

TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Informaatika eriala

Jaana Metsamaa

**MICROSOFT SILVERLIGHT
MEEDIARAKENDUSTE ARENDUSVAHENDINA**

Bakalaureusetöö (4 AP)

Juhendaja: dotsent Helle Hein

Autor: “.....“ mai 2008

Juhendaja: “.....“ mai 2008

Lubada kaitsmisele

Professor “.....“ mai 2008

TARTU 2008

Sisukord

1	Sissejuhatus	4
2	Silverlight arhitektuur ning rakenduse ülesehitus	5
2.1	Arhitektuur	5
2.2	Töövahendid	6
2.3	Silverlight rakenduse ülesehitus	8
3	Kasutajaliidese elemendid.....	14
3.1	Transformatsioonid	14
3.2	Animatsioonid.....	15
3.3	Paigutushaldurid	19
3.4	Stiilid.....	22
3.5	Mallid.....	24
3.6	Kasutajaelemendid (UserControls).....	26
4	Meediaelemendid	30
4.1	Toetatud meediaformaadid	30
4.2	Video hetkepositsiooni kuulamine.....	33
4.3	Täisekraan ning selle piirangud	34
5	Publitseerimine.....	36
5.1	Silverlight komponendi lisamine veebilehele.....	36
5.2	Initsialiseerimisparameetrid	38
5.3	Veebi laadimine	39
5.4	Silverlight Streaming	39
6	Silverlight plussid ja miinused	43
7	Näidisrakendus	45
7.1	Kirjeldus.....	45
7.2	Rakenduse ülesehitus	46
	Kokkuvõte	48

Summary.....	50
Kirjanduse loetelu.....	52
Lisad	55
Lisa 1 – Silverlighti toetavad platvormid ning veebilehitsejad	55
Lisa 2 – Vajalikud vahendid Silverlighti kasutamiseks.....	55
Lisa 3 – Vajalikud vahendid Silverlight rakenduste arendamiseks	55
Lisa 4 - Silverlight 1.0 ning Silverlight 2 Beta 1 võrdlus [32]	57
Lisa 5 - CD	58

Sissejuhatus

Microsoft Silverlight (edaspidi Silverlight) on mitmeplatvormiline tehnoloogia, mis võimaldab luua ning kasutada interaktiivseid ja meediarikkaid veebirakendusi. See on analoogiks sellistele tehnoloogiatele nagu näiteks Flash, Flex, Java Fx jt.

Käesoleva ajani on Microsoft välja lasknud kaks erinevat versiooni tehnoloogiast Silverlight – Silverlight 1.0 ja Silverlight 2 Beta 1. Antud töö keskendub versioonile Silverlight 2, mille võimalused on võrreldes eelmise versiooniga laiemad, kuna ta võimaldab arendada keeltes C# ja VB ning sisaldab ka alamhulka .NET raamistikust.

Käesoleva töö eesmärgiks on uurida Microsoft Silverlight arhitektuuri ja vahendeid, mis on vajalikud meediarakenduste loomiseks veebis, samuti luua näidisrakendus, mis kasutaks Silverlight erinevaid võimalusi.

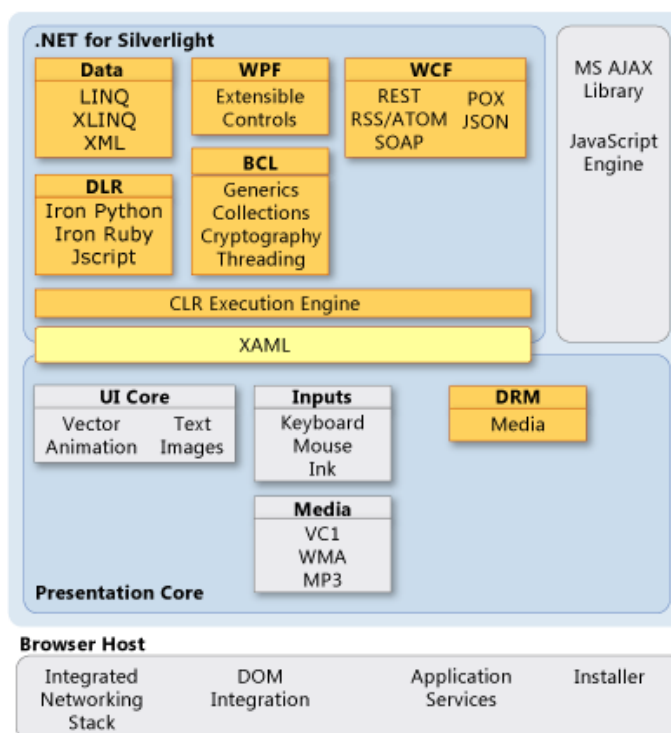
Töö esimeseks peatükis käsitletakse Silverlight arhitektuuri ja töövahendeid ning rakenduse ülesehitust. Kasutajaliidese elemente vaadeldakse teises peatükis, mis koodinäidete abil demonstreerib transformatsioonide, animatsioonide jms. loomist. Kolmas peatükk on pühendatud tööle meediaelementidega. Neljandas peatükis käsitletakse Silverlight rakenduste publitseerimist. Viies peatükk annab ülevaate Silverlight plussidest ning miinustest.

Viimases peatükis antakse ülevaade käesoleva töö raames loodud rakendusest.

1 Silverlight arhitektuur ning rakenduse ülesehitus

1.1 Arhitektuur

Silverlight tehnoloogia [1-5] võib jaotada kolmeks komponendiks – CPF (*Core Presentation Framework*), *.NET for Silverlight* ning *Browser Host* (Joonis 1). Lisas 4 on esitatud Silverlight versioonide 1.0 ja Silverlight 2 Beta 1 võrdlus.



Joonis 1. Silverlight arhitektuur [1]

- CPF alla kuulub kõik visuaalne – vektorgraafika, animatsioonid, multimeedia, interaktiivsus (hiir, klaviatuur ning tugi tahvelarvutitele), XAML (*Extensible Application Markup Language*) parser, samuti ka meediafailide esitamine ning DRM (*Digital Rights Management*).
- .NET for Silverlight – Silverlight sisaldab alamhulka (~5MB) .NET raamistikust (~50MB) objektidest ning tekidest. See võimaldab Silverlight rakendusi arendada .NET keeltes C# ja VB. Andmetega töötamiseks on LINQ (*Language Integrated Query*) ja XML tugi ning suhtlust teiste teenustega (HTTP, RSS, JSON jt.) lihtsustab WCF (*Windows Communication Foundation*). Moodul WPF (*Windows Presentation Foundation*) toob kaasa hulga juba valmisolevaid kasutajaliidese elemente – paigutushaldurid, nupud jms. Lisaks on olemas tugi paljudele .NET põhiklassidele –

loendid, magasinid, lõimed, stringid, regulaaravaldised jne. Enamik .NET teeke on integreeritud kasutaja veebilehitseja Silverlight lissasse, osa neist lisatakse koos Silverlight rakendustega, mis just neid lisateeke vajavad.

- Kolmas komponent – *Browser Host* - vastutab esmakordse paigalduse ning edaspidiste (automaatsete) uuenduste eest. Silverlight oluline osa on koostöö veebilehitsejaga ning veebilehega, mille sisse Silverlight rakendus on lisatud. Rakenduse programmikoodis on võimalik muuta selle hostveebilehe dokumendi objektidemudelit (*DOM*). Kuigi kõik kliendimasinas töötavad Silverlight rakendused töötavad väga rangelt piiritletud õigustega (kokku 500MB suuruses „liivakastis“), on võimalus kliendi masinasse salvestada ajutisi faile (*Isolated Storage*).

1.2 Töövahendid

1.2.1 Kasutajale

Kasutamaks Silverlight tehnoloogiaga loodud veebilehti, on vaja alla laadida Silverlight veebilehitseja lisa. Hetkel on ametlikuks versiooniks Silverlight 1.0 ning avalikult on arendajatele saadaval ka versioon Silverlight 2 Beta 1, mis on välja antud *go-live* litsentsiga. Lisa on mahult veidi üle 2MB (uuem 4MB) ning tema installeerimisele kulub umbes üks minut. Paigaldamiseks on vaja administraatori õigusi. Veebilehitsejaid ei pea pärast paigaldamist taaskäivitama.

Kui Silverlight on arvutis juba olemas, paigaldatakse kasutaja soovi korral uued versioonid automaatselt, kasutajale märkamatuks – see annab arendajatele ka kindluse, et kasutajatel on alati uusim versioon ning ei ole vaja muretseda versioonidevaheliste erinevuste pärast.

1.2.2 Arendajale

Enamike arendusülesannete lahendamise juures saame öelda, et vaja läheb vaid tekstiredaktorit ning tarkvaraarenduspaketti (*SDK*), nii on ka tehnoloogiaga Silverlight, ent siiski on mugavam kasutada spetsiaalseid tööriistu.

Silverlight rakenduste loomiseks on vaja paigaldada vähemalt kaks komponenti – Microsoft Visual Studio 2008 ning Silverlight Tools Beta 1 for Visual Studio 2008.

1.2.3 Microsoft Visual Studio 2008

Visual Studio 2008 on arenduskeskkond, mis on peamiselt suunatud .NET keeltele. Selle tugevaimateks külgedeks Silverlight arenduse kontekstis on kindlasti *IntelliSense* ehk koodivihjamine. Silverlight 1.0 rakenduste juures on Visual Studio suureks trumbiks JavaScript silumine (*debug*), mis on muidu suhteliselt ebamugav protsess. Keskkonna Visual Studio kasutamine Silverlight rakenduste silumiseks võimaldab lihtsasti kinnitada veebilehitseja protsesside külge ning seda ka näiteks üle võrgu Macintosh arvutil jooksva veebilehitseja korral.

Paigaldades paketi Silverlight Tools Beta 1 for Visual Studio 2008 lisanduvad uute projektide valimise dialoogi ka Silverlight rakenduste mallid.

Visual Studio on peamiselt mõeldud programmi loomiseks. Kasutajaliidese loomiseks on võimalik kasutada spetsiaalseid vahendeid nagu Expression Blend ning Expression Design.

1.2.4 Expression Blend 2.5 March Preview

Kuigi programmis Visual Studio 2008 on olemas ka visuaalne koodiredigeerija ehk disainerivaade, on siiski mugavam vahend Silverlight rakenduse kasutajaliidese loomiseks Expression Blend (hetkel uusim versioon Expression Blend 2.5 March Preview).

Erinevalt Visual Studio disainerivaatest, kus arendaja sisestab XAML koodi ning saab koheselt visuaalse tagasiside, on rakenduses Expression Blend võimalik kasutajaliides luua ka analoogselt joonistusprogrammidele ning Expression Blend loob vastava XAML koodi automaatselt.

Antud töö käigus täideti nii arendaja kui ka disaineri rolli ning kasutajaliidese loomiseks kasutati vaheldumisi rakendusi Visual Studio ja Expression Blend. Rakenduses Expression Blend on küll väga mugav töötada hiirega ja uute objektide loomiseks tuleb need vaid töölauale lohutada, kuid mõnikord on vaja luua objekti, mille jaoks ei ole malli Expression Blend tööriistakastis. Sellistel juhtudel tuleb see ise luua kirjutades XAML koodi. Kuna Expression Blend on siiski veel Beta toode, siis ei ole temas kahjuks IntelliSense tuge XAML koodi jaoks. Selle olemasolu vähendaks oluliselt tekkivate vigade arvu ning kiirendaks tööd. IntelliSense on aga olemas rakenduses Visual Studio.

1.2.5 Koostöö programmide Expression Blend ning Visual Studio vahel

Kahe programmi, Expression Blend ja Visual Studio, on sujuv – rakendust arendades saab kasutada mõlemat korraga. Kui disainer teeb muudatusi, siis annab Expression Blend sellest teada rakendusele Visual Studio ja vastupidi. See toimub vaid juhul, kui muudatused arendaja tööd ning faile puudutavad. Peamiselt tulevad teated stiilis „Seda projekti on väljaspool antud programmi muudetud, kas võtame muudatused vastu?“ ette vaid koodi kustutamise juures. Seega ei teki olukorda, kus arendaja ega disainer ei saa korralikult tööd teha, kuna arendusvahendid ei sünkroniseeri enda koodi.

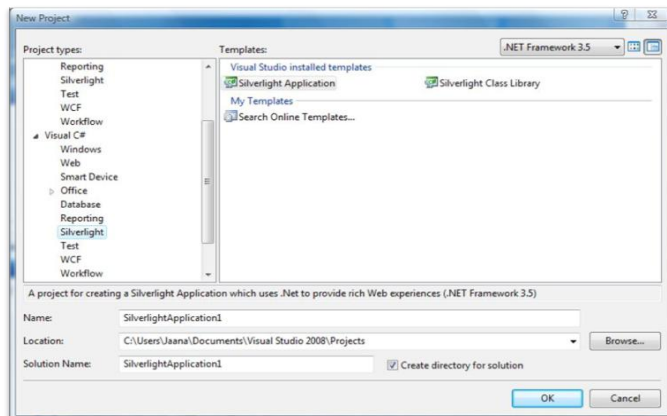
Antud töös on abivahenditena kasutatud vähesel määral ka programme Expression Media Encoder ning Expression Design – esimene võimaldab mugavalt töödelda veebi laetavaid videoid ning teine sarnaneb programmile Expression Blend, ent annab disainerile veelgi vabamad käed kasutajaliidese elementide loomiseks.

