Paavo Ertolahti

Internetpohjaisen musiikkipelin toteutus flash tekniikalla

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikan koulutusohjelma
Insinöörityö
25.3.2011

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Paavo Ertolahti Internetpohjaisten musiikkipelien toteutus flash tekniikalla 29 sivua + 1 liite 15.3.2011
Tutkinto	insinööri
Koulutusohjelma	tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	Sovelluskehitys
Ohjaaja(t)	tuottaja Susanna Ertolahti-Mertanen koulutuspäällikkö Markku Karhu

Insinöörityn tavoitteena oli toteuttaa Espoon musiikkiopiston verkkosivuille lapsille ja nuorille suunnattu teoriapeli usealle eri soittimelle ja audio- ja videotoistimet. Tämän jälkeen jatkoprojektiksi tuli tehdä lisää musiikin teoriapelejä musiikkikirjan verkkosivuille.

Teoriapelissä ruutuun tulee nuotteja ja pelaajan on tarkoitus painaa nuottia vastaavaa nappia. Pelin teemaksi valittiin kalastuspeli, koska siihen oli helppo integroida tahdotunlainen käytös. Peli tehtiin usealle eri soittimelle. Jokaisen soittimen pelissä on kolme tai neljä vaikeustasoa ja vaikeammilla vaikeustasoilla erilaisten kalojen määrä kasvaa.

Audiotoistinta tarvittiin, jotta oppilaat voivat kuunnella miltä tietyt nuottipätkät kuulostavat oikein soitettuna. Videoina haluttiin soittimien hoito- ja käsittelyohjeita.

Työ aloitettiin vähäisellä flash/actionscript-kokemuksella, mutta kohtalaisella Java/C++-kokemuksella. Actionscript 3 oli kuitenkin riittävän samankaltainen jo tunnettuihin kieliin verrattuna, että projekti sujui ilman suurempia ongelmia.

Työn kaikki suurimmat tavoitteet saavutettiin, mutta aikataulu venyi kesäisen ajankohdan ja siitä johtuvan materiaalien hitaan saatavuuden takia. Joidenkin soittimien nuotit saatiin vasta syksyllä opettajien palatessa kesälomiltaan.

Avainsanat	flash, actionscript, xml, javascript

Metropolia

Abstract

Author(s) Title Number of Pages Date	Paavo Ertolahti Internet based music bames with Flash 29 pages + 1 appendice 15 March 2011
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Susanna Ertolahti-Mertanen, Producer Markku Karhu, Lecturer

The aim of this final year project was to develop a music theory game with Adobe Flash targeted to young teens and audio- and video players for Espoo Music Academy and theory games for a music book's internet page.

In the Music Academy's theory game the players gets one note at a time to the screen and has to push a button with the corresponding note's name. The game was themed as a fishing game, because of the theme's easy integration into the game. The game was made for multiple instruments and every instrument has three or four levels of difficulty.

The music book's theory practice program is a platform for multiple games in which you earn money to buy instruments for playing songs.

Previous experience with Flash and ActionScript was relatively low, but more extensive knowledge of Java and C++ helped a great deal, since the third version of ActionScript is very similar to Java. Thus the project was manageable.

All parts of the project were completed successfully, even though the initial timescale did not hold. The acquisition for the games took longer than expected.

Keywords	Flash, Actionscript, XML, Javascript

Sisällys

1 Johdanto

Insinöörityön tavoitteena oli tehdä Espoon Musiikkiopistolle internet-pohjaisen pelin, sekä musiikki- ja videotoistimet internet-sivuille nuorien musiikinharrastajien mielenkiinnon lisäämiseksi musiikin teoriaa kohtaan. Erillisenä projektina oli vuonna 2010 julkaistun Musiikkiseikkailu 2 -nimisen musiikkikirjan internet-sivuille useita Flash-pelejä WSOY:n kirjallisuussäätiöltä saaduilla tukirahoilla.

Musiikkiopiston projekti oli aikataulutettu alkamaan kesäkuussa 2010 ja kestämään vain syksyyn asti. Julkaisutilaisuus olikin elokuussa, mutta koko projekti valmistui vasta marraskuussa 2010.

Musiikkiopiston pelin aiheeksi valittiin yksinkertainen kalastuspeli. Musiikkikirjan peliprojektiin rakennettiin monimutkaisempi, Commodore 64:n Aku Ankka peliin pohjautuva, rahanansaitsemisohjelma, joka toimii alustana muille peleille.

Molempien pelien ja mediatoistimien alustaksi valittiin Adobe Flash, koska se on hyvin laajasti levinnyt ja löytyy lähes jokaiselta tietokoneelta.

Työn tekijän flash-kokemus ennen projektia oli vain harrastuspohjaista, joten siirtyminen actionscript 3:een oli hieman sekava. Omat tiedot flashin monipuolisuudesta olivat etukäteen pääosin teoriapohjaisia, ja tekemiseen tarvittavat opetusmateriaalit etsittiin internetistä. Suurelta osin tarvittavat tiedot ja esimerkit löytyivät Adoben referenssisivustolta, mutta myös muiden sivustojen tutoriaaleja käytettiin kehityksessä ilmenneiden ongelmien ratkomiseen.

2 Flashin historia

Flash sai alkunsa vuonna 1996 kun Macromedia julkaisi Future Splash Animatorin, joka myöhemmin nimettiin Macromedia Flash 1:ksi. Nykyään Flashia kehittää muun muassa Photoshop-ohjelmasta tuttu Adobe Systems.

Flashin edeltäjä oli piirto-ohjelma Smartsketch, joka oli tarkoitettu kynällä ohjattaville kosketusnäyttökoneille, joissa oli PenPoint-käyttöjärjestelmä. SmartSketchin kehittäjä oli Jonathan Gay, joka vei idean Silicon Beach Software -yhtiölle. Penpoint-käyttöjärjestelmä ei kuitenkaan saanut tuulta alleen, joten Smartsketch käännettiin Microsoft Windowsille ja Mac OS:lle. Internetin yleistymisen myötä SmartScetch julkaistiin uudelleen nimellä FutureSplash ja se kilpaili Macromedia Shockwaven kanssa. Samalla siihen lisättiin "frame-by-frame" animointiominaisuuksia. (2, 3)

Joulukuussa 1996 Macromedia osti FutureSplashin ja leikkasi nimen keskeltä kirjaimia lyhentäen sen vain Flashiksi. Myöhemmin, vuonna 2005, Adobe Systems osti Macromedian ja julkaisi vuonna 2007 seuraavan version Flashista nimellä Adobe Flash CS3 Professional.

