TAMPEREEN
AMMATTIKORKEAKOULU

TUTKINTOTYÖRAPORTTI

ANALYSTE OYJ:N TUOTE-ESITYS WWW-SIVUILLA FLASHILLÄ TOTEUTETTUNA

Samuli Rajala

Tietojenkäsittelyn koulutusohjelma
Huhtikuu 2006
Työn ohjaaja: Paula Hietala

TAMPERE 2006

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t) Samuli Rajala

Koulutusohjelma(t) Tietojenkäsittely

Tutkintotyön nimi Analyste Oyj:n tuote-esitys WWW-sivuilla Flashillä toteutettuna

Työn valmistumis-kuukausi ja -vuosi Huhtikuu 2006

Työn ohjaaja Paula Hietala **Sivumäärä:** 69

TIIVISTELMÄ

Tutkintotyöni tavoitteena on luoda Analyste-ratkaisusta Flash-esitys Analyste Oyj:n WWW-sivuille. Esityksessä kerrotaan lyhyesti Analyste-ratkaisun toiminta-ajatus, eli sen tavoitteena on herättää kiinnostusta.

Tarkoitukseni on tehdä esityksestä teknisesti parempi kuin Analyste eOffice -ohjelmiston nykyinen Flash-esitys, joka on neljä vuotta vanha. Esitykselle asettamiani vaatimuksia ovat esimerkiksi navigoinnin helppous, latausajan minimoiminen, tekstien päivittämisen vaivattomuus ja esityksen upottaminen WWW-sivuille yleisten WWW-standardien mukaisesti.

Käyn tutkintotyössäni läpi käytettävyyttä yleisellä tasolla ja tarkemmin Flashin tiettyihin osa-alueisiin liittyen. Sovellan näitä ohjeita Analyste-ratkaisun esityksen luomisessa, jotta se noudattaisi mahdollisimman tarkasti käytettävyyden määritelmän mukaisia ohjeita.

Arvioin itse molemmat esitykset käytettävyyden kannalta ja lisäksi haastattelen kahta henkilöä esitystä kohden. Otan myös kantaa tekniseen toteutukseen. Huomioin Analyste-ratkaisun esitystä luodessani käytettävyyden lisäksi Analyste eOffice -esityksen arvioinnissa esille tulevat seikat.

Flash tukee hyvin suunnittelua ja käytettävyyttä. Flash-esityksen voi toteuttaa usealla eri tavalla, ja valittiinpa mikä tapa tahansa, kannattaa suunnittelussa ja toteutuksessa lähteä liikkeelle kohderyhmästä ja tavoitteista.

Avainsanat Flash käytettävyys esitykset tuotteet Analyste

TAMPERE
POLYTECHNIC

Author(s)	Samuli Rajala
Degree Programme(s)	Business Information Systems
Title	A product presentation in Flash for the website of Analyste Corporation
Month and year	April 2006
Supervisor	Paula Hietala

Pages: 69

ABSTRACT

The purpose of this thesis is to create a product presentation in Flash from Analyste solution. The presentation will be used on the Web and its goal is to attract attention, while it briefly shows the main idea of the solution.

My intention is to make a better presentation than Analyste eOffice software's Flash presentation, which was made four years ago. I aim to make navigating and updating texts as easy as possible, minimize the download time and embed the presentation into a web page while supporting common standards.

This thesis handles usability at first in general and then more specific regarding Flash. I will adapt these guidelines when I create the presentation of the Analyste solution.

I will evaluate usability on both presentations and in addition I will interview two persons per presentation. I will also give my opinion on the technical implementation. When creating the presentation of the Analyste solution I will take usability into consideration as well as the interviews.

Flash supports design and usability well. Flash presentation can be created in many ways. However it is wise to begin the design and implementation by defining target group and goals.

Keywords Flash usability presentations products Analyste

Sisällysluettelo

Sanastoa

ActionScript	Skriptauskieli, jolla voidaan vaikuttaa toimintoihin, ja jota voidaan liittää avainkehykseen, näyttämöllä olevaan painikkeeseen ja näyttämöllä olevaan elokuvaleikkeeseen.
aikajana	Kehyksistä koostuvalla lineaarisella aikajanalla esitetään tapahtumat tietyllä ajanjaksolla.
alpha	Alpha-kanavalla säädetään objektin läpinäkyvyyttä.
avainkehys	Kehys, jossa on määritelty animaatioon vaikuttavia toimintoja.
elokuvaleike	Symboli, joka on itsessään pieni Flash-esitys. Elokuvaleikkeelle voi määritellä interaktiivisia toimintoja, ääniä ja muita tapahtumia. Elokuvaleikkeen aikajana etenee itsenäisesti pääesityksen aikajanasta riippumatta.
esiintymä	Yhdestä symbolista voi olla useita esiintymiä näyttämöllä. Symboli yksilöidään esiintymänimellä.
esilataaja	Esilataaja eli preloader ilmoittaa elokuvan lataamisesta.
julkaiseminen	HTML-sivuun linkitettävä SWF-tiedosto tehdään julkaisu-toiminnolla Flashin lähdetiedostosta eli FLA-tiedostosta.
kehys	Ohjelman oletusarvoilla 12 kehystä vastaa ajassa yhtä sekuntia.
kirjasto	Flashin kirjastossa on kaikki käytettävät objektit.
kohtaus	Kohtauksella on oma aikajana ja niitä voi sijoittaa peräkkäin tarpeen mukaan.
käyttäjä	Esitystä katsova henkilö.
näyttämö	Näyttämö on alue, joka näkyy käyttäjälle. Objekteja voi sijoittaa myös näyttämön ulkopuolelle.
symboli	Symboli voi olla elokuvaleike, painike tai graafinen symboli.
taso	Aikajanalla voi olla useampia tasoja. Ohjelman oletusarvoilla ylempi taso näkyy aina alemman tason päällä.
toistopää	Kohta, jossa aikajanalla kulloinkin ollaan.
upottaminen	Flashin linkittäminen HTML-sivuun.

1 Johdanto

Flashiä voidaan käyttää useisiin erilaisiin käyttötarkoituksiin. Flashiä voidaan upottaa WWW-sivuille pienissä erissä, mutta sillä voidaan myös toteuttaa kokonaisia WWW-sivustoja. Lisäksi Flashiä voidaan käyttää mainosbannereissa, tuote-esityksissä, koulutuksissa tai puhtaasti viihteessä kuten peleissä.

Analyste Oyj:n WWW-sivuilla on ollut pitkään Analyste eOffice -ohjelmistosta Flashillä tehty esitys, jossa esitellään selkeästi ja havainnollisesti laskun eri käsittelyvaiheet. Olen tehnyt kyseisen esityksen vuosina 2002–2003. Tämän jälkeen esitykseen on vielä tehty joitain muutoksia, jotka ovat vaikuttaneet sen nykyiseen toimintaan.

Esityksen katsottuaan käyttäjällä on mahdollisuus jättää yhteystietonsa, jotta häneen voidaan ottaa yhteyttä ja kertoa tarkemmin ohjelmistosta. Flash-esityksen rooli on siis toimia mielenkiinnon herättäjänä. Myyjä esittelee ohjelmiston toimintaa tarkemmin erillisessä tapaamisessa.

Tutkintotyöni tavoitteena oli luoda Analyste-ratkaisusta Flash-esitys, joka olisi teknisesti parempi kuin Analyste eOffice -ohjelmistosta tehty esitys. Analyste sai tutkintotyöni myötä Analyste-ratkaisustaan Flash-esityksen WWW-sivuilleen. Käyn tutkintotyössäni läpi molempien esitysten toteutustavat.

Olen työskennellyt Analystessä useamman vuoden, joten aihepiiri oli minulle entuudestaan tuttu. Olen myös toteuttanut Flashillä erilaisia esityksiä eOfficen esityksen teon jälkeen ja oppinut uusia toteutustapoja. Myös Analyste-ratkaisun esitystä tehdessäni opin lisää uusia toteutustapoja.

Käyn luvussa 3 läpi Flashiin liittyviä käytettävyystekijöitä ja toteutustekniikoita. Näitä hyödyntäen arvioin ensin vanhan eOfficen esityksen. Tämän jälkeen toteutan uuden esityksen Analyste-ratkaisusta ottaen huomioon sekä luvussa 3 että eOfficen esityksen arvioinnissa ja haastatteluissa esille tulevat asiat.

Tutkintotyötä tehdessäni BasWare Oyj osti Analyste Oyj:n 31.1.2006. Yritysten välinen fuusio tapahtuu kesään 2006 mennessä. Tämä ei kuitenkaan vaikuttanut liiemmin tutkintotyöhöni. Siksi puhun tutkintotyössäni edelleen Analystestä.

Tästä tutkintotyöstä saa apua minkä tahansa Flash-ratkaisun suunnitteluun ja toteutukseen. Jotta toteutustavoista saisi kuitenkin mahdollisimman paljon irti, on Flashin peruskäyttö oltava hyvin hallussa, koska en työssäni kaikkia perustoimintoja selitä.

2 Analyste Oyj:n esittely

Verkkolaskutuksen yleistyminen on muuttanut taloushallintoa merkittävästi. Taloushallinnon sähköistymisen ja automatisoinnin myötä saadaan selviä säästöjä ja tehokkaammat toimintatavat. Näin työssä voidaan keskittyä yrityksen kannalta tuottavampiin asioihin. (Tehokkaan sähköisen taloushallinnon opas 2005: 3)

Analyste Oyj kehittää sähköisen taloushallinnon ohjelmistoja yritysten ja pankkien väliseen maksuliikenteeseen, hankintojen ja ostolaskujen sähköiseen käsittelyyn, kassa- ja rahoitussuunnitteluun sekä arkistointiin. (The Art of Cash Management 2005: 2)

Analyste on perustettu vuonna 1981. Sen ohjelmistot sopivat kaikille yrityksille ja yhteisöille koosta ja toimialasta riippumatta. Analyste on Suomessa markkinajohtaja, jonka tuotteita ja palveluita on käytössä yli 7 500 yhteisössä. Analysten henkilöstömäärä on noin 110. (The Art of Cash Management 2005: 2.)

Analysten tuoteperheeseen kuuluvat taulukossa 1 olevat ohjelmistot, jotka ovat yhteensopivia myös muiden taloushallinnon järjestelmien kanssa.

Taulukko 1 Analysten tuoteperhe (The Art of Cash Management 2005: 2)

Analyste-ratkaisu	integroitu taloushallinnon ratkaisu, joka kattaa maksuliikenteen, laskujen sähköisen käsittelyn, kassasuunnittelun ja arkistoinnin
Analyste Maksuliikenne	kaikki pankkien palvelut maksuliikenteeseen liittyen
Analyste eOffice	ostolaskujen ja hankintojen sähköinen käsittely ja arkistointi
Analyste In-House Banking	maksuliikenteen ja likviditeetin hallinnan tehostaminen
Analyste Finance	kassasuunnittelu ja rahoituksen sekä sijoitusten hallinta
Analyste Palvelukassa	palvelupisteessä tapahtuva myynti ja maksujen vastaanotto
Analyste iBanking	yrityksen taloushallinto Internetissä

Analystellä on Internet-sivujen lisäksi erittäin monipuoliset asiakkaille ja jälleenmyyjille tarkoitetut ekstranet-sivut. Kaikista ohjelmistoista on myös demoversiot, joita myyjät käyttävät ohjelmistoja esitellessään. Analystellä on myös asiakaslehti, Analyste Cash Manager, joka ilmestyy neljä kertaa vuodessa.

2.1 Analyste eOffice

Analyste eOffice on ostolaskujen, matkalaskujen ja muiden taloushallinnon aineistojen sähköiseen käsittelyyn ja arkistointiin tarkoitettu ohjelmisto. Se mahdollistaa paperi- ja verkkolaskujen vastaanottamisen, tarkastuksen, hyväksynnän, tiliöinnin sekä siirron maksatukseen. Laskut voidaan tarkastaa ja hyväksyä WWW-selaimella. eOfficen avulla voidaan myös arkistoida elektronisesti kaikki taloushallinnon tositteet. Tästä sähköisestä arkistosta on mahdollista etsiä haluamiaan tietoja WWW-selaimella. (Viitanen 2005: 1)

2.2 Analyste-ratkaisu

Analyste-ratkaisu on integroitu taloushallinnon kokonaisuus, joka kattaa maksuliikenteen, laskujen ja matkalaskujen sähköisen käsittelyn, kassasuunnittelun sekä arkistoinnin. Analyste-ratkaisu koostuu siis eri osa-alueista, jotka muodostavat saumattoman kokonaisuuden. (The Art of Cash Management 2005: 2)

Analyste-ratkaisu voidaan ottaa käyttöön vaihe kerrallaan yrityksen tilanteen mukaan. Jokainen osa-alue tuo mukanaan tiettyjä hyötyjä. (Tehokkaan sähköisen taloushallinnon opas 2005: 13)

Analyste-ratkaisun keskeisimmän osan muodostaa Analyste eOffice, jolla hoidetaan laskujen sähköinen käsittely kierrätyksineen. eOfficella tehdään myös matkasuunnitelmat ja käsitellään matkalaskut. Lisäksi eOfficella hoidetaan aineistojen arkistointi. Analyste Maksuliikenne -ohjelmistolla hoidetaan pankkiyhteydet eli maksamiset ja verkkolaskujen noudot pankista. Analyste Finance pitää samalla ajantasaista tietoa yrityksen rahatilanteesta.

3 Käytettävyys ja Flash

Flash mahdollistaa animaatioiden upottamisen WWW-sivuille tiedostokoon pysyessä pienenä. Flash mahdollistaa myös interaktiivisuuden tason lisäämisen ja sellaisten asettelujen käyttämisen, joiden toteuttaminen pelkällä HTML-kielellä olisi hyvin vaikeaa. (Bhangal 2001: 1)

Flashin näkyminen käyttäjän WWW-selaimessa edellyttää selaimelta Flash-laajennusta. Macromedian tilastojen mukaan syyskuussa 2005 vaadittava Flash Player oli jo 97,3 %:lla käyttäjistä (Macromedia Flash Player Statistics 2005).

Flash tukee suunnittelua ja käytettävyyttä hyvin, kun sitä käytetään viisaasti. Flashin avulla saadaan visuaalisesti näyttäviä esityksiä ja silti samalla voidaan parantaa käytettävyyttä. (Airgid & Reindel 2002: 18)

Käsittelen seuraavassa luvussa käytettävyyttä yleisesti, minkä jälkeen paneudun käytettävyyteen nimenomaan Flashin eri osa-alueissa.

3.1 Käytettävyys

Käytettävyyden voidaan ajatella olevan osa tuotteen laatua. Tuote voi olla esimerkiksi tietokoneohjelma, WWW-sivusto, matkapuhelin, vesihana tai ovi. (Sinkkonen, Kuoppala, Parkkinen & Vastamäki 2006: 11.) Käytettävyydestä on hieman erilaisia määritelmiä näkökulmasta riippuen. Jakob Nielsenin ja ISO 9241-11 -standardin määritelmät ovat kuitenkin kaksi yleisimmin käytettyä. Jakob Nielsen (1993: 24–25) määrittelee käytettävyyden osaksi tuotteen hyödyllisyyttä.

Hyödyllisyydessä on kyse siitä, päästäänkö tuotteen avulla haluttuun tavoitteeseen. Hyödyllisyys jakautuu kahteen osaan, toiminnallisuuteen ja käytettävyyteen. Toiminnallisuuden mukaan tuotteen pitää tehdä se, mikä on tarkoituskin. Käytettävyys puolestaan kertoo, kuinka hyvin käyttäjät voivat hyödyntää tätä toiminnallisuutta. Näin ollen käytettävyys on tuotteen ja käyttäjän välistä vuorovaikutusta. (Nielsen 1993: 24–25)

ISO 9241-11 -standardissa ”Näyttöpäätteillä tehtävän toimistotyön ergonomiset vaatimukset, Osa 11” käytettävyyden määritelmään on lisätty vielä käyttötilanne. Käytettävyys on siis riippuvainen nimenomaan tietystä tilanteesta. (Sinkkonen ym. 2006: 11)

Käytettävyys ei kuitenkaan ole vain yksi ominaisuus, vaan se koostuu useista osa-alueista, joita ovat opittavuus, tehokkuus, muistettavuus, virheettömyys ja miellyttävyys. Ensinnäkin tuotteen käyttötilanteen

pitää olla helposti opittavissa, jotta käyttäjä pääsee nopeasti alkuun. Tämän lisäksi opittuaan tuotteen käytön hän saavuttaa tehokkaasti mahdollisimman suuren tuottavuuden. Tuotteen käyttämisen pitää myös olla helposti muistettavissa, jotta pidemmän käyttötauon jälkeen opettelua ei tarvitse aina aloittaa alusta. Näiden lisäksi tuotteen toiminnassa pitää olla mahdollisimman vähän virheitä. Virhetilanteen sattuessa käyttäjän on kuitenkin pystyttävä helposti jatkamaan toimintaansa. Viidennen osa-alueen mukaan tuotteen pitää lisäksi olla käyttäjälle mieluinen. (Nielsen 1993: 26)

Etenkin käyttöliittymien kohdalla edellä mainittujen ominaisuuksien lisäksi joukkoon voidaan lisätä vielä kieli, johdonmukaisuus ja palaute. Käyttöliittymän ilmoitusten pitäisi olla selviä sanoja ja lauseita, sillä kaikki käyttäjät eivät välttämättä ymmärrä tuotekeskeistä sanastoa. Mikäli käyttöliittymässä käytetäänkin symboleja, niiden pitäisi olla niin selkeitä, että jokainen käyttäjä ymmärtää ne. On myös erittäin tärkeää, että käyttöliittymän toimintaperiaatetta ei vaihdeta yhtäkkiä, koska tällöin käyttäjän pitäisi kesken kaiken opetella käyttäminen uudestaan. Lisäksi käyttöliittymän pitää kertoa käyttäjälle, mitä on tapahtumassa. Käyttöliittymän on siis reagoitava käyttäjän tekemisiin kohtuullisessa ajassa ja annettava palautetta käyttäjän toiminnan mukaan. (Nielsen 1993: 20.)

Käytettävyys on tärkeää, koska jos käyttäjä kokee jonkin tuotteen käyttämisen epämiellyttäväksi, hän ei käytä sitä uudestaan. Hän mitä todennäköisimmin jättää vielä ensimmäisenkin käyttökerran kesken. Vastaavasti jos käytettävyys on käyttäjän mieleen, hän käyttää tuotetta uudestaankin. Hän saattaa jopa kertoa ystävilleen siitä. Hyvä käytettävyys heijastuu myös yrityksen imagoon positiivisesti.

Ensimmäinen askel käytettävyyden huomioon ottamisessa on oppia tuntemaan käyttäjät ja tuotteen käyttötarkoitus (Nielsen 1993: 73). Jokainen käyttäjä on erilainen ja kokee asiat omalla tavallaan. Siksi on tärkeää saada muodostettua yhtenäinen käsitys käyttäjistä, heidän kyvyistään ja rajoitteistaan sekä tarpeistaan ja odotuksistaan (Airgid & Reindel 2002: 64). Näin saadaan määriteltyä kyseisen tuotteen kohderyhmä. Kohderyhmän lisäksi on tiedettävä tuotteen käyttötarkoitus ja tavoitteet parhaimman mahdollisen käytettävyyden saavuttamiseksi.

Miten käytettävyyteen voi sitten vaikuttaa käytännön tasolla? Käytettävyyden määritelmän mukaan puhutaan tietystä tilanteesta, jossa tietty käyttäjä voi suorittaa tietyn tehtävän tehokkaasti ja häntä miellyttävällä tavalla saavuttaakseen halutun tavoitteen. Esimerkiksi Flash-esitystä tehtäessä on muistettava, että käytettävyyteen vaikuttavia osa-alueita ovat eri WWW-selaimet, esilataaja, navigointipalkki, tekstit, kuvat ja lomakkeet. Nämä on siis pyrittävä toteuttamaan aiemmin mainitut käytettävyyden ominaisuudet huomioiden. On myös

tiedettävä, mikä esityksen käyttötarkoitus on ja mikä on sen kohderyhmä.

3.1.1 Käyttökohteet

Kun kohderyhmä ja tuotteen käyttötarkoitus ovat selvillä, pitää tuotetta suunniteltaessa määritellä tavoite. Miksi esimerkiksi Flash-toteutus tehdään? Tavoite auttaa toteutuksen vaatimusten määrittelyssä eli siinä, mitä toteutuksen pitää tehdä. (Airgid & Reindel 2002: 42.) Jotta vaatimuksia voitaisiin määritellä, on myös tiedettävä, mihin ympäristöön Flash-toteutusta ollaan tekemässä. Kyseessä voi esimerkiksi olla CD-ROM, kokonainen WWW-sivusto tai sen osa, WWW-sivuille tuleva peli, mainos tai esitys.

Kohderyhmä vaikuttaa myös siihen, kuinka paljon ja minkälaista toiminnallisuutta toteutuksessa kannattaa käyttää, miltä se näyttää ja minkälaista sisältöä siinä käytetään (Airgid & Reindel 2002: 42). Näin ollen käytettävän Flashin määrä voi vaihdella kohderyhmän mukaan (Airgid & Reindel 2002: 64).

Alkuvaiheessa onkin siksi kysyttävä, tarvitaanko esityksen tekemiseen Flashiä? Tuoko Flash jotain lisäarvoa tai mahdollistaako se kenties esityksen tekemisen halutulla tavalla? Kaikkeen sisältöön sitä ei välttämättä tarvita (McGregor ym. 2002: 89).

Flashiä voidaan sekoittaa tavalliseen HTML-sivuun helposti ja monipuolisesti. ”Tavallinen käyttäjä ei välttämättä erota Flashiä WWW-sivulla, eikä hänen tarvitsekaan” (McGregor ym. 2002: 39).

Airgid ja Reindel (2002: 43) ovat jakaneet WWW-sivustojen päätavoitteet seuraaviin kategorioihin.