Rakendused Expression Blend ning Visual Studio on Tartu Ülikooli Matemaatika-informaatikateaduskonna üliõpilastele ja töötajatele tasuta kättesaadavad ka MSDN AA (*Microsoft Developers Network Academic Alliance*) raames. Rakendustest Expression Design ning Expression Media on saadaval prooviversioonid.

Viited kõikide eelnevate programmide kodulehtedele ning prooviversioonide allalaadimislinkidele võib leida Lisades 2 ja 3.

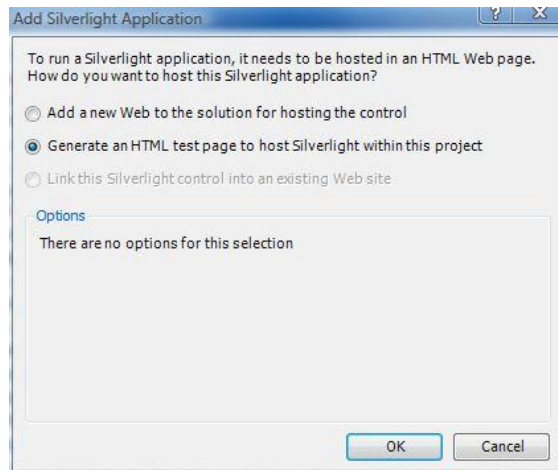
1.3 Silverlight rakenduse ülesehitus

Lihtsaima Silverlight rakenduse loomiseks avame programmi Visual Studio või Expression Blend ning loome uue *Silverlight Application* projekti nimega Veebileht nagu on näidatud Joonisel 2.



Joonis 2. Uue projekti loomine programmis Visual Studio 2008

Luues projekti programmis Visual Studio, küsitakse järgmisena, kas lahendusse lisatakse ka ASP.NET veebiprojekt või genereeritakse testimiseks igal kompileerimisel projekti üks HTML-veebileht (Joonis 3).



Joonis 3. Test-veebilehe valimine.

Kuna käesoleva töö eesmärgiks on uurida Silverlight tehnoloogiat, siis koostööl teiste tehnoloogiatega siin ei peatuta ning loome Silverlight projekti koos lihtsa HTML-lehega, mis hakkab sisaldama meie Silverlight rakendust.

Loodud minimaalne Silverlight 2 Beta 1 rakendus (nimega Veebileht) koosneb järgnevatest failidest [8-11]:

- Page.xaml;
 - Page.xaml.cs;
- App.xaml;
 - App.xaml.cs;
- Properties\AppManifest.xaml;
- Default.html,

mis veebi avaldamiseks pakendatakse *.xap* faili (see on sisuliselt *.zip* fail, lihtsalt teise laiendiga), kuhu kuuluvad

- Veebileht.dll;
- System.Windows.Controls.dll;

- Veebileht.Windows.ControlsExtended.dll;
- ning lisaks – video-, pildi- ja lisa .dll failid, mida rakenduses kasutatakse.

1.3.1 XAML

XAML (*Extensible Application Markup Language*) [6,7] on XML põhine deklaratiivne keel, mis leiab laialdast kasutust .NET tehnoloogiates nagu WPF ning WF (*Workflow Foundation*). WPF rakendustes kasutatakse keelt XAML kasutajaliidese elementide defineerimiseks, nende sidumiseks sündmuste ning andmetega. Silverlight on veebipõhine alamosa WPF raamistikust ning XAML mängib Silverlight rakendustes väga olulist rolli, kuna suurem osa kasutajaliidest kirjeldatakse XAML failides. XAML loob ka võimaluse paremini eraldada disaineri ning arendaja tööd, kuna kõike, mis disainer on XAML failis loonud, on arendajal võimalik koodiga kontrollida, seejuures disaineri koodi puutumata. Selline töövoog teeb võimalikuks disaineri ja arendaja üheaegse töö samade failide kallal; samuti on programmikood paremini loetav, sest kasutajaliides ning loogika on väga selgelt eraldatud.

1.3.2 Failid

1.3.2.1 Page.xaml

Andmaks loodud minimaalsele rakendusele ka sisu, peab muutma faili Page.xaml, mille esialgne kuju on näha Näites 1.

Näide 1

```

1 <UserControl x:Class="Veebileht.Page"
2 xmlns="http://schemas.microsoft.com/client/2007"
3 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4 Width="400" Height="300">
5     <Grid x:Name="LayoutRoot" Background="White"/>
6 </UserControl>

```

Esimene rida Näites 1 viitab C# klassile, kus on sellele XAML failile vastav programmikood, teine ja kolmas rida viitavad XML skeemidele (*schema*), millele fail peab vastama. Neljandas reas on määratud vaikimisi kõrgus ja laius, mille antud rakendus hõivab. Kui mõõtmed on veebilehitseja aknast suuremad, siis ülejääv osa „lõigatakse“ ära.

Järgmisena (rida 5) tuleb igas Silverlight rakenduses mingit tüüpi paigutus-lõuend, mis sisaldab kõiki rakenduse kasutajaliidese elemente - vaikimisi on selleks *Grid* ehk ruudustik. Erinevatest paigutushalduritest on juttu peatükis 2.

Rakendusele uute objektide lisamine on lihtne. Näites 2 on lisatud tekstiväli.

Näide 2

```
1 <TextBlock x:Name="tekst" Text="Tere maailm!" FontSize="36"/>
```

Lisame selle *Grid* märgendite vahele (Näide 3) ning paneme programmi tööle (klahvikombinatsioon CTRL + F5) - ekraanile ilmub tervitus.

Näide 3

```
1 <UserControl x:Class="Veebileht.Page"
2 xmlns="http://schemas.microsoft.com/client/2007"
3 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4 Width="400" Height="300">
5     <Grid x:Name="LayoutRoot" Background="White">
6         <TextBlock
7             x:Name="tekst"
8             Text="Tere maailm!"
9             FontSize="36"/>
10    </Grid>
11 </UserControl>
```

Näites 3 *TextBlock* atribuut *x:Name* on vajalik antud objektile programmikoodis viitamiseks ning animatsioonide lisamiseks. Kui XAML koodis ei ole *x:Name* määratud, siis ei ole objekt koodifailis *Page.xaml.cs* nähtav. Luues animatsioone programmiga Expression Blend, lisatakse *x:Name* atribuudid automaatselt, nendeks saavad *TextBlock1*, *TextBlock2* jne.

Lisaks Näites 3 toodud atribuutidele on olemas veel palju teisi – värvid, kaugused lõuendi äärtest, läbipaistvus jne. ning tänu *IntelliSense* tehnoloogiale on kõikvõimalikud objektid ning nende atribuudid lihtsasti leitavad.

1.3.2.2 Page.xaml.cs

Igale kasutajaliidese *.xaml* failile vastab *.cs* fail. Selles on kirjeldatud programmi loogika, käsitletakse sündmusi, vigu jms. Minimaalselt sisaldab fail *Page.xaml.cs* vähemalt konstruktorit (Näide 4).

Näide 4

```
1 using System.Windows.Controls;
2 namespace Veebileht
3 {
4     public partial class Veebileht : UserControl
5     {
6         public Veebileht()
7         {
8             InitializeComponent();
9         }
10    }
11 }
```

1.3.2.3 App.xaml

Selles failis defineeritakse kogu rakenduse piires kehtivad ressursid – näiteks rakenduse kujunduselementide stiilikirjeldused, mida vaadeldakse teises peatükis.

1.3.2.4 App.xaml.cs

Antud fail sisaldab koodi rakenduse kui terviku kohta. Siin defineeritakse, mis juhtub rakenduse laadimisel, sulgemisel ning erindi korral, mida mujal rakenduses kinni ei püütud. Samuti saab määrata, milline on *.xaml* fail, mis rakenduse käivitamisel esimesena laetakse. Vaid *App* klassi *Startup* sündmust haldavas meetodis on kättesaadavad rakenduse initsialiseerimisparameetrid.

1.3.2.5 Properties/AppManifest.xml

Rakenduse kompileerimise järel on selles failis loetletud kõik *.dll* failid, millest rakendus koosneb. Samuti on defineeritud, millisest failist tuleb rakenduse laadimisel alustada.

1.3.2.6 Default.html

Vaikimisi HTML-fail, mis luuakse Silverlight projekti testimiseks. Sisus on kaks *div* elementi – üks Silverlight objekti jaoks ning teine tekkinud vigade kuvamiseks. Juhul, kui

kliendil ei ole paigaldatud Silverlight lisa veebilehitsejale, siis on siin failis ka kood näitamaks standardset „*Paigalda Silverlight*“ viidet ja pilti. Selleks, et muuta veebilehte, mis automaatselt rakenduse testimiseks avatakse, on kaks võimalust. Üheks võimaluseks on rakenduse Visual Studio 2008 Project menüüst valida *Properties* ning avanenud aknast *Debug* paani alt muuta *Start Action* kindlaks veebileheks. Teise võimalusena jõuab samale rakenduse seadete leheküljele topeltklikiga projekti kataloogpuus kataloogil *Properties*.

2 Kasutajaliidese elemendid

2.1 Transformatsioonid

Silverlight erinevad transformatsioonid [12] võimaldavad teisendada graafilisi objekte ühest asendist teise. Transformatsioonidega on võimalik objekte pöörata, suurendada/vähendada, liigutada ning kallutada. Lisaks saab transformatsioonimaatriksi abil ka ise teisendusi luua.

2.1.1 Võimalikud teisendused

- *RotateTransform* – objekti pööramine etteantud nurga võrra.
- *ScaleTransform* – laiuse ja/või pikkuse suurendamine/vähendamine antud kordaja võrra.
- *TranslateTransform* – objekti liigutamine etteantud pikslite võrra üles, alla, vasakule või paremale. Vasakule või üles liikumiseks tuleb ette anda negatiivsed väärtused.
- *SkewTransform* – objekti kallutamine etteantud kraadide võrra x- ja y-telje suhtes.
- *MatrixTransform* – arendaja enda loodud teisendus.

Näide 5

```
1 <Rectangle x:Name="ruut" Height="100" Width="100"
  Fill="Red">
2     <Rectangle.RenderTransform>
3         <ScaleTransform x:Name="suurem" ScaleX="2"
  ScaleY="2"/>
4     </Rectangle.RenderTransform>
5 </Rectangle>
```

Näites 5 on punase ristküliku suuruseks määratud 100 x 100 (rida 1), kuid kuna lisatud on ka *ScaleTransform* (rida 3), kuvatakse see ekraanile suurusega 200 x 200 pikslit. Määratud on ka atribuut *x:Name*, mis võimaldab programmi töö ajal muuta suurendusfaktoreid.

Ühele objektile mitme transformatsiooni lisamiseks tuleb need rühmitada, kasutades elementi *TransformGroup*, nagu on tehtud Näites 6 (rida 4).

Näide 6

```
1 <Rectangle x:Name="ruut" Height="100" Width="100"
  Fill="Red"
2     RenderTransformOrigin="0.5,0.5" >
3     <Rectangle.RenderTransform>
```

```

4         <TransformGroup>
5             <ScaleTransform ScaleX="0.5" ScaleY="0.5"/>
6             <RotateTransform Angle="45"/>
7         </TransformGroup>
8     </Rectangle.RenderTransform>
9 </Rectangle>

```

Iga transformatsiooni (v.a *TranslateTransform*) puhul saab muutujatega *CenterX* ja *CenterY* määrata, mis punkti loetakse teisenduse alguspunktiks (vaikimisi punkt 0,0 – ülemine vasak nurk). Samas võib objekti enda juures defineerida muutuja *RenderTransformOrigin* ning seejärel kõik teisendused algavad just sellest punktist.

2.2 Animatsioonid

Animatsioonidega [13-15] on võimalik luua interaktiivseid ning pilkupüüdvaid eriefekte. Sisuliselt on animatsioonid transformatsioonid, mis toimuvad teatud aja vältel. Animeerida saab objektide neid atribuute, mis on kas *Double*, *Point* või *Color* tüüpi.

Animeerimiseks tuleb kõigepealt luua süžeeahvel (*Storyboard*), mis ülekantud tähenduses võiks olla filmilint, millele lisatakse animatsioonid ehk film.

Näide 7

```

1 <Rectangle x:Name="ruut" Height="10" Width="10"
  Fill="Blue">
2     <Rectangle.Resources>
3         <Storyboard x:Name="Kuma">
4             <DoubleAnimation x:Name="Animatsioon"
5                 Storyboard.TargetName="ruut"
6                 Storyboard.TargetProperty="Opacity"
7                 From="0" To="1"
8                 Duration="00:00:05" />
9         </Storyboard>
10    </Rectangle.Resources>
11 </Rectangle>

```

Kood Näites 7 muudab 5 sekundi jooksul (rida 8) sinise ruudu läbipaistvust täiesti läbipaistvast kuni läbipaistmatuni (read 6. ja 7). Ühes XAML failis võib olla mitu süžeeahvli ning igaühes neist võib olla mitu animatsiooni.

Ükski animatsioon ei alga automaatselt, animatsioonid tuleb programmikoodis ise käivitada. Näites 7 defineeritud animatsioon käivitatakse käsuga `käsuga Animatsioon.Begin()`.