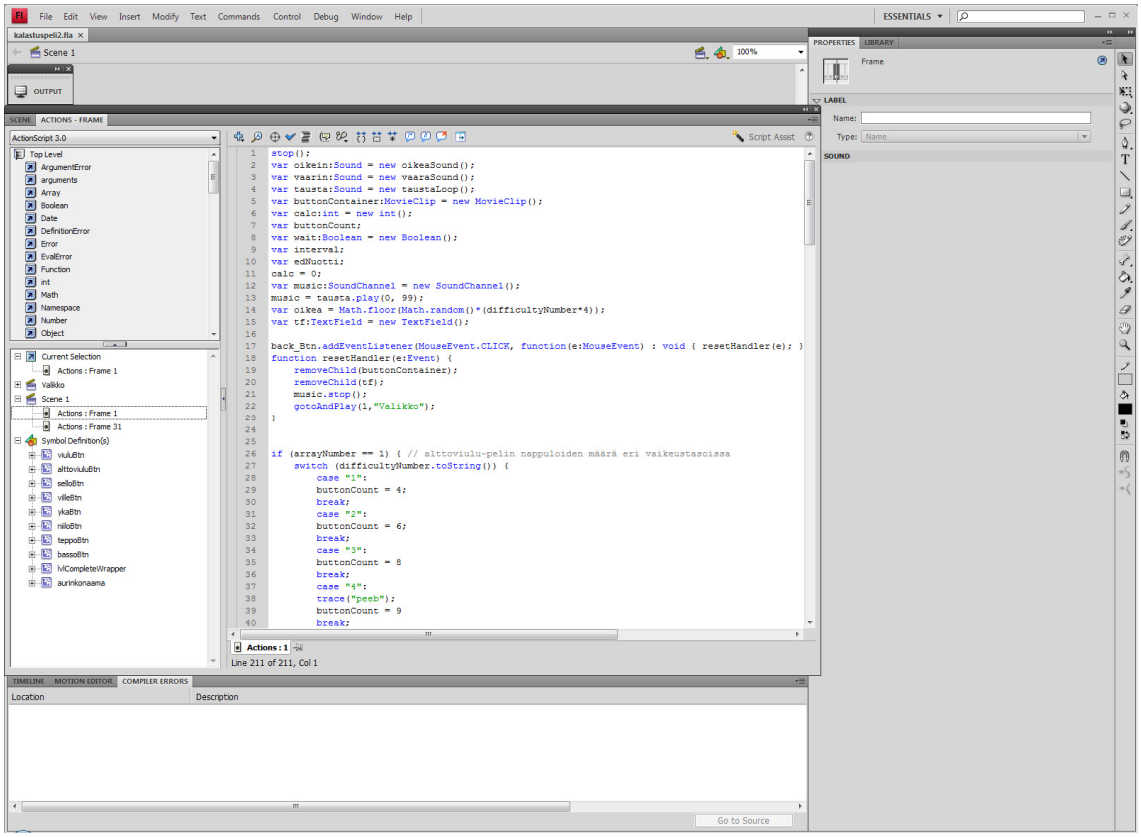
Adobe Flash CS3:n myötä flash alkoi tukea actionscript 3:a, joka oli suuri askel actionscript 2:sta eteenpäin. Kieli muuttui tehokkaammaksi ja lähemmäksi tavanomaisia korkeamman asteen koodikieliä, kuten C++:aa tai Javaa. Actionscript 3 mahdollisti kokonaisten ohjelmien kirjoittamisen actionscriptillä, eikä graafista kehitintä ollut pakko käyttää enää ollenkaan.

3 Työkalujen esittely

Adobe Flash CS4

Adobe Flash on flash-tiedostojen tekemiseen suunniteltu ohjelma. Alun perin flash oli tarkoitettu internet-videoiden ja interaktiivisuuden lisäämiseen, mutta nykyään flashilla voi tehdä myös työpöytäsovelluksia. Flashilla voi tehdä interaktiivisia ohjelmia ilman actionscript-osaamista WYSIWYG-tyylisessä kehitysympäristössä. 12. huhtikuuta 2010 Adobe julkaisi Adobe Flash CS5:n, joka on tällä hetkellä flashin uusin versio. Projekteissa käytettiin Adobe Flashin CS4 versiota, jonka actionscript kehitysikkuna näkyy kuvassa 1.