- tiedon esittäminen
- palvelun tarjoaminen
- tuotteiden myyminen
- koulutus
- yrityksen tai tuotteen markkinointi
- viihde
- portaali

Suurin osa kaupallisista WWW-sivustoista kuuluu viidenteen kategoriaan, jolloin päätavoite on kertoa kävijöille yrityksestä tai sen tuotteista ja palveluista. Tällaisilla sivustoilla käytetään useita erilaisia sisältötyyppejä, kuten tekstiä, 2D- ja 3D-kuvia, esityksiä ja niin edelleen. Flashin käyttö vaihtelee tapauksittain. Sisältötyypistä riippumatta on muistettava säilyttää toteutuksen käytettävyys. Käyttäjän kokemus ei silti muodostu pelkästään toteutuksen toiminnallisuudesta, sillä to-

teutuksen visuaalisuus vaikuttaa kokemukseen yhtä lailla. (Airgid & Reindel 2002: 52–54)

Flashillä on mahdollista tehdä monenlaisia toinen toistaan hienompia tehosteita. Kaikkia mahdollisia tehosteita ei silti kannata käyttää etenkään samanaikaisesti, koska silloin olennainen tieto hukkuu helposti tehosteiden sekaan. Tällöin käyttäjän huomio ei kiinnity sellaiseen tietoon, mihin oli tarkoitus.

3.1.2 Selaimet

Airgidin ja Reindelin mukaan (2002: 66) yksi Flashin suurimmista vahvuuksista on sen selainriippumattomuus. Tällaisen selainriippumattomuuden saavuttaminen ei useinkaan ole niin yksinkertaista kuin SWF-tiedoston julkaiseminen Flashin lähdetiedostosta. Flash-esitys näyttää samalta kaikissa selaimissa. Tosin näin yksinkertaista se ei kuitenkaan ole, sillä eri selaimissa saattaa näkyä pieniä eroja. Internet Explorer - ja Netscape-selaimissa elementtien sijoittelussa ero voi olla jopa viisi pikseliä.

Nämä erot johtuvat selaimien erilaisista työkalupalkeista ja näkymistä. Esimerkiksi työkalupalkkeja voi olla useampia kuin yksi, mikä voi viedä näkymästä jo kymmenen pikseliä. Kaikki käyttäjät eivät edes välttämättä osaa ottaa ylimääräisiä työkalupalkkeja pois käytöstä. (Airgid & Reindel 2002: 67)

Tällaisista eroista johtuen koko Flash-esitys ei välttämättä edes mahdu kerralla näkyviin. Jos taas esitys skaalautuisi koko näytön kokoiseksi, ei tarvitsisi huolehtia siitä, jääkö joku osa näkemättä eli ruudun ulkopuolelle. Tällöin animaatiot saattavat kuitenkin toimia hieman hitaammin, koska näyttö täytyy päivittää isolta alueelta. Lisäksi bittikarttakuvat näyttävät helposti huonolaatuisilta ja alkavat vääristyä. Siksi esitystä pitääkin testata niin monella erilaisella selaimella kuin mahdollista. (Airgid & Reindel 2002: 68)

Lisäksi Flash-esitystä suunniteltaessa täytyy pitää mielessä, että käyttäjällä saattaa olla käytössään hitaampi Internet-yhteys kuin itse suunnittelijalla. Flash-esitystä kannattaa testata myös erilaisilla Internet-yhteyksillä. Macromedia Flash -ohjelman testaustilassa pystyy kokeilemaan esityksen toimimista eri nopeuksilla. Myös koneen laitekokoonpano voi vaikuttaa Flash-esityksen toimimiseen. Jos koneessa on todella vähän muistia, ei sitä välttämättä riitä tarpeeksi isojen Flash-animaatioiden näyttämiseksi. (Airgid & Reindel 2002: 68–69)

3.1.3 Esilataaja

Käyttäjät eivät yleensä ole kärsivällisiä. Airgidin ja Reingelin mukaan käyttäjä jaksaa odottaa maksimissaan noin kymmenen sekuntia sivun tai elementin latautumista. Jos käyttäjä uskoo odottamisen olevan hänelle sen arvoista, hän jaksaa odottaa kauemminkin. Käyttäjän odottamispäätökseen auttaa se, että kerrotaan odotusaika. (Airgid & Reindel 2002: 71)

Esilataajaa tehtäessä on otettava huomioon kohderyhmä, sillä esilataajalla voidaan myös parantaa käyttäjäkokemusta. Flash-esityksen lataamisen yhteydessä on hyvä ilmoittaa jollain tavalla, kuinka kauan käyttäjän pitää odottaa, sillä pelkkä vilkkuva latausilmoitus ei kerro tarpeeksi. Hyviä tapoja voivat olla arvioidun latausajan ilmoittaminen tai latauksen edistymistä kuvaava palkki, jonka sijasta voi käyttää mitä tahansa muutakin kuviota. Ei kuitenkaan kannata ilmoittaa kaikkea mahdollista tietoa esityksen lataamisesta, koska tällöin käyttäjä ei välttämättä enää tiedä mikä tarkoittaa mitäkin. Jopa karkea arvio latausajasta on parempi kuin ei mitään. (Capraro & McAlester 2002: 58–59)

Latausajan ilmoittamisen lisäksi käyttäjälle voi tarjota jotain tekemistä esityksen latausajaksi. Lataus-sivulla voi olla esimerkiksi jokin pieni huomion kiinnittävä animaatio, joka näytetään toistamiseen. Sivulla voi olla myös jotain tekstiä luettavaksi. Myös hiiren liikuttamisella voidaan aiheuttaa jokin tehoste, jonka katsominen vie käyttäjän huomion latausajan pituudesta. (Airgid & Reindel 2002: 72)

Jos varsinaisen Flash-sisällön lataamiseen menee todella paljon aikaa, yksi hyvä tapa pitää käyttäjä sivulla on tehdä hyvin yksinkertainen peli tai muu sovellus, jota voi käyttää samalla, kun varsinainen esitys latautuu (Keating 2002: 442). Kun esitys on latautunut, voidaan asiasta ilmoittaa tuomalla näkyviin painike varsinaiseen sisältöön siirtymiseksi. Sisältöön ei kannata siirtyä automaattisesti sen latauduttua, koska tällöin käyttäjän peli voi keskeytyä käyttäjäkokemukseen negatiivisesti vaikuttavalla tavalla.

Koko sivustoa tai esitystä ei kannata ladata ensin muistiin, koska käyttäjä ei välttämättä tarvitse kaikkea sivustolta löytyvää tietoa. Kannattaa siis ladata sivustoa sen mukaan, mitä käyttäjä haluaa. Tämä tarkoittaa käytännössä sitä, että sivusto toteutetaan usealla SWF-tiedostolla. Tiedostot voi pilkkoa osiin myös sen takia, että niitä on helpompi päivittää. (Capraro & McAlester 2002: 58–59.)

Latausaikoja silmällä pitäen Flash-sisältö voidaan streamata eli ladata sisältöä samalla kun sitä näytetään. Esimerkkinä tulee mieleen video, jota aletaan näyttää jo ennen kuin se on kokonaan latautunut. Sivustoilla voidaan myös ladata navigointipalkki ennen varsinaisen kuvan

tai animaation latausta. Näin käyttäjä näkee navigointipalkin ensin ja voi halutessaan mennä sivustolla suoraan haluamalleen sivulle. Alussa voidaan myös ladata navigointipalkin lisäksi otsikko ja pieniä kuvia. Ääniraidat puolestaan ladataan yleensä vasta, kun kaikki muu sisältö on ladattu. Streamauksen pääajatus on, että käyttäjän ei tarvitse odottaa, kunnes koko sisältö on latautunut sataprosenttisesti. Lataamalla osioita kerrallaan käyttäjälle tarjotaan sisältöä mahdollisimman pian. (Airgid & Reindel 2002: 73)

3.1.4 Navigointipalkki

Navigointipalkin merkitystä ei voi vähätellä käytettävästä tekniikasta riippumatta, koska navigointipalkin avulla sivustoa pääasiassa kontrolloidaan. Pelkästään navigointipalkin näyttävä ulkoasu ei riitä, vaan sitä on pystyttävä myös käyttämään. Navigointipalkin suunnittelussa on jälleen otettava huomioon kohderyhmä. (Airgid & Reindel 2002: 78)

Navigointipalkin sijainnista on käyty paljon keskustelua suunnittelijoiden ja käytettävyysasiantuntijoiden välillä. Joidenkin mielestä ainoa oikea tapa on laittaa navigointipalkki ylös ja vasemmalle. Flashin antamien mahdollisuuksien myötä tulee helposti kokeiltua mitä erilaisimpia ratkaisuja navigointipalkin suhteen. Se, mikä on paras sijainti navigointipalkille, riippuu paljolti itse esityksestä, jota ollaan tekemässä. Navigointipalkin fyysinen sijainti ei kuitenkaan ole kaikkein tärkein tekijä käytettävyyttä silmällä pitäen. (Airgid & Reindel 2002: 85)

Navigointipalkin käytettävyydessä nousee esiin neljä merkittävää tekijää. Navigointipalkin pitää olla tunnistettava, opittavissa oleva, johdonmukainen ja jäljitettävä. (Airgid & Reindel 2002: 85)

Kun käyttäjä tulee sivulle, hänen pitää pystyä tunnistamaan navigointipalkki, jota pitkin hän pääsee sisällössä eteenpäin. Monissa Flash-toteutuksissa on oltu ehkä liiankin luovia, jonka vuoksi käyttäjä ei tunnista navigointipalkkia. Se voi olla piilotettu, jolloin se paljastuu vasta, kun käyttäjä vie hiiren sen päälle. Navigointipalkki voi myös olla epälooginen rakenteeltaan tai nimiöinneiltään. Lisäksi navigointipalkki voi myös olla liikkuva, jolloin käyttäjän on vaikea päästä siihen käsiksi. (Airgid & Reindel 2002: 85–86)

Navigointipalkin pitää myös olla opittavissa. Käyttäjän pitää oppia ja muistaa navigointipalkin käyttäminen ilman, että siihen tarvitsisi kiinnittää erityistä huomiota tai lisäponnistusta. (Airgid & Reindel 2002: 86–87)

Näiden tekijöiden lisäksi navigointipalkin pitää olla johdonmukainen. Jos navigointipalkki ei ole johdonmukainen, se vaikeuttaa jo kahden edellisen kohdan täyttymistä. Tiedon pitää olla samalla tavalla lajiteltuna myös navigointipalkin alakohdissa. Esimerkiksi jos asiakas menee levykauppaan, hän olettaa levyjen olevan loogisessa ja johdonmukaisessa järjestyksessä. Kun asiakas menee rock-hyllylle, hän huomaa, että CD-levyt on jaoteltu omaan osioonsa ja DVD-levyt omaan osioonsa molemmat aakkosjärjestyksessä. Sitten asiakas menee katsomaan suomipop-hyllyä. Tässä hyllyssä CD-levyt ja DVD-levyt ovatkin keskenään sekaisin ja julkaisupäivän mukaan järjestettynä. Kolmannessa hyllyssä levyt on järjestetty levy-yhtiön mukaan. Tällaisista hyllyistä on todella vaikea löytää etsimäänsä. (Airgid & Reindel 2002: 87–88)

Neljäntenä kohtana navigointipalkin pitää olla jäljitettävissä. Käyttäjän pitää nähdä, missä hän on ja mistä hän on tullut, jotta hän osaa tarvittaessa palata takaisin haluamaansa kohtaan. (Airgid & Reindel 2002: 88–89)

Flash-esityksissä ei tyypillisesti voida käyttää selaimen Takaisin-painiketta. Jos käyttäjä painaa sitä, hän joutuu koko esityksestä pois. Yksi vaihtoehto tällaisen tilanteen estämiseksi tai mahdollisuuden pienentämiseksi on tehdä oma Takaisin-painike Flash-esitykseen. Tällaisen merkitys korostuu etenkin silloin, kun selaimen Takaisin-painike on poistettu käytöstä esimerkiksi JavaScriptillä. (Airgid & Reindel 2002: 90–91.) Takaisin-painikkeesta on myös muistettava tehdä tunnistettava muun navigointipalkin tapaan.

Navigointipalkin nimeämisessä pitää noudattaa selkeää logiikkaa. Tämän takia navigointipalkin tekstin pitää kertoa tarpeeksi selvästi, mitä painikkeen takaa löytyy. On myös huomioitava, että käytettävästä kielestä riippuen osan sanoista voi tulkita monella tavalla. Tekstin pitää olla sellaista, jonka käyttäjät tunnistavat eivätkä voi kovin helposti tulkita väärin. (Airgid & Reindel 2002: 91–92)

Tekstin sijasta navigointipalkissa voi käyttää myös ikoneja eli pieniä kuvia, sillä kuvat jäävät tekstiä helpommin mieleen. Ikonien käytön ongelmana on kuitenkin se, että ikonin kuvasta ei välttämättä tunnista, mitä se tarkoittaa. Kuva voi myös tarkoittaa montaa eri asiaa. Jos tekstin sijasta välttämättä haluaa käyttää ikoneja, pitää niiden olla tarpeeksi selkeitä, yksiselitteisiä ja helposti tunnistettavissa. (Airgid & Reindel 2002: 92–93.)

3.1.5 Tekstit

Värejä valittaessa on ajateltava, onko teksti luettavaa. Yleisesti ajatellaan, että musta teksti valkoisella on helpoiten luettavissa. Tämä on

totta luettaessa paperilta. Tekstiä, jossa on iso kontrasti, on helpompi lukea kuin tekstiä, jossa on pieni kontrasti. Kun henkilö katsoo näyttöä, hän katsoo suoraan eräänlaiseen valonlähteeseen. Siksi musta teksti valkoisella ei olekaan kaikkein miellyttävintä katsottavaa etenkään yhtäjaksoisesti pitkiä aikoja. Näytöltä lukemista voidaan helpottaa vähentämällä kontrastia aavistuksen verran etenkin silloin, kun luettavaa on paljon. Esimerkiksi valkoisen värin sijasta voi käyttää hieman eri sävyistä valkoista tai muuta vaaleaa väriä. (McGregor ym. 2002: 239)

HTML-kielessä on perinteisesti rajoituttu käyttämään esimerkiksi kirjasimia Arial, Verdana, Times tai Courier, koska erikoisemmat kirjasimet eivät välttämättä näy käyttäjän koneella. Flashissä voidaan käyttää myös erikoisempia kirjasimia, koska julkaistaessa kirjasimet sisältyvät SWF-tiedostoon. Tällöin kaikki näkevät kirjasimet sellaisina kuin tekijä on ne halunnut. Välttämättä kaikki kirjasimet eivät kuitenkaan ole helposti luettavia. (Airgid & Reindel 2002: 118)

Kun suunnittelee tietyn kirjasimen käyttämistä Flashissä, on tärkeää testata sitä erilaisissa näytöissä ja resoluutioissa. Pitää myös muistaa, että tekstin tausta vaikuttaa luettavuuteen. Jos taustalla on jokin kuva, voi tekstin lukeminen olla huomattavasti vaikeampaa kuin silloin, kun tumma teksti on vaalealla pohjalla tai vaalea teksti tummalla pohjalla. Tietyissä kirjasimissa myös tekstin antialiasointi eli reunojen tasoitus voi vaikeuttaa lukemista. (Airgid & Reindel 2002: 118–120)

Flashissä on kolmenlaisia tekstikenttiä, staattisia, dynaamisia ja syötekenttiä. Staattiseen tekstikenttään kirjoitetaan teksti suoraan Flashissä, kun taas dynaamiseen tekstikenttään voidaan tuoda teksti Flashin sisältä jostain muuttujasta tai ulkoisesta tiedostosta. Syötekenttä puolestaan on kenttä, johon käyttäjä voi kirjoittaa.

3.1.6 Kuvat

Flashin merkittävimpiä etuja animaatioiden lisäksi on vektorigrafiikan käyttö, minkä etuja puolestaan on, että kohdetta voi suurentaa ja pienentää ilman, että laatu huononisi. Kaikki ohjelman sisällä tehdyt tekstit ja grafiikat ovat vektorigrafiikkaa. Flashiin voi myös tuoda bittikarttakuvia, joita ovat esimerkiksi valokuvat. Tällaisten kuvien kokoa muutettaessa kuvan laatu kärsii.

Jo yhden bittikarttakuvan tuominen Flashiin saattaa kasvattaa tiedostokokoa helposti jopa 50 kilotavulla (Lever 2004: 185). Siksi tällaiset kuvat kannattaa optimoida kuvankäsittelyohjelmalla jo valmiiksi oikean kokoisiksi.

Flash tukee yleisimpien JPG- ja GIF-kuvamuotojen lisäksi PNG-kuvamuotoa, joka tukee läpinäkyviä taustoja läpinäkyvien GIF-kuvien tapaan. PNG-kuvasta voi tehdä kuitenkin huomattavasti monimutkaisemman, eli se voi sisältää enemmän värejä ja useampia tasoja. PNG-kuvien käyttäminen Flashissä mahdollistaa erilaisten tasojen hyödyntämisen suunnittelussa. (Airgid & Reindel 2002: 115.) PNG-kuvan tiedostokoko on kuitenkin isompi kuin JPG-kuvan, mikä kannattaa pitää mielessä, jos tavoitteena on mahdollisimman pieni tiedostokoko. Oikean kuvamuodon valitseminen riippuu viime kädessä käyttötarkoituksesta.

3.1.7 Lomakkeet

Jos lomakkeet on tehty hyvin, käyttäjä voi tehokkaasti kommunikoida sivuston omistajan kanssa. Käyttäjä ei välttämättä osaa käyttää huonosti toteutettua lomaketta ja voikin jättää siksi sen täyttämisen kesken. (Airgid & Reindel 2002: 141)

Lomakkeiden suunnitteluun pätevät lähes samat ohjeet kuin navigointipalkillekin (vrt. luku 3.1.4). Lomakkeen pitää olla tunnistettavissa, opittavissa oleva ja jäljitettävissä. Käyttäjän pitää pystyä helposti tunnistamaan lomake ja sen kentät. Samoin käyttäjän pitää ilman suurempia ponnisteluja ymmärtää lomakkeen toiminta. Tähän auttaa kenttien sijoittaminen loogiseen järjestykseen ja niiden nimeäminen selkeästi. Jos lomake on pidempi ja jaettu useammalle sivulle, pitää käyttäjän myös nähdä, missä hän on menossa. Lyhyessä lomakkeessa jäljitettävyydestä ei juurikaan tarvitse välittää. (Airgid & Reindel 2002: 142–145.)

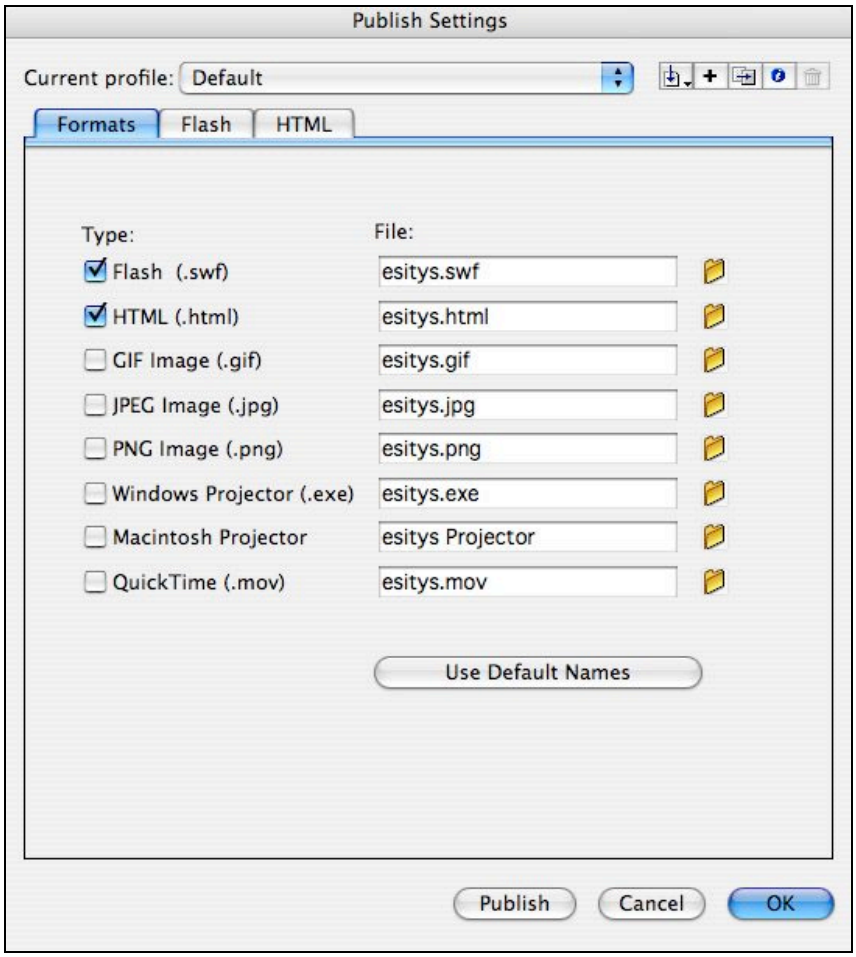
Käyttäjäkokemusta voi parantaa myös se, että kursori vilkkuu lomakkeen ensimmäisessä kentässä sivulle tultaessa. Näin käyttäjän ei tarvitse erikseen siirtyä oikeaan tekstikenttään hiirellä valitsemalla. Jotta tämä toimisi Flashissä, käyttäjän on ennen lomakkeelle siirtymistä painettava hiirellä missä tahansa Flash-sisällössä. Tällöin Flash saa kohdistuksen itseensä ja osaa sijoittaa lomakkeelle siirryttäessä kursorin määrättyyn kenttään. Lisäksi on hyvä ilmoittaa käyttäjälle jo valmiiksi, mitkä kentät on täytettävä ennen lomakkeen lähettämistä.

Flash-sisällössä voi hypätä kentästä toiseen HTML-sivuilta totuttuun tapaan painamalla TAB-näppäintä. Flashissä voi määritellä erikseen indeksit elementeille, jolloin TAB-näppäintä painettaessa liikutaan indeksissä määritettyyn seuraavaan kohtaan. Ilman näitä määrityksiä vähänkin monimutkaisemmassa asettelussa TAB-näppäimen painaminen voi viedä eri paikkaan kuin mihin oli tarkoitus.

3.2 Flashin upottaminen WWW-sivuille

Flashin näkyminen WWW-sivuilla edellyttää SWF-tiedoston tekemistä ja linkittämistä HTML-tiedostoon. Jos selaimessa on vaadittava Flash Player -laajennus, se osaa näyttää SWF-tiedoston, kunhan se on linkitetty HTML-tiedostoon tavalla, jonka selain ymmärtää.