2.2.1 Võtmekaadrid

Näites 8 esitatud animatsioon liigub ühtlase kiirusega, kuid mõnikord on tarvis ise paika panna, kus täpselt (või mis seisus) objekt mingil ajahetkel olema peab. Selleks võib kasutada võtmekaadritega (*Keyframe*) animatsioone [14-15].

Näide 8

```
1 <Rectangle Height="5" Width="5" Fill="Blue"
  x:Name="rectangle">
2   <Rectangle.Resources>
3     <Storyboard x:Name="Liiguta">
4       <DoubleAnimationUsingKeyFrames
5         Storyboard.TargetName="ruut"
6         Storyboard.TargetProperty="(Canvas.Left)">
7         <SplineDoubleKeyFrame KeyTime="00:00:01"
  Value="50"/>
8         <SplineDoubleKeyFrame KeyTime="00:00:03"
  Value="100"/>
9       </DoubleAnimationUsingKeyFrames>
10    </Storyboard>
11  </Rectangle.Resources>
12 </Rectangle>
```

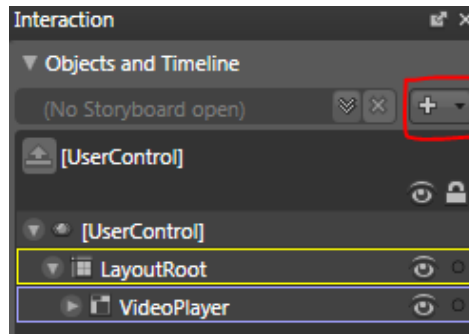
Animatsioon Näites 8 liigutab ruudu lõuendi (*Canvas*) äärest 100 piksli kaugusele. Seejuures esimesed 50 piksli läbitakse 1. sekundi lõpuks ning ülejäänud 50 piksli 2 korda aeglasemalt (read 7 ja 8).

Eelnevates näidetes lisasime süžeetahvlid animeeritavate ressursside sisse. Teine, levinum võimalus on lisada süžeetahvlid juurelemendi *UserControl* ressursside hulka. Sinna lisatakse automaatselt näiteks keskkonnas Expression Blend loodud animatsioonid.

Tuleb arvestada, et *UserControl.Resources* süžeetahvlid on Expression Blend süžeetahvlite tööriistapaanis leitavad, ent individuaalsete objektide ressursside hulka lisatud seal ei kuvata.

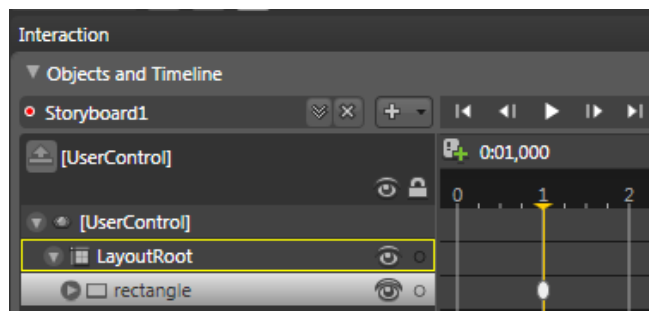
Lihtsamaid animatsioone on lihtne kirjutada kasutades keelt XAML, ent vähegi keerulisemate süžeetahvlite korral, kus animeeritakse erinevate objektide mitmeid atribuute ning kus animatsioonid kasutavad võtmekaadreid, muutub käsitsi kirjutamine tülikaks.

Mugavam viis animatsioonide kirjeldamiseks Silverlight rakenduses on kasutada programmi Expression Blend. Süžeetahvli loomine on näidatud Joonisel 4.

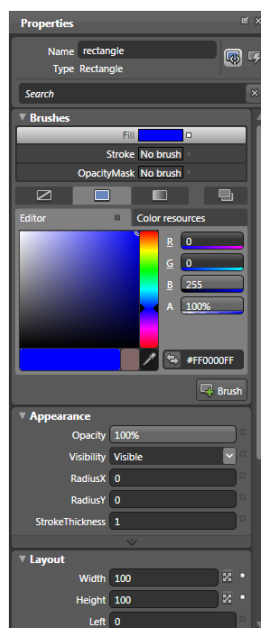


Joonis 4. Uue süžetahvli loomine

Animatsiooni loomiseks tuleb esmalt valida objekt, mida soovitakse animeerida. Seejärel liigutada kollane marker ajaribal punkti, mil animatsioon peaks jõudma mingisse kindlasse seisule. Objekti atribuutide paneelis (Joonis 6) tuleb seejärel väärtused muuta selliseks nagu nad tol hetkel olema peavad. Nii lisatakse valitud süžetahvlile uus võtmekaader. Eelneva protsessi tulemus on näha Joonisel 5.



Joonis 5. Ristkülikule on lisatud 1 sekundi pikkune animatsioon.

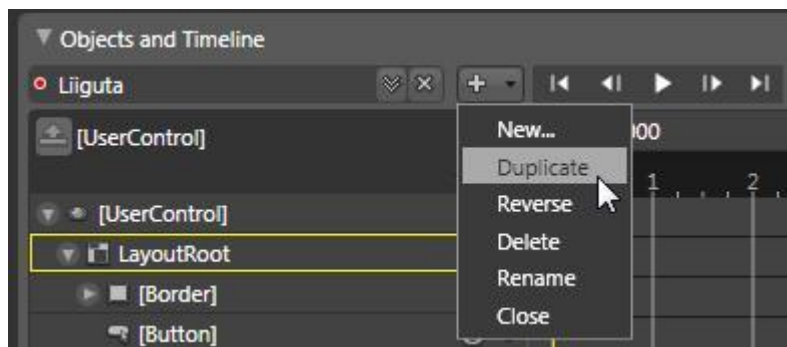


Joonis 6. Osa atribuutide paanist.

2.2.2 Animatsiooni ümberpööramine

Luues animatsioonidega rakendust, on tõenäoline, et kui miski liigub seisust A seisu B, siis mingil hetkel peab see ka tagasi muutuma. Selle asemel, et luua täiesti uus süžeetahvel ning taas paika panna võtmekaadreid, võime seda teha ka palju lihtsamini.

Selleks tuleb võtta esialgne süžeetahvel ning see kopeerida nagu on tehtud Joonisel 8.



Joonis 7. Animatsiooni dubleerimine.

Kopeeritud süžeetahvli juures (nimeks saab *Nimi_Copy1*) valida *Reverse behavior* ning meil ongi vastupidine animatsioon olemas. Nüüd võib lihtsama viitamise jaoks loodud süžeetahvlile anda ka uue nime.

2.2.3 Animatsioonide korduvkasutamine

Ühe animatsiooni mitme objekti juures kasutamiseks, ei piisa programmikoodis muutuja *Storyboard.TargetName* määramisest, vaid tuleb *TargetName* XAML koodis ära jätta ning see iga uue animatsiooni alguses programmikoodis uuesti määrata (Näide 9) [15]:

Näide 9

```
1 Animatsioon.Stop();
2 Animatsioon.SetValue(Storyboard.TargetNameProperty,
  uus.Name);
3 Animatsioon.Begin();
```

Korruga saab üks süžeetahvel olla seotud vaid ühe objektiga. Kui mitmel objektil peab samaaegselt toimuma ühesugune animatsioon, siis on üks võimalus luua animatsioon programmikoodis, mida on tehtud Näites 10. Määratud on samad atribuudid, mis eelnevates animatsioonide näiteski. Tähele tuleb panna, et lisaks tuleb loodud animatsioon lisada, kas juurelemendi või rakenduse globaalsete ressursside hulka (rida 10).

Näide 10

```
1 Storyboard filmilint = new Storyboard();
2 DoubleAnimation animatsioon = new DoubleAnimation();
3 Duration kestvus = new
  Duration(TimeSpan.FromSeconds(0.5));
4 filmilint.Duration = kestvus;
5 animatsioon.Duration = kestvus;
6 filmilint.Children.Add(animatsioon);
7 Storyboard.SetTargetProperty(animatsioon,
  "(Canvas.Left)");
8 animatsioon.To = 200;
9 Storyboard.SetTarget(animatsioon, ruut);
10 LayoutRoot.Resources.Add(filmilint);
11 filmilint.Begin();
```

Kasutades tsüklit võib Näite 10 koodi abil lisada ühesuguse animatsiooni mitmele erinevale objektile. Programmikoodis võib uusi animatsioone nii luua kui ka juba olemasolevaid muuta. Selleks peab süžeetahvlil olema määratud *x:Name* atribuut.

2.3 Paigutushaldurid

Iga Silverlight XAML fail sisaldab üht juurelementi – paigutushaldurit (*Layout Control*), millele lisatakse kõik kasutajaliidese objektid. Paigutushaldurid on näiteks *Canvas*, *Border*, *StackPanel* [16 - 18].

2.3.1 Canvas

Canvas on lihtsaim paigutushaldur, mille korral objekti asukohta lõuendil mõjutavad vaid kaugus vasakust ülemisest servast. Kui mitu elementi on üksteise peal, sõltub nende järjekord objekti muutuja *ZIndex* suurusest. *Canvas* enda juures on peale kõrguse ja laiuse muudetav vaid tausta välimus. Eelnevates näidetes oleme objekte värvinud vaid ühe kindla värviga, kirjutades *Background="Blue"* või *Fill="Pink"* need on Näites 11 kirjutatuga samaväärsed.

Näide 11

```
1 <Canvas.Background>
2     <SolidColorBrush Color="Pink"></SolidColorBrush>
3 </Canvas.Background>
```

Silverlight objekte on võimalik värvida ka teist tüüpi pintslitega. Üks nendest on *LinearGradientBrush*, mida on kasutatud Näites 12. Millise nurga all pintsliga värvitakse, määravad reas 3 *StartPoint* ning *EndPoint*. Kasutatavad värvid määratakse *GradientStop* väärtustega (rida 4 ja 5), muutuja *Offset* näitab seejuures seda, kui kaugel alguspunktist peab selle värvini sujuvalt jõudma.

Näide 12

```
1 <Canvas x:Name="canvas" Width="100" Height="100" >
2   <Canvas.Background>
3     <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
4       <GradientStop Color="White"/>
5       <GradientStop Color="DeepPink" Offset="1"/>
6     </LinearGradientBrush>
7   </Canvas.Background>
8   <Rectangle x:Name="ruut" Width="50" Height="50"
9     Canvas.Left="10" Canvas.Top="10"
10    Canvas.ZIndex="0" Fill="Pink"/>
11 </Canvas>
```

2.3.2 Border

Paigutushalduril *Border* on eelmisega rohkem võimalusi välimuse määramiseks. Raamil (*Border*) on seatavad pikkus, laius, raami laius, värv, sisu värv ning nurkade ümarus. Sisu on võimalik nii horisontaalselt kui ka vertikaalselt joondada kas üles, alla, keskele või „venitada“ (*stretch*), võib olla määratud ka vooderdus (*padding*). Joonisel 8 on keerukam näide *Border* elemendist – nii tausta- kui ka raamivärviks on kasutatud gradiente.



Joonis 8. Gradient raami ja taustaga *Border*.

Erinevalt teistest paigutushalduritest võib raamil olla vaid üks alluv. See ei ole aga suureks takistuseks, sest alluvaks võib seada *Canvas* halduri ning selle alla koondada kõik ülejäänud. Näites 13 on loodud peenikese raamiga ning ümarate nurkadega (rida 3) *Border*, mille sisse on lisatud nupp (rida 4 ja 5).

Erinevalt *Canvas* paigutushaldurist, ei pea *Border* elemendi puhul seadma staatilisi koordinaate vaid on võimalik kasutada joondamist (rida 5).

Näide 13

```
1 <Border x:Name="raam"  
2 Height="50" Width="50"  
3 CornerRadius="10,10,10,10" BorderThickness="5,5,5,5" >  
4 <Button x:Name="Nupp" Height="40" Width="40"  
   Content="Nupp"  
5 HorizontalAlignment="Center"/>  
6 </Border>
```

2.3.3 StackPanel

Paigutushaldur *StackPanel* võimaldab temas sisalduvaid objekte paigutada vastavalt muutuja *Orientation* väärtusele, kas ühte ritta või veergu. Analoogselt halduriga *Border* saab siingi alluvate joondust mõjutada muutujatega *Horizontal-* ning *VerticalAlignment*. Näites 14 on loodud lihtne *StackPanel* ning koodi tulemus on näha Joonisel 10.

Näide 14

```
1 <StackPanel x:Name="pinu" Orientation="Horizontal">  
2     <Button Width="100" Height="20" Content="Esimene"/>  
3     <Button Width="100" Height="20" Content="Teine"/>  
4     <Button Width="100" Height="20" Content="Kolmas"/>  
5 </StackPanel>
```



Joonis 9: Horisontaalne Stackpanel.

2.3.4 ScrollViewer

Tavalisete paigutushaldurite korral, juhul kui sisu on halduri mõõtmetest suurem, lõigatakse üle jääv osa ära. *ScrollViewer* puhul tekivad kogu sisu vaatamiseks kerimisribad.

2.3.5 Grid

Eelnevad paigutushaldurid pakuvad mitmekülgseid võimalusi sisu kuvamiseks. Suuremat kontrolli selle üle, kuidas kuvatav sisu täpselt positioneeritud on, pakub siiski vaid *Grid*. Seepärast on näidisrakenduses peamiseks paigutushalduriks *Grid*, millele on vajadusel lisatud teisi – *StackPanel*, *Canvas*, *Border*, mille tugevusteks on seevastu just sisu presenteerimine.