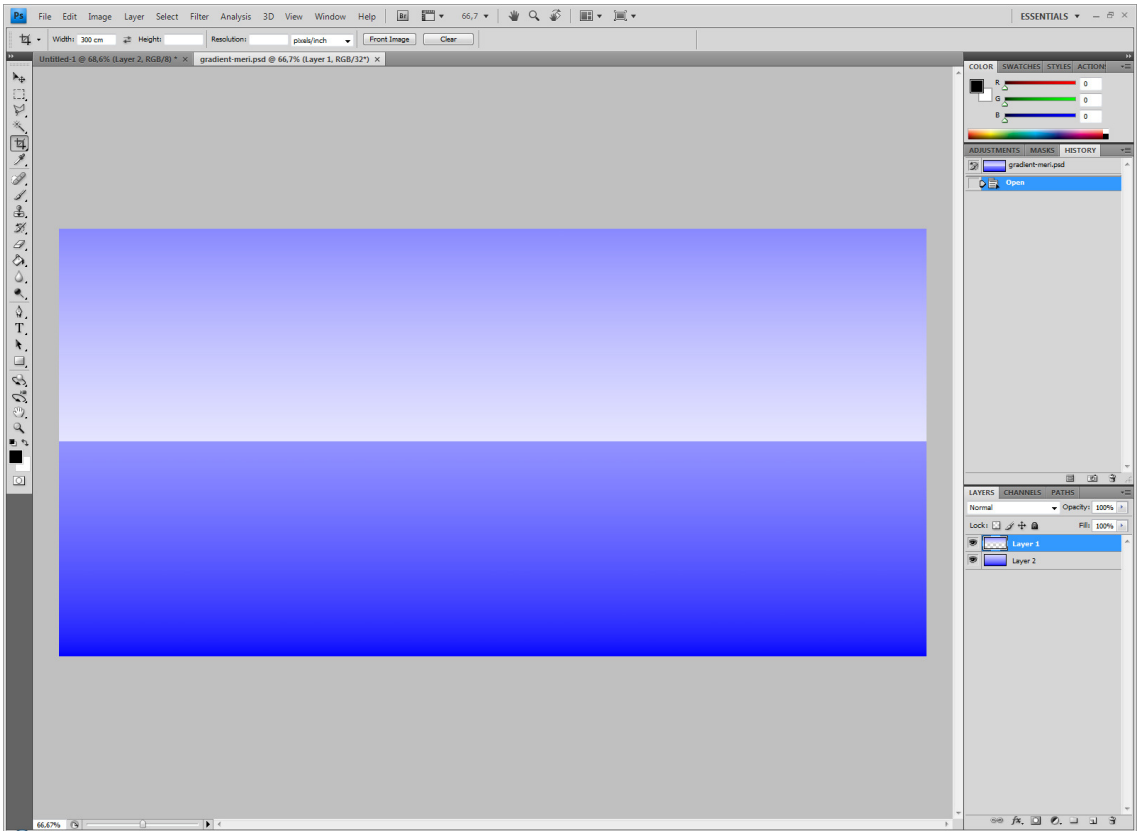
Flash-ohjelmia voi kehittää myös kolmannen osapuolen ohjelmistoilla, joista monet ovat huomattavasti halvempia kuin Adobe Flash. Näitä ohjelmistoja ei kuitenkaan projekteissa käytetty.



Kuva 1. Adobe Flash CS4

Adobe Photoshop CS4

Photoshop on Adoben hyvin suosittu kuvankäsittelyohjelma, jolla voi hyvin monipuolisesti ja nopeasti muokata kuvia ja luoda grafiikkaa. Photoshopille on tehty hyvin paljon liitännäisiä, joista monien on tarkoitus yksinkertaistaa kuvanparannuksia ja muokkauksia vielä enemmän niin, että aloittelevatkin käyttäjät pystyvät luomaan näyttäviäkin efektejä. Adobe julkaisi samanaikaisesti Adobe Flash CS5:n kanssa myös uusimman Adobe Photoshop CS5:n. Projekteissa käytettiin Photoshopin versiota CS4. Kuvassa 2 Photoshopissa on auki kalastuspelin meritausta.



Kuva 2. Adobe Photoshop CS4

4 ActionScriptin vertailu yleisesti tunnetuimpiin kieliin

Actionscriptin syntaksi on kolmannessa versiossa muuttunut huomattavasti skriptikielestä oikean koodikielen suuntaan. Aikaisemmissa versioissa actionscript oli huomattavasti rajoittuneempaa eivätkä yhtäläisyydet tunnettuihin koodikieliin olleet suuret.

4.1 C++-kieli

Suurin ero täysin koodipohjaisen kielen, kuten C++ tai java ja actionscript 3:n välillä on actionscript 3:n jaksopohjaisuus. Actionscript 3 on jaettu osiin kehyksillä (frame) ja kohtauksilla (scene). Kehykset ovat tietyllä hetkellä näkyviä ohjelman osia. Kehykseen liitetty koodi ajetaan silloin, kun kyseinen kehys on ruudulla. Ruudulla voi kuitenkin liikkua animaatiota, ilman että ohjelman sisäiset kehykset vaihtuvat.

Ohjelma tulkitsee kohtauksen mielestä täysin samanlaiseksi kuin kehyksen. Se on tehty vain helpottamaan kehittäjän työtä. Niiden avulla on helppo jaksottaa koodia ja animaatioita.

Flashissa ohjelma pyörii täysin yhdellä säikeellä. Tätä on osittain korvattu monipuolisilla asynkronisilla funktiokutsuilla ja käyttäjän näkökulmasta ohjelma voi tuntua yhtä sulavalta kuin säikeistetty ohjelma. Esimerkiksi timer-luokka auttaa tähän ongelmaan hyvin paljon. Sillä voi ajastaa funktiokutsuja, joiden ei kuitenkaan tulisi viedä kovin paljoa aikaa, koska ohjelma voi vaikuttaa jäätyneen niiden suorituksen aikana. Säikeistetyssä ohjelmassa useampia koodipätkiä voisi ajaa yhtä aikaa, jolloin taustalla ajettavan koodin suorittamisen nopeus ei ole yhtä aikakriittistä.

Tavallisessa C- tai C++-ohjelmassa ohjelmalla on täysi kontrolli viestisilmukkaan (message loop). Flash/actionscript-ohjelmassa taas ei. Viestisilmukkaa hallitsee flash-ohjelman ladannut ohjelma kuten selain, itsenäinen flash-toistin tai jokin muu ohjelma. Flash toimii täysin tapahtumien (event) kautta. Jos flash-ohjelmalla kestää jonkin asian laskennassa hyvin kauan, ohjelma ja sitä pyörittävä toistin eivät voi suorittaa muita

toimintoja laskennan aikana. Flashin täytyy siis tehdä kaikki asynkronisten palautekutsujen (callback) kautta. (4.)

Tyypillisessä C- tai C++ -ohjelmassa suorittamisen voi keskeyttää dialogin ajaksi ja jatkaa kun käyttäjä on valmis ilman, että koko ohjelma jäätyy. Näin ei voi tehdä actionscriptillä. Actionscriptissä dialogiin syötetyt tiedot luetaan vasta kun palautekutsu niin tekee. Tästä syystä C tai C++:ssa usein käytetyt hälytysikkunat (alert) eivät ole yhtä hyviä virheiden etsimiseen actionscriptissä, koska ohjelma ei pysähdy hälytysikkunan kohdalle, vaan jatkaa suorittamista.