Flash-esitystä tehtäessä työversio eli lähdeversio tallennetaan aina FLA-tiedostomuodossa. Kyseessä on Macromedia Flash -ohjelman oma tiedostomuoto. Ohjelmassa on oma toimintonsa esityksen julkaisemiseksi. Julkaisuasetuksiin pääsee kohdasta Tiedosto | Julkaisuasetukset eli File | Publish Settings. Tässä ikkunassa on mahdollista valita muun muassa, millaisia tiedostoja julkaistaan ja mitä Flash Playerin versiota käytetään (Kuvio 1). Ohjelmassa on mahdollista julkaista tarvittavan SWF-tiedoston lisäksi HTML-tiedosto.



Kuvio 1 Julkaisuasetukset

Flash-ohjelman tuottaman HTML-sivun <body>-osion HTML-koodi on koodiesimerkin 1 mukainen.

```
<body bgcolor="#d9d9d9">
<!--url's used in the movie-->
<!--text used in the movie-->
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-
444553540000"
codebase="http://download.macromedia.com/pub/shockwave/
cabs/flash/swflash.cab#version=7,0,0,0" width="640"
height="480" id="esitys" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="esitys.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#d9d9d9" />
<embed src="esitys.swf" quality="high"
bgcolor="#d9d9d9" width="640" height="480"
name="esitys" align="middle"
allowScriptAccess="sameDomain" type="application/x-
shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplaye
r" />
</object>
</body>
```

Koodiesimerkki 1 Osa HTML-koodista, jonka Flash-ohjelma on tehnyt

Kehitetyt standardit mahdollistavat, että toteutetut WWW-sivut näkyvät kaikissa selaimissa erilaisilla alustoilla ja laitteilla. Tällaista yhteensopivuutta on mahdotonta saavuttaa ilman sovittuja standardeja. Ne mahdollistavat, että sivut näkyvät myös tulevaisuudessa, jotta niiden ylläpito ja korjaaminen eivät silloin sido niin paljon resursseja. (Zeldman 2003: 17)

World Wide Web Consortium eli W3C on luonut määrittelyt ja ohjeistuksen, joiden tarkoituksena on ollut mahdollistaa Internetin kehitys ja varmistaa, että eri Internet-teknologiat toimivat hyvin yhdessä. W3C:n kehittämät määrittelyt koskevat muun muassa HTML-, CSS-, XML- ja XHTML-kieliä. Aluksi W3C käytti määrityksistään vain nimitystä suositukset. Mahdollisesti tämän takia eri selainvalmistajat eivät ole ottaneet määrityksiä tosissaan, minkä vuoksi eri selaimet saattavatkin näyttää saman asian hieman eri tavalla. Nykyään W3C käyttääkin suosituksistaan nimitystä standardit. (Zeldman 2003: 16–17)

W3C:n standardit eivät suoranaisesti koske itse Flash-sovelluksia. Standardit tulevat kyseeseen vasta upotettaessa Flash-sovellusta WWW-sivulle. Flash-ohjelmalla voi siis sovellusta julkaistaessa luoda myös HTML-sivun. Valitettavasti tämä HTML-koodi ei ole W3C:n standardien mukaista, koska standardissa ei ole mukana <embed>-elementtiä. W3C suosittelee käytettäväksi <object>-elementtiä. Vieläkään ei ole kehitetty mitään yksiselitteistä tapaa upottaa Flash-sovellus WWW-sivulle standardien mukaisesti. Tällaisen kehittäminen takaisi toimivuuden ja luotettavuuden kaikissa selaimissa ja alustoissa. (Zeldman 2003: 291–292)

W3C:n WWW-sivuilla, http://validator.w3.org, on validaattori, jolla voi tarkistaa tekemänsä koodin oikeellisuuden. Yksi tapa saada HTML-koodi validaattorista läpi on linkittää SWF-tiedosto JavaScriptillä. Tämä ei kuitenkaan ole paras ja siistein vaihtoehto. Drew McLellan on kehittänyt tavan, jota nimitetään Satay-tavaksi. Sen avulla linkitys voidaan tehdä standardien mukaisesti, mutta sekään ei ole silti täysin ongelmaton. Kerron tästä tavasta lisää luvussa 6.2.

3.3 Flashin ylläpito

Periaatteessa suurin osa muutoksista pitää tehdä Macromedia Flash -ohjelmassa, mikä edellyttää ohjelman käytön osaamista. Tietyt toiminnot voi kuitenkin tehdä niin, että päivittäminen onnistuu osittain Flashin ulkopuolisista tiedostoista käsin.

Flash-esitykset voi rakentaa useammalla eri tavalla. Toteutustapa voi riippua tavoitteista ja siitä, tehdäänkö Flashillä WWW-sivustoa, esitystä vai pientä mainosta. Perusajatus esityksen koostamisessa on tiedostokokojen pitäminen mahdollisimman pienenä, jotta käyttäjä ei joudu odottamaan latausta liian kauan. Hyvä keino on pilkkoa esitys osiin eli useampiin SWF-tiedostoihin. Näin esitystä voidaan ladata käyttäjän valintojen mukaan. Tässä on vielä se hyvä puoli, että yksittäistä osaa on helpompi päivittää ilman, että tarvitsisi koskea esityksen muuhun rakenteeseen.

Lisäksi toteutuksen koodien sijoittelulla voi helpottaa ylläpitoa. Elokuvaleikkeen kehyksiin voi sijoittaa koodia, joka lukee talteen käyttäjän syötteitä tai liikkeitä. Kaikki vähänkin monimutkaisempi toiminnallisuus kannattaa sijoittaa funktioihin, joita voidaan kutsua esimerkiksi elokuvaleikkeiden sisältä. (Lever 2004: 151.) Nämä funktiot voi sijoittaa yhteen paikkaan, esimerkiksi ulkoiseen AS-tiedostoon, joka otetaan käyttöön Flashissä yhdellä #include-komennolla. Käytännössä tällaisia AS-tiedostoja voi olla useampiakin, jolloin kukin voi sisältää tietyn aihealueen funktiot.

Tekstit on mahdollista tuoda ulkoisesta TXT- tai XML-tiedostosta, mikä tekee tekstien ylläpidosta todella helppoa. Tällaisten tekstitiedostojen käyttämisessä on muistettava tallentaa tiedostot UTF-8-koodattuna. Myös bittikarttakuvat on mahdollista ladata ulkoisista tiedostoista ilman, että tuo niitä Flash-ohjelmaan. Tämä voi toimia joissain tapauksissa hyvin. Näin kuvan saa helposti vaihdettua, kun tallentaa uuden kuvan tiettyyn paikkaan tietyllä nimellä. FLA-tiedostoon ei tarvitse tällöin tehdä muutoksia.

4 Tutkimus

Arvioin tutkintotyössäni vanhaa Analyste eOfficen Flash-esitystä sekä nyt tekemääni uutta Analyste-ratkaisun Flash-esitystä. Halusin omien mielipiteideni tueksi kommentteja myös muutamalta muulta henkilöltä. Uuden Flash-esityksen tekemisessä otin huomioon oman arviointini ja tekemäni haastattelut. Esitysten arvioinneissa otin huomioon esityksen toimivuuden, käytettävyyden ja teknisen toteutustavan. Tarkastelin esityksiä luvussa 3 esiteltyjen käytettävyyskriteerien pohjalta.

Halusin saada kommentteja esityksistä niiden varsinaisilta käyttäjiltä. Haastattelin omassa tutkimuksessani kahta henkilöä esitystä kohden, koska tarkoitukseni ei ollut kuitenkaan tehdä täydellistä käytettävyystutkimusta, josta olisi voinut saada jo oman tutkintotyön aiheen. Tavoitteena oli ennemminkin saada tukea omalle arvioinnille ja selvittää, tuleeko esiin sellaisia asioita, joita en itse ollut huomannut ajatella. Tein heti aluksi haastattelut Analyste eOfficen esitykseen liittyen. Tein samanlaiset haastattelut myös lopuksi Analyste-ratkaisun esitykseen liittyen haastattelemalla kuitenkin eri henkilöitä.

Haastateltaviksi valitsin yhden Analysten myyjän ja yhden Analysten ulkopuolisen henkilön, jolle taloushallinto on tuttu aihepiiri. Myyjään päädyin, koska hän on tuotteen kanssa koko ajan tekemisissä ja esitys on yksi myyjän tukimateriaaleista. Ulkopuolinen henkilö kuuluu juuri Analysten kohderyhmään, joka koostuu yrityksen taloushallinnossa vaikuttavista ihmisistä. Esitystä katsovat taloushallinnon ammattilaiset, joten on tärkeää tietää, miten kyseisen alan edustaja näkee ja kokee esityksen.

Annoin haastateltaville tehtäväksi katsoa esityksen läpi haastattelijasta välittämättä. Lopuksi haastateltavien piti lähettää testitiedot lomakkeella. Ennen lomakkeen täyttämistä ja lähetystä lisätehtävänä oli kuitenkin katsoa eOfficen esityksessä hyödyt. Analyste-ratkaisun esityksessä hyödyt ovat selkeästi mukana navigointipalkissa, joten lisätehtäväksi valitsin ostolaskut-kohdan esimerkin katsomisen. Muuten tehtävänanto ja tekemäni kysymykset olivat samoja.

Koska molemmat haastateltavat ovat esityksen suhteen erilaisessa roolissa, kysyin heiltä hieman eri asioita. Käytin alla olevia kysymyksiä haastattelun runkona keskustelun aikaansaamiseksi.

Myyjä:
- Tukeeko esitys myyntiä?
- Jos tukee, mikä erityisesti? Miksi?
- Jos ei tue, mikä erityisesti? Miksi?
- Mikä esityksessä on hyvää?
- Mikä esityksessä on huonoa?

Potentiaalinen asiakas:

- Miltä esitys näytti visuaalisesti?
- Miten esitys mielestäsi toimi?
- Ehditkö lukea tekstit?
- Ymmärsitkö esityksen perusteella eOfficen toiminta-ajatuksen?
- Minkälainen mielikuva sinulle muodostui Analystestä?
- Mitä mieltä olet esityksen esilataajasta (preloaderista)?
- Mitä mieltä olet esityksessä käytetyistä näyttökaappauksista?

5 Vanha Analyste eOfficen Flash-esitys

Analyste eOfficen tuote-esityksen tavoitteena on kertoa ohjelmiston toiminta selkeästi ja yksinkertaisesti sekä toimia mielenkiinnon herättäjänä, joka rohkaisisi käyttäjää ottamaan yhteyttä ja pyytämään lisätietoa.

Kuvaan tässä luvussa mielestäni merkittävimmät kohdat siitä, miten Analyste eOfficen Flash-esitys on tehty ja miten se on upotettu WWW-sivuille. Lopuksi esitän oman arvioni sekä haastattelemieni henkilöiden arviot esityksestä.

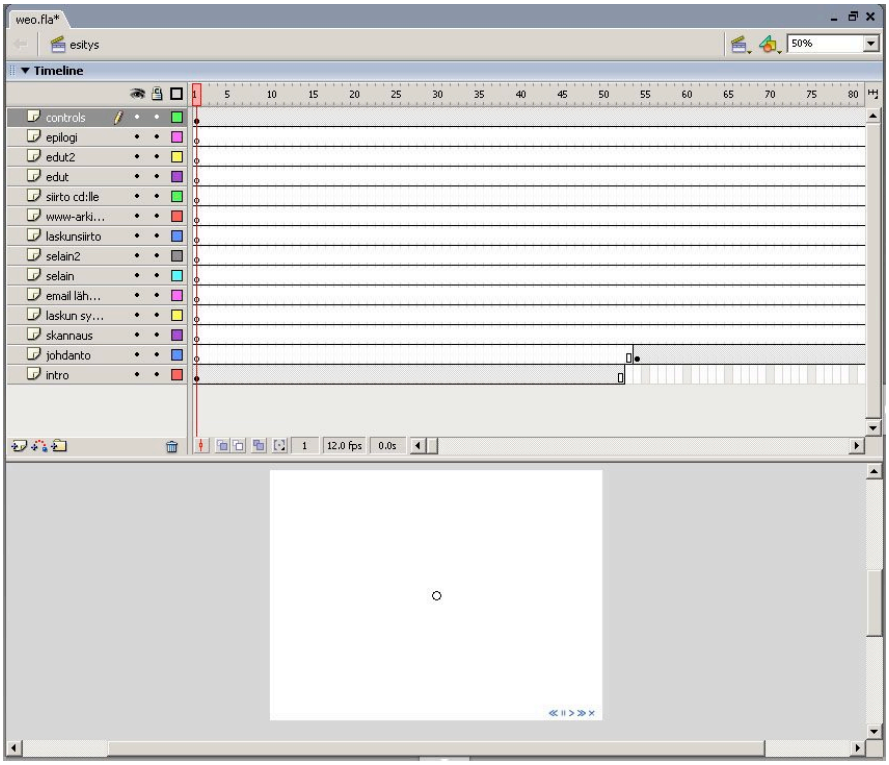
5.1 Toteutustavat

eOfficen esitys avautuu uuteen ikkunaan, joka on kooltaan 640 * 480 pikseliä. Esitys koostuu yhdestä SWF-tiedostosta, *weo.swf*, jonka tiedostokoko on 830 952 tavua eli 812 kt.

Esityksen nopeutena on Flashin oletusnopeus 12 FPS eli 12 kehystä sekunnissa. Koko esitys kestää pääaikajanalla kehykseen 2314 asti. Esityksen kesto ajallisesti on 192,8 sekuntia eli hieman yli kolme minuuttia.

5.1.1 Esitysrakenne

Esityksessä on kaksi kohtausta, ”pleasewait” ja ”esitys”. Näistä ensimmäisessä on esityksen esilataaja ja toisessa varsinainen sisältö. *Esitys*-kohtauksessa on 14 tasoa, joista ylimmäisenä on *controls* (Kuvio 2). Se sisältää navigointipalkin, joka on näkyvillä koko esityksen ajan.



Kuvio 2 Esityksen pääaikajana

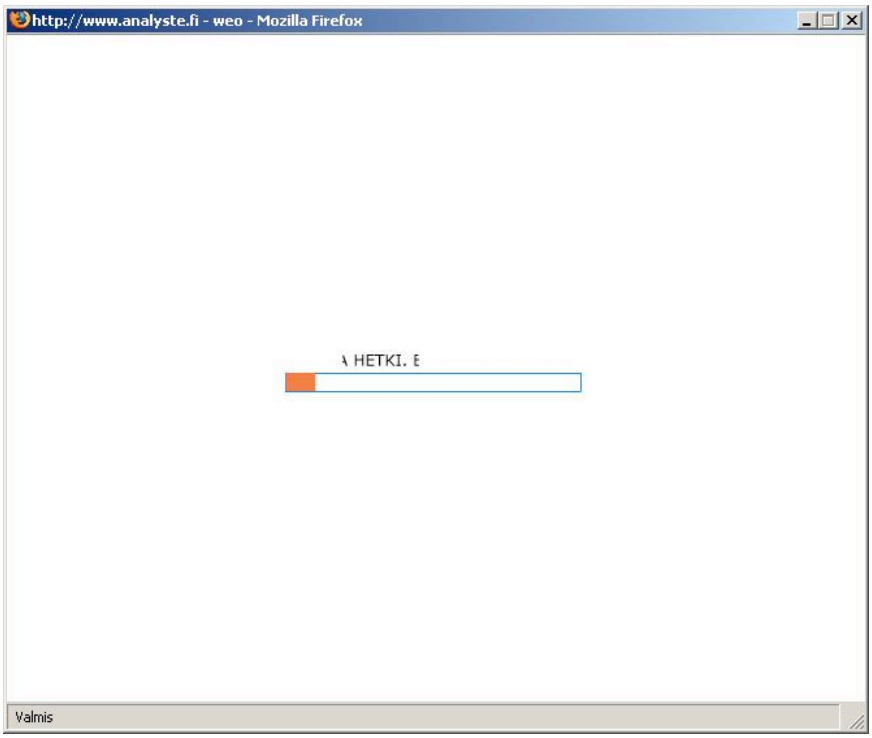
Pääaikajanalla on peräkkäin 13 osiota, jotka ovat kukin omassa elokuvaleikkeessään ja sijoiteltu vastaavasti omalle tasolleen. Alimmaisin taso eli ensimmäinen elokuvaleike sisältää intron, jonka kesto on 52 kehystä. Näyttämöllä olevan elokuvaleikkeen esiintymänimeksi on annettu *intro*. Tässä elokuvaleikkeessä Analysten logo sekä tekstit ilmestyvät ja häviävät. Kun elokuvaleikkeen aikajanalla saavutaan viimeiseen kehykseen, siirrytään pääaikajanalla seuraavana olevaan elokuvaleikkeeseen eli johdantoon. *Intron* lopussa on erikseen määritelty siirtyminen pääaikajanalla seuraavaan kehykseen.

Johdanto on 260 kehyksen pituinen. *Johdannossa* tulee näkyviin yksi kuva ja useita lauseita vuoron perään. Tämän leikkeen lopussa ei ole erikseen määritelty seuraavaa kohtaa, koska luonnollisesti tämän leikkeen lopussa siirrytään pääaikajanalla seuraavaan kehykseen. Seuraavassa kehyksessä alkaa *skannaus*-elokuvaleike, joka on 241 kehyksen pituinen.

Samalla tavalla siirrytään aina pääaikajanalla seuraavana olevaan elokuvaleikkeeseen, kunnes viimeisessä elokuvaleikkeessä pysäytetään koko esityksen eteneminen. Merkittävintä tässä on huomata, että pääaikajanalla elokuvaleikkeen pituus on sama kuin elokuvaleike itsessään on.

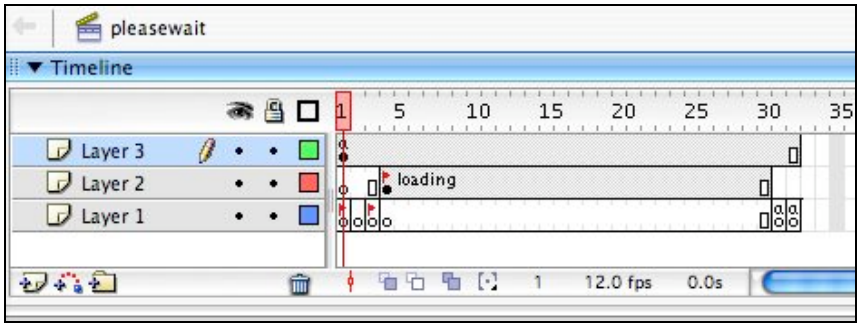
5.1.2 Esilataaja

Esilataajassa näkyy keskellä pieni osa kerrallaan tekstistä ”Odota hetki. Esitys latautuu…” (Kuvio 3). Tekstin alla on palkki, joka kasvaa esityksen latautuessa. Kun esitys on latautunut, alkaa varsinaisen esityksen toisto.



Kuvio 3 Analyste eOfficen esitys latautuu

Esilataajan aikajanalla on kolme tasoa, jotka kestävät kehykseen 32 asti (Kuvio 4). Ylimmäisellä tasolla on esityksen latauksen edistymistä kuvaava palkki, kun taas toisella tasolla on esilataajassa näytettävä teksti. Alin taso puolestaan sisältää avainkehyksiä ja ActionScriptiä.



Kuvio 4 Esilataajan aikajana

Ylimmäisen tason ensimmäisessä avainkehyksessä on esilataajan merkittävin ActionScript (Koodiesimerkki 2).

```
// Specify how many frames to load before playing.
var loadAmount = _totalframes;

// If the required number of frames
// have finished loading...
if (_framesloaded == loadAmount) {
  // ...start the movie
  gotoAndPlay("StartMovie");
} else {
  // ...otherwise, display the load status
  // then go back and check load progress again.

  // First, determine the loaded and total kilobytes.
  loaded = Math.round(getBytesLoaded() / 1024);
  total = Math.round(getBytesTotal() / 1024);
  percent = Math.round((loaded/total) * 100);

  // Display the loaded kb, total kb, and percentage of
  // kb loaded in text fields.
  loaded_bytes = loaded;
  total_bytes = total;
  percent_done = percent + "%";

  progressbar._width = kehys._width * (percent / 100);

  // Now go back and check load progress.

  gotoAndPlay ("loading");
}
```

Koodiesimerkki 2 Esilataajan ActionScriptiä

Koodissa otetaan aluksi talteen Flash-esityksen kokonaiskehysmäärä, jota sen jälkeen verrataan ladattuun kehysmäärään. Jos määrä on sama, siirrytään aikajanalla *StartMovie*-nimiseen avainkehykseen, jossa on komento siirtyä *esitys*-kohtauksen ensimmäiseen kehykseen. Muussa tapauksessa Flash-esityksen ominaisuuksista otetaan omiin muuttujiin talteen tiedostokoko ja paljonko on ladattu siihen mennessä.

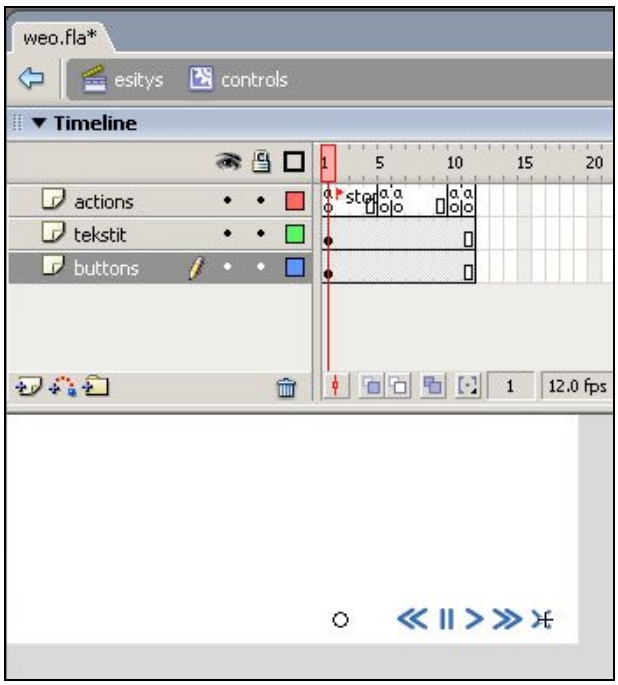
Palkin, jonka esiintymänimi on *progressbar*, leveydeksi määritellään ladattu prosenttimäärä palkin kehyksen leveydestä. Tämän jälkeen siirrytään *loading*-nimiseen avainkehykseen. Tästä alkaa lataustekstin animaatio, joka kestää kehykseen 30 asti. Avainkehyksessä 31 siirrytään takaisin ensimmäiseen avainkehykseen.