Grid sarnaneb HTML-tabelile – ta sisaldab ridu ja veerge, ent märgenduses on päris palju erinevusi.

Näide 15

```
1 <Grid Height="100" Width="100" ShowGridLines="True">
2   <Grid.RowDefinitions>
3     <RowDefinition Height="0.5*" />
4     <RowDefinition Height="0.5*" />
5   </Grid.RowDefinitions>
6   <Grid.ColumnDefinitions>
7     <ColumnDefinition Width="0.5*" />
8     <ColumnDefinition Width="Auto" />
9   </Grid.ColumnDefinitions>
10  <Button x:Name="nupp" Grid.Row="0" Content="nupp" />
11  <Button x:Name="nupp2" Grid.Row="0" Grid.Column="1"
12    Content="nupp" />
13 </Grid>
```

Esimene erinevus HTML-tabeliga seisneb selles, et sisu ei lähe mitte tabeli definitsiooni, vaid see lisatakse hiljem, viidates lisatava objekti juures *Grid* nende rea ning tulba numbritele, kuhu sisu minema peab. Teine erinevus on, et ridade ja tulpade definitsioonid ei pea sisaldama staatilisi väärtusi, vaid suurused võib defineerida ka proportsioone kasutades. Näite 15 ridades 3 ja 4, on loodud read, mille kõrgus on pool *Grid* kõrgusest. Reas 8 on loodud tulp, mille laius arvutatakse automaatselt. Lisaks toetab *Grid* ka atribuute *Column-* ning *RowSpan*.

2.4 Stiilid

Veebilehtedel on palju erinevaid komponente – nupud, tekstikastid, pildid jms, millest igapähele on hulk omadusi – kõrgus, laius, värv jne. Seni oleme välimuse kirjeldused lisanud

objektide XAML koodi. Mitme ühesuguse elemendi korral on välimus defineeritud iga objekti juures eraldi. Sellise stiili korral langeb aga loetavus ja tekib palju üleliigset ning raskesti hallatavat koodi. Lihtsam ning otstarbekam on korduvad stiilikirjeldused kokku koondada. Stiiliga [20] saab määrata kõiki samu objekti omadusi nagu XAML koodiski.

Stiilikirjeldused võib lisada kas rakenduse globaalsete ressursside hulka failis App.xaml või individuaalse objekti ressurssidesse. Esimesel juhul on stiilid kasutatavad üle rakenduse, teisel juhul vaid ühe konkreetse objekti ja ta alluvate juures.

Erinevalt tavalisest XAML koodist on stiili identifitseeriv atribuut *x:Key*, mitte *x:Name*. Näites 16 on faili *App.xaml* lisatud *TextBlock* elemendile välimuskirjeldus *menu*. Seatud on tekstisuurus (rida 3) ning värv (read 4-11).

Näide 16

```
1 <Application.Resources>
2   <Style x:Key="menu" TargetType="TextBlock">
3     <Setter Property="FontSize" Value="24" />
4     <Setter Property="Foreground">
5       <Setter.Value>
6         <LinearGradientBrush>
7           <GradientStop Color="LightBlue"
8             Offset="0"></GradientStop>
9           <GradientStop Color="SteelBlue"
10            Offset="1"></GradientStop>
11          </LinearGradientBrush>
12        </Setter.Value>
13      </Setter>
14    </Style>
15  </Application.Resources>
```

Selleks, et määrata objektile stiili, tuleb lisada *Style* atribuut nagu on näidatud Näites 17.

Näide 17

```
1 <TextBlock Text="Tere!" Style="{StaticResource menu}"/>
```

Sama välimuse kirjeldus XAML koodis jaotuks kaheksale reale. Kui rakenduses oleks viis sellist tekstikasti, võtaks kood ruumi 40, stiile kasutades aga vaid viis rida.

Juhul kui ühe objekti välimuses on vaja teha vaid väikene muudatus – näiteks suurendada teksti suurust selle esiletõstmiseks, siis ei pea looma täiesti uut stiili, vaid saab ka juba defineeritud stiili atribuute üle kirjutada või siis lisada.

Kuigi teksti suurus on Näites 16 loodud stiilis *menu* määratud olema 24, saab Näites 18 teksti suuruseks 30 (rida 2) ning lisaks tehakse tekst rasvaseks, mis on uus omadus, mida stiilis määratud ei ole. Juhul kui selliseid muudatusi vajavad mitmed objektid, tuleks koodiliiasuse vältimiseks luua uus stiil.

Näide 18

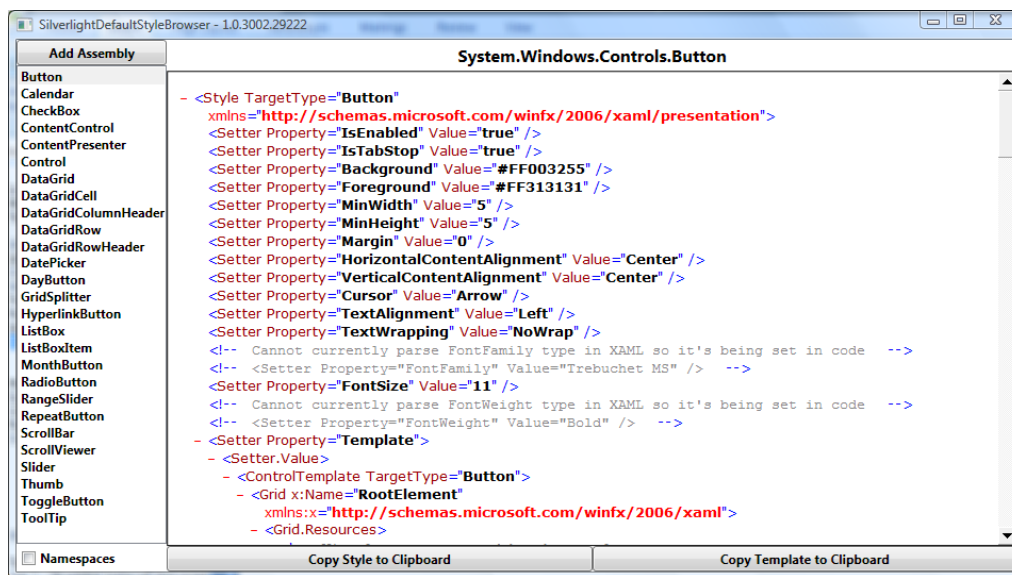
```
1 <TextBlock Text="Tere!" Style="{StaticResource menu}"
2   FontSize="30"   FontWeight="Bold" />
```

2.5 Mallid

Stiilide eesmärgiks on korduva koodi kokkukoondamine ning rakenduse kasutajaliidese muutmise ning isikupärastamise lihtsustamine. Stiilidega saab muuta kõiki atribuute, mis on defineeritavad XAML koodis.

Mallidega [19] on aga võimalik muuta ka objektide neid väärtusi, mida muidu ei saa – kuju, elemendiga seotud animatsioone ja seda ilma funktsionaalsust kaotamata. Mall on objekti üks paljudest omadustest, mida saab defineerida stiilis.

Mallide loomisel on otstarbekas võtta aluseks objektide vaikimisi mallid – neid võib leida dokumentatsioonist. Nii sisseehitatud kui ka kasutaja poolt defineeritud malle on mugav sirvida programmiga *SilverlightDefaultStyleBrowser*. Joonisel 10 on selle rakenduse ekraanipilt, millelt on näha, et igal objektil on mall ja vaikimisi stiil.



Joonis 10. Stiili sirvimine SilverlightDefaultStyleBrowser rakenduses

Malli luues võib olla väga põhjalik - pannes paika kõikvõimalikd atribuudid või määrata vaid mõned ning ülejäänud jätta vaikimisi mallis määratuks.

Näite 19 esimeses reas on loodud stiil nupule. Teises reas määratakse nupu taustavärviks tumehall. Kolmandas reas loodava nupu malli juurelemendiks saab Grid, millele lisatakse 7. reas ellips, mis määrab nupu kuju. Ellipsi (nupu) kõrgus ja laius saavad väärtuse vastavalt sellele, mis on XAML koodis seatud. Olenemata suurusel on nupu piirjoon musta värvi (rida 10).

Nupu sisu välimuse kirjeldamine algab 13. reast. Nupu taustavärv saab väärtuseks selle, mis XAML koodis paika on pandud (rida 14). Kui XAML koodis ei ole värv defineeritud, siis jääb taustavärviks tumehall, mis on määratud selle stiili alguses (rida 2). Juhul kui peale 14. rida oleks veel taustavärvi määratud, saaks värviks viimasena defineeritud värv.

Nupu sisu (tekst, pilt vms.) saab samuti väärtuse XAML koodist. Vertikaalselt kuvatakse sisu keskele (rida 16). Horisontaalselt joondatakse sisu vasakule, sest antud omadus ei ole määratud ei XAML koodis, stiilis ega ka mallis. Juhul kui mallis oleks kirjas, et horisontaalne joondus peab tulema XAML koodist ja seda ei oleks seal määratud, joondataks sisu siiski keskele, sest sel juhul saab horisontaalne joondus Silverlight vaikimisi väärtuse. Omadusi, mida ei ole mallis kirjas, ei saa objekti juures määrata, sest neid ignoreeritakse. Sellel mallil puuduvad animatsioonid, mis eristaksid nupu erinevaid seisundeid – vajutatud, all jne, funktsionaalselt töötab see aga niisamuti nagu tavaline nupp.

Näide 19

```
1 <Style x:Key="NupuMall" TargetType="Button">
2   <Setter Property="Background" Color="DarkGray" />
3   <Setter Property="Template">
4     <Setter.Value>
5       <ControlTemplate TargetType="Button">
6         <Grid>
7           <Ellipse
8             Width="{TemplateBinding Width}"
9             Height="{TemplateBinding Height}"
10            Stroke="Black"
11          >
12        </Ellipse>
13        <ContentPresenter
14          Background="{TemplateBinding Background}"
15          Content="{TemplateBinding Content}"
16          VerticalContentAlignment="Center">
```

```

17         />
18     </Grid>
19 </ControlTemplate>
20 </Setter.Value>
21 </Setter>
22 </Style>

```

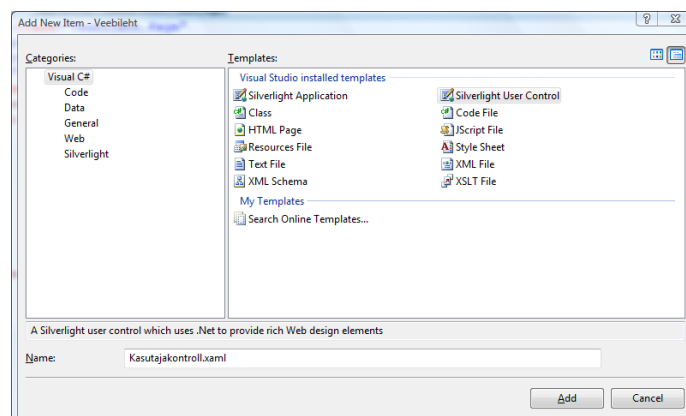
2.6 Kasutajaelemendid (UserControls)

Control on üks kasutajaliidese juhtelement – nupp, kerimisriba jms, millel on välimus, kindel funktsionaalsus ning otstarve.

Rakendustes on tihti olukordi, kus kindla funktsionaalsuse ei loo mitte üks objekt, vaid mitu. Selline on näiteks veebipoes aadressivorm – tekstikastid, nende selgitused, nupp kinnitamiseks. Kasutades stiile, väheneb välimuse kirjeldamiseks vajaminev kood, kuid näiteks veebipoes võib vaja minna kaht sellist komplekti – üks kohaletoitmetamise- ning teine maksjaaadressi kirjeldamiseks. Taas tekib koodiliiasus, mille vältimiseks võib luua *UserControl* [21] elemendi, mis võimaldab objekte grupeerida ühtseks hallatavaks osaks ning mida saab taaskasutada ka sellest konkreetsest rakendusest väljaspool. Sõltumata sellest, mitu elementi *UserControl* sisaldab, on see hiljem rakendusse lisatav vaid ühe reaga.

2.6.1 Kasutajaelemendi loomine

Esimene võimalus kasutajaelemendi loomiseks on lisada projekti uus XAML fail, kasutades Visual Studio 2008 *Add-> New Item* menüüs *UserControl* malli (Joonis 11). Loodav *UserControl* fail on sarnane failiga *Page.xaml* (Näide 1), mis luuakse kasutades Visual Studio 2008 Silverlight rakenduse projekti malli (Joonis 11).

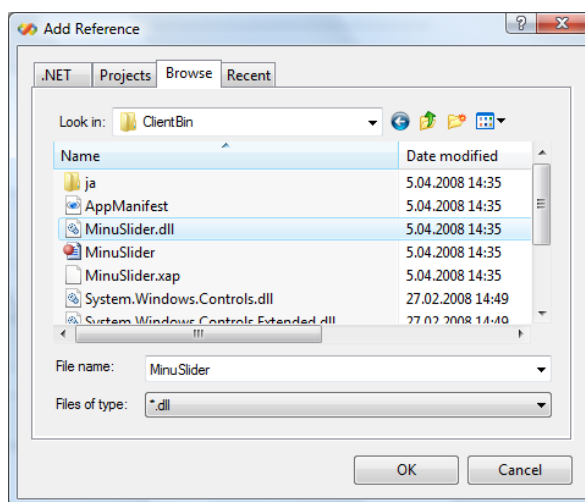


Joonis 11. Kasutajakontrolli loomine.