Actionscript 3:ssa ei myöskään ole enumeraattoria. Vastaavanlaisen toteutuksen joutuu tekemään itse, tai käyttämään jonkun muun jo valmiiksi luomaa luokkaa. Kääntäjä ei kuitenkaan näe näiden käytössä olevia virheitä, koska actionscript 3:ssa on vain primitiivimuuttujia, mikä voi haitata virheiden paikallistamista.

Actionscript 3:ssa kaikki luokkamuodostimet (constructor) ovat julkisia. Pääosin siksi, että ECMAScript 4 ei ole täysin valmis ja actionscript 3:ssa päätettiin olla käyttämättä muita muodostimia (protected ja private), kunnes ECMAScript 4 saataisiin valmiiksi. ECMAScript 4 kuitenkin hylättiin kehittäjien toisistaan eriävien mielipiteiden vuoksi ja actionscript 3 jäi ainoaksi ECMAScript 4:ää käyttäväksi kieleksi, joskin sekin vain osittain. (6.)

Actionscript 3:ssa tarpeettomien tietojen posto muistista toimii automaattisesti, eikä hajoittimista (destructor) tarvitse huolehtia. Vain dynaamisien luokkien objekteja ja niiden arvoja voi poistaa delete-funktiolla. Dynaamiset luokat ovat luokkia, joihin voi lisätä funktioita ja muuttujia ajon aikana. Funktiot ja muuttujat voivat kuitenkin olla vain julkisia ja dynaamiset luokat vievät niiden takia enemmän resursseja, koska tiedoston kääntämisen aikana tuntemattomille funktioille ja muuttujille täytyy luoda hash-taulu. Myös kaikki primitiivimuuttujat ovat olioita actionscriptissä. Kaikki oliot toimivat referensseillä (osoitin) ja delete poistaa vain referenssin. Jos kuitenkaan objektilla ei ole enää yhtään referenssiä, sekin poistetaan. C/C++:ssa ei käytetä globaaleja muuttujia, actionscript 3:ssa taas hyvin monet muuttujat ovat globaaleja. (5.)

4.2 Java

Actionscript 2 oli lähempänä javascriptiä kuin javaa ulkonäöltään, mutta tämä muuttui actionscript 3:n myötä. Nykyään koodi on hyvin samanlaista javan kanssa koska ECMAScript, johon actionscript perustuu, on muuttunut javankin kaltaisemmaksi. Myös tarpeettomien tietojen poisto muistista on javassa lähempänä flashia kuin C++:ssa. (7.) Kuvassa 3 näkyy javan ja actionscript 3:n vertailua.

Java

```
package foo;

import somepackage.*;

public class Foo
        extends FooParent
        implements IFoo
{
  private String blah = "blah";Â
  /**
   * bar performs important string-
   * setting functionality
   *
   * @param b the String to be set
   */
  public boolean bar(String b) {
    blah = b;
    return true;
  }
}
```

ActionScript 3

```
package foo {

import somepackage.*;

public class Foo
        extends FooParent
        implements IFoo
{
  private var blah:String = "blah";
Â  /**
   * bar performs important string-
   * setting functionality
   *
   * @param b the String to be set
   */
  public function bar(b:String):Boolean {
    blah = b;
    return true;
  }
}
}
```

Kuva 3. Javan ja actionscript 3:n syntaksin vertailua

5 Pedagogia

Hyvin montaa peliä on kutsuttu opettavaksi, ns. edutainmentiksi, sillä useimmissa peleissä käsitellään yhteiskuntaa, tiedettä tai muita elämän osa-alueita faktapohjaisesti. Useimmissa peleissä oppiminen on kuitenkin täysin toissijaista viihteen rinnalla.

Pelejä, joissa opettaminen on pääasia, kutsutaan pelipohjaiseksi opettamiseksi (game based learning). Näissä peleissä opettaminen liittyy usein asioitten kontekstualisoimiseen. Pelit voivat vaikuttaa oppimismotivaatioon itse pelin hauskuuden lisäksi näyttämällä käytännön sovellutuksia opetettaville taidoille ja tiedoille. Musiikkipelissä esimerkiksi soittamalla musiikkia eri soittimilla. (15.)

Pelien lähestymistapa opetukseen on hyvin erilainen tavanomaiseen opetukseen verrattuna. Pelit voivat auttaa alhaiseen motivaatioon lisäämällä oppimiseen oppilasta kiinnostavia asioita ja interaktiivisuutta. Oppimisalueet ovat kuitenkin useimmiten varsin rajattuja ja pelit monesti vaativat vähintään alkeelliset tiedot käsiteltävästä asiasta.

Jos opiskelu tai harjoittelu on tylsää, ei opiskeltavaan materiaaliin syvenny eikä opiskelu ole motivoivaa. Materiaalin oppii muistamaan ulkoa, mutta sitä ei sisäistä. Opittujen asioiden soveltamistaidot jäävät vajavaisiksi, sekä luetun tarpeellisuuden vähentyessä se voi helposti täysin unohtua.

Hyvä opetuspeli pystyy vetämään opiskelijan virtuaalisiin ympäristöihin jotka näyttävät ja tuntuvat tutuilta ja olennaisilta. Tämä motivoi hyvin, koska oppija huomaa helposti opittavan asian ja käytännön sovellutuksien yhteydet. Opittava asia ei tunnu turhalta, kun oikean elämän hyödyt tulevat heti esille. (16.)

Pelipohjaista oppimista puolustavat tahot sanovat, että se yhdistää perinteisen opiskelun ja työharjoittelun parhaita puolia. Se on halpaa eikä siihen sisälly työtapaturmariskejä, niin kuin työharjottelussa. Saman tilanteen voi käydä monta kertaa läpi ja palaute on välitöntä. Esimerkiksi simulaattoreita on käytetty lentämisen harjoitteluun jo monta kymmentä vuotta. Taulukossa 1 näkyy New Media Instituten

vertailua eri opetustavoista, jossa tuodaan esille kuinka pelipohjainen opetus yhdistää tavanomaisen opetuksen ja työharjottelun hyviä puolia.(13.)