5.1.3 Navigointipalkki

Navigointipalkki sijaitsee esityksen oikeassa alakulmassa. Se koostuu viidestä painikkeesta (Kuvio 5), jotka ovat vasemmalta oikealle Taaksepäin, Pysäytä, Jatka, Eteenpäin ja Lopeta.

Kuvio 5 Navigointipalkin painikkeet

Koko navigointipalkki on yhden elokuvaleikkeen sisällä. Jokaisesta painikkeesta on tehty oma painikesymboli. Tekstit ovat omassa elokuvaleikkeessään, jonka esiintymänimeksi on annettu *tekstit*. Se sijaitsee painikkeiden vasemmalla puolella ja näkyy kuviossa 6 pienenä tyhjänä pallona.



Kuvio 6 Controls-elokuvaleikkeen sisältö

Actions-tasolla eteneminen pysäytetään heti ensimmäisessä avainkehyksessä, jolle on annettu nimeksi *stop*. Viides avainkehys, *fastforward*, sisältää komennot siirtyä esityksessä seitsemän kehystä eteenpäin (Koodiesimerkki 3). Kuudennessa avainkehyksessä on komento siirtyä avainkehykseen nimeltä *fastforward* eli aikajanalla on tällä kohdalla silmukka. Tämä tarkoittaa, että näiden kehysten välillä liikutaan toistamiseen niin kauan kun painike on painettuna pohjaan.

```
_parent.intro.gotoAndPlay
(_parent.intro._currentframe+7);
_parent.johdanto.gotoAndPlay
(_parent.johdanto._currentframe+7);
_parent.skannaus.gotoAndPlay
(_parent.skannaus._currentframe+7);
_parent.laskunsyotto.gotoAndPlay
(_parent.laskunsyotto._currentframe+7);
_parent.sophallinta.gotoAndPlay
(_parent.sophallinta._currentframe+7);
_parent.selain.gotoAndPlay
(_parent.selain._currentframe+7);
_parent.selain2.gotoAndPlay
(_parent.selain2._currentframe+7);
_parent.laskunsiirto.gotoAndPlay
(_parent.laskunsiirto._currentframe+7);
_parent.wwwarkisto.gotoAndPlay
(_parent.wwwarkisto._currentframe+7);
_parent.siirtocd.gotoAndPlay
(_parent.siirtocd._currentframe+7);
_parent.edu1.gotoAndPlay
(_parent.edut1._currentframe+7);
_parent.edut2.gotoAndPlay
(_parent.edut2._currentframe+7);
_parent.lomake.gotoAndPlay
(_parent.lomake._currentframe+7);
_parent.gotoAndPlay (_parent._currentframe+7);
```

Koodiesimerkki 3 Eteenpäin-painikkeen ActionScript

ActionScriptissä on mainittu kukin elokuvaleike, koska kelaustoiminnon pitää vaikuttaa läpi koko esityksen nykyisestä sijainnista riippumatta.

Kymmenes avainkehys on *rewind*, joka sisältää samanlaisen ActionScriptin kuin avainkehyksessä *fastforward* sillä erolla, että sijaintia vähennetään kymmenellä kehyksellä. Seuraavassa avainkehyksessä on komento siirtyä avainkehykseen *rewind*, jolla toteutetaan kehysten välinen silmukka.

Taaksepäin- ja Eteenpäin-painikkeissa on määritelty, että painiketta painettaessa siirrytään aikajanalla esimerkiksi *rewind* avainkehykseen, jolloin päädytään kahden kehyksen silmukkaan ja aiheutetaan kelaustoiminto. Kun painikkeesta päästää irti, siirrytään aikajanalla *stop*-avainkehykseen, jolloin päästään pois silmukasta ja kelaustoiminto loppuu.

Pysäytä-painike puolestaan laittaa esityksen taukotilaan, jolloin esitys ei etene. Toiminto on toteutettu niin, että kun painiketta painetaan, annetaan pysähtymiskomento kaikille elokuvaleikkeille. Kun hiiren painike vapautetaan, annetaan myös komento siirtyä yksi kehys eteenpäin. Pysäytä-painiketta painamalla voidaan siis siirtyä yksi kehys eteenpäin. Kuitenkin painikkeen suurin vaikutus on esityksen pysäyttäminen.

Jatka-painike jatkaa esityksen toistoa. Painikkeessa on komennot kullekin elokuvaleikkeelle jatkaa eteenpäin.

Lopeta-painike aiheuttaa koko esityksen sulkemisen. Tämä on toteutettu niin, että painiketta painettaessa annetaan komento `fscommand ("close");`, joka välittää close-komennon HTML-koodissa olevaan JavaScriptiin (Koodiesimerkki 4).

```
function weo_DoFSCommand(command, args) {
    window.close();
}
```

Koodiesimerkki 4 Osa HTML-koodissa olevasta JavaScriptistä

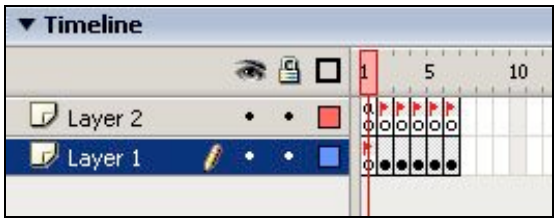
Flashistä voidaan julkaista HTML-sivu FSCommand-tuen kanssa, mikä tarkoittaa sitä, että HTML-koodiin tulee yllä oleva JavaScript-funktio, johon pitää vain lisätä `window.close()`-rivi. Funktiossa on paljon muutakin valmiina, mutta sillä ei ole tässä tapauksessa merkitystä. Funktio osaa ottaa Flashistä välitetyn komennon vastaan ja suorittaa tässä tapauksessa `window.close()`-komennon.

Kussakin navigointipalkin painikkeessa vaikutetaan *tekstit*-elokuvaleikkeen käyttäytymiseen tarkastamalla, ollaanko painikkeen päällä vai ei (Koodiesimerkki 5).

```
on (rollOver) {
    tekstit.gotoAndStop("edellinen");
}
on (rollOut) {
    tekstit.gotoAndStop("blank");
}
```

Koodiesimerkki 5 Painikkeissa oleva ActionScript

Tekstit-elokuvaleike koostuu kuudesta avainkehyksestä (Kuvio 7), joista viidessä on teksti, joka vastaa kutakin painiketta. Ensimmäinen avainkehys on tyhjä ja siihen tullaan silloin, kun ei olla painikkeiden päällä.

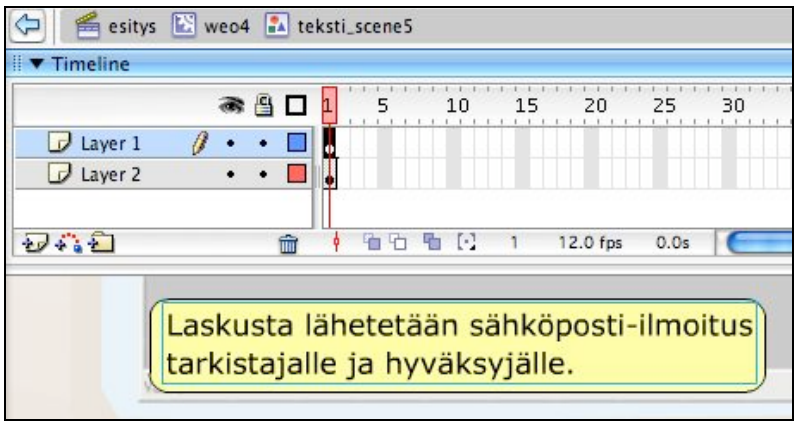


Kuvio 7 Tekstit-elokuvaleikkeen aikajana

5.1.4 Tekstit

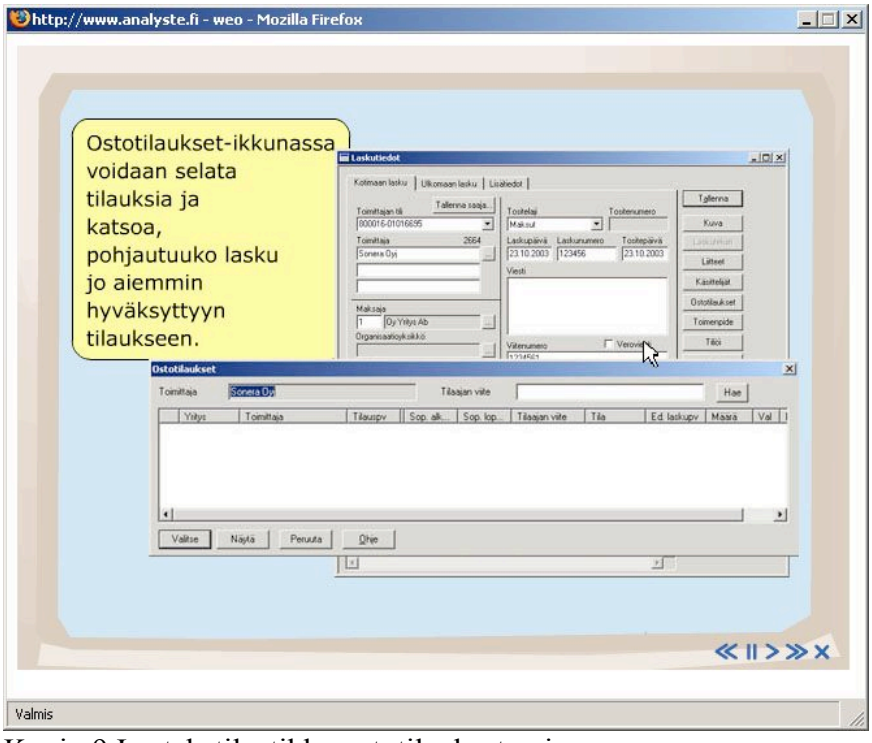
Esityksen tekstit on toteutettu tekemällä graafinen symboli, jonka sisään on tehty omalle tasolleen keltainen pohja ja toiselle tasolle kir-

joitettu teksti staattiseen tekstikenttään (Kuvio 8). Tekstin muotoilut on tehty suoraan Flashissä. Tekstin kirjasimeksi on valittu Verdana pistekooltaan 16. Tekstiä ei voi valita hiirellä esitystä katsottaessa.



Kuvio 8 Esityksen tekstilaatikko

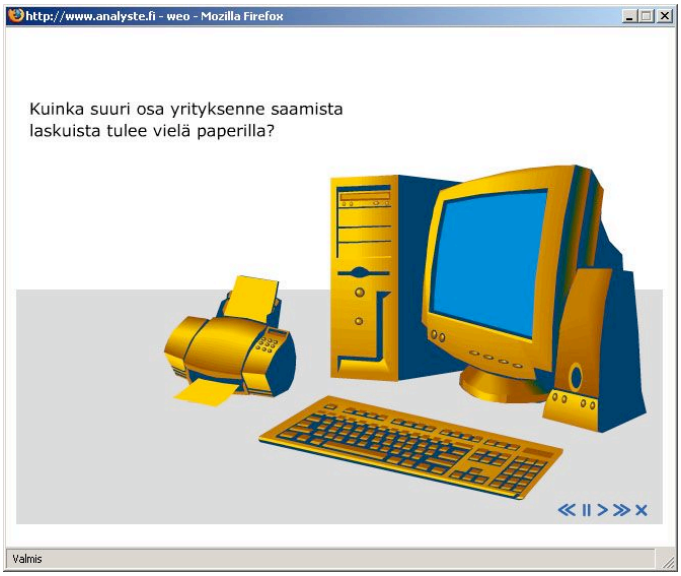
Jokaisesta tekstiosiosta on tehty oma graafinen symboli, jonka alpha-ominaisuutta on animoitu osion elokuvaleikkeen aikajanalla. Jokaisen tekstilaatikon pohja on tehty juuri sen kokoiseksi, että teksti on siihen mahtunut (Kuvio 9).



Kuvio 9 Iso tekstilaatikko ostotilaukset-osiossa

5.1.5 Kuvat

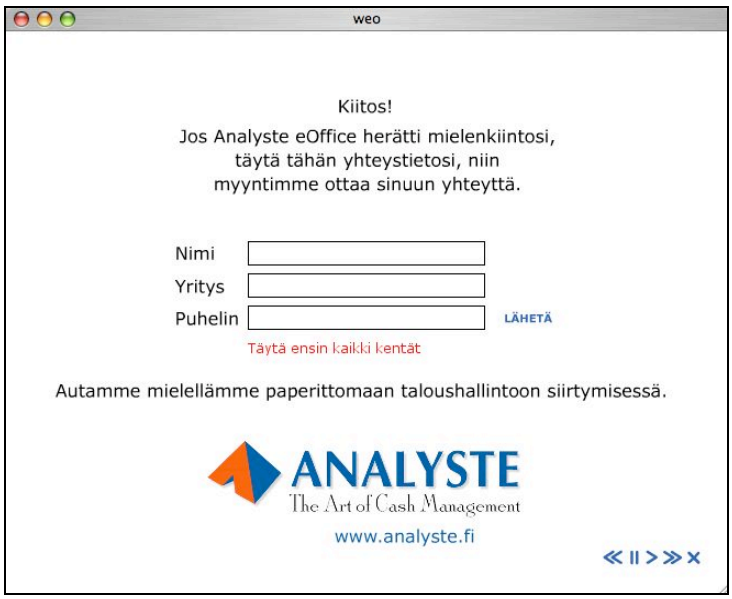
Esityksessä on käytetty paljon varsinaisen ohjelman näyttökaappauksia. Näiden lisäksi esityksessä on käytetty paljon piirrosgrafiikkaa havainnollistamaan työasemia ja tietokoneen näyttöä (Kuvio 10). Kuvat on ensin käsitelty Photoshopissa, minkä jälkeen ne on tuotu Flashiin.



Kuvio 10 Esityksessä on käytetty piirrosgrafiikkaa

5.1.6 Lomake

Esityksen yhteydenottolomakkeessa on kolme kenttää: *nimi*, *yritys* ja *puhelin* (Kuvio 11). Jos käyttäjä yrittää lähettää tyhjää lomaketta, pyydetään häntä täyttämään ensin kaikki kentät.



Kuvio 11 Yhteydenottolomake

Lähetä-painiketta painettaessa kutsutaan getURL()-metodilla *mailer.php*-tiedostoa, joka käsittelee Analysten WWW-sivuilla kaikki lomakkeet (Koodiesimerkki 6).

```
on (release) {
    if (nimi ne "" and yritys ne "" and puhelin ne "")
    {
        trace(nimi);
        getURL("https://www.analyste.fi/mailer.php",
"_self", "POST");
    } else {
        setProperty(fillall, _visible, true);
    }
}
```

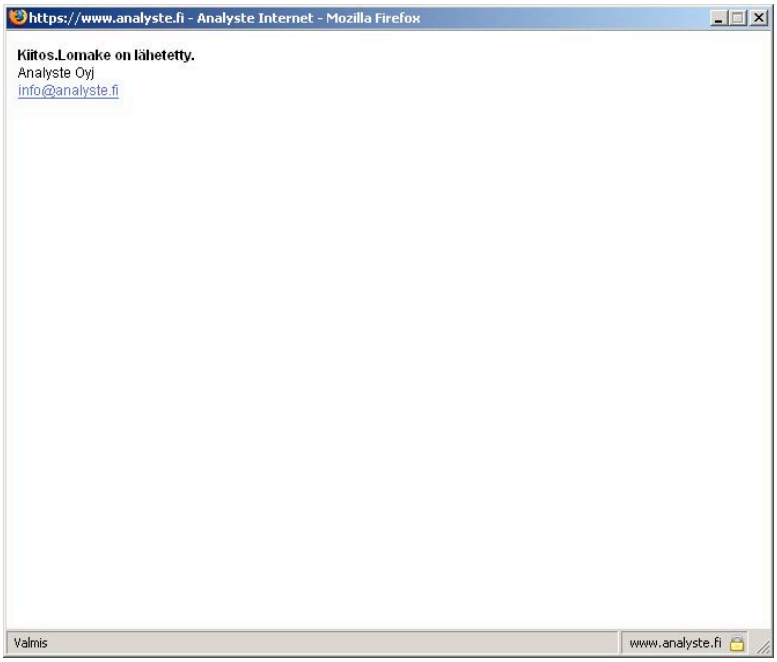
Koodiesimerkki 6 Lähetä-painikkeen ActionScript

Painiketta painettaessa tarkistetaan ensin, onko kaikki kentät täytetty. Jos ei ole, näytetään kenttien alapuolella *fillall*-niminen elokuvaleike, joka sisältää punaisen tekstin ”Täytä ensin kaikki kentät”. Lomake-sivulle tultaessa *fillall* ei näy, koska elokuvaleikkeen load-tapahtumankäsittelijään on lisätty koodiesimerkin 7 mukainen ActionScript. Elokuvaleikkeen näkyvyysominaisuus alustetaan ladattaessa.

```
onClipEvent (load) {
    setProperty(fillall, _visible, false);
}
```

Koodiesimerkki 7 Fillall-elokuvaleikkeessä oleva ActionScript

Kun käyttäjä lähettää lomakkeen, hän ohjautuu kiitos-sivulle, joka ei enää ole Flash-esityksen sisällä (Kuvio 12). *Mailer.php* tulostaa näytölle tiedon lomakkeen lähettämisestä.



Kuvio 12 Kiitos-sivu

5.2 Esityksen upottaminen WWW-sivuille

Esityksen SWF-tiedoston upotus HTML-sivuun on toteutettu Flashillä julkaisemalla. ”ODOTA HETKI. ESITYS LATAUTUU…” ja ”Analyste eOffice - hyödyt” toistuvat sivun HTML-koodissa niin monta kertaa, että olen poistanut yli sivun verran tekstiä alla olevasta koodiesimerkistä 8. Selaimessa pitää olla Flash Playeristä vähintään versio 5, jotta selain osaa näyttää esityksen.

```
<HTML>
<HEAD>
<TITLE>weo</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<!-- URL's used in the movie-->
<!-- text used in the movie-->
<!--ODOTA HETKI. ESITYS LATAUTUU... ODOTA HETKI. ESITYS
LATAUTUU... ODOTA HETKI. ESITYS LATAUTUU... ODOTA
HETKI. ESITYS LATAUTUU... ODOTA HETKI. ESITYS
LATAUTUU... ODOTA HETKI. ESITYS LATAUTUU... ODOTA
HETKI. ESITYS LATAUTUU... Analyste eOffice - hyödyt
Analyste eOffice - hyödyt Analyste eOffice - hyödyt
Analyste eOffice - hyödyt Analyste eOffice - hyödyt
Analyste eOffice - hyödyt Analyste eOffice - hyödyt
Analyste eOffice - hyödyt Analyste eOffice - hyödyt --
><OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-
444553540000"
codebase="http://download.macromedia.com/pub/shockwave/
cabs/flash/swflash.cab#version=5,0,0,0"
 WIDTH=640 HEIGHT=480>
 <PARAM NAME=movie VALUE="weo.swf"> <PARAM NAME=quality
VALUE=high> <PARAM NAME=bgcolor VALUE=#FFFFFF> <EMBED
src="weo.swf" quality=high bgcolor=#FFFFFF  WIDTH=640
HEIGHT=480 TYPE="application/x-shockwave-flash"
PLUGINSPAGE="http://www.macromedia.com/shockwave/downlo
ad/index.cgi?P1_Prod_Version=ShockwaveFlash"></EMBED>
</OBJECT>
</BODY>
</HTML>
```

Koodiesimerkki 8 Analyste eOffice -esityksen HTML-sivun koodi

5.3 Haastattelut

Analyste eOfficen esityksen arviointia varten haastattelin myyntipäällikkö Mika Tolosta Analysteltä ja talouspalvelusihteeri Erkka Koskista Tampereen kaupungin taloushallinnon palvelukeskukselta.

Myyjän näkökulmasta katsottuna esitys tukee myyntiä. Se olisi tosin voinut olla helpommin löydettävissä WWW-sivuilta. Esitys näyttää yksinkertaisesti laskun käsittelyn alusta loppuun ja kuinka helppoa se on.

Myyjän mielestä huonoa oli se, että esityksestä ei selvinnyt tarpeeksi selvästi Maksuliikenteen ostoreskontrapiirre. Mahdollisuus arkistoida muutakin aineistoa ostolaskujen lisäksi jäi myös epäselväksi.

Asiakkaan näkökulmasta katsottuna esitys näytti selkeältä ja siitä sai hyvän kuvan ohjelmasta. Toiminta-ajatus oli helposti ymmärrettävissä. Esitys antoi myös positiivisen mielikuvan Analystestä. Esityksen esilataaja oli havainnollistava, sillä lataaminen kesti jonkin aikaa. Lisäksi esityksessä käytetyt näyttökaappaukset olivat selkeitä ja hyviä. Esitys oli kokonaisuudessaan sopivan pituinen.

Haastateltavien mielestä esityksen navigointipalkki olisi voinut olla sellainen, missä pääsee suoraan tiettyyn aihealueeseen. Lisäksi tekstit olisivat voineet edetä ehkä aavistuksen verran hitaammin.

Kumpikaan haastateltavista ei kommentoinut esityksen piirroskuvia. He eivät myöskään ymmärtäneet, että navigointipalkin Taaksepäin- ja Eteenpäin-painikkeet olivat kelauspainikkeita. Molemmat olettivat, että painikkeista olisi päässyt isomman askeleen eteen- tai taaksepäin. Jos he olisivat painaneet kelauspainikkeita esityksen edetessä, he mitä todennäköisimmin olisivat ymmärtäneet painikkeiden toiminnan.

5.4 Oma arviointi

Haastattelujen lisäksi arvioin Analyste eOfficen esityksen myös itse. Arvioinnissa otan kantaa käytettävyyteen, ulkoasuun ja tekniseen toteutukseen.

Mielenkiinnon herättäjänä esitysten pitäisi mielestäni olla ajan tasalla olevia ja ajan hengen mukaisesti tehtyjä. Esimerkiksi koko esityksen rakenteen, esilataajan, navigointipalkin ja yhteydenottolomakkeen toteutuksen voisi tehdä teknisesti paremmin. Myös koko esityksen visuaalinen ilme kaipaisi uudistamista.

