}
```

//Play the selected video from the playlist.

```
function setVideo(videoName){
  var lastVideo = currentVideo;
  for (i=0;i < training.length;i++){
    if ((videoName) == training[i]){
      currentVideo = i;
      plugin.Content.mediaControl.setVideo(training[currentVideo]);
      naturalDuration = 0;
      changeColor(lastVideo, i); //Indicate the playing video in the playlist by changing the color.
    }
  }
}
```

demoPage.html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Labwing demo</title>
  <script type="text/javascript" src="Silverlight.js"></script>
  <script type="text/javascript" src="DemoPage.html.js"></script>
  <script type="text/javascript" src="videoConf.js"></script>
  <!-- Initialise the Silverlight plugin. -->
  <style type="text/css">
    .silverlightHost { width: 640px; height: 600px; }
  </style>
  <script type="text/javascript">
    setVideos(new Array("videos/1","videos/2","videos/3")); //Fill the playlist (order is important).
    setBandwidthTestFile("videos/1MED.wmv", z.036);
    //disableBandwidthMeasurement();
    setLimits(14, 20); //0-14 kB/s: LOW quality, 14-20 kB/s: MEDIUM quality, 20+ kB/s: HIGH quality.
    enableLoop(); //Loop the playlist.
    enableAutoNext(); //Automatically play the next video in the playlist.
  </script>
</head>
<body bgcolor="#ffe9a6">
<table width="856" border="0">
<tr>
  <td width="177" align="left" valign="bottom">
    <script type="text/javascript">
      createButtons(); //Create the playlist menu.
    </script>
  </td>
  <td width="669">
    <div id="SilverlightControlHost" class="silverlightHost" >
      <script type="text/javascript">
        initSilverlight(); //Load the videoplayer inside the Silverlight plugin.
      </script>
    </div>
  </td>
</tr>
</table>
</body>
</html>
```