Taulukko 1: New Media Instituten näkemys opetustapojen vertailusta. (13.)

	Tavanomainen opetus	Työharjoittelu	Pelipohjainen opetus
Tuottava hintaan nähden	x		x
Alhainen tapaturmariski	x		x
Standardisoitu arvostelu	x		x
Mukaansatempaavaa		x	x
Oppilaisiin suhteutettu opettamisvauhti		x	x
Välitön palaute virheistä		x	x
Oppilas voi helposti siirtää opitun tosielämään		x	x
Oppija on aktiivisesti mukana		x	x

Kalastuspelissä ajatuksena oli lähinnä paikata puuttuvaa motivaatiota teorian opiskelun suhteen. Peli on suunnattu varsin nuorille lapsille, joilla tosielämän sovellutukset eivät vaikuttaisi motivaatioon yhtä paljon kuin lapsien vanhemmilla. Taitojen hyödyt tulevat kuitenkin ilmi soittoharjoituksissa. Korttelipeliin lisättiin myös ajatus oman yhtyeen perustamisesta ja soittamisesta kavereiden kanssa, sekä suurempi saavutuksen tunne soittimien ja kappaleiden ostamisen kautta.

6 Musiikkiopiston ohjelmat

Ennen sopimuksen tekemistä tehtiin tekniikkademoksi peli, jossa oli neljä nappia ja neljä eri nuottikuvaa. Oikeaa nappia painamalla kuului ääni ja tikkumainen kalastaja nosti onkeaan. Demon tarkoitus oli todistaa pelien kehityksen osaaminen ennen sopimuksen tekemistä, koska taidoista ei muuten ollut juurikaan näyttöä.

6.1 Vaatimukset

Musiikkiopisto pyysi valmistettavaksi yksinkertaisen internet-pohjaisen musiikkipelin, jossa opetellaan erilaisten puhallin- ja jousisoittimien teoriaa, sekä ohjelmat ääni- ja videotiedostojen liittämiseen internet-sivuille.

Ohjelmien tuli täyttää seuraavat ehdot:

- Äänientoistin joka toimii sekä mäkissä, että PC:ssä. Asiakkaan tulee pystyä helposti muokkaamaan toistettavia musiikkilistoja.
- Videotoistin, joka toimii sekä Macissa, että PC:ssä. Asiakkaan tulee pystyä helposti muokkaamaan toistettavia videolistoja.
- Peli nuottien kyselemistä varten. Pelin pohja sama, mutta jokaiselle musiikki-instrumentille tulee olla omat nuotit ja vastausnapit.

Alustaksi valittiin peliin flash, koska se on asennettu erittäin kattavasti internetiin liitettyihin tietokoneisiin, niin OSX- kuin Windows-koneisiinkin. Myös soittimiin flash sopi paremmin kuin HTML5, sillä HTML5:n video- ja äänituki on vielä melko hajanaista.

6.2 Kalastuspeli

6.2.1 Pelin ulkonäkö ja toiminta

Pelin aiheeksi valittiin kalastus, koska sen yhdistäminen nuottikysymyksiin oli yksinkertaista ja graafinen toteutus nopeaa.

Kalastaja istuu laiturilla ja pilvessä näkyy nuotteja yksi kerrallaan. Pelaaja painaa nuottia vastaavaa nappia ja oikein painettaessa kalastaja nostaa merestä kalan. Peli käännettiin yhdeksälle soittimelle, joilla on jokaisella noin 30 eri nuottia. Peli jaettiin kahteen eri ohjelmaan, joista toisessa on jousisoittimet ja toisessa puhaltimet.

Käynnistettäessä pelissä tulee näkyviin valikko, missä on neljä soitinta jousisoitinversiossa ja viisi soitinta puhaltimien puolella. Kuvassa 4 näkyy jousisoitinpelin valikko. Valikosta valitaan soitin ja vaikeustaso jonka jälkeen painetaan "Aloita peli"-nappia. Tällöin peli siirtyy kalastusnäkymään, jossa pilviin tulee näkyviin kysytyn nuotin kuva ja vedessä näkyy vastausnapit. Oikein vastattaessa kalastaja nostaa vedestä kalan, joka arvotaan valitun vaikeustason mukaan. Ensimmäisellä vaikeustasolla voi saada vain yhtä kalalajia. Lajien määrä kasvaa vaikeustason kasvaessa. Neljännellä tasolla erilaisia kaloja on neljä.

Kuva 4. Jousisoittimien kalastuspelin valikko.

6.2.2 Pelin toteutus

Kalastuspelissä jokaisen soittimen nuottikuvat sijoitettiin omaan movieclip-objektiin (movieclip). Jokaisella nuottikuvalla on oma kehyksensä objektin sisällä ja uutta nuottia arvottaessa arvotaan vain, mikä kehys movieclip-objektista näytetään. Nuottien nimet sijoitettiin kehysten leima-kenttään (label).

Peliä kokeiltuaan osa opettajista oli sitä mieltä, että peli on liian vaikea heidän nuorille oppilailleen, joten korotukset ja alennukset otettiin nuottien nimistä pois, mutta nuottikuvat jätettiin peliin. Tästä seurasi, että usealla kehyksellä oli sama leima-kenttä. Kehyksiä voi kutsua myös käyttämällä leima-kenttää, joten tämä ei ollut enää hyvä ratkaisu. leima-kenttiä ei kuitenkaan käytetty kehysten kutsumiseen ollenkaan, joten ongelma ei estänyt peliä toimimasta normaaliin tapaan, mutta tällainen toteutus ei ole kovinkaan kaunis.

Nuottien nimeäminen olisi alun perinkin kannattanut toteuttaa lisäämällä movieclip-luokkaan muuttuja, joka jokaisen nuottia esittävän kehyksen kohdalla saa arvokseen nuotin nimen.