Avautuvan ikkunan otsikkona on ”weo”. Tavallinen käyttäjä tuskin tietää, mitä weo tarkoittaa, koska kyseessä on Analysten sisäinen tuotekoodi. Otsikkona olisi siis järkevämpää käyttää tekstiä, jonka käyttäjäkin ymmärtää.

Kun käyttäjä painaa hiiren oikealla esityksen päällä, avautuu valikko, josta voi vaikuttaa esityksen toistoon. Tätä valikkoa ei ole rajattu mitenkään (Kuvio 13).

Kuvio 13 Hiiren oikealla painettaessa avautuva valikko

Esitysrakenne

Varsinaisessa esityksessä kukin osio on sijoitettu pääaikajanalle sen pituisena kuin elokuvaleikkeen oma aikajana on. Elokuvaleikkeen kokonaan näyttämiseksi näin ei tarvitsisi tehdä, sillä elokuvaleike näytetään kokonaan, vaikka se pääaikajanalla kestäisi vain yhden kehyksen. Tällöin aikajanalla etenemistä pitäisi kontrolloida enemmän ActionScriptillä, jotta seuraavaan elokuvaleikkeeseen ei siirryttäisi ennen kuin parhaillaan katsotussa elokuvaleikkeessä on päästy loppuun. eOffice-esityksen toteutuksessa on kuitenkin tehty aikajanatoteutus toisin, jotta navigoinnin pikakelaukset eteen- ja taaksepäin toimisivat. Tällöin on voitu käyttää hyvin yksinkertaista ActionScriptiä.

Intro on ainoa osio, jossa on ActionScriptillä määritelty, mihin siirrytään kyseisen elokuvaleikkeen lopussa. Muissa osioissa näin ei ole tehty, sillä elokuvaleikkeen lopussa siirrytään automaattisesti pääaikajanalla seuraavaan kehykseen, jossa alkaa seuraava elokuvaleike. Jostain syystä *introon* on kuitenkin kyseinen komento vielä jäänyt.

Esilataaja

Esityksen esilataaja kertoo hyvin esityksen latautumisesta. Esilataajan toiminnassa on kuitenkin pieniä ongelmia, sillä palkki ei kasva yhtäjaksoisesti vaan isoin hyppäyksin. Esilataajan aikajanalla on useita kehyksiä pitkä animaatio tekstin näkymiseen liittyen. Tämä animaatio menee aina kerran läpi. Vasta sen jälkeen palkin tilanne päivittyy. Esityksen latausaika on isosta tiedostokoosta huolimatta kohtuullinen.

Navigointipalkki

Navigointipalkki on selkeä ja tunnistettavissa. Käytetyt symbolit ovat mielestäni selkeitä ja ymmärrettäviä. Kelauspainikkeiden toimintaa ei kuitenkaan välttämättä ymmärrä, ellei satu painamaan sitä esityksen sellaisessa kohdassa, missä tapahtuu jotain. Navigointipalkin avulla ei ainakaan suoraan pääse haluamaansa kohtaan.

Lopeta-painikkeesta ikkunan pitäisi sulkeutua. Tämä ei enää toimi, koska esityksen linkityksen yhteydessä HTML-koodissa pitäisi vastaanottaa JavaScriptillä FSCommand-metodi. Tarvittavan JavaScriptin saisi lähes automaattisesti julkaisemalla esityksen HTML-sivu FSCommand-tuen kanssa. Todennäköisesti jossain vaiheessa esityk-

sen julkaisun yhteydessä on kuitenkin julkaistu tavallinen HTML-sivu, joka on tallennettu alkuperäisen päälle.

Navigointipalkin muissa painikkeissa on monta riviä ActionScript-koodia, jolla on vaikutettu kuhunkin osioon. Ylläpitoa silmällä pitäen näin ison ja toistuvan koodin käyttö on hankalaa.

Tekstit

Esityksen tekstit ovat selkeitä, koska ne ovat tarpeeksi isolla ja sopivalla pohjalla luettavuuden kannalta. Ainoa ongelma tekstien suhteen on, että kukin lause on omana graafisena symbolina. Tämän takia mahdolliset tekstimuutokset pitää tehdä suoraan graafiseen symboliin, johon kyllä pääsee helposti käsiksi Flashin kirjastosta. Kukin lause on kuitenkin eri paikassa. Jos teksti muuttuu paljon, pitää myös pohjan kokoa muuttaa käsin samassa graafisessa symbolissa.

Kuvat

Esityksen piirrosgrafiikkakuvat ovat havainnollistavia ja tukevat hyvin esitystä, mutta ne ovat mielestäni silti aikansa eläneitä. Kuvat ovat Clip Art -kuvia. Analysten WWW-sivuilla käytetään paljon valokuvia, joten esitys on kuvien suhteen täysin eri henkinen. Esityksessä käytetyt näyttökaappaukset ovat selkeitä ja toimivia.

Lomake

Yhteydenottolomake on selkeä. Käyttäjän ei tarvitse miettiä, mihin tietoja pitäisi täyttää. Tyhjää lomaketta ei voi lähettää, vaan siitä tulee selkeä ilmoitus. Tosin ilmoituksen kirjasin olisi voinut olla saman tyylistä kuin sivulla muutenkin. Käyttäjän toiminnan helpottamiseksi kursori olisi voinut vilkkua valmiiksi ensimmäisessä kentässä sivulle tultaessa.

Kun sivulle tullaan, ilmestyvät tekstit omalla vuorollaan. Viimeisen lauseen ilmestyessä lause myös liikkuu muista poiketen hieman alaspäin. Lauseen näkymätöntä esiintymää ei ole siirretty oikealle kohdalle siinä vaiheessa, kun yläpuolelle on lisätty tilaa ”Täytä ensin kaikki kentät” -tekstille. Tästä muodostuu hieman huolimaton vaikutelma.

Lomakkeen lähetyksessä on käytetty samaa PHP-tiedostoa, jota käytetään muidenkin Analysten WWW-sivujen lomakkeiden käsittelyyn. Koska PHP-tiedostoa kutsutaan getURL()-metodilla, ohjaudutaan Flashistä kokonaan pois omalle PHP:n luomalle kiitos-sivulle. Tämän takia ei esitystä voi katsoa uudelleen muuten kuin avaamalla alkuperäinen linkki uudestaan.

Flashin upottaminen WWW-sivulle

HTML-koodi, johon esitys on linkitetty, ei ole standardien mukaista eikä se mene W3C:n validaattorista läpi. W3C:n validaattori löytää 11 virhettä. Tämän lisäksi koodissa on kommenteissa olevaa turhaa tekstiä sivukaupalla (ks. luku 5.2).

6 Uusi Analyste-ratkaisun Flash-esitys

Analyste-ratkaisun Flash-esityksessä kerrotaan lyhyesti ja selkeästi ratkaisun kokonaisajatus ja sen osioiden toiminta-ajatus. Esityksen tavoitteena on toimia tehokkaan sähköisen taloushallinnon laajan ratkaisun mielenkiinnon herättäjänä ja aktivoida potentiaalinen asiakas pyytämään lisätietoja. (Mäki-Petäjä 2006)

Kuvaan tässä luvussa tarkemmin esityksen merkittävimmät osa-alueet ja niiden toteutukset. Lopuksi esitän oman arvioni sekä haastattelemieni henkilöiden arviot esityksestä.

Analyste-ratkaisun Flash-esityksen toteuttamisessa huomioin eOfficen esityksessä havaitut puutteet ja haastatteluissa esille nousseet asiat. Asetin uudelle esitykselle kolmeksi tärkeimmäksi tavoitteeksi selkeän navigointipalkin, latausajan minimoinnin ja tekstien päivityshelppouden. Muita tavoitteita olivat toimiva esilataaja, esityksen upottaminen WWW-sivuille standardien mukaisesti ja visuaalisuuden parantaminen. Esityksen tekstit tulivat Analysten viestintäpäällikkö Maiju Mäki-Petäjältä.

Aloitin esityksen toteuttamisen suunnittelemalla ensin pääaikajanan ja tiedostojen rakenteen sekä toteuttamalla karkean ulkoasun esitykselle. Tämän jälkeen toteutin navigointipalkin, jonka toimintaa hioin lisää projektin edetessä. Sijoitin aluksi navigointipalkin tekstit suoraan painikkeisiin, mutta myöhemmin muutin tekstit tulemaan ActionScriptin kautta taulukosta. Kun navigointipalkki oli teknisesti toimiva, oli seuraava askel viimeistellä esityksen ulkoasu lähemmäksi lopullista tavoitetta.

Näiden vaiheiden jälkeen ryhdyin toteuttamaan tekstien ilmestymistä. Tämä oli ehkä työn isoin haaste, koska halusin saada tekstit Flashiin lähes automaattisesti niin, että tekstin määrästä riippumatta tekstit näkyisivät kokonaisuudessaan ja sijoittuisivat oikein muihin elementteihin nähden. Tekstien luonnin jälkeen etsin oikeat kuvat kuhunkin osioon. Seuraava luonnollinen askel olikin kuvien ja tekstien animointi.

Esilataajan ja yhteydenottolomakkeen toteutin muiden osioiden valmistuttua. Kun esityksen perusrunko oli valmis, toteutin erilliset *esimerkki*-elokuvaleikkeet. Tämän jälkeen varmistin, että navigointipalkin kelaus- ja tauko-painikkeet toimivat kaikissa tilanteissa. Eli kun esimerkkiä näytettiin, tauko-painikkeen piti vaikuttaa vain *esimerkki*-elokuvaleikkeeseen eikä pääesitykseen.

Viimeisiä vaiheita oli upottaa esitys yleisten WWW-standardien mukaisesti WWW-sivulle ja varmistaa, että esitys toimi eri selaimissa. Lisäksi säädin esityksen tekstien ajoitusta hitaammaksi ja varmistin, että esityksen kaikki osa-alueet toimivat niin kuin pitikin.

6.1 Toteutustavat

Analyste-ratkaisun esitys avautuu uuteen ikkunaan, joka on kooltaan 640 * 480 pikseliä. Esitys koostuu useasta SWF-tiedostosta, jotka hyödyntävät kahta AS-tiedostoa ja yhtä XML-, CSS- sekä PHP-tiedostoa (Taulukko 2).

Taulukko 2 Esityksen tiedostorakenne

Tiedosto	Merkitys
blank.swf index.htm	Esityksen avaamiseen liittyvät tiedostot, joista kerrotaan tarkemmin luvussa 6.2.
eoffice7.swf esitys.htm esitys.swf lomake.php navigointi.as tekstienlataus.as tekstit.xml tekstityyli.css esimerkit/ arkistoiminen.swf hankinta.swf maksaminen.swf matkalaskut.swf ostolaskut.swf pankkilogot.swf rahatilanne.swf	Esitykseen liittyvät tiedostot.

Esityksen perusrunko on *esitys.swf*-tiedostossa, jonka koko on 191 448 tavua eli 188 kt. Esimerkkien SWF-tiedostojen koko vaihtelee 20 kilotavusta 188 kilotavuun.

Index.htm-tiedosto sisältää vain linkin varsinaiseen esitykseen. Sivulla olevasta linkistä kutsutaan JavaScript-funktiota (Koodiesimerkki 9). Esitys voidaan linkittää *index.htm*-tiedostossa olevalla tavalla haluttuun kohtaan WWW-sivuilla.

```
function avaa_esitys() {
window.open("esitys.htm","_blank","toolbar=no,
location=no, directories=no, status=no, menubar=no,
scrollbars=no, resizable=no, copyhistory=no, width=640,
height=480");
}
```

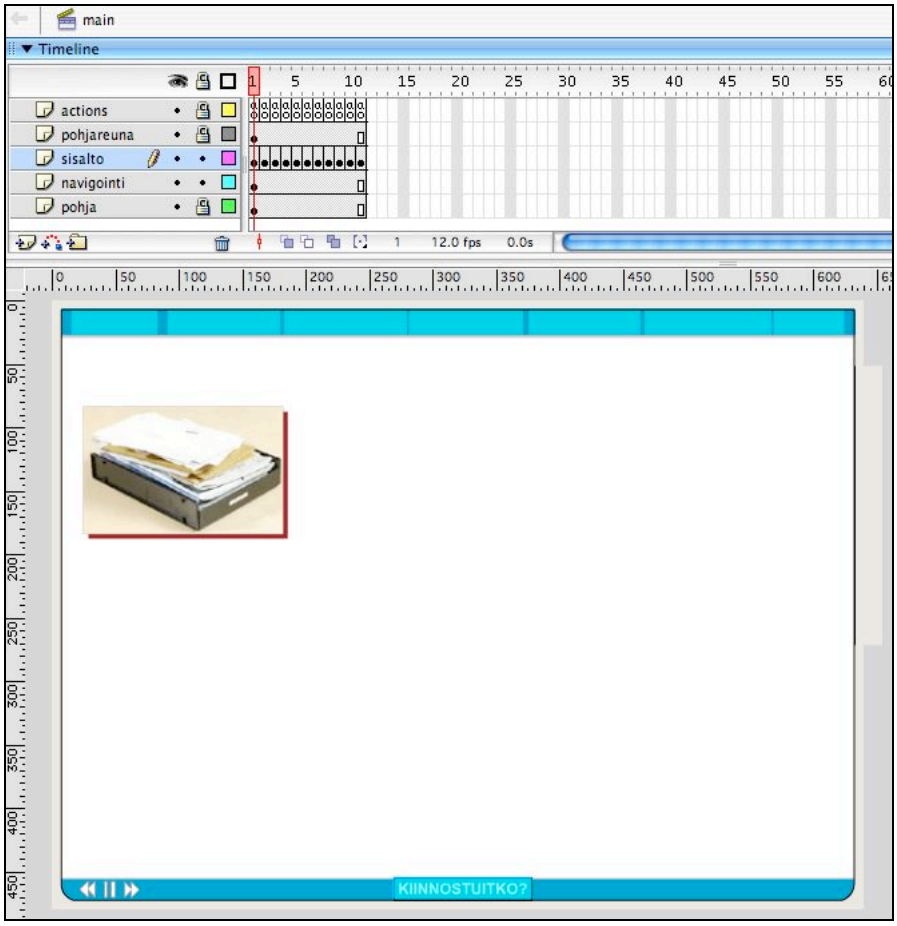
Koodiesimerkki 9 JavaScript-funktio ikkunan avaamiseen

Esityksen nopeutena on Flashin oletusnopeus 12 FPS eli 12 kehystä sekunnissa. Koko esitys kestää pääaikajanalla kehykseen 11 asti. Esi-

tyksen kesto ajallisesti on 142,9 sekuntia eli noin 2 minuuttia 20 sekuntia. Tähän lisätään vielä käyttäjän niin valitessa esimerkit, joiden kesto vaihtelee 6–12 sekunnin välillä.

6.1.1 Esitysrakenne

Esityksessä on kaksi kohtausta, ”preloader” ja ”main”. Ensimmäisessä on nimensä mukaisesti esilataaja, kun taas toisessa on esityksen varsinainen sisältö. *Main*-kohtauksessa on viisi tasoa (Kuvio 14), joista ylin *actions* on varattu ActionScripteille ja toiseksi alimmainen taso navigointipalkkia varten. Keskimmäinen taso on puolestaan sisältöä varten. Tässä tasossa on 11 avainkehystä, joista kukin sisältää tietyn osion esityksessä. Jäljelle jäävät kaksi tasoa muodostavat esityksen pohjan ja reunaviivan.



Kuvio 14 Esityksen pääaikajana ja näyttämö

Actions-tasolla on kussakin avainkehyksessä ActionScript, joka pysäyttää pääaikajanan ja määrittelee *sijainti*-muuttujan sekä kutsuu funktiota, joka vaikuttaa navigointipalkkiin (Koodiesimerkki 10).

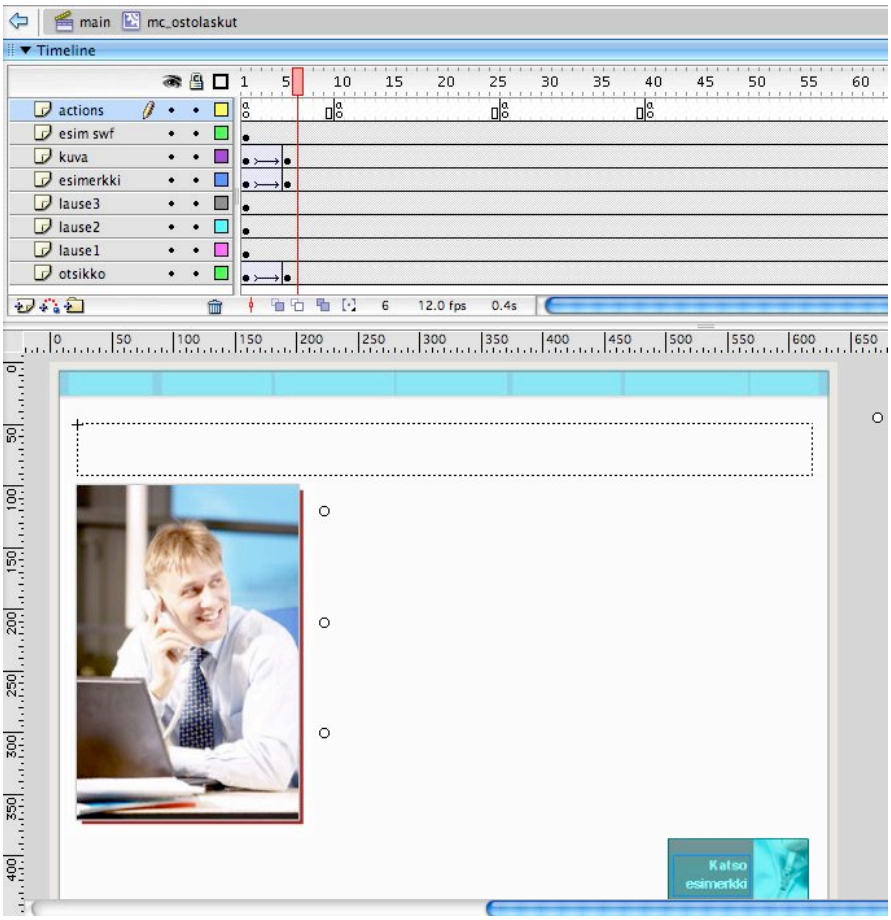
```
stop();
// OSTOLASKUT
var sijainti:String = "ostolaskut";

_root.navigointi.changeStatus(1);
```

Koodiesimerkki 10 Ostolaskut-kehyksessä oleva ActionScript

Kolmessa ensimmäisessä avainkehyksessä vaikutetaan lisäksi navigointipalkin näkyvyyteen. Avainkehykset on nimetty järjestyksessä ensimmäisestä viimeiseen *intro*, *tulevaisuus*, *hankinta*, *ostolaskut*, *matkalaskut*, *rahatilanne*, *maksaminen*, *arkistoiminen*, *hyodyt*, *lomake* ja *kiitos*.

Kussakin avainkehyksessä *sisalto*-tasolla on elokuvaleike, joka sisältää kyseisen osion asian. Kaikki elokuvaleikkeet on nimetty samalla logiikalla eli lisäämällä ”mc_” kunkin osion nimen eteen, esimerkiksi *mc_intro*.

Kukin elokuvaleike sisältää omalla aikajanallaan kyseisen osion tapahtumat kuten kuviossa 15, jossa on *ostolaskut*-elokuvaleikkeen aikajana.



Kuvio 15 Ostolaskut-elokuvaleikkeen aikajana ja näyttämö

Kunkin elokuvaleikkeen ensimmäisessä avainkehyksessä *actions*-tasolla on määritelty vielä *location*-muuttuja, joka on sama kuin *sijainti*-muuttuja pääaikajanalla. Lisäksi samalla ladataan ulkoinen *tekstienlataus.as*-tiedosto, jossa on määritelty tekstien näyttämiseen liittyvät toiminnot (Koodiesimerkki 11).

```
var location = "hankinta";

#include "tekstienlataus.as"

_root.navigointi.lataaSwf();
```

Koodiesimerkki 11 #include-komennolla ladataan ulkoinen AS-tiedosto

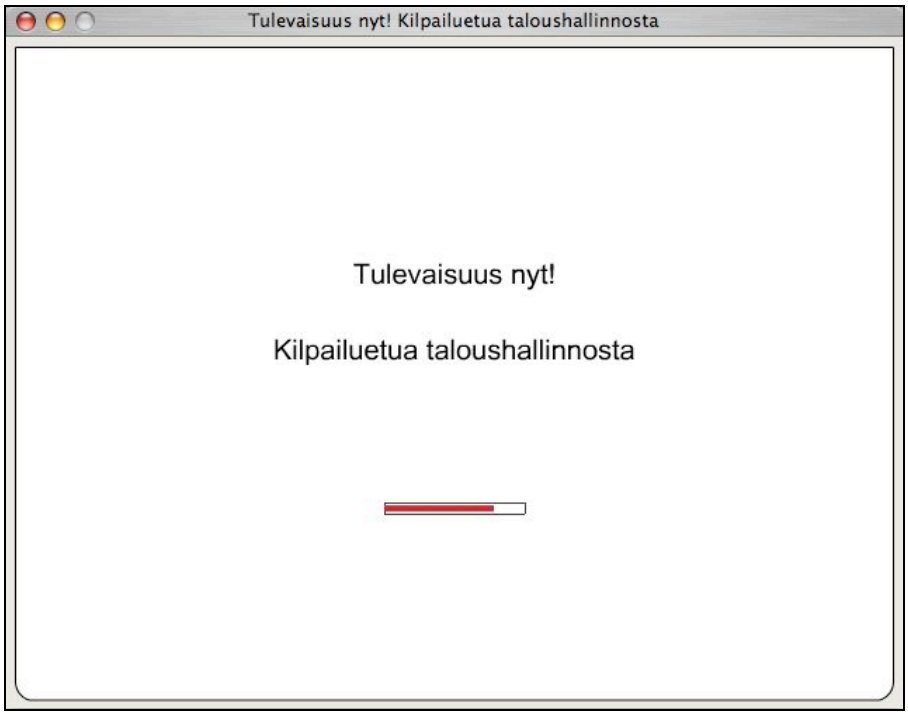
Kuusi elokuvaleikettä, *mc_hankinta*, *mc_ostolaskut*, *mc_matkalaskut*, *mc_rahatilanne*, *mc_maksaminen* ja *mc_arkistoiminen*, sisältävät myös painikkeen, josta saa näkyviin esimerkin varsinaisesta ohjelmasta näyttökaappauksineen. Esimerkit ovat erillisiä SWF-tiedostoja, joiden lataaminen on toteutettu siten, että heti ensimmäisessä avainkehyksessä kutsutaan *lataaSwf()*-funktiota. Kyseinen funktio löytyy navigointipalkista, jonka alussa on ladattu ulkoinen *navigointi.as*-tiedosto. Funktio lataa oikean SWF-tiedoston ja sijoittaa sen näyttämön ulkopuolelle. Kun käyttäjä painaa *Katso esimerkki* -painiketta, tulee esimerkki keskelle näyttämöä. Esimerkin edettyä loppuun se siirtyy taas näyttämön ulkopuolelle.