Kasutajaelemendi XAML faili ning sellele vastavasse programmikoodi võib kirjutada kogu funktsionaalsuse, mida soovitakse taaskasutada.

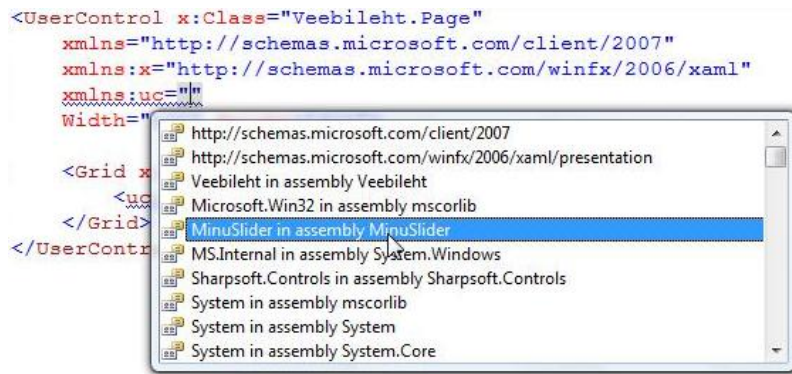
Loodud kasutajaelemendi rakendamiseks tuleb XAML faili lisada viide selle nimeruumile. Viide kasutajaelemendile, mis asub samas lahenduses, on toodud Näite 21 neljandas reas. Selline märgendus teeb antud failis kasutatavaks kõik kasutajakontrollid, mis asuvad nimeruumis Veebileht.

Kui on vaja rakendada kasutajaelementi, mis ei asu samas lahenduses (*Solution*), siis tuleb projektis esmalt lisada viide (*Reference*) selle .dll failile. Viite saab lisada valides menüüst *Project Add Reference*. Avanenud aknas saab kataloogipuust liikuda vajaliku .dll failini.



Joonis 12. Viitamine kasutajaelemendile, mis ei asu rakendusega samas lahenduses.

Joonisel 12 lisatakse viide projektile *MinuSlider*. Lisaks tuleb ka XAML faili lisada viide selle .dll faili nimeruumile (Joonis 13, Näide 20). Edaspidi on kasutatavad kõik *UserControl* elemendid, mis on nimeruumis *MinuSlider*.



Joonis 13

Näide 20

```
1 xmlns:uc="clr-namespace:MinuSlider;assembly=MinuSlider"
```

Kasutajaelemendi kasutamine failis *Page.xaml* on toodud Näites 22 – *uc* on vabalt valitud märgend, mis on defineeritud Näites 20. Lisaks võiks Näites 22 lisada või üle kirjutada ka kasutajaelemendi *Kasutajakontroll* laiuse, kõrguse ja kõik muu, mis ühe *UserControl* objekti juures võimalik on. Lisades mitu ühesugust kasutajaelementi, tuleb nende eristamiseks määrata *x:Name* atribuut.

Näide 21

```

1 <UserControl x:Class="Veebileht.Page"
2     xmlns="http://schemas.microsoft.com/client/2007"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:uc="clr-namespace:Veebileht"
5     Width="100" Height="100">

```

Näide 22

```

1 <uc:Kasutajakontroll />
2 <uc:TeineKasutajakontroll />

```

Teine võimalus kasutajaelemendi loomiseks on programmi Expression Blend abil. Juba olemasolevas XAML failis võib valida teatud objektid ning paremklikuga avanevas kontekstimenüüs valida *Make control*. Selle tulemusena luuakse projekti uus *UserControl* fail, mille sisuks saavad eelnevalt valitud objektid. Võimalik on esialgsed objektid jätta lähtefaili esialgsel kujul või asendada need viitega loodud kasutajaelemendile (analoogselt Näitele 22). Viimasel juhul lisatakse lähtefaili päisesse viide kasutajaelemendi nimeruumile automaatselt.

Eelnevalt kirjeldatud meetod on mugav olukordades, kus alles töö käigus selgub, et mõningad objektid võiksid olla kasutajakontrollid – nii ei pea neid uues failis taas looma või mehhaaniliselt ümber tõstma.

Analoogselt programmiga Visual Studio saab ka rakenduses Expression Blend luua kasutajaelementi, lisades projekti uue *UserControl* XAML faili, kasutades menüüd *Add New Item*.

Kasutajaelementide defineerimine on vältimatu vähegi keerulisemate ja dünaamilisemate rakenduste juures. Kasutajaelemendid teevad võimalikuks OOP lähenemise Silverlight kasutajaliidese arendusele – dünaamiliselt on võimalik luua näiteks menüüsid, menüüpunkte, taasesitlusloendi elemente jms.

3 Meediaelemendid

Silverlight üks tugevusi on oskus pakkuda multimeediat paljudele platvormidele. Silverlight suudab esitada kõiki enda poolt toetatavaid meediaformaate nõudmata koodeki või meediamängija olemasolu kliendi masinas [22-25].

3.1 Toetatud meediaformaadid

Toetatakse järgmisi formaate [23]:

Video:

- WMV1: Windows Media Video 7
- WMV2: Windows Media Video 8
- WMV3: Windows Media Video 9
- WMVA: Windows Media Video Advanced Profile, non-VC-1
- WMVC1: Windows Media Video Advanced Profile, VC-1

Heli:

- WMA 7: Windows Media Audio 7
- WMA 8: Windows Media Audio 8
- WMA 9: Windows Media Audio 9
- MP3: ISO/MPEG Layer-3 (*piirangutega*)

Lisaks on toetatud *http*, *https*, *mms* protokollid ning *ASX* (*Advanced Stream Redirector*) taasesitusloendid (*playlist*).

Näide 23:

```
1 <MediaElement Height="100" Width="100"  
2 Source="Butterfly.wmv" Stretch="Fill" />
```

Näites 23 lisatakse videofail suurusega 100 x 100 pikslit. *Source* on videofaili relatiivne asukoht võrreldes antud XAML failiga. Parameetri *Stretch* väärtus määrab, kuidas video etteantud raamidesse (100 x 100) paigutatakse.

- Stretch = “None“ korral on videost nähtav nii palju kui etteantud raamidesse mahub – väga suure video korral vaid vasak ülemine nurk.
- Stretch = “Uniform“ (vaikimisi) puhul lisatakse raamidesse maksimaalse suurusega video nii, et säiliks video proportsioonid.
- Stretch = “UniformToFill“ korral säilitab video oma proportsioonid, kuid raami mittemahtuv osa eemaldatakse.
- Stretch = “Fill“ lisab video, surudes ta etteantud mõõtmetesse kokku.

MediaElement tüüpi objekte saab animeerida ning transformeerida analoogselt tavaliste objektidega. Lisaks võib aga *MediaElement*i kasutada ka pintslina nagu on näha Näites 24. Kõigepealt lisame lehele video, mis jääb esialgu nähtamatuks (rida 1). Seejärel loome elemendi *TextBlock* tekstiga Tere! (rida 2) ning lisame sellele *VideoBrush* tüüpi pintslile, mille nõ. värviks saab näite esimeses reas loodud film (rida 5). Tulemuseks saame teksti Tere! Mille sees mängib film (Joonis 15).

Näide 24

```

1 <MediaElement x:Name="film"
   Source="Butterfly.wmv"Opacity="0"/>
2 <TextBlock Text="Tere!" FontSize="80" FontWeight="Bold"
3           FontFamily="Comic Sans MS">
4 <TextBlock.Foreground>
5   <VideoBrush SourceName="film"/>
6 </TextBlock.Foreground>
7 </TextBlock>

```



Joonis 14: *TextBlock*, mille sees mängib video.

Suuremate videofailide allalaadimine on ajamahukas. Kinnitamaks kasutajale, et rakendus siiski töötab ning laetakse videofaili, on meil vaja anda neile tagasisidet, mida rakendus hetkel teeb ning millal video on vaadatav. Seda tagasisidet aitab anda sündmusekuular *MediaElement.CurrentStateChanged*, mis annab teada antud hetkel kehtiva seisundi

meediaelemendi seitsmest võimalikust seisundist (*MediaElement.State*) – *Closed, Opening, Buffering, Playing, Paused, Stopped, Error*.

Multimeedia sujuvaks esitluseks vajaliku puhvri suuruse arvutab Silverlight ise, võttes arvesse faili suurust ning kasutuses olevat ribalaiust. Puhvri võib ka ise määrata, näiteks defineerides mitu sekundit meediat peab puhvris olema enne esitluse alustamist.

Teine oluline sündmusekuular on *DownloadProgressChanged*, mis teatab, kui palju on antud hetkel meediafailist alla laetud. See info on kasulik näiteks progressiriba kuvamiseks.

3.1.1 Markerid

Markeriteks [25] nimetakse metaandmeid, mis on seotud teatud kindla punktiga videofailis. Markereid kasutatakse näiteks video peatükkideks jaotamisel ja kontekstitundlike reklaamide kuvamiseks internetivideodes. Markerid lisatakse tavaliselt mõne tarkvara abil nt. Expression Media, kuid markereid saab lisada ka programmikoodis. Näites 25 on filmi Butterfly.wmv esimesele sekundile lisatud marker tekstiga „Tere“, markeri tüübiks on seatud „minuMarker“.

Näide 25

```
1 <MediaElement x:Name="film" Source="Butterfly.wmv" >
2 <MediaElement.Markers>
3 <TimelineMarker Text="Tere" Time="0:0:1"
   Type="minuMarker"/>
4 </MediaElement.Markers>
5 </MediaElement>
```

Juhul kui Silverlight rakenduses mängiv video jõuab markerini, tekib sündmus *MarkerReached*, millele on võimalik programmikoodis reageerida. *TimelineMarker* tüüpi objekt sisaldab alati teksti, tüüpi ning *TimeSpan* tüüpi aega, kus see marker asub. Nii teksti kui ka tüüpi võib arendaja ise defineerida. Näites 26 lisatakse Näites 25 loodud filmile *MarkerReached* sündmusekuular, mida haldab Näites 27 meetod *OnMarkerReached*, mis kuvab ekraanile iga markeri kõik andmed.

Näide 26

```
1 film.MarkerReached +=  
2 new TimelineMarkerRoutedEventHandler (OnMarkerReached);
```

Näide 27

```
1 public void OnMarkerReached(object sender,  
2                             TimelineMarkerRoutedEventArgs  
3     e)  
4 {  
5     debugText.Text = e.Marker.Time.Seconds.ToString();  
6     debugText.Text += " " + e.Marker.Type.ToString();  
7     debugText.Text += " " + e.Marker.Text;  
8 }
```

3.2 Video hetkepositsiooni kuulamine

Iga videorakendus eeldab, et kasutaja näeb video kulgu – kui palju on mängitud; kui palju on veel mängida jne. Võiks eeldada, et Silverlighti on sisseehitatud lihtne meetod video positsiooni kuulamiseks, ent kahjuks see nii ei ole.

Näidisrakenduses on see ülesanne lahendatud kasutades XAML süžeetahvleid. XAML koodis (Näide 28) on loodud üks *Canvas*, millel ei ole muid atribuute määratud kui vaid *x:Name*, mis võimaldab tema poole programmikoodis pöörduda.

Näide 28

```
1 <Canvas x:Name="dummy"/>
```

Seejärel on Näites 29 loodud *Storyboard dummyStory*, mis kestab ühe sekundi ning mille lõppedes kutsutakse välja meetod *updateVideoTimeline()*.

Näide 29

```
1 <Storyboard x:Name="dummyStory"  
2     Storyboard.TargetName="dummy">  
3     <DoubleAnimationUsingKeyFrames  
4         Storyboard.TargetName="dummy"  
5         Storyboard.TargetProperty="(UIElement.Opacity)"  
6         BeginTime="00:00:00">  
7         <SplineDoubleKeyFrame  
8             KeyTime="00:00:01"
```

```

9         Value="0"/>
10     </DoubleAnimationUsingKeyFrames>
11 </Storyboard>

```

Meetod `updateVideoTimeline` Näites 30 loeb video positsiooni ning vastavalt sellele uuendab mängiva video kella ning progressi näitavat joont

Näide 30:

```

1 void updateVideoTimeline (object sender, EventArgs e)
2 {
3     VideoProgress.Text = Video1.Position.ToString();
4     Slider1.Value =
5     Convert.ToDouble(Video1.Position.Seconds);
6     dummyStory.Begin();
7 }

```

See on suhteliselt ebamugav viis väga lihtsa asja tegemiseks. Põhjus, miks ei ole muutujat võimalik siduda `MediaElement.Position` atribuudi külge, on selle atribuudi muutumise kiirus, nimelt muutuja `Position` väärtus uueneb umbes iga 100 nanosekundi järel, mis tekitaks väga palju (ebavajalikke) uuendusi [26].

3.3 Täisekraan ning selle piirangud

Võimalus rakendus tööle panna terve ekraani ulatuses on meediarakenduste juures väga oluline, sest parim kasutajaelamus saadakse videot vaadates võimalikult suurelt ja võimalikult hea kvaliteediga.

3.3.1 Rakendus täisekraanile

Rakenduse täisekraani tööd märgib muutuja `Application.Current.Host.Content.IsFullScreen`. Täisekraani kasutamiseks (Näide 31) tuleb lisada import teegile `System.Windows.Interop`.