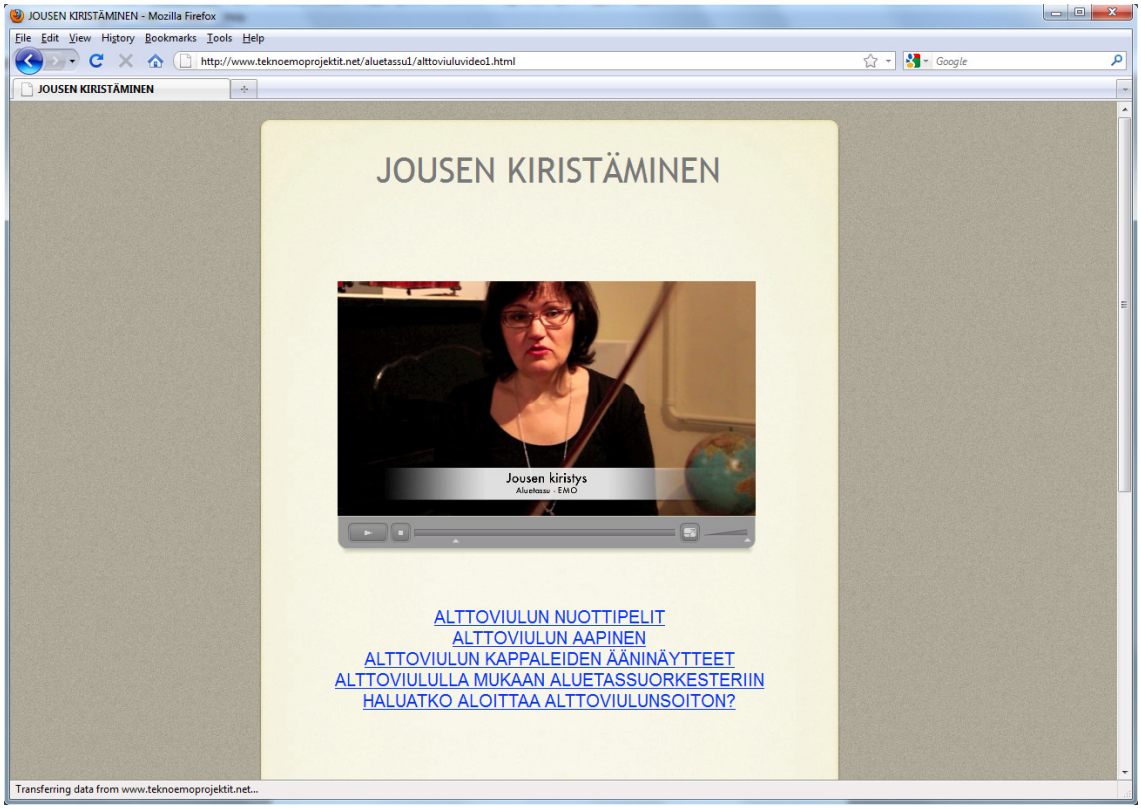
Peli tehtiin ensin täysin valmiiksi jousisoittimille, jonka jälkeen se käännettiin puhaltimille. Näin ei tarvinnut kehittää kahta eri versiota samasta pelistä.

Toteutus kuitenkin venyi kesäisen aikataulun takia. Jokaisella eri soittimella oli eri ihminen opettajana ja heidän tuli toimittaa omaan soitinpeliin haluamansa nuotit ja monet toimittivatkin ne vasta syksyllä.

6.3 Videotoistin

Musiikkiopiston opetussivuille haluttiin videoita muun muassa erilaisista soittimien huoltoon ja hoitoon liittyvistä toimenpiteistä. Videot olivat jo valmiiksi olemassa AVI-enkapsuloituina ja flashin ymmärtämällä kodekilla enkoodattuina. Videosoittimena käytettiin muokattua Flash CS4:n valmista videotoistinta.

Videotiedostojen valinta tehdään flash-tiedoston sisältävään html-tiedostoon kirjoitetulla javascript-funktiolla, jota videotoistin kutsuu. Javascript-funktio palauttaa flash-tiedostolle toistettavan videotiedston nimen. Ratkaisussa yhdellä sivulla voi olla vain yksi video. Määrä kuitenkin oli riittävä, sillä videot ovat koulutusmateriaalia ja jokaisella koulutusteemalla on oma sivunsa. Tähän hieman kömpelöön ratkaisuun päädyttiin, koska haluttiin saada videotoistimen liittäminen internet-sivuun mahdollisimman yksinkertaiseksi. Internet-sivustojen tekijät käyttivät pääasiassa WYSIWYG-tyyppisiä kehittimiä, eivätkä osaa hyvin lukea HTML-koodia. Kuvassa 5 näkyy videosoitin internetsivuun upotettuna.



Kuva 5. Videosoitin internet-sivustolla.

Myöhemmin lisättiin tuki myös videotiedoston nimen välittämiseen flashvarsia käyttämällä, sillä uusien sivusuunnitelmien myötä tuli tarve myös useammalle videolle sivua kohden.

Videoiden ei tarvitse olla flv- tai f4v-enkapsuloituja, mutta niiden täytyy sijaita tietyssä hakemistossa. Toistinta rajattiin näin, ettei toistimen kautta voisi katsoa videoita, joita sillä ei ole tarkoitus katsoa.

6.4 Audiotoistin

Audiotoistinta tarvittiin musiikkiopiston opetussivuille tulevien nuottien soitattamiseen. Sivulla näkyy sarja nuotteja ja soittimella voi kuunnella, miltä niiden soittamisen tulisi kuulostaa. Toteutus haluttiin flashilla sivustolla hieman käytetyn Quicktimen sijaan, koska Quicktimeä ei ole kovinkaan monella Windows-koneella asennettuna.

Audiosoittimessa käytettiin melko samaa lähestymistapaa kuin videosoittimessa. Musiikkia tarvitsi kuitenkin useamman soittimen ja äänitiedoston samalle sivulle, joten käyttöön otettiin xml-formaatissa oleva soittolista. Audiotoistin saa soittolistan nimen query stringinä ja lataa soittolistassa olevat äänitiedostot soittimeen.