Esimerkit ovat rakenteeltaan hyvin yksinkertaisia. Niiden pääaikajanalla on liikkeet animoitu suoraan. Kustakin objektista tekstilaatikot mukaan lukien on tehty omat elokuvaleikkeet.

Esimerkit ovat tiedostokooltaan eri kokoisia eli toisen lataaminen kestää kauemmin kuin toisen. Kussakin esimerkissä on oma esilataaja, joka on samanlainen kuin pääesityksessäkin pohjaa lukuun ottamatta. Esimerkki ei myöskään ala toistaa itseään näyttämön ulkopuolella sen jälkeen, kun se on latautunut. Kun esitys ladataan heti osion alussa, se ehtii mitä todennäköisimmin latautua valmiiksi jo ennen kuin käyttäjä painaa *Katso esimerkki* -painiketta. Jos se ei ole ehtinyt latautua kokonaan, se on kuitenkin latautunut jo osittain. Käyttäjän ei siis tarvitse missään tapauksessa odottaa esimerkin latautumista alusta asti.

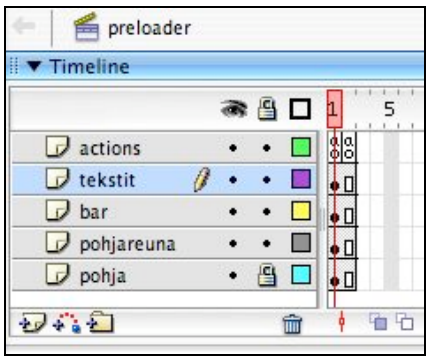
6.1.2 Esilataaja

Esilataajassa näkyy keskellä palkki, joka kasvaa. Palkin yläpuolella on kaksi lausetta ”Tulevaisuus nyt!” ja ”Kilpailuetua taloushallinnosta” (Kuvio 16).



Kuvio 16 Analyste-ratkaisun esitys latautuu

Esilataajan aikajana on kahden kehyksen pituinen ja koostuu viidestä tasosta (Kuvio 17). Kaksi tasoa muodostavat pohjan, yksi sisältää tekstit, jotka ovat staattisia tekstikenttiä ja yksi taso sisältää latautumista osoittavan palkin. Ylimmäinen taso on varattu ActionScripteille.



Kuvio 17 Esilataajan aikajana

Palkin sisällöstä on tehty elokuvaleike, jolle on annettu esiintymänimeksi *mc_bar*. Elokuvaleike koostuu sadasta kehyksestä, joista ensimmäisessä avainkehyksessä palkin sisältöä kuvaava grafiikka on yhden pikselin levyinen. Viimeisessä avainkehyksessä grafiikka on täyttänyt latautumista kuvaavan palkin kehyksen. Nämä avainkehykset on yhdistetty tekemällä väliin shape tween eli muodonmuutos, joka siis muuttaa muodon määrätyllä aikavälillä alkutilanteesta lopputilanteeseen. Tärkeintä tässä toteutustavassa on, että muodonmuutoksen on määritelty kestävän sata kehystä.

Esilataajan ensimmäisessä avainkehyksessä on määritelty esityksen latautumiseen liittyviä muuttujia (Koodiesimerkki 12). *Ladattu*-muuttujaan tallennetaan tieto siitä, paljonko on ladattu tähän mennessä. *Kaikki*-muuttujaan taas tallennetaan tieto esityksen koosta. Lisäksi *prosentti*-muuttujassa lasketaan yksinkertaisesti prosenttimäärä ladatusta määrästä ja tulos muutetaan vielä kokonaisluvuksi. Tämän jälkeen käsketään *mc_bar*-elokuvaleikkeen toistopään mennä ja pysähtyä omalla aikajanallaan *prosentti*-muuttujassa määriteltyyn kohtaan. Koska *prosentti*-muuttujan arvo on 0–100, *mc_bar*-elokuvaleikkeen aikajanan pitää kestää sata kehystä.

```
ladattu = getBytesLoaded();
kaikki = getBytesTotal();
prosentti = Int((ladattu/kaikki)*100);
mc_bar.gotoAndStop(prosentti);
```

Koodiesimerkki 12 Esilataajan ensimmäinen avainkehys

Toisessa avainkehyksessä tarkistetaan, onko esitys latautunut kokonaan vertaamalla *ladattu*-muuttujaa *kaikki*-muuttujaan. Jos ehto täyttyy, siirrytään *main*-kohtauksen ensimmäiseen kehykseen. Muussa tapauksessa siirrytään takaisin ensimmäiseen kehykseen nykyisessä kohtauksessa. Tätä toistetaan, kunnes esitys on latautunut.

Tällainen yksinkertainen esilataaja toimii hyvin, koska esitys on tiedostokooltaan riittävän pieni. Jos esityksen koko olisi ollut huomattavasti isompi, olisi esilataaja kannattanut tehdä siten, että esitystä olisi näytetty sitä mukaa, kun se latautuu.

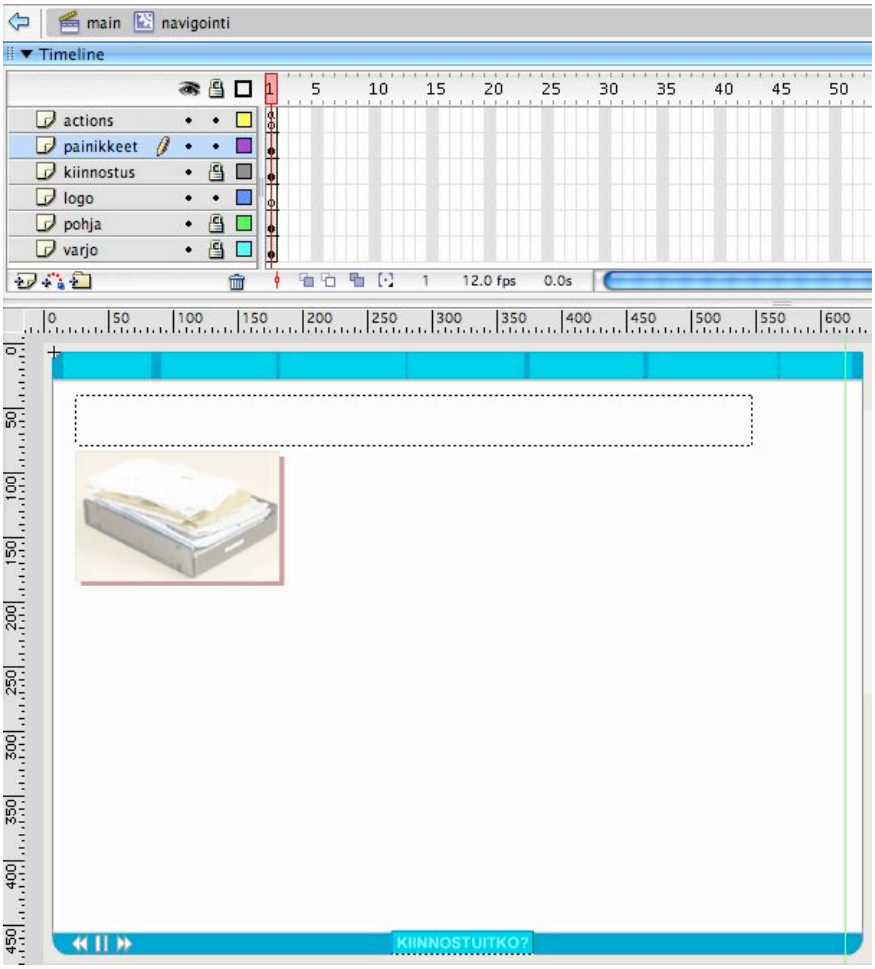
6.1.3 Navigointipalkki

Navigointipalkki esitetään toisessa osiossa heti *intron* jälkeen. Navigointipalkki on sen jälkeen koko ajan näkyvissä. Se koostuu yläpalkissa olevista painikkeista, vasemmassa alakulmassa olevista eteen- ja taaksepäin-painikkeista sekä toisto- ja tauko-painikkeista. Lisäksi alapalkissa on keskellä painike yhteydenottolomakkeeseen.

Navigointipalkissa painike muuttuu oranssiksi, kun se on valittu. Tällöin kyseinen painike myös poistetaan käytöstä eli yritettäessä painaa samaa valittua painiketta uudestaan ei tapahdu mitään, koska painike ei enää käyttäydykään kuin painike. Painiketta voi taas painaa, kun se muuttuu takaisin normaalitilaan. Tämä edellyttää, että jokin toinen painike tulee valituksi.

Navigointipalkki on tehty omaan elokuvaleikkeeseen, jonka esiintymänimeksi on annettu yksinkertaisesti *navigointi*. *Navigointi*-elokuvaleike koostuu kuudesta tasosta (Kuvio 18). Ylimmällä *actions*-tasolla ladataan *navigointi.as*-tiedosto komennolla `#include "navi-`

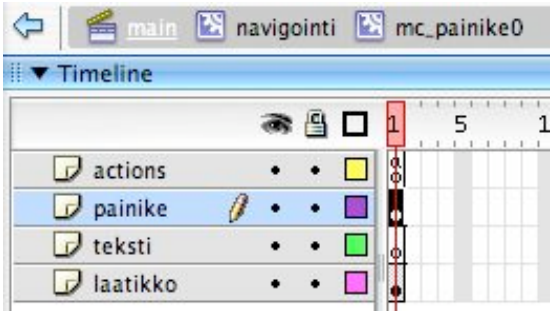
gointi.as". Tähän tiedostoon on sijoitettu kaikki navigointipalkin toimintaan liittyvät funktiot ja määrittelyt.



Kuvio 18 Navigointi-elokuvaleikkeen aikajana

Kaksi alimmaista tasoa muodostavat navigointipalkkien taustavärit ja varjot. Logoa ei vielä ole lisätty tapahtuneen yrityskaupan takia, mutta sille on varattu oma taso. Logon paikaksi ajateltiin vasenta alakulmaa painikkeiden yläpuolelle. Alapalkissa oleva *Kiinnostuitko*-painike on laitettu myös omalle tasolleen. *Painikkeet*-tasolle on sijoitettu kaikki muut painikkeet.

Yläpalkin painikkeet ovat omia elokuvaleikkeitään, joiden esiintymät on nimetty numerojärjestyksessä alkaen nimestä *mc_painike0*. *Mc_painike*-elokuvaleikkeet sisältävät neljä tasoa (Kuvio 19). *Laatikko*-tasolla on *pohja*-elokuvaleike, jolle on annettu esiintymänimeksi *pohja*. Siinä on määritelty pohjan värin muutokset sinisestä oranssiin ja takaisin. *Teksti*-tasolla ei ole mitään, koska tekstit tulevat ActionScriptin kautta *text_navi*-nimisiin tekstikenttiin.



Kuvio 19 Mc_painike0-elokuvaleikkeen aikajana

Actions-tasolla on määritelty tekstin väri valkoiseksi ja varmistettu, että tekstikentällä ei ole reunoja näkyvissä. ActionScriptillä on luotu väriolio, joka on sen jälkeen liitetty koskemaan tekstikenttää (Koodiesimerkki 13).

```
var my_color:Color = new Color(text_navi1);
my_color.setRGB(0xffffff);
this.text_navi1.border = false;
```
Koodiesimerkki 13 Väriolion luominen

Läpinäkyvässä painikkeessa, jolle on annettu esiintymänimeksi *btn*, on toimintoja liittyen painikkeen eri tiloihin (Koodiesimerkki 14). Kun hiiren kursori tuodaan painikkeen päälle, muuttuu tekstin väri oranssiksi. Samalla määritetään painikkeen *status_over*-muuttujan boolean-arvo todeksi. Vastaavasti painikkeen päältä pois mentäessä muutetaan väri takaisin valkoiseksi ja boolean-arvo epätodeksi.

```
on (release) {
    if (!_parent.painikkeenTila[0]) {
        _parent.painikkeenTila[0] = true;
        my_color.setRGB(0xffffff);
        _root.gotoAndStop("hankinta");
        _parent.play_status = true;
        _parent.mc_playpause.gotoAndStop("playing");
    }
}
on (rollOver) {
    my_color.setRGB(0xff861d);
    this.status_over = true;
}
on (rollOut) {
    my_color.setRGB(0xffffff);
    this.status_over = false;
}
```
Koodiesimerkki 14 Navigointipalkin painikkeen ActionScript

Kun painiketta painetaan ja päästetään irti, tarkistetaan painikkeen tila. Jos se on epätosi, se muutetaan todeksi. Samalla muutetaan väri valkoiseksi ja liikutaan pääaikajanalla oikeaan kohtaan. Lisäksi muutetaan *play_status*-muuttujan boolean-arvo todeksi ja tauko-painike näkyväksi.

Alapalkissa on kolme toistoon vaikuttavaa painiketta. *Taaksepäin*-elokuvaleike eli *mc_previous*-esiintymä sisältää painikkeen, jossa on määritelty liikkuminen edelliseen kehykseen pääaikajanalla. Ehtolauseessa estetään liikkuminen *preloader*-kohtaukseen (Koodiesimerkki 15).

```
on (release) {
    if (_root._currentframe > 3) {
        var nbr = _root._currentframe - 1;
        _root.gotoAndStop(nbr);
        _parent.play_status = true;
        _parent.mc_playpause.gotoAndStop("playing");
    }
}
```

Koodiesimerkki 15 Ehtolauseella rajoitetaan liikkumista

Samanlainen komento on *Eteenpäin*-elokuvaleikkeen eli *mc_next*-esiintymän painikkeessa, jonka erona on vain se, että *nbr*-muuttujassa yhdellä vähentämisen sijaan lisätään yksi. Ehtolauseessa estetään menemästä *kiitos*-sivulle.

Näiden painikkeiden väliin jää elokuvaleike, jonka esiintymänimi on *mc_playpause*. Se sisältää kaksi peräkkäin olevaa avainkehystä, joista toisessa on *Tauko*-painike ja toisessa on *Toista*-painike.

Kun esitys pysäytetään *Tauko*-painikkeella, siirrytään toiseen kehykseen, jolloin näkyviin tulee vain *Toista*-painike (Koodiesimerkki 16). Sama tapahtuu toisin päin. Painikkeita painettaessa tarkistetaan *esimerkki_status*-muuttujan boolean-arvo. Se on tosi silloin, kun käyttäjä on painanut *Katso esimerkki* -painiketta eli tarkempi esimerkki on näyttämöllä. Tällöin toistossa vaikutetaan esimerkin etenemiseen. Kun esimerkki ei ole näyttämöllä, vaikutetaan varsinaisen esityksen toistoon.

```
on (press, keyPress "<Space>") {
    gotoAndStop(2);
    if (_parent.esimerkki_status == false) {
        _parent.play_status = false;
    }
    else if (_parent.esimerkki_status == true) {
        _root["mc_" + _root.sijainti].esimerkki.stop();
    }
    _parent.pauseandplay();
}
```

Koodiesimerkki 16 Tauko-painikkeen ActionScript

Navigointi.as-tiedoston alussa määritellään *play_status*-muuttujan boolean-arvo todeksi ja *esimerkki_status*-muuttujan epätodeksi.

Yläpalkin painikkeiden tekstejä varten on luotu taulukko, jonka käsittelyssä on käytetty samaa tekniikkaa kuin esityksen varsinaisissa

teksteissä (Koodiesimerkki 17). Tästä tekniikasta kerrotaan tarkemmin luvussa 6.1.4.

```
// luodaan taulukko, jossa on navigoinnin tekstit
var navitekstit:Array = new Array(7);
navitekstit[0] = "HANKINTA";
navitekstit[1] = "OSTOLASKUT";
navitekstit[2] = "MATKALASKUT";
navitekstit[3] = "RAHATILANNE";
navitekstit[4] = "MAKSAMINEN";
navitekstit[5] = "ARKISTOIMINEN";
navitekstit[6] = "HYÖDYT";
```

Koodiesimerkki 17 Navitekstit-taulukon luominen

Painikkeella on kaksi varsinaista erilaista tilaa. Se on valittu tai sitä ei ole valittu. Tätä varten on luotu oma *painikkeenTila*-taulukko, jossa on tieto kunkin painikkeen tilasta. Painikkeen värin vaihtumista varten on puolestaan tehty kaksi eri funktiota. Tämän lisäksi painikkeen liikkumista varten on tehty kolme erilaista funktiota. Painikkeiden toimintaan vaikuttavat funktiot ovat liitteessä 1.

Kun käyttäjä painaa painiketta tai kun painike muuttuu automaattisesti aktiiviseksi siirryttäessä seuraavaan osioon, kutsutaan *changeStatus()*-nimistä funktiota, jolle välitetään aina painikkeen indeksiä kuvaava numero parametrina. *changeStatus()*-funktio muuttaa kyseisen painikkeen tilan todeksi ja kutsuu edelleen *checkStatus()*-funktiota välittäen sille saman indeksitiedon.

checkStatus()-funktiossa käydään for-silmukalla läpi *painikkeenTila*-taulukko. Kun käsittelyssä on painike, jota ei ollut valittu, tarkistetaan sen tila. Jos painike oli aiemmin valittuna, muutetaan sen tila epätodeksi. Samalla kutsutaan *colorToNormal()*-funktiota, joka muuttaa painikkeen värin takaisin siniseksi. Lisäksi painike aktivoidaan, jotta sitäkin voidaan taas painaa. Painiketta käsketään myös siirtymään takaisin ylös (Koodiesimerkki 18).

```
this["mc_painike" + i].onEnterFrame = function() {
    _parent.moveBack(i);
}
```

Koodiesimerkki 18 Painike siirretään takaisin normaalitilaan

moveBack()-funktiota kutsutaan hieman eri tavalla hyödyntäen elokuvaleikkeen *onEnterFrame*-ominaisuutta, joka toimii tapahtumankäsittelijänä. Tällöin funktio suoritetaan elokuvan kehysnopeuden eli FPS:n mukaan. Jos *moveBack()*-funktiota kutsuttaisiin suoraan ilman *onEnterFrame*-tapahtumankäsittelijää, ei tehosteen välivaiheita eli liikettä näkyisi ollenkaan.

Kun silmukassa saavutaan juuri valitun painikkeen kohdalle, kutsutaan *colorToOrange()*-funktiota, joka muuttaa painikkeen värin orans-

siksi. Samalla painike myös otetaan pois käytöstä eli painiketta ei voi painaa uudestaan. Lisäksi painikkeen tekstin väri muutetaan valkoiseksi ja varmistetaan, että *esimerkki_status*-muuttujan boolean-arvo on epätosi.

Kiinnostuitko-painikkeelle tehdään oma tarkastus, koska se ei liiku missään vaiheessa. Painike muuttaa ainoastaan väriä.

Painikkeiden liikkumista varten on määriteltävä liikkeiden rajat. Liikkeet tapahtuvat hidastuvalla liikkeellä. Painike liikkuu sitä hitaammin, mitä lähempänä se on kohderajaa (Koodiesimerkki 19).

```
var targetY:Number = 10;

// painikkeen alas liikuttaminen
function moveDown(nbr) {
    currentY = this["mc_painike" + nbr]._y;
    difY = currentY - targetY;
    setProperty(this["mc_painike" + nbr], _y, currentY
- (difY/3));
}
```

Koodiesimerkki 19 Painikkeen hidastuva liike

6.1.4 Tekstit

Esityksen perusrungon tekstit ovat *tekstit.xml*-tiedostossa. Liitteessä 2 on lyhyt esimerkki tiedoston rakenteesta. Tekstien näyttämiseen liittyvät toiminnot on sijoitettu *tekstienlataus.as*-tiedostoon, joka otetaan käyttöön #include-komennolla halutuissa kohdissa aikajanaa.

Otsikoille on tehty oma elokuvaleike, jonka esiintymänimeksi on aina kussakin osiossa annettu *mc_otsikko*. Elokuvaleike sisältää yhden dynaamisen tekstikentän, jonka esiintymänimeksi on annettu *otsikko*. Koska otsikoiden muotoilut on määritelty *tekstityyli.css*-tiedostossa, pitää tekstikenttää pystyä käsittelemään HTML:n tapaan eli ominaisuuksista pitää olla valittuna *Render text as HTML*.

Tyylitiedosto otetaan käyttöön ja liitetään haluttuun kohteeseen koodiesimerkin 20 mukaisesti.

```
tekstityyli = new TextField.StyleSheet();
tekstityyli.load("tekstityyli.css");
mc_otsikko.otsikko.styleSheet = tekstityyli;
```

Koodiesimerkki 20 Tyylitiedoston käyttöön ottaminen

Otsikot on määritelty <h1>-elementtien sisään, joten <h1>-elementille on annettu oma muotoilu tyylitiedostossa (Koodiesimerkki 21).

```
h1 {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 24px;
    font-weight: bold;
    color: #000000;
}
```

Koodiesimerkki 21 Otsikon muotoilu tyylitiedostossa

Muiden tekstien muotoilut on määritelty suoraan ActionScriptissä. XML-tiedoston lukeminen Flashiin on toteutettu määrittämällä ensin XML-olio (Koodiesimerkki 22).

```
esitysdata = new XML();
esitysdata.ignoreWhite = true;
esitysdata.onLoad = loadXmlData;
esitysdata.load("tekstit.xml");
```

Koodiesimerkki 22 XML-olion määrittäminen

Tämän jälkeen suoritetaan funktio *loadXmlData(success)*. Funktiossa luetaan XML-tiedosto läpi kahdella eri for-silmukalla. Ensin kuitenkin tallennetaan XML-tiedoston lapsielementtien määrä *pituus*-nimiseen muuttujaan.