Näide 31

```

1 Content content = Application.Current.Host.Content;
2 content.IsFullScreen = true;

```

Kuid niisamuti nagu täisekraan avardab rakendusele võimalusi, võib see olla ka ohtlik – nii võiks pahalane luua näiteks kloni operatsioonisüsteemi ekraanist ning meelitada kasutajat

sinna sisestama hinnalist informatsiooni. Selliseid olukordi ette nähes on Silverlighti täisekraanile ülemineku juures mitmeid piiranguid.

3.3.2 Piirangud tööle täisekraanil

Silverlight rakendab järgmisi piiranguid [27]:

- Täisekraanile ei saa minna rakenduse laadimisel – koodile Näites 32 ei anta viga, vaid seda ignoreeritakse.
- Kui rakendus töötab täisekraanil, ei ole võimalik püüda klahvivajutusi.
- Sisenedes töösse täisekraanil või sealt väljudes suureneb märgatavalt rakenduse tööpind, kuid objekti suurused ja muud parameetrid Silverlight rakenduse sees ei muutu automaatselt. Seda tuleb teha käsitsi.

Näide 32

```
1 void Page_Loaded(object sender, RoutedEventArgs e) {
2     Content content = Application.Current.Host.Content;
3     content.IsFullScreen = true;
4 }
```

Näites 33 on välja arvatud suurendusfaktor (read 3, 4) ning seejärel on suurenduse läbiviimiseks kasutatud *ScaleTransform* tüüpi transformatsiooni. *ScaleTransform* transformatsiooni eeliseks on, et kasutades seda paigutushalduri muutmiseks suurendatakse/vähendatakse selle halduri kõiki alluvaid sama faktori võrra – seega ei pea ükshaaval muutma kõiki kasutajaliidese elemente.

Näide 33

```
1 void fullscreenButton_Click(object sender, RoutedEventArgs
   e) {
2     content.IsFullScreen = true;
3     double heightRatio = content.ActualHeight / this.Height;
4     double widthRatio = content.ActualWidth / this.Width;
5     ScaleTransform scale = new ScaleTransform();
6     scale.ScaleX = widthRatio;
7     scale.ScaleY = heightRatio;
8     this.RenderTransform = scale;
9 }
```

4 Publitseerimine

4.1 Silverlight komponendi lisamine veebilehele

Silverlight rakenduse lisamiseks veebilehele on kaks võimalust – kasutades HTML märgist *object* [28] või *Silverlight.js* abiskriptis olevaid JavaScript funktsioone *CreateSilverlight()* ja *CreateSilverlightEx()* [29]. Nii *object* märgise kui ka JavaScript funktsioonide korral defineeritakse parameetrid, mille järgi Silverlight veebilehitseja lisa kuvab rakendust.

4.1.1 Objekt märgendi kasutamine

Näite 34 esimene rida viitab antud objekti andmete asukohale (Silverlight veebilehitseja lisa), teine rida defineerib andmete tüüpi (hetkel Silverlight 2 Beta 1 tüüpi rakendus) ning kolmas rida viitab XAML failile, mida Silverlight kuvab. Juhul kui kliendil ei ole Silverlight installeeritud, näidatakse standardset „*Get Microsoft Silverlight*“ pilti koos allalaadimisviitega, mille kood on toodud ridades viis kuni kaheksa. Silverlight 2 Beta 1 puhul on see tekst alati ingliskeelne. Silverlight 1.0 paigaldaja on kättesaadav 33 keeles. Hetkel nende seas eesti keelt ei ole.

Näide 34

```
1 <object data="data:application/x-silverlight,"
2         type="application/x-silverlight-2-b1">
3   <param name="source" value="Veebileht.xap"/>
4
5   <a href="http://go.microsoft.com/fwlink/?LinkId=108182">
6     
9   </a>
10 </object>
```

4.1.2 JavaScript funktsiooni kasutamine

JavaScript funktsiooni kasutamiseks Silverlight rakenduse lisamiseks veebilehele tuleb esmalt projekti lisada fail *Silverlight.js* ning viidata sellele HTML-faili päises. Fail *Silverlight.js* asub kataloogis *%ProgramFiles%\Microsoft SDKs\Silverlight\v2.0\Tools*. Rakenduse initsialiseerimiseks on kaks funktsiooni: *CreateSilverlight()* ning *CreateSilverlightEx()*.

`CreateSilverlightEx()` funktsioonis edastatakse rakenduse instantsi loomiseks vajalikud parameetrid kasutades JSON märgendust. `CreateSilverlight()` on tavaline JavaScript funktsioon ning saab samad väärtused ette parameetritena. Nende erinevus seisneb vaid koodi loetavuses, funktsionaalsus on sama.

Näide 35

```
1 function createSilverlightEx() {
2     Silverlight.createObjectEx({
3         source: "Veebileht.xap",
4         parentElement:
5             document.getElementById("SilverlightHost"),
6         properties: {
7             width: '600',
8             height: '200',
9             version: '2.0'
10        },
11        events: {}
12    });
13 }
```

Näites 35 on toodud minimaalne vajalik kood Silverlight objekti lisamiseks veebilehele, mis töötab IE 7-ga. Erinevalt märgisest *object* Näites 34, kus Silverlight rakendus lisati selle märgise kohale, tuleb JavaScript funktsioone kasutades määrata ka kuvamiskoht (4. rida).

Rakenduse laitmatuks tööks kõigis toetatavates veebilehitsejates tuleks määrata *parentElement* atribuudi järel ka rakenduse *id*, mis on vajalik erinevate Silverlight rakenduste eristamiseks ühel veebilehel. Dokumentatsioon soovib lisada ka JavaScript funktsioonid, mis haldavad sündmuseid *Error* ja *Resize*. Tähele tuleb panna, et Visual Studio 2008 poolt automaatselt Silverlight projekti testimiseks loodud veebilehel on sündmuse *OnError* haldaja lisatud, kuid juures on kommentaar, et see on vaid testimiseks ning tuleks enne publitseerimist eemaldada.

Versiooninumbri atribuudi kohta on dokumentatsioonis öeldud, et see ei ole vajalik. Märgist *object* kasutades saab Silverlight veebilehitseja lisa vajaliku versiooni teada atribuudist *type*. JavaScript funktsioone kasutades tuleb versioon siiski määrata, vastasel juhul kuvatakse tühi leht.

JavaScript funktsiooni ning märgise *object* tulemus on Silverlight veebilehitseja lisa jaoks sama. Otsus kumb valida, sõltub väga palju olukorrast. Märgist *object* on kindlasti lihtsam kasutada, piisab vaid ühest märgisest ja skriptifaile ei ole vaja lisada. Seega on märgise

object puhul oluliselt lihtsam pakkuda rakenduse *embedded* linke, selle kuvamiseks mõnel teisel veebilehel. Miinuseks on aga see, et vähemasti IE blokeerib *object* märgisega lisatud Silverlight rakenduse kui potentsiaalselt ohtliku ActiveX komponendi.

Teisalt võib veebilehitsejas olla ka JavaScript keelatud ning Silverlight rakendust ei laetaks ka sel juhul. Tänapäevases Web 2.0/AJAX ajastus on see aga ebatõenäoline. Samas võib vaid 6.4KB suurune Silverlight.js suure kasutajakonna juures andmemahatu oluliselt paisutada.

4.2 Initsialiseerimisparameetrid

Silverlight rakendusele saab edasi anda ka kasutaja defineeritud initsialiseerimisparameetreid [30], JavaScript funktsioonide puhul tuleks parameetrid lisada komadega eraldatult *events: {}* järele, nagu on näidatud Näites 36.

Näide 36

```
1 ...
2 events: {},
3 „muutuja1=väärtus, muutuja2=väärtus“
4 ...
```

Märgist *object* kasutades tuleb lisada parameeter nimega *initParams alphanumeric* väärtusega, mida on tehtud Näites 37.

Näide 37

```
1 <param name="initParams" value="muutuja1=Leht,muutuja2=12" />
```

Initsialiseerimisparameetrite väärtuseid saab lugeda vaid *App.cs Startup* sündmuse kuularis ning üle kirjutada neid ei saa.

Näide 38

```
1 using System.Windows.Browser;
2 private void Application_Startup(object sender, StartupEventArgs e)
3 {
4     HtmlPage.Window.Alert(e.InitParams["muutuja"].ToString());
5     HtmlPage.Window.Alert(e.InitParams["muutuja1"].ToString());
6     this.RootVisual = new Page();
7 }
```

Näite 38 esimeses reas on lisatud viide *System.Windows.Browser* teegile, mis sisaldab meetodeid HTML-lehe dokumendiobjektmudeli lugemiseks ning kirjutamiseks. Meetodis *Application_Startup* (rida 4 ja 5) luuakse tüüpilised JavaScript aknad, mis kuvavad Näites 36 defineeritud initsialiseerimisparameetrite väärtused – vastavalt *Leht* ja *12*. Kuues rida initsialiseerib Silverlight rakenduse, selle juurleheküljeks saab Page, millele vastab *Page.xaml* fail.

4.3 Veebi laadimine

Veebiserverisse tuleb laadida kaks faili – veebilehekülg, kuhu on Silverlight rakendus paigutatud ning rakenduse *.xap* fail, mis vaikimisi paigutatakse projekti *ClientBin* kataloogi.

Nõuded veebiserverile puuduvad, kuna kogu rakendus laetakse kliendi masinasse, milles sellega töötab Silverlight veebilehitseja lisa. Kui server suudab esitada HTML lehte, suudab ta kuvada ka Silverlight objekti.

4.4 Silverlight Streaming

Silverlight Streaming by Windows Live [31-32] on teenus Silverlight rakenduste täielikuks või osaliseks hostimiseks. See annab igapäevase võimaluse luua meediarikkaid rakendusi – kuna kuus on tasuta võimalik hea ribalaiusega liigutada 5TB andmeid ning kontrol on rakenduste jaoks kuni 10GB ruumi. Tulevikus lisatakse tasuta teenust kasutavatele videotele kontekstitundlike reklaame, juurde tuleb ka reklaamideta tasuline versioon, milles on lisaks ka rohkem kettaruumi ning piiramatut andmesidemaht. Silverlight reklaamid hakkavad töötama analoogselt *Google AdSense* programmiga.

Silverlight Streaming parameetrid on järgmised:

- ruumi 10GB;
- 5TB andmesidemahtu;
- video pikkus kuni 10 min;
- ühe video maksimumsuurus 105MB.

4.4.1 Rakenduse ettevalmistamine

Tavaliselt piisab Silverlight rakenduse publitseerimiseks *.xap* faili veebi laadimisest. Silverlight Streaming kasutamiseks tuleb lisaks luua fail *manifest.xml*, kus on kirjas rakenduse initsialiseerimisparameetrid. Seejärel tuleb loodud fail *manifest.xml* ning *.xap* pakend kokku pakkida.

4.4.1.1 *manifest.xml*

Failis, mille nimi peab kindlasti olema *manifest.xml*, on kirjas samad rakenduse initsialiseerimisparameetrid, mis tavaliselt asuvad *CreateSilverlight()* funktsioonis või märgise *object* sees. Teenust *Silverlight Streaming* kasutades tuleks need tõsta faili *manifest.xml*. Minimaalne fail *manifest.xml* on esitatud Näites 39.

Näide 39

```
1 <SilverlightApp>
2   <source>Veebileht.xap</source>
3   <version>2.0</version>
4   <width>100%</width>
5   <height>100%</height>
6 </SilverlightApp>
```

Failid *manifest.xml* ning *Veebileht.xap* tuleb seejärel pakkida *.zip* faili.

4.4.2 Kasutamine

Silverlight Streaming teenuse kasutamiseks tuleb sisse logida <http://streaming.live.com> lehele, kasutades *Windows Live* ID-d. Konto loomiseks tuleb avanenud leheküljel vajutada *Get it free* linki, mille järel saab sisselogitud *Live ID* siduda uue *Silverlight Streaming* kontoga – luuakse id ning võti, mis on vajalikud antud konto rakenduste kasutamiseks veebilehtedel.

Uue rakenduse lisamiseks tuleb *Silverlight Streaming* administreerimismenüüst valida *Manage Applications-> Upload Applications*, määrata rakendusele nimi ning see üles laadida. Faili laadimise lõppedes kontrollitakse üle pakendis olev fail *manifest.xml*. Juhul kui see on korrektne, kuvatakse automaatselt genereeritud HTML-kood, mis võimaldab lihtsasti seda konkreetset Silverlight rakendust veebilehele lisada. Kasutades selleks kas *Silverlight Streaming Control* või märgist *iFrame* nagu on tehtud Näidetes 40 ja 41. Juhul, kui fail *manifest.xml* puudub või selles on vigu, võimaldab Silverlight Streaming selle lühikese veebivormi täitmisega uuesti luua. Üles võib laadida ka ainult *.xap* pakendi, sel juhul on samuti võimalik *manifest.xml* veebis teha.

4.4.2.1 iFrame

Lihtsaim viis Silverlight Streaming teenuse abil rakenduse kasutamiseks on märgis *iFrame*. Näites 40 luuakse *iFrame*, mille sisuks saab Silverlight rakendus. Näite 40 teises reas on *Silverlight Streaming* konto ID (60908) ning Veebileht on selle konto all oleva rakenduse nimi.