Query string tai flashvars ovat huomattavasti parempia vaihtoehtoja dynaamisen datan välittämisestä flashille kuin javascript-funktio, koska sama Flash-tiedosto voi esiintyä samalla HTML-sivulla useampaan kertaan ilman, että jokainen flash-tiedosto saisi väkisin saman HTML-tiedostosta välitetyn tiedon, kuten XML-tiedoston nimen. Toistin itse toteutettiin muutamalla käsinluodulla napilla ja list-elementillä. Listaan ladataan kappaleiden nimet ja nimiä painamalla voi valita soitettavan kappaleen. Lista venyy sen mukaan, kuinka pitkä soittolista ladataan. Kuvassa 6 näkyy audiosoitin kahden kappaleen soittolistalla ladattuna.

Kuva 6. Audiotoistin

Soittimesta haluttiin mahdollisimman yksinkertainen, joten siinä on vain soittolista ja play- ja pause-napit. Väriteema tehtiin vastaamaan sivuston teemaa. Musiikin täytyy olla mp3-formaatissa.

6.5 Vaihtoehtoisia toteutustapoja musiikkiopiston ohjelmiin

Videotoistin

Videoita voi toistaa hyvin monella eri selainliitännäisellä. Muun muassa Quicktimella, Windows Mediaplayerilla ja Silverlightilla. Edellä mainituissa on kuitenkin ongelmana, ettei jokaisessa tietokoneessa voi olettaa näitä olevan asennettuna. Flash on kuitenkin niin laajasti levinnyt, että lähes jokaisessa internetiin kytkettyyn pöytäkoneeseen se on asennettuna.

Nykyään myös HTML5 tukee natiivisti videotoistoa. HTML5:n ongelmana on kuitenki vielä toistaiseksi hajanainen videokoodekkituki. Eri selaimet tukevat eri koodekeilla tehtyjä videoita, eikä ole yhtä, jota kaikki suosituimmat selaimet tukisivat. Applen kämmentietokoneet kuitenkin tukevat HTML5:ttä, joten tulevaisuudessa internetvideot olisi parempi lisätä sivuihin käyttämällä sitä.(9.)

Audiotoistin

Äänitiedostoja voi toistaa selaimessa kaikilla samoilla liitännäisillä kuin videotakin ja samasta syystä Flash on parempi vaihtoehto.

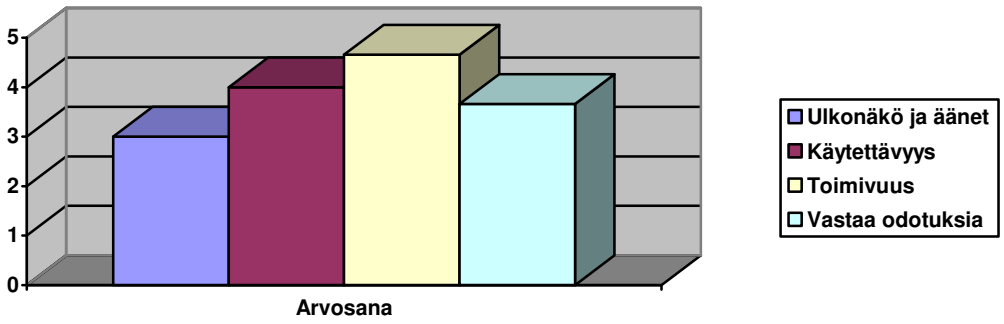
HTML5 ontuu äänientoistossa samalla tavalla kuin videotoistossa. Äänitiedostojen tallentaminen monella eri formaatilla eri selaimien tukemiseksi ei kuitenkaan olisi yhtä suuri ongelma, sillä äänitiedostot ovat huomattavasti pienempiä kuin videot. Kuvassa HTML5:llä toteutettu äänitoistin.(10.)

Kuva 7. Audiotoistin HTML5:llä toteutettuna.

6.6 Musiikkiopiston palaute

Palautteen antamista varten tehtiin lomake, jossa oli neljä kysymystä joilla jokaisella oli vastausvaihtoehdot 1-5. Vaihtoehto 5 tarkoittaa, että pelin kyseinen ominaisuus vastaa odotuksia ja vaihtoehto 1, että parantamisen varaa olisi paljonkin. Tämän lisäksi annettiin rajoittamaton kenttä vapaamuotoiselle palautteelle.

Opettajista vain kolme yhdeksästä vastasi palautekyselyyn. He olivat pääsääntöisesti tyytyväisiä ja sitä mieltä, että tuote vastaa odotuksia. Palautteiden keskiarvot näkyvät kuvassa 8.



Kuva 8: Palautteiden keskiarvot

Vapaasta palautteesta ilmenee, että osa opettajista haluaisi peruuttaa aikaisemmin sovittuja yksinkertaistuksia. Trumpettiin haluttaisiin oktaavimerkinnät takaisin. Ohjelma toimii kuitenkin palautteen mukaan luotettavasti ja jotkut oppilaat ovat selkeästi kehittyneet sen avulla. Pelin vaatimatonta ulkonäköä arveltiin vähäisen budjetin syyksi. Palautteessa tulleet muutosehdotukset viittaisivat siihen, ettei informaatio kulkenut projektin edetessä täydellisesti opettajilta kehittäjille asti. Syynä lienee ollut epäselvä informaation reitti. Opettajat valitsivat yhden opettajan hoitamaan projektia, jonka seurauksena vähäinen kokemus projektityöskentelystä hieman kostautui. Palautteessa mainitaan myös erään oppilaan kehittyneen jo "aika mestariksi" pelin avulla.

Palautteesta ilmenevät myös eri opettajien hieman erilaiset näkemykset siitä millainen pelin tulisi olla, sekä että osa opettajista haluaisi kehittää projektia vielä eteenpäin. Määrärahojen puute kuitenkin käytännössä estää tämän.