Ensimmäisessä for-silmukassa tarkistetaan, missä osiossa ollaan, minkä jälkeen määritellään *otsikko*-tekstikentälle XML-tiedostosta oikea otsikko, joka on *otsikko*-attribuuttina (Koodiesimerkki 23). Samalla poimitaan saman kohtauksen lapsielementit omaan *kaikkilauseet*-muuttujaan.

```
for (var i = 0; i < pituus; i++) {
    // jos osion kohtaus vastaa sijaintia,
    // joka on maaritelty movie clipin alussa
    if
(this.firstChild.childNodes[i].attributes.kohtaus ==
location) {
        mc_otsikko.otsikko.htmlText = "<h1>" +
this.firstChild.childNodes[i].attributes.otsikko +
"</h1>";
        kaikkilauseet =
this.firstChild.childNodes[i].childNodes;
    }
}
```

Koodiesimerkki 23 XML-tiedoston läpikäyminen

Toisessa for-silmukassa poimitaan kukin lause *tekstit*-taulukkoon.

Funktiossa määritellään myös *my_fmt*-niminen TextFormat-olio, jota hyödynnetään lauseiden näyttämisessä (Koodiesimerkki 24).

```
var my_fmt:TextFormat = new TextFormat();

// maaritellaan objektin muotoiluasetuksia
my_fmt.font = "Arial";
my_fmt.size = 20;
my_fmt.leading = 1;
my_fmt.color = 0x000000;

if (location == "tulevaisuus") {
    my_fmt.align = "center";
}
```

Koodiesimerkki 24 TextFormat-olion määrittäminen

Lauseet olisi voitu näyttää suoraan XML-tiedostosta otsikon tapaan, jolloin myös lauseiden muotoilut olisi voitu toteuttaa CSS-tyylitiedoston kautta. Lauseet näytetään kuitenkin TextFormat-olion avulla, koska tällöin saadaan luotua tekstikenttien koot juuri tekstin määrän mukaan. Näin tekstin päivityksen yhteydessä ei tarvitse välittää siitä, mahtuuko teksti kokonaan tekstikenttään vai ei, koska tekstikentästä tulee juuri oikean kokoinen.

Toisaalta herää kysymys, miksi otsikoita ei tehty samalla tavalla. Niiden muotoilut on tehty CSS-tyylitiedoston kautta, koska tällainen mahdollisuus on hyvä pitää mielessä. Kun tekniikka on jo toteutettu, sitä on helppo soveltaa laajemmin. Jos esimerkiksi tuleekin tarve muuttaa esityksen luonnetta ja tekstien määrää tai ilmestymistä niin paljon, että TextFormat-oliota ei tarvittaisikaan, on muotoilut helppo muuttaa tulemaan CSS-tyylitiedoston kautta.

Kussakin osiossa on oikea määrä elokuvaleikkeitä, joiden esiintymät on nimetty tyyliin *mc_lause1*. Ne ovat alkutilanteessa tyhjiä, koska niiden sisältö luodaan vasta ActionScriptillä. Tekstikenttien leveys on määritetty omassa muuttujassa, jotta teksti saadaan mahtumaan näyttämölle. Jos tekstiä on enemmän kuin yhdelle riville määrättyyn leveyteen mahtuu, tekstikentän koko kasvaa alaspäin.

Tekstikentän leveyteen vaikuttaa kussakin osiossa olevan kuvan leveys. Jos osiossa on kapea pystysuunnassa oleva kuva, leveys on 400 pikseliä. Myös *introssa* ja *tulevaisuus*-osiossa on omat tietyt leveydet. Nämä tarkistukset suoritetaan ehtolauseissa muuttujan määrittelyn jälkeen.

Metrics-taulukkoon tallennetaan kunkin lauseen ulottuvuudet eli leveys ja korkeus (Koodiesimerkki 25). Lauseen pitää mahtua *tekstikentanLeveys*-muuttujassa määriteltyyn leveyteen.

```
for (var i = 0; i < metrics.length; i++) {
    metrics[i] = my_fmt.getTextExtent(tekstit[i],
tekstikentanLeveys);
}
```

Koodiesimerkki 25 Lauseiden ulottuvuuksien mittaaminen

Tekstikenttä luodaan kussakin osiossa olevaan elokuvaleikkeeseen *createTextField*-komennolla, jossa samalla annetaan esiintymänimeksi vastaavaa numerointia mukaillen *lause1*. Sulkeissa olevat tiedot ovat järjestyksessä esiintymän nimi, syvyys, x- ja y-koordinaatti, leveys ja korkeus (Koodiesimerkki 26).

```
mc_lause1.createTextField("lause1",
getNextHighestDepth(), 0, 0, tekstikentanLeveys,
metrics[0].textFieldHeight);
```

Koodiesimerkki 26 Tekstikentän luominen elokuvaleikkeen sisään

Leveydeksi tulee siis itse aiemmin määritelty arvo. Korkeudeksi tulee se, mikä *getTextExtent*-kohdassa laskettiin. Tekstikentissä on oltava myös rivinvaihto päällä, mikä onkin varmistettu koodiesimerkissä 27 olevalla tavalla kullekin lauseelle erikseen. Jälkimmäinen rivi määrittää tekstien ominaisuudet niin, että niitä ei pysty valitsemaan hiirellä.

```
mc_lause1.lause1.wordWrap = true;
mc_lause1.lause1.selectable = false;
```

Koodiesimerkki 27 Tekstikentän ominaisuuksien määrittäminen

Kun oikean kokoiset tekstikentät on luotu, niihin sijoitetaan oikeat lauseet *tekstit*-taulukosta, johon ne aiemmin sijoitettiin. Lisäksi määritellään, että tekstikenttä ottaa aiemmin luodun *my_fmt*-olion käyttöön (Koodiesimerkki 28).

```
mc_lause1.lause1.text = tekstit[0];
mc_lause1.lause1.setTextFormat(my_fmt);
```

Koodiesimerkki 28 Tekstin sijoittaminen my_fmt-oliolle

TextFormat-olion käyttämisen huonona puolena on animoinnin vaikeutuminen. Kun lauseet sisältäville elokuvaleikkeille yritti laittaa häivytysanimaation aikajanalla, ei tehoste toiminutkaan. Tähän löytyi ratkaisu, joka toimii hyvin tämän tapaisessa esityksessä. Häivytystehoste voidaan toteuttaa tekstin päällä olevan valkoisen laatikon avulla. Tekstikenttien lisäksi luodaan siis oikea määrä valkoisia laatikoita. Flashin kirjastossa on *valkoinen_laatikko*-niminen elokuvaleike, joka liitetään näyttämölle *whitebox*-nimellä (Koodiesimerkki 29).

```
for (var j = 0; j < kaikkilauseet.length; j++) {
    _root["mc_" +
_root.sijainti].attachMovie("valkoinen_laatikko",
"whitebox" + j, j + 1);
}
```

Koodiesimerkki 29 Valkoisia laatikoita tuodaan näyttämölle

Kustakin laatikosta tehdään juuri tekstikentän kokoinen hyödyntäen aiemmin saatuja mittoja. Laatikon x- ja y-koordinaatit määritellään samoiksi kuin tekstikentänkin koordinaatit. Ensimmäinen tekstikentän sisältävä elokuvaleike sijoitetaan samalle korkeudelle kuin kuva ja 15

pikselin päähän kuvasta sivusuunnassa. Seuraavat *mc_lause*-elokuvaleikkeet tulevat sivusuunnassa samalle kohdalle ja pystysuunnassa siten, että edelliseen elokuvaleikkeeseen jää tyhjää tilaa *offset*-muuttujassa määritellyt 18 pikseliä.

Valkoisia laatikoita häivytetään näkyvistä silloin, kun halutaan tekstin tulevan näkyviin. Vastaavasti laatikko tuodaan esiin, kun tekstin halutaan häipyvän. Koska nämä tehosteet koskevat useaa elokuvaleikettä, on funktioista tehty omat prototyypit (Koodiesimerkki 30).

```
MovieClip.prototype.fadeOut = function() {
    this.onEnterFrame = function() {
      if (this._alpha > 0) {
          this._alpha -= 7;
      }
    }
}
```

Koodiesimerkki 30 Yleisen funktion tekeminen

Tällöin funktion koodi on helppo liittää haluttuun elokuvaleikkeeseen käyttämällä funktiokutsussa elokuvaleikkeen esiintymänimeä (Koodiesimerkki 31). Tätä funktiota kutsutaan aikajanalta halutusta kohdasta.

```
whitebox0.fadeOut();
```

Koodiesimerkki 31 Funktiokutsussa käytetään esiintymänimeä

Introssa lauseet liikkuvat sivusuunnassa. Tätä liikettä varten on myös rakennettu prototyyppifunktio, johon välitetään lauseen indeksi, koska lauseen päällä olevaa valkoista laatikkoa on myös liikutettava samaan aikaan (Koodiesimerkki 32).

```
MovieClip.prototype.moveText = function(nbr) {
    this.onEnterFrame = function() {
      var targetX:Number = 248;
      currentX = this._x;
      difX = currentX - targetX;
      setProperty(this, _x, currentX - (difX/3));
      setProperty(_root.mc_intro["whitebox" + nbr], _x,
currentX - (difX/3));
    }
}
```

Koodiesimerkki 32 Yleinen funktio tekstin liikuttamiseksi

Lauseen liikkumista ja esiin tulemista varten funktiota kutsutaan aikajanalla halutussa kohdassa koodiesimerkin 33 mukaisella tavalla.

```
mc_lause1.moveText(0);
whitebox0.fadeOut();
```

Koodiesimerkki 33 Funktiokutsussa käytetään jälleen esiintymänimiä

6.1.5 Kuvat

Esityksessä on käytetty kuvituksena bittikarttakuvia, joille on Flashissä tehty reunat ja varjo kuvan korostamiseksi taustasta. Kuvat on optimoitu ensin Photoshopissa, minkä jälkeen ne on tuotu Flashiin. Kustakin kuvasta reunoineen ja varjoineen on tehty oma elokuvaleike. Lisäksi *introa* ja *tulevaisuus*-osiota lukuun ottamatta jokaisen kuvan esiintymänimeksi on annettu *mc_kuva*. Tällöin sivulla olevat lauseet osaavat sijoittua oikealle kohdalle.

Esimerkeissä on käytetty näyttökaappauksia varsinaisesta ohjelmasta. Niiden näyttämiseen liittyvät funktiot sijaitsevat *navigointi.as*-tiedostossa. Esimerkit ladataankin *esimerkki*-nimiseen elokuvaleikkeeseen *lataaSwf()*-funktiota kutsumalla (Koodiesimerkki 34). Tämä elokuvaleike sijaitsee alkutilanteessa näyttämön ulkopuolella.

```
function lataaSwf() {
    loadMovie("esimerkit/" + _root.sijainti + ".swf",
_root["mc_" + _root.sijainti].esimerkki);
}
```

Koodiesimerkki 34 Funktio esimerkkien lataamiseksi

Kun käyttäjä painaa *Katso esimerkki* -painiketta, kutsutaan *katsoEsimerkki()*-funktiota, joka pysäyttää pääesityksen toiston ja aloittaa esimerkin toiston kutsumalla kuitenkin ensin *sijoitaKeskelle()*-funktiota, joka tuo esimerkin näyttämölle käyttäjän nähtäväksi. Esimerkkien näyttämisen yhteydessä pitää lisäksi vaihtaa elokuvaleikkeen syvyysarvoa, jotta se näkyisi myös tekstien päällä (Koodiesimerkki 35).

```
_root["mc_" + _root.sijainti].esimerkki.swapDepths(10);
```

Koodiesimerkki 35 Elokuvaleikkeen syvyysarvon vaihtaminen

Kun esimerkki siirretään piiloon, syvyysarvoksi vaihdetaan -15 000.

Maksaminen-osiossa pankkien logot ladataan omasta SWF-tiedostosta kutsumalla *lataaPankit()*-funktiota, joka on määritelty *tekstienlataus.as*-tiedostossa (Koodiesimerkki 36). Funktiossa luodaan uusi *pankit*-niminen elokuvaleike, joka osion lopussa poistetaan `pankit.removeMovieClip();` -komennolla.

```
function lataaPankit() {
    createEmptyMovieClip("pankit", 9);
    loadMovie("esimerkit/pankkilogot.swf", "pankit");
    pankit._x = 260;
    pankit._y = 164;
}
```

Koodiesimerkki 36 Pankkien logojen lataaminen

6.1.6 Lomake

Yhteydenottolomakkeessa on kuusi kenttää, jotka ovat *nimi*, *tehtävä*, *yritys*, *sähköposti*, *puhelin* ja *viesti* (Kuvio 20). Kursori vilkkuu sivulle tultaessa ensimmäisessä kentässä. Tämä edellyttää kuitenkin, että käyttäjä on jossain vaiheessa esitystä edes kerran painanut hiiren näppäintä Flashin sisällä. Tyhjää lomaketta ei voi lähettää, koska pakollisia kenttiä ovat *nimi* ja *sähköposti* tai *puhelin*. Jos pakollisia kenttiä ei ole täytetty, tulee lomakkeen alapuolelle näkyviin ilmoitus punaisella tekstillä.

Kuvio 20 Yhteydenottolomake

Ilmoitus puuttuvien kenttien täyttämisestä on elokuvaleike, jonka esiintymänimi on *mc_error*. Lomakkeen aikajanalla *actions*-tasolla on määritelty alkutilanne, jossa elokuvaleikkeen näkyvyysarvo on määritelty epätodeksi (Koodiesimerkki 37).

```
// virheilmoitus ei nay
mc_error._visible = false;

// asetetaan kursori ensimmaiseen tekstikenttaan
Selection.setFocus("nimi_txt");
```
Koodiesimerkki 37 Alkuasetuksia lomakkeelle tultaessa

Selection.setFocus()-funktiolla saadaan kursori kohdistettua haluttuun kohtaan.

Lomake tarkistetaan, kun käyttäjä painaa Lähetä-painiketta (Koodiesimerkki 38). Tällöin ehtolauseessa tarkistetaan kolmen kentän si-

sällön pituus. Jos lomake on täytetty muodollisesti oikein, ladataan *lomake.php*-tiedosto *loadVariablesNum()*-metodilla, jolloin Flash käsittelee PHP-tiedoston suoraan. Tämän jälkeen siirrytäänkin pääaikajanalla *kiitos*-sivulle.

```
on (release) {
    if (nimi.length > 0 && email.length > 0 ||
puhelin.length > 0) {
        loadVariablesNum("lomake.php", 0, "POST");

        _root.gotoAndStop("kiitos");
    } else {
        mc_error._visible = true;
        var emptyField:String;

        // ensimmaisesta tyhjasta kentasta emptyField
        if (nimi == EMPTY) {
            emptyField = "nimi_txt";
        }
        else if (email == EMPTY) {
            emptyField = "email_txt";
        }
        else if (puhelin == EMPTY) {
            emptyField = "puhelin_txt";
        }

        // asetetaan kursori tyhjaan kenttaan
        Selection.setFocus(emptyField);
    }
}
```

Koodiesimerkki 38 Lähetä-painikkeen ActionScript

Jos ehtolause ei täyty, näytetään *mc_error*-elokuvaleike. Samalla tarkistetaan, mikä pakollisista kentistä on tyhjä, ja siirretään kursori ensimmäiseen tyhjään pakolliseen kenttään.

Lomakkeen käsittely hoidetaan PHP:lla, jossa on määritelty vastaanottaja, viestin aihe ja otsikkotiedot, sekä viestin sisältö. Viestin sisältöön poimitaan lomakkeelta täytetyt tiedot, minkä jälkeen viesti lähetetään *mail()*-funktiolla (Koodiesimerkki 39).

```
<?php

// viestin vastaanottaja ja aihe
$to = "info@analyste.fi";
$subject = "Yhteydenotto Flash-esityksesta";

// viestin otsikkotiedot
$headers = "From: Flash-esitys\r\n";
$headers .= "Reply-To: " . $_POST["email"] . "\r\n";

// viestin sisalto
$message = "Nimi: " . $_POST["nimi"] . "\r\n";
$message .= "Tehtava: " . $_POST["tehtava"] . "\r\n";
$message .= "Yritys: " . $_POST["yritys"] . "\r\n";
$message .= "Sahkoposti: " . $_POST["email"] . "\r\n";
$message .= "Puhelin: " . $_POST["puhelin"] . "\r\n";
$message .= "Viesti: " . $_POST["viesti"] . "\r\n";

// viestin lahetys
mail($to, $subject, $message, $headers);

?>
```

Koodiesimerkki 39 Lomakkeen lähetys hoidetaan PHP:lla

6.2 Esityksen upottaminen WWW-sivuille

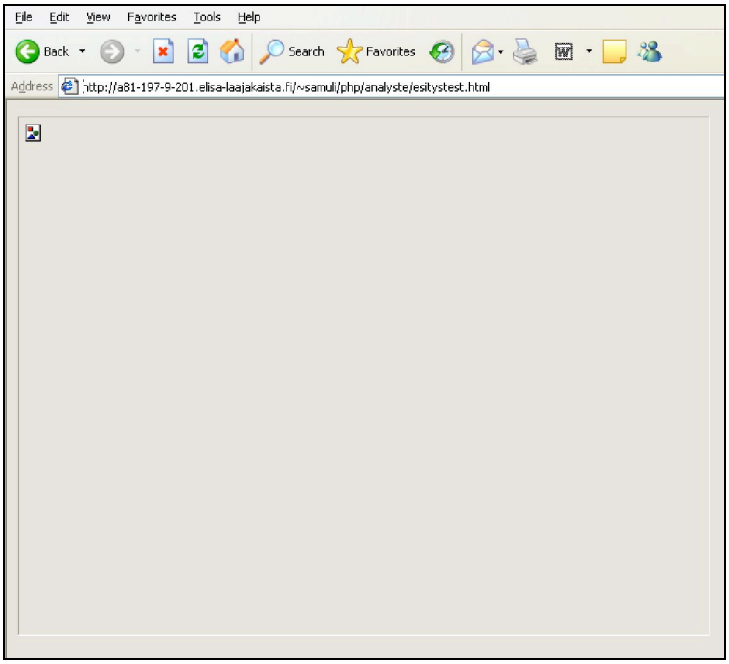
Koska Flashillä julkaistu HTML-tiedosto ei ole yleisten WWW-standardien mukainen, on SWF-tiedoston linkittäminen toteutettu hyödyntämällä Drew McLellanin keksimää Satay-tapaa (McLellan 2002) koodiesimerkin 40 mukaisesti.

```
<!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"
xml:lang="fi" lang="fi">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
<title>Tulevaisuus nyt! Kilpailuetua
taloushallinnosta</title>
<style type="text/css">body { margin: 0; }</style>
</head>
<body bgcolor="#e8e5de">
<object type="application/x-shockwave-flash"
data="eoffice7.swf?path=esitys.swf" width="640"
height="480">
<param name="movie"
value="eoffice7.swf?path=esitys.swf" />
<param name="menu" value="false" />
</object>
</body>
</html>
```

Koodiesimerkki 40 Esityksen HTML-koodi

Linkityksessä on käytetty vain <object>-elementtiä, jonka sisällön tyypiksi on asetettu *application/x-shockwave-flash*. Jotta esitys näkyisi myös Netscapessa, on lisättävä vielä *data=esitys.swf*.

Koodiesimerkissä 40 on tiedostoviittauksina *eoffice7.swf?path=esitys.swf*. Jos viittauksissa käyttäisi pelkästään varsinaisen esityksen tiedostonimeä eli tässä tapauksessa *esitys.swf*, ei Flashin streamaus toimisi Windowsin Internet Explorer -selaimissa. Esitys latautuisi kokonaan ennen kuin näyttäisi sitä yhtään. Normaalissa tilanteessa pitäisi siis näkyä esilataaja, mutta nyt näkyisikin vain tyhjä laatikko kuvion 21 mukaisesti.



Kuvio 21 Ongelma Internet Explorer -selaimessa

Satay-tavan mukaan ongelman voi ratkaista niin, että tehdään pieni tyhjä SWF-tiedosto, joka lataa varsinaisen esityksen. McLellan käytti tästä pienestä tyhjästä SWF-tiedostosta nimeä säiliö. Tämän ohjeen mukaisesti tehtiin tyhjä SWF-tiedosto, joka oli saman kokoinen kuin *esitys.swf*. Ensimmäisessä ja ainoassa avainkehyksessä on hyvin lyhyt ActionScript-koodi, joka lataa *path*-muuttujassa määritellyn kohteen tasolle 0 (Koodiesimerkki 41).

```
loadMovieNum(path,0);
```

Koodiesimerkki 41 Säiliönä toimivassa SWF-tiedostossa on vain yksi ActionScript

Tehty säiliö julkaistiin nimellä *eoffice7.swf*, jonka tiedostokooksi tuli vain 63 tavua. Tästä johtuen lopullisessa linkityksessä pitää tiedostoviittauksissa käyttää yhdistelmää *eoffice7.swf?path=esitys.swf*, jolloin Flash osaa poimia linkistä *path*-muuttujan arvon.

Tässä ratkaisussa oli vielä yksi iso ongelma. *Codebase*-attribuutin puuttuminen aiheuttaa sen, että käyttäjää ei kehoteta lataamaan tarvittavaa Flash-laajennusta, jos selaimessa ei sellaista ole. McLellanin ratkaisun mukaisesti on tehty yksi ylimääräinen tyhjä *blank.swf*-tiedosto, jonka linkitys on tehty Flash-ohjelman julkaisemalla HTML-koodilla. Tällöin *codebase*-attribuutti oli mukana. *Blank.swf*-tiedoston koko oli vain 25 tavua ja se on sijoitettu WWW-sivustolla sen linkin yhteyteen, mistä varsinainen esitys olisi tarkoitus avata uuteen ikkunaan eli tässä tapauksessa *index.htm*-tiedostoon. Ratkaisu ei ollut puhtain mahdollinen vaikka olikin käytännöllinen.

Tämän lisäksi on rajattu esityksen päällä hiiren oikealla painettaessa avautuvaa valikkoa lisäämällä HTML-koodiin `<param name="menu" value="false" />`. Näin käyttäjä ei voi valikon kautta vaikuttaa esityksen toistoon, koska tarkoitus on, että hän käyttää navigointipalkkia siihen tarkoitukseen (Kuvio 22).

Kuvio 22 Suppea valikko

6.3 Haastattelut

Analyste-ratkaisun esityksen arviointia varten haastattelin Analysten yhteyspäällikköä Tuija Hatakkaa ja DokuMentorin teknistä dokumentoijaa Heli Schroderusta, joka tittelistään huolimatta hoitaa DokuMentorin taloushallintoa osto- ja myyntireskontran sekä laskutuksen osalta.