Näide 40

```
1 <iframe
2   src="http://silverlight.services.live.com/invoke/60908/Vee
   bileht/iframe.html"
3   scrolling="no" frameborder="0" style="width:100%;
4   height:100%">
5 </iframe>
```

4.4.2.2 Silverlight Streaming Control

Komponendi *Silverlight Streaming Control* kasutamiseks tuleb Silverlight rakendust sisaldavat veebilehekülge rohkem muuta. Näite 41 esimeses reas defineeritakse uus nimeruum, mis võimaldab hiljem 4. reas kasutada märgendit *devlive*. Märgis *devlive* sisaldab viidet rakenduse poolt kasutatavale Silverlight versioonile ning rakendusele endale. Viide rakendusele koosneb kahes osast - Silverlight Streaming konto ID-st ning selle konto all paikneva rakenduse nimest. Lisada tuleb ka viited kahele JavaScript failile – read 2 ja 3.

Näide 41

```
1 <html xmlns:devlive="http://dev.live.com">
2   <script type="text/javascript"
   src="https://controls.services.live.com/scripts/base/v0.3/
   live.js"></script>
3   <script type="text/javascript"
   src="https://controls.services.live.com/scripts/base/v0.3/
   controls.js"></script>
4   <devlive:slscontrol
5     silverlightVersion="2.0"
6     src="/60908/Veebileht/">
7 </devlive:slscontrol>
```

4.4.3 Rakenduse osaline hostimine

Silverlight Streaming teenust võib kasutada ka rakenduse osaliseks paigutamiseks – näiteks rakendus asub ettevõtte ja selle videod *Silverlight Streaming* serveris. Videoid võib seejärel kasutada *Silverlight* rakenduses või lisada need veebilehele, kasutades märgist *iFrame*. Sel juhul luuakse video vaatamiseks ka lihtne videopleier – hea lahendus tavalisele arvutikasutajale, kes soovib oma videoid veebis kasutada ent ei taha kaotada kvaliteedis, millele teevad piiranguid sellised videopangad nagu youtube.com, toru jms.

Video või muu ressursi kasutamiseks *Silverlight Streaming* keskkonnas on m:

- streaming:/kontoId /rakendus/failinimi
- key=streaming:/kontoId/rakendus /failinimi

Need märgendid ei viita aga otse failile, vaid *Silverlight Streaming* skriptile, mis teisendab need http:// lingiks, mille saab seada *MediaElement source* väärtuseks. Failidele viidatakse märgise *streaming* abil, sest genereeritavad „otselingid“ kehtivad vaid mõne tunni, julgustamaks *Silverlight* rakenduste arendamist, mitte ainult voogesitusteenuse kasutamist.

Antud märgised tuleb rakendusele edasi anda initsialiseerimisparameetrite kaudu ning seejärel omistada näiteks *MediaElement* allikaks.

5 Silverlight plussid ja miinused

Tehnoloogia Silverlight suurimaks plussiks on sõltumatus operatsioonisüsteemist - võimalus arendada veebirakendusi, mis töötavad ühtmoodi Windows, Linux ja Mac platvormil. Seejuures saab arendamisel kasutada samu komponente, mida .NET arenduses juba aastaid kasutatud on, mis tähendab, et Silverlight rakenduste loomise alustamine on .NET arendajatele vaid väikese sammu tegemine, sama lihtsasti peaks toimuma ka vastupidine liikumine.

See, mis annab Silverlight platvormile võimekuse on .NET raamistik, mis on võimalusterohke ja kiire. Veebiaadressil bubblemark.com on võimalik katsetada lihtsat 2D animatsiooni erinevatel platvormidel (JavaScript, Flash (Flex), Java, Silverlight jt). Vaatamata sellele, et see test näitab vaid raamistiku arvutusjõudlust ning animatsiooni kiirust on tulemused raamisagedustest kõnekad: JavaScript ~55 fps, Flash ~55 fps, Silverlight 1.0 ~60 fps, Java ~180 fps ning Silverlight 2 Beta 1 ~650 fps.

Silverlight sisaldab võimalusi koostööks veebilehega, mille sisse ta paigutatud on, mis tähendab, et piir RIA (*Rich Internet Application*) ning tüüpilise Web 2.0 veebirakenduse vahel hägustub – vajadusel võib kasutada C# koodi, et modifitseerida veebilehe dokumendiobjekti mudelit.

Koostöö teiste teenustega ning töö andmetega. Veebirakendus võib koosneda väga mitmetest osadest – Flash kasutajaliidese jaoks, PHP mis vahendab andmeid Flash rakenduse ja veebiserveri vahel jne. Silverlight täidab aga eelnevad nõudmised üksi, lisaks suudab ta suhelda näiteks REST teenustega.

Veebis asuvaid videoid seostatakse tavaliselt Flash tehnoloogiaga, ent siingi on Silverlight platvormil eeliseid, mis on pannud mitmed suured kompaniid sellele üle minema. Flash video kasutab videoformaati, mida mujal ei kasutata. Silverlight toetab aga VC-1 formaati, mis on laialt levinud standard ning lisaks on olemas tasuta teegid, mis võimaldavad mitte-toetatud formaatide ümberkodeerimist. Ümberkodeerimiseks võib kasutada ka rakendust, Windows Movie Marker, mis on olemas praktiliselt iga Windows platvormiga arvutis.

Arendaja seisukohast on Silverlight väga hea, võrreldes tehnoloogiatega nagu Flash on näha, et on püütud lahendada paljusid selles esinevaid kitsaskohti. Kuivõrd versioon Silverlight 2 Beta 1 tuli välja vaid 3 kuud tagasi, ei ole erilisi miinuseid ega kitsaskohti esilekerkinud. Üheks suureks (ning potentsiaalselt saatuslikuks) miinuseks on

arendusvahendite (eriti disainitööriistade) Macintosh platvormi toe puudumine. Etteheidetakse ka seda, et Silverlight rakendustel ei ole ligipääsu veebikaamerale ning mikrofonile.

6 Näidisrakendus

6.1 Kirjeldus

Antud töö üks eesmärk oli luua rakendus, mis tutvustaks tehnoloogia Silverlight võimalusi meediarakenduste arendamisel. Loodud näidisrakenduse sisuks on videod (ka HD), mida on võimalik erinevatel viisidel tarbida. Rakendus koosneb kahest vaatest – „Vaata“ ning „Loo“. Rakenduse lähtekood asub CD-1 Lisas 5 ja on kättesaadav aadressilt <http://www.ut.ee/~texmex5>.

Vaikimisi vaates „Vaata“ saab videoloendist valida video ning seda vaadata, kas tavasuurusel või täisekraanil. Vaikimisi ei toeta Silverlight video aegluubis näitamist, näidisrakenduses aga on videoid võimalik vaadata tavakiirusest kuni kümme korda aeglasemalt.

Vaates „Loo“ saab videoloendist lisada videoid töölauale ning lisatud käsitletakse kui üht tervikut. Vaadates lisatud videoid täisekraanil, mängitakse kõik videod järjest vastavalt horisontaalsele paigutusele. Samas ei pea vaatama kõiki videoid terviklikult, vaid iga video juures on „käärid“, millega saab video algusest ja lõpust kaadreid „lõigata“. Töölauale võib lisada mitu videot ning üht videot ka mitu korda – seega on teostatav lihtsakoeline videotöötlus.

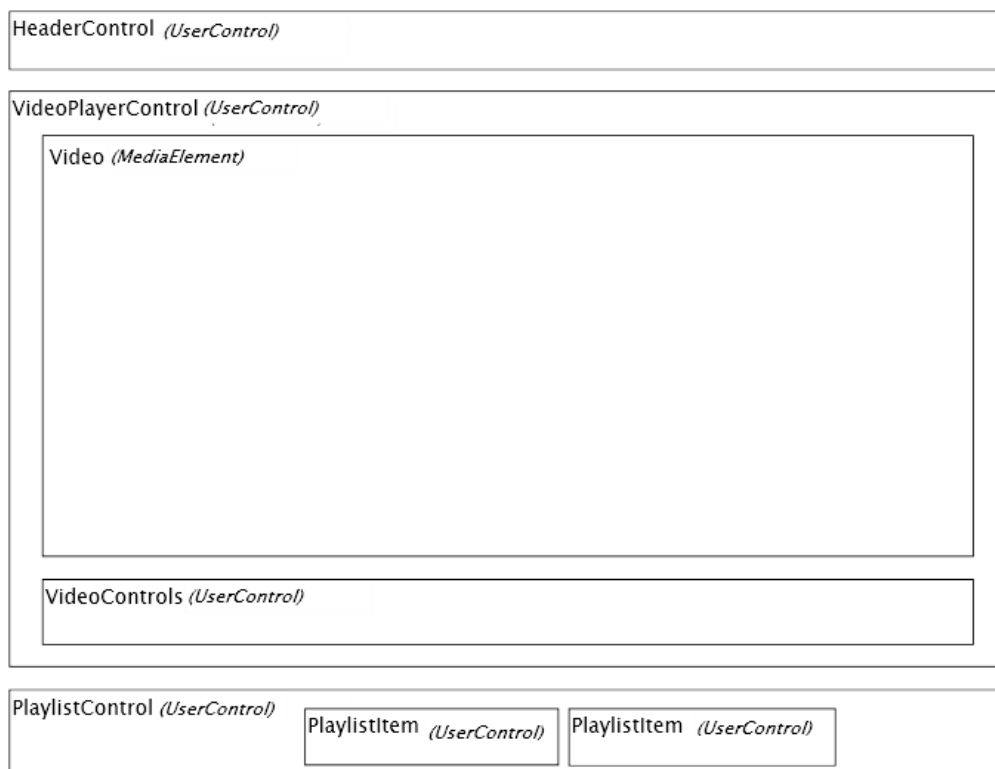
Rakendus on lisatud HTML-lehele, kasutades *JavaScript* funktsiooni `CreateSilverlight()`. *JavaScript* funktsiooni kasutamise eelis märgise *object* ees on see, et märgise *object* puhul IE vaikimisi blokeerib Silverlight rakenduse kui potentsiaalselt ohtliku ActiveX objekti, *JavaScript* funktsioone kasutades seda aga ei tehta.

Rakendus hõlmab kogu HTML-lehe nii, et kerimisribasid ei teki – objektide suurused Silverlight rakenduses sõltuvad reaalistest lehe mõõtmetest, mille sees rakendus asub.

6.2 Rakenduse ülesehitus

6.2.1 Page

Rakenduse juurelement on fail *Page.xaml*, milles on kolmerealine *Grid* tüüpi paigutushaldur. Paigutushalduri esimeses reas asub kasutajakontroll *HeaderControl*, teises *VideoPlayerControl* ning kolmandas *PlaylistControl*, millesse on omakorda lisatud hulk *PlaylistItemControl* kasutajaelemente. Paigutushalduri ridade kõrgused on vastavalt 10%, 80% ning 10% Silverlight rakenduse kõrgusest. Joonisel 15 on näha eelnevalt tutvustatud kasutajaelementide paigutust rakenduses.



Joonis 15. Faili *Page.xaml* sisu.

6.2.2 HeaderControl

HeaderControl sisaldab logo ning viiteid „Vaata“ ja „Loo“, mille abil toimub navigeerimine rakenduse kahe vaate vahel. Vajutades viidet „Loo“ lisandub *PlaylistItem* tüüpi kasutajakontrollidele „+“ märk, mis võimaldab neid lisada töölauale. Töölaud lisatakse juurelemendi teise ritta, seal algselt asetsenud *VideoPlayerControl* eemaldatakse seniks kuni ollakse vaates „Loo“.

6.2.3 VideoPlayerControl

VideoPlayerControl kasutajaelemendi puhul on kõige enam kasutatud Silverlight erinevaid võimalusi kasutajaliidese omanäoliseks tegemiseks. Paigutushaldur on *Border* tüüpi, mis annab videole ümarate nurkade ning *gradiendiga* värvitud raami. Video esitluse kontrollimiseks mõeldud „pult“ koosneb paljudest nuppudest ja kerimisribadest (*Slider*). Puldi näitamiseks ja peitmiseks on lisatud animatsioonid. Nuppude jaoks on defineeritud stiil – näiteks on ühest tavalisest kandilisest nupust on saadud teokujuline nupp.

6.2.4 PlaylistControl

PlaylistControl on lihtne kasutajaelement *PlaylistItem* elementide jaoks. Ta sisaldab animatsioone nende elementide kuvamiseks juhul kui neid on rohkem kui ekraanile mahub.

6.2.5 PlaylistItem

PlaylistItem esindab üht videot videonimekirjas. Ülesehituselt on see kasutajakontroll lihtne, sisaldades vaid videot ning pluss-märki, mis näitab, et videot saab töölauale lisada. Programmikoodis on meetodit *PlaylistItem*i lisamiseks töölauale ning meetodid videote ühiseks mängimiseks ning kuvamiseks.

Kokkuvõte

Käesolevas bakalaureusetöös uuritakse, kuidas kasutada MS Silverlight tehnoloogiat interaktiivsete ja meediat sisaldavate veebirakenduste programmeerimiseks. Töö eesmärgiks oli tutvustada Silverlight tehnoloogiat ja selgitada välja selle tugevad ja nõrgad küljed.

Töö esimeses peatükis tutvustatakse Silverlight veebilehitseja lisa arhitektuuri, arenduseks vajalikke töövahendeid ning luuakse esimene Silverlight rakendus, mille põhjal uuritakse tüüpilise rakenduse ülesehitust. Selgitatakse ka arendusvahendite Expression Blend ning Visual Studio omavahelist koostööd ning kuidas see mõjutab disaineri ja arendaja tööd. Konkreetse ülesande jaoks õige arendusvahendi valimine on teema, millel peatutakse töö jooksul korduvalt.

Teises peatükis antakse ülevaade Silverlight rakenduse kasutajaliidese loomisest ning selle isikupärastamisest. Käsitletakse transformatsioone ja animatsioone ja nende loomist. Selgub, et nende loomiseks on kindlasti paremaks tööriistaks rakendus Expression Blend, sest see võimaldab mõningate tegevuste nagu näiteks animatsioonide ümberpööramine automatiseerimist. Leitakse lahendus olukorrale, kus mitmele objektile on vaja lisada ühesugune animatsioon.

Teises peatükis tutvustatakse ning võrreldakse ka erinevaid paigutushaldureid, mis pakuvad mitmekülgeid võimalusi rakenduse sisu kuvamiseks. Jõutakse järelduseni, et parim paigutushaldur on *Grid* ning ülejäänud on pigem sobilikud sisu presenteerimiseks.

Peatüki lõpus vaadeldakse stiile, malle ning kasutajaelemente, mis kõik aitavad vältida koodiliiasust ning seeläbi parandada koodi hallatavust. Kasutajaelementide juures leitakse, et nende abil korduvate elemendigruppide kokkukogumine on hädavajalik vähegi keerukamate rakenduste juures.

Kolmandas peatükis peatutakse pikemalt objektidel *MediaElement* ning videofailide kasutamisel rakendustes. Lahendatakse esmapilgul lihtne ülesanne – video mängimisel saada teada tema positsioon ning kuvada see ajateljel. Selgitatakse, miks see ei ole lahendatav kasutades andmete sidumist (*DataBind*). Edasi käsitletakse ka Silverlight rakenduse tööd täisekraanil ning selle piiranguid.

Neljas peatükk keskendub sellele, mida teha siis, kui rakendus on avaldamiseks valmis. Antakse ülevaade ning võrreldakse erinevaid võimalusi Silverlight rakenduse lisamiseks

veebilehele. Selgitatakse nõudeid veebiserverile ning tutvustatakse teenust *Silverlight Streaming*, mis on tasuta teenus Silverlight rakenduste hostimiseks.

Viiendas peatükis tuuakse välja Silverlight tehnoloogia tugevad ja nõrgad küljed. Tehnoloogia Silverlight tugevuseks on kiire ning laialdaste võimalustega .NET raamistik, puuduseks on asjaolu, et puudub toetus veebikaamerale ja mikrofonile. Miinusena nähakse ka arendusvahendite puudumist Macintosh platvormile.

Töö käigus valmis näidisrakendus, mida kirjeldatakse kuuendas peatükis.

Developing media applications using MS Silverlight

Bachelor Thesis

Jaana Metsamaa

Summary

The purpose of this Bachelor Thesis is to introduce and study MS Silverlight and to establish strong and weak features of this technology.

An overview of Silverlight plug-in architecture, file structure of a typical Silverlight application and tools needed to get started with Silverlight development is presented in the first chapter.

The second chapter concentrates on developing and customizing user interfaces using transformations, animations, styles and themes. In this chapter the different possibilities for creating transformations and animations are introduced. It is concluded that the best way to work with animations is to use Expression Blend. An overview of styles, templates and *UserControl* creation is given. It is recommended to use these structures as it enhances readability and minimizes code duplication. Different layout controls are compared and it is concluded that *Grid* is the best in positioning objects and the rest layout controls find a better use in simple content presentation.

In the third chapter *MediaElement* object and its different usages are covered as *MediaElement* plays an important part in the sample application. Related to videos Silverlight full screen support and its limitations are discussed briefly.

The fourth chapter focuses on different possibilities for Silverlight application initialization and publication. After comparing using JavaScript and using the object tag for application initialization, it is concluded that the choice between them is tightly tied to a situation, but in not very big applications the user experience is highest using the JavaScript function. Demands for server infrastructure are also explained and the usage of Silverlight Streaming service is covered.

In the fifth chapter the strong and weak features of Silverlight are discussed. It is concluded that the strong features of Silverlight technology are fast .NET framework and

its many features. The weak features of Silverlight appear the facts that currently there is no webcam or microphone support. Lack of design applications is also seen as a possibly fatal flaw.

The last chapter gives an overview of the example application developed during the paper and explains how different aspects introduced in the paper are used in the application.

Kirjanduse loetelu

- [1] <http://silverlightguy.com/2007/09/22/silverlight-introduction/> (28.05.2008)

- [2] Silverlight Overview
<http://microsoft.com/Silverlight> (28.05.2008)

- [3] Silverlight Architecture Overview
[http://msdn.microsoft.com/en-us/library/bb404713\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/bb404713(VS.95).aspx) (28.05.2008)

- [4] Silverlight Architecture Overview
<http://msdn.microsoft.com/en-us/library/bb428859.aspx> (28.05.2008)

- [5] Silverlight 2 Developer Poster
<http://blogs.msdn.com/brada/archive/2008/03/16/silverlight-2-developer-poster.aspx>
(28.05.2008)

- [6] Xaml Overview
<http://msdn.microsoft.com/en-us/library/ms752059.aspx> (28.05.2008)

- [7] <http://en.wikipedia.org/wiki/XAML> (28.05.2008)

- [8] Application class
[http://msdn.microsoft.com/en-us/library/system.windows.application\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/system.windows.application(VS.95).aspx)
(28.05.2008)

- [9] Anatomy of an .XAP file
<http://blogs.msdn.com/katriend/archive/2008/03/16/silverlight-2-structure-of-the-new-xap-file-silverlight-packaged-application.aspx> (28.05.2008)

- [10] App.xaml events
<http://silverlightuk.blogspot.com/2008/03/silverlight-2-appxaml-events.html>
(28.05.2008)

- [11] Packaging and Application Startup in Silverlight 2 Beta 1
http://community.irritatedvowel.com/blogs/pete_browns_blog/archive/2008/03/05/Xap_2100_-App_2100_-Pow_2100_-Packaging-and-Application-Startup-in-Silverlight-2-Beta-1-_2D00_-Part-2.aspx (28.05.2008)

- [12] Transforms Overview
<http://msdn.microsoft.com/en-us/library/bb979976.aspx> (28.05.2008)
- [13] Animations Overview
[http://msdn.microsoft.com/en-us/library/cc189019\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189019(VS.95).aspx) (28.05.2008)
- [14] Key-Frame Animations (Silverlight 2)
[http://msdn.microsoft.com/en-us/library/cc189038\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189038(VS.95).aspx) (28.05.2008)
- [15] Working with Animations Programmatically (Silverlight 2)
[http://msdn.microsoft.com/en-us/library/cc189069\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189069(VS.95).aspx) (28.05.2008)
- [16] Object Positioning and Layout (Silverlight 2)
[http://msdn.microsoft.com/en-us/library/cc189087\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189087(VS.95).aspx) (28.05.2008)
- [17] Layout controls in Silverlight 2 Beta 1
<http://www.silverlightshow.net/items/Layout-controls-in-Silverlight-2-Beta-1.aspx>
(28.05.2008)
- [18] Using Layout management
<http://weblogs.asp.net/scottgu/pages/silverlight-tutorial-part-2-using-layout-management.aspx> (28.05.2008)
- [19] Styling and Templating overview
[http://msdn.microsoft.com/en-us/library/cc189093\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189093(VS.95).aspx) (28.05.2008)
- [20] Using Control Templates to Customize a Controls Look and Feel
<http://weblogs.asp.net/scottgu/pages/silverlight-tutorial-part-7-using-control-templates-to-customize-a-control-s-look-and-feel.aspx> (28.05.2008)
- [21] Using User Controls to Implement Master/Detail Scenarios
<http://weblogs.asp.net/scottgu/pages/silverlight-tutorial-part-6-using-user-controls-to-implement-master-detail-scenarios.aspx> (28.05.2008)
- [22] Audio and Video Overview
[http://msdn.microsoft.com/en-us/library/bb980141\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/bb980141(VS.95).aspx) (28.05.2008)
- [23] Supported Media Formats and Protocols
[http://msdn.microsoft.com/en-us/library/bb980063\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/bb980063(VS.95).aspx) (28.05.2008)
- [24] MediaElement States
[http://msdn.microsoft.com/en-us/library/bb980165\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/bb980165(VS.95).aspx) (28.05.2008)
- [25] Markers
[http://msdn.microsoft.com/en-us/library/bb979836\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/bb979836(VS.95).aspx) (28.05.2008)

- [26] Silverlight 2 Databinding to the MediaElement.Position Property
<http://www.silverlightshow.net/items/Silverlight-2-Databinding-to-the-MediaElement.Position-property.aspx> (28.05.2008)
- [27] Full-Screen Support
[http://msdn.microsoft.com/en-us/library/cc189023\(vs.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189023(vs.95).aspx) (28.05.2008)
- [28] Instantiating a Silverlight Plug-In (Silverlight 2)
[http://msdn.microsoft.com/en-us/library/cc189089\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189089(VS.95).aspx) (28.05.2008)
- [29] Using Silverlight.js
[http://msdn.microsoft.com/en-us/library/cc265155\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc265155(VS.95).aspx) (28.05.2008)
- [30] Specifying and Retrieving Custom Initialization Parameters (Silverlight 2)
[http://msdn.microsoft.com/en-us/library/cc189004\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/cc189004(VS.95).aspx) (28.05.2008)
- [31] Silverlight Streaming SDK
<http://msdn.microsoft.com/en-us/library/bb851621.aspx> (28.05.2008)
- [32] Silverlight Overview
<http://silverlight.net/getstarted/overview.aspx> (28.05.2008)

Lisad

Lisa 1 – Silverlighti toetavad platvormid ning veebilehitsejad

Opsüsteem*/brauser	Internet Explorer 6 SP2	Internet Explorer 7.0	FireFox 1.5.x.x	FireFox 2.0	Mac Safari 2.0.4
Windows XP SP2	+	+	+	+	
Windows Vista		+	+	+	
Windows 2000	kavas	kavas	kavas	kavas	
Mac OS 10.4.8 PPC			kavas	kavas	+
Mac OS 10.4.8 Intel			+	+	+

Lisa 2 – Vajalikud vahendid Silverlighti kasutamiseks

Silverlight 2 Beta 1 Runtime Windows ja Mac

<http://www.microsoft.com/silverlight/resources/installationFiles.aspx?v=2.0>

Silverlight 1 Runtime Windows ja Mac

<http://www.microsoft.com/silverlight/resources/installationFiles.aspx?v=1.0>

Silverlight Linuxile – Moonlight Projekt

<http://www.novell.com/products/desktop/moonlight.html>

Lisa 3 – Vajalikud vahendid Silverlight rakenduste arendamiseks

Microsoft Visual Studio 2008

<http://msdn2.microsoft.com/en-us/vstudio/bb964524.aspx>

Microsoft Silverlight Tools Beta 1 for Visual Studio 2008

<http://www.microsoft.com/downloads/details.aspx?FamilyId=E0BAE58E-9C0B-4090-A1DB-F134D9F095FD&displaylang=en>

Expression Blend 2.5 March 2008 Preview

<http://www.microsoft.com/expression/products/download.aspx?key=blend2dot5>

Expression Encoder

<http://www.microsoft.com/expression/products/Overview.aspx?key=encoder>

Expression Design

<http://www.microsoft.com/expression/products/Overview.aspx?key=design>

SilverlightDefaultStyleBrowser

<http://blogs.msdn.com/delay/archive/2008/03/22/improving-everyone-s-access-to-silverlight-2-s-generic-xaml-resources-silverlightdefaultstylebrowser-tool-and-source-code.aspx>

Lisa 4 - Silverlight 1.0 ning Silverlight 2 Beta 1 vördlus [32]

Features	Silverlight 1.0	Silverlight 2 Beta 1
2D Vector Animation/Graphics	●	●
AJAX Support	●	●
Cross-Browser (Firefox, IE, Safari)	●	●
Cross-Platform (Windows, Mac)	●	●
Framework Languages (Visual Basic, Visual C#, IronRuby, Ironpython)	-	●
HTML DOM Integration	●	●
HTTP Networking	●	●
Isolated Storage	-	●
JavaScript Support	●	●
JSON, REST, SOAP/WS-*, POX, and RSS Web Services (as well as support for Sockets)	-	●
Cross Domain Network Access	-	●
LINQ to Objects	-	●
Canvas Layout Support	●	●
StackPanel, Grid and Panel Layout Support	-	●
Managed Control Framework	-	●
Full suite of Controls (TextBox, RadioButton, Slider, Calendar, DatePicker, DataGrid, ListBox, and others)	-	●
Deep Zoom Technology	-	●
Managed HTML Bridge	-	●
Managed Exception Handling	-	●
Media – Content Protection	-	●
Media – 720P High Definition (HD) Video	●	●
Media – Audio/Video Support (VC-1, WMV, WMA, MP3)	●	●
Media – Image Support (JPG, PNG)	●	●
Media Markers	●	●
Rich Core Framework (e.g. Generics, collections)	-	●
Security Enforcement	-	●
Silverlight ASP.NET Controls (asp:media, asp:xaml)	●	●
Type Safety Verification	-	●
Windows Media Server Support	●	●
XAML Parser (based on WPF)	●	●
XMLReader/Writer	-	●

Lisa 5 - CD