Analysten näkökulmasta katsottuna esitys tukee hyvin myyntiä. Tämä kuitenkin edellyttää, että käyttäjä tuntee aihealueen. Asiakkaan on helppo saada käsitys Analyste-ratkaisusta katsomalla Flash-esitys WWW-sivuilta ennen kuin on nähnyt varsinaista ohjelmistoa myyjän esittelemänä. Esitys toimii myös hyvin messuilla ja seminaareissa.

Esityksen ulkoasu on molempien haastateltavien mukaan selkeä, esimerkit ovat hyviä ja esitys etenee loogisesti. Pysäytyspainikkeetkin ovat hyvät, koska käyttäjä voi helposti kontrolloida esitystä niiden avulla.

Ensivaikutelmana oli, että käyttäjä ei ehdi lukea tekstejä, koska alussa tulee niin monta lausetta nopealla aikataululla. Loppujen lopuksi tekstien ajoitus oli kuitenkin ihan hyvä, koska ne ovat näkyvissä tarpeeksi kauan. Huonona puolena oli esityksen pieni koko.

Asiakkaan näkökulmasta katsottuna esitys oli selkeä ja se toimi hyvin. Esityksen perusteella ymmärsi myös hyvin Analyste-ratkaisun toiminta-ajatuksen. Lisäksi esimerkeissä olleet tekstilaatikot ohjasivat hyvin katsetta oikeaan paikkaan. Esityksen latautumista ei tarvinnut juurikaan odottaa. Siksi esilataajaan ei oikeastaan edes ehtinyt kiinnittää huomiota. Näiden lisäksi esimerkeissä käytetyt näyttökaappaukset olivat selkeitä.

Molemmat haastateltavat osasivat käyttää hyvin esityksen navigointipalkkia ja navigoida oikeaan kohtaan pyytäessäni heitä esityksen lopussa katsomaan vielä ostolaskujen esimerkin. Myös esityksen aikana he käyttivät navigointipalkkia ja *Katso esimerkki* -painikkeita tottuneen oloisesti.

Molemmat haastateltavat osasivat navigoida ostolaskujen esimerkkiin nopeasti. Tämän jälkeen asiakas ei kuitenkaan heti osannutkaan siirtyä takaisin lomakkeelle. Hän päätti katsoa ensin *hyödyt*-osion, minkä jälkeen lomake tuli automaattisesti. Vasta sen jälkeen hän huomasi, että alapalkista olisi päässyt lomakkeeseen suoraan.

6.4 Oma arviointi

Haastattelujen lisäksi arvioin Analyste-ratkaisun esityksen myös itse. Arvioinnissa otan kantaa käytettävyyteen, ulkoasuun ja tekniseen toteutukseen.

Avautuvan ikkunan otsikkona on ”Tulevaisuus nyt! Taloushallinnon haaste”. Otsikko ei sisällä nyt mitään tavalliselle käyttäjälle vieraita lyhenteitä. Käyttäjä ei myöskään pysty vaikuttamaan esityksen toistoon, kun hän avaa valikon hiiren oikeanpuoleista painiketta painamalla. Näin käyttäjää ohjataan tarkoituksenmukaisesti käyttämään esityksen omaa navigointipalkkia.

Esitys aukeaa saman kokoiseen ikkunaan kuin Analyste eOfficenkin esitys. Mielestäni ikkunan koko on riittävä ja toimii tässä tapauksessa hyvin. Ikkunan kokoa mietittäessä tavoitteena kannattaa edelleenkin pitää sitä, että se mahtuu hyvin näytölle, kun näytön resoluutiona on 800 * 600.

Esitysrakenne

Esitysrakenne mahdollistaa helpon päivittämisen. Etenkin tekstien muutokset ovat helposti tehtävissä, koska ei tarvitse etsiä oikeaa kohtaa FLA-tiedostosta. Lisäksi toiminnallisuudet on koottu kahteen AS-tiedostoon, mikä helpottaa toiminnallisuuden päivittämistä. Jatkokehitystoimenpiteinä voisi miettiä, voiko funktioista tehdä luokkia ja voiko prototyypit toteuttaa erilaisilla funktioilla.

Latausaikaa silmällä pitäen esitys toimii erittäin hyvin. Mielestäni on hyvä, että esimerkkiä ladataan jo samalla, kun käyttäjä katsoo jotain muuta.

Esilataaja

Esilataaja on hyvin informatiivinen, koska palkki päivittyy koko ajan. Lisäksi esilataajassa siirretään hyvin käyttäjän huomio latauksesta muualle kahdella lauseella. Tosin todella nopeilla Internet-yhteyksillä esilataajassa olevia lauseita ei välttämättä ehdi lukea, koska esitys latautuu pienen kokonsa ansiosta todella nopeasti. Esilataajassa olevat lauseet ovat kuitenkin avautuvan ikkunan otsikkona.

Navigointipalkki

Navigointi esityksessä on helppoa, koska navigointipalkista näkee koko ajan, missä osa-alueessa kulloinkin ollaan. Navigointipalkki on mielestäni tunnistettava, sen toiminnan oppii todella nopeasti eikä sen toimintaa tarvitse opetella alusta asti myöhemmin. Navigointipalkki on mielestäni myös johdonmukainen, vaikka siinä on hieman erilaisia painikkeita.

Tekstit

Esityksessä käytetty teksti on selkeää ja hyvin erotettavissa. Vaikka pohjana on käytetty täysin valkoista, ei se mielestäni haittaa lukemista, koska teksti on tarpeeksi isoa, eikä sitä ole liian paljon. Esimerkkien tekstit ovatkin keltaisella pohjalla, mikä erottaa ne hyvin näyttökaappaustaustoista.

Tekstien päivittäminen on tehty helpoksi, koska tekstien muuttamisen yhteydessä ei tarvitse miettiä, mahtuuko teksti kokonaan näkyviin tekstikenttään. Ainoastaan esimerkkien tekstit on tehty suoraan esimerkkeihin omina elokuvaleikkeinään. Koska esimerkkien rakenne on hyvin yksinkertainen, on niihinkin helppo tehdä tarvittavat tekstimuutokset.

Kuvat

Kuvat ovat raikkaita ja tukevat hyvin kutakin osiota. Kuvat on vielä pienellä tehosteella tuotu paremmin esiin taustasta. Lisäksi kuvat ovat saman henkisiä kuin Analysten ja BasWaren WWW-sivuilla käytetyt kuvat. Näyttökaappaukset ovat myös onnistuneita ja selkeitä.

Lomake

Yhteydenottolomake toimii hyvin ja lomakkeen kentät ovat hyvin tunnistettavissa. Tyhjää lomaketta ei pysty lähettämään, tosin pakolliset kentät voisi osoittaa käyttäjälle jo valmiiksi esimerkiksi pienellä tähdellä. Käyttäjä ei myöskään joudu nyt esityksestä pois lomakkeen lähettämisen jälkeen. Esitys on siis mahdollista katsoa uudestaan lomakkeen lähettämisen jälkeenkin.

Lomakkeelle tultaessa kursori vilkkuu jo valmiiksi ensimmäisessä kentässä. Yleensä käyttäjä saattaa ryhtyä kirjoittamaan heti tarkistamatta ensin kursorin sijaintia, joten jos kursori ei vielä alkuvaiheessa vilku missään kentässä, käyttäjän kirjoitus ei tallennu mihinkään. Kur-

sorin kohdistaminen valmiiksi ensimmäiseen kenttään säästää myös käyttäjältä ylimääräisen hiiren liikuttamisen ja painikkeen painamisen.

Kiitos-sivulla on painike, josta on mahdollista sulkea koko esitys. Tämä on toteutettu `getURL("javascript:window.close()");`-metodin avulla. Analyste eOfficen esityksessähän esityksen sulkemiseen käytettiin FSCommand()-metodia. getURL()-metodin avulla JavaScript-koodia ei tarvitse liittää erikseen esityksen HTML-koodiin.

Flashin upottaminen WWW-sivulle

Esityksen linkitys menee W3C:n validaattorista läpi, sillä validaattori ei löydä yhtään virhettä. Mielestäni on hyvin pieni kauneusvirhe, jos WWW-sivustolle joutuu laittamaan yhden tyhjän esityksen, jossa linkitys on tehty standardien vastaisesti. Kuitenkin päätavoite esityksiä ja WWW-sivuja tehdessä on se, että mahdollisimman moni pystyy ne vielä näkemäänkin. Siksi on hyvä, että vielä jossain kohtaa tarkistetaan tarvittavat selainlaajennukset. Parempaakaan tapaa tehdä linkitys täysin standardien mukaisesti ei ole vielä kehitetty.

7 Tuote-esitysten erot

Ensimmäinen silmiin pistävä ero vanhan Analyste eOfficen esityksen ja uuden Analyste-ratkaisun esityksen välillä on ulkoasu. Molemmat esitykset aukeavat saman kokoisiin ikkunoihin. Uudemmassa esityksessä käytettiin kuitenkin enemmän väriä ja bittikarttakuvia, kun taas eOfficen esityksessä tyydyttiin piirrosgrafiikan käyttöön.

Analyste-ratkaisun esityksessä myös navigointipalkin rooli on isompi. Navigointipalkissa on selkeästi eroteltu eri osa-alueet, joita esityksessä käydään läpi. Lisäksi käyttäjä tietää koko ajan, missä osiossa ollaan menossa ja hän voi myös helposti siirtyä haluamaansa osioon navigointipalkin avulla. eOfficen esityksessä navigointipalkissa oli kelauspainikkeet, joilla esitystä pystyi kelaamaan eteenpäin ja taaksepäin. Analyste-ratkaisun esityksessä navigointipalkin vastaavista painikkeista siirrytään toiseen osioon.

Merkille pantava ero on myös esitysten esilataajissa, koska eOfficen esityksessä se ei toimi täysin oikein. Toinen merkille pantava seikka on lomakkeiden toiminta. eOfficen esityksessä lomakkeen lähetyksen jälkeen käyttäjä joutuu pois esityksestä, jolloin esityksen uudestaan katsominen vaatii alkuperäisen linkin uudelleen avaamisen. Analyste-ratkaisun esityksessä lomakkeen lähetyksen jälkeen käyttäjä on edelleen esityksessä. Kiitos-sivulla on vielä erikseen linkki esityksen uudestaan katsomiseksi, mutta myös navigointipalkin kautta käyttäjän on mahdollista katsoa haluamansa kohta uudestaan.

Pintaa syvemmältä katsottuna esitysten tekniset toteutustavat eroavat todella paljon. eOfficen esitys koostuu yhdestä tiedostosta, ja esityksen osioiden elokuvaleikkeet on sijoiteltu pääaikajanalle peräkkäin elokuvaleikkeiden omien aikajanojen pituisina. Lisäksi kukin teksti on omassa graafisessa symbolissaan, eli tekstipäivitystä tehtäessä on aina ensin etsittävä oikea symboli kirjastosta.

Analyste-ratkaisun esitys puolestaan koostuu useasta tiedostosta, sillä se on pilkottu osiin aihealueiden mukaan. Kustakin esimerkistä on tehty oma tiedosto. Lisäksi tekstien päivitys onnistuu helposti yhden XML-tiedoston kautta. Tekstien näkyminen on vielä toteutettu niin, että esityksen tekstikenttien koot muodostuvat automaattisesti juuri sen kokoisiksi, mitä tekstin määrä vaatii. Näiden lisäksi esityksen kaikki toiminnallisuudet funktioineen on sijoitettu kahteen AS-tiedostoon, jolloin esityksessä joko aikajanalta, elokuvaleikkeistä tai painikkeista kutsutaan tiedostoissa määriteltyjä funktioita.

Analyste-ratkaisun esitysrakenteen vuoksi esityksen lataamista ei tarvitse odottaa kovin kauaa. Latausaikaan vaikuttaa kuitenkin luonnollisesti käyttäjän Internet-yhteyden nopeus.

8 Yhteenveto

Flash-esitykset on mahdollista toteuttaa käytettävyyttä unohtamatta. Flashin käyttäminen ei siis automaattisesti tarkoita, että käytettävyys olisi huono. Flash-esityksen eri osa-alueet voidaan toteuttaa järkevästi käytettävyyteen vaikuttavat seikat huomioiden. Toteutustapoja on varmasti monia, ja tässä tutkintotyössä onkin esitelty yksi tapa tehdä esitys.

Flashin upottaminen WWW-sivuille standardien mukaisesti ei vielä onnistu käden käänteessä, mutta tähänkin toivon mukaan tulevaisuudessa kehitetään yksi yhtenäinen tapa, joka toimisi kaikissa selaimissa.

Flash-esitystä kuten mitä tahansa muutakin tuotetta tai esitystä tehtäessä on lähdettävä liikkeelle kohderyhmästä ja tavoitteista. Ne auttavat selvittämään projektin laajuuden ja esityksen vaatimukset. On kuitenkin pidettävä mielessä, että Flashiä ei kannata käyttää kaikkeen mahdolliseen sisältöön. Joissain tapauksissa selvitään myös ilman Flashiä.

Flash-esityksen tehosteita kannattaa käyttää maltilla. Jos esitys on täynnä mitä erilaisimpia tehosteita, menettävät tehosteet merkityksensä, ja varsinainen sisältö hukkuu niiden sekaan. Oikein käytettynä ne parantavat käyttäjäkokemusta ja tuovat sisältöä jopa paremmin esiin.

Lisäksi Flash-esityksen rakenne on syytä pitää selkeänä ja varmistaa, että mahdolliset navigointipalkit täyttävät käytettävyyden ehdot. Flash-esityksistä voi tehdä myös helposti ylläpidettäviä. Flashillä tehtäessä on muistettava myös testata tuotosta mahdollisimman monessa eri selaimessa ja vielä erilaisilla Internet-yhteyksillä. On pidettävä mielessä, että kaikki selaimet eivät näytä asioita samalla tavalla.

Flashillä on siis mahdollista tehdä hyvin selkeitä ja monipuolisia esityksiä oikeastaan tuotteesta kuin tuotteesta. Flash toimii loistavasti myös opetusvälineenä. Yleisesti Flash-esitykset toimivat hyvin mielenkiinnon herättäjinä.

Lähteet

Airgid, Kevin & Reindel, Stephanie 2002. Flash 99 % good: a guide to Macromedia Flash usability. Berkeley: McGraw-Hill.

Bhangal, Sham 2001. Inside Flash ActionScript. Helsinki: Edita, IT Press.

Capraro, Michelangelo & McAlester, Duncan 2002. Skip intro: Flash usability and interface design. Indianapolis: New Riders Publishing.

Keating, Jody 2002. Inside Flash. Indianapolis: New Riders Publishing.

Lever, Nik 2004. Flash MX 2004 Games: Art to ActionScript. Oxford: Focal Press.

Macromedia Flash Player Statistics 2005. [online] [viitattu 21.1.2006]. www.macromedia.com/software/player_census/flashplayer/

McGregor ym. = McGregor, Chris, Waters, Crystal, Doull, David, Regan, Bob, Kirkpatrick, Andrew & Pinch, Peter 2002. The Flash usability guide: interacting with Flash MX. Birmingham: Friends of ED.

McLellan, Drew 2002. Flash Satay: Embedding Flash While Supporting Standards. [online] [viitattu 4.1.2006]. www.alistapart.com/articles/flashsatay/

Nielsen, Jakob 1993. Usability Engineering. San Diego: Academic Press.

Sinkkonen ym. = Sinkkonen, Irmeli, Kuoppala, Hannu, Parkkinen, Jarmo & Vastamäki, Raino 2006. Psychology of usability. Helsinki: IT Press.

Tehokkaan sähköisen taloushallinnon opas 2005.

The Art of Cash Management 2005. Analyste Cash Manager 4, 2.

Viitanen, Jari 2005. Analyste eOffice -tuotekuvaus.

Zeldman, Jeffrey 2003. Designing with Web standards. Indianapolis: New Riders Publishing.

Haastattelut

Hatakka, Tuija. Analyste Oyj, yhteyspäällikkö. Haastattelu 8.3.2006. Tampere.

Koskinen, Erkka. Tampereen kaupungin taloushallinnon palvelukeskus, talouspalvelusihteeri. Haastattelu 2.1.2006. Tampere.

Mäki-Petäjä, Maiju. Analyste Oyj, viestintäpäällikkö. Haastattelu 17.1.2006. Tampere.

Schroderus, Heli. DokuMentori Oy, tekninen dokumentoija. Haastattelu 9.3.2006. Tampere.

Tolonen, Mika. Analyste Oyj, myyntipäällikkö. Haastattelu 22.12.2005. Tampere.

Liitteet

Liite 1: Navigointipalkin painikkeisiin vaikuttavia funktioita

```
// funktiolle valitetaan numero, joka tulee painikkeen
// nimessa olevasta numerosta
function checkStatus (nbr) {

    // kaydaan painikkeiden tilat lapi
    for (var i = 0; i < painikkeenTila.length; i++) {

           // jos painike, jota ei klikattu
           if (i !== nbr) {

               // jos painike on ollut valittuna
               if (painikkeenTila[i] == 1) {

                      // muutetaan tila
                      painikkeenTila[i] = false;
                      // kasketaan siirtya ylos alkuperaiselle paikalle
                      this["mc_painike" + i].status_over = false;
                      colorToNormal(i);
                      this["mc_painike" + i].onEnterFrame = function() {
                          _parent.moveBack(i);
                      }
                      // aktivoidaan painike
                      this["mc_painike" + i].btn.enabled = true;
               }
           }

           // jos valittu painike, poistetaan
           // painike kaytosta ja vaihdetaan vari
           else if (i == nbr) {

               colorToOrange(i);
               this["mc_painike" + i].btn.enabled = false;
               this["mc_painike" + i].my_color.setRGB(0xffffff);
               esimerkki_status = false;
           }
    }

    if (this.mc_painike7.tila) {

           this.mc_painike7.btn.enabled = true;
           if (this.mc_painike7.pohja._currentframe == 6) {

               this.mc_painike7.pohja.gotoAndPlay("toNormal");
           }
    }
}

// painikkeen tilan muuttaminen
function changeStatus(nbr) {
    _root.navigointi.painikkeenTila[nbr] = true;
    _root.navigointi.checkStatus(nbr);
}
```

jatkuu

```
// painikkeen vari normaaliksi
function colorToNormal(nbr) {
    if (2 <= this["mc_painike" + nbr].pohja._currentframe <= 6) {
           this["mc_painike" + nbr].pohja.gotoAndPlay("toNormal");
    }
}

// painikkeen vari oranssiksi
function colorToOrange(nbr) {
    this["mc_painike" + nbr].pohja.gotoAndPlay("toOrange");
}

// painikkeiden liikkeiden rajat
var targetY:Number = 10;
var targetYover:Number = 7;
var targetYnormal:Number = 0;

// painikkeen alas liikuttaminen
function moveDown(nbr) {
    currentY = this["mc_painike" + nbr]._y;
    difY = currentY - targetY;
    setProperty(this["mc_painike" + nbr], _y, currentY - (difY/3));
}

// painikkeen liikuttaminen takaisin ylos
function moveBack(nbr) {
    currentY = this["mc_painike" + nbr]._y;
    difYnormal = currentY - targetYnormal;
    setProperty(this["mc_painike" + nbr], _y, currentY - (difYnormal/3));
}

// kun hiiri painikkeen paalla
function moveWhenOver(nbr) {
    currentY = this["mc_painike" + nbr]._y;
    difY = currentY - targetYover;
    setProperty(this["mc_painike" + nbr], _y, currentY - (difY/3));
}
```

Liite 2: Esimerkki tekstit.xml-tiedoston rakenteesta

```
<?xml version="1.0"?>
<sisalto>
   <osio kohtaus="intro" otsikko="Taloushallinnon haaste">
           <rivi>"Tarkistatko tämän?"</rivi>
           <rivi>"Hyväksyn!"</rivi>
           <rivi>"Ulkomaan päiväraha?"</rivi>
           <rivi>"Mikä on kassatilanteemme huomenna?"</rivi>
           <rivi>"Ehditkö hakea minulle sen paperin?"</rivi>
   </osio>
   <osio kohtaus="tulevaisuus" otsikko="Tulevaisuus nyt!">
           <rivi>Siirry kerralla tehokkaan sähköisen taloushallinnon
todellisuuteen</rivi>
   </osio>
   <osio kohtaus="hankinta" otsikko="Automatiikkaa hankintojen
käsittelyyn">
           <rivi>Hyödynnä laskujen käsittelyssä jo sopimus- ja
tilausvaiheessa syntyvät tiedot.</rivi>
           <rivi>Ohjelmisto vertaa saapunutta laskua tilaukseen tai
sopimukseen ja käsittelee ne automaattisesti.</rivi>
           <rivi>Turhat rutiinitarkastukset jäävät pois ja vain poikkeukset
käsitellään.</rivi>
   </osio>
   <osio kohtaus="ostolaskut1" otsikko="Aika on rahaa ostolaskujen
käsittelyssä">
           <rivi>Kun käsittelet laskut sähköisesti, saat toimintoihin
tarkkuutta, nopeutta ja vaivattomuutta.</rivi>
           <rivi>Ratkaisu kestää pitkälle tulevaisuuteen.</rivi>
   </osio>
   <osio kohtaus="ostolaskut2" otsikko="Aika on rahaa ostolaskujen
käsittelyssä">
           <rivi>Ratkaisulla hoidat verkkolaskujen vastaanottamisen,
tarkastamisen, tiliöinnin ja siirron maksatukseen.</rivi>
           <rivi>Tarkastat ja hyväksyt laskut sekä sen liitteet
selaimella.</rivi>
           <rivi>Ohjelmisto seuraa eräpäiviä ja valvoo ostolaskujen
käsittelyä puolestasi.</rivi>
   </osio>
   <osio kohtaus="matkalaskut1" otsikko="Vauhtia ja vaivattomuutta
matkalaskujen käsittelyyn">
           <rivi>Tehoa palkkahallintaan ilman erillistä
matkalaskusovellusta. Käsittelet matkakulut ja maksat korvaukset -
sähköisesti ja helposti.</rivi>
           <rivi>Tiedot matkasta ja kuluista syötetään selaimella.</rivi>
           <rivi>Vaivatonta sekä matkustavalle henkilölle että hänen
esimiehelleen!</rivi>
   </osio>
</sisalto>
```

Liite 3: Flash-esitykset Analyste eOffice -ohjelmistosta ja Analyste-ratkaisusta

erillinen CD