7 WSOY:n korttelipeli

Pelin teemaksi valittiin 80-luvulta peräisin olevasta Commodore 64:llä olleesta Aku Ankka -pelistä, nimeltä Donald Duck's Playground, musiikkiin sopivaksi muokattu versio. Pelissä Aku Ankka tekee erilaisia töitä ansaitakseen rahaa leikkikentän laitteisiin, joissa veljenpojat voivat leikkiä. Kuvassa 9 Aku on töissä hedelmien lajittelijana.

Kuva 9. Donald Duck's Playground.

Korttelipelissä pelaaja tienaa rahaa, jolla voi ostaa soittimia pelissä olevaan musiikkihuoneeseen. Mitä useampi soitin pelaajalla on, sitä useamman soittimen äänet kuuluvat musiikkia kuunnellessa. Myös eri kappaleita voi ostaa. Tarkoitus olisi, ettei yhdellä pelikerralla saa kaikkea ostettua, ettei pelaajalle tulisi läpipääsyn tunnetta ja on mielekästä palata pelin ääreen vielä myöhemminkin.

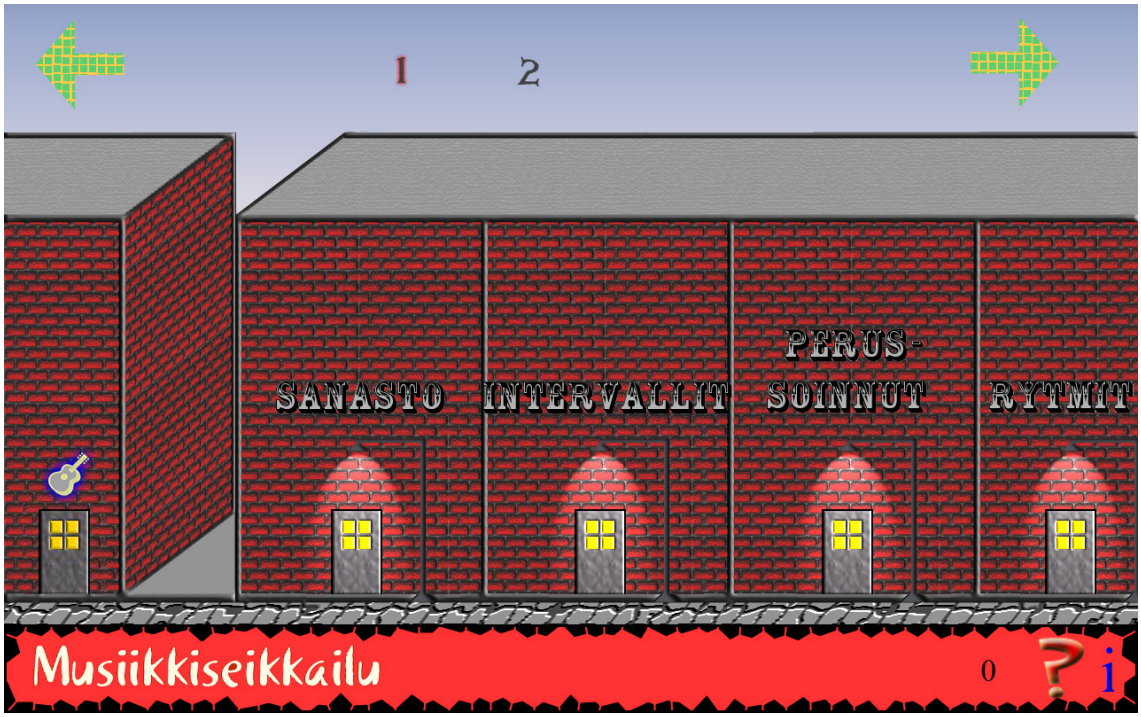
Pelin suunnittelu aloitettiin piirtämällä kortteli, jossa on useita ovia. Ovista yhdestä pääsee musiikkihuoneeseen ja muissa ovissa on erilaisia työpaikkoja esittäviä pelejä. Pelejä on helppo lisätä, kun ei tarvitse kuin lisätä uusi ovi kortteliin josta pääsee uuteen peliin.

Kaikki ansaittu raha merkitään erilliseen laskuriin, joka piirretään joka näkymän päälle. Peleissä ansaitessa pelit lisäävät funktiolla ansaitut rahat laskuriin ja soittimia sekä kappaleita ostettaessa laskuri tarkistaa riittävätkö rahat.

7.1 Pelin toteutus ja ilmenneet ongelmat

Kortteli

Aluksi peliin luotiin kuva talon seinästä, jossa oli useita ovia. Yksi ovi on musiikkihuonetta varten ja muut peleille. Alun perin suunnitelma oli tehdä joka ovesta linkki uuteen kohtaukseen ja tehdä pelit suoraan niihin. Tämä olisi kuitenkin johtanut siihen, että kaikki pelit ladattaisiin heti ohjelman käynnistyttyä. Siitä olisi seurannut pitkä latausaika erityisesti hitaammilla internet-yhteyksillä ja suurempi muistin kulutus. Kuvassa 10 näkyy pelin korttelinäkymä. Vasemmaisimmasta ovesta pääsee musiikkihuoneeseen. Neljä muuta ovea ovat ansaitsemispelejä varten.



Kuva 10. Korttelinäkymä.

Lopulliseksi ratkaisuksi muodostui tehdä pelit erillisiin flash-tiedostoihin, jotka ladataan korttelin ovien kautta. Kun kyseisen pelin ovesta mennään sisään, peli hakee

palvelimelta xml-tiedoston määräämän flash-tiedoston. Latausajat jaetaan näin osiin, ettei mikään yksi lataus veny haitallisen pitkäksi. Myös pelien korjaaminen ja muuntaminen helpottuu modulaarisuuden vuoksi. Ei tarvitse kääntää kaikkia osia uudelleen ja lähettää niitä taas palvelimelle.

Myös uusien pelien lisääminen on modulaarisuuden myötä helpompaa. Pelien tiedot lisätään xml-tiedostoon, josta kortteli-ohjelma lukee pelitiedostojen nimet ja lataa pelit. Uusia pelejä voi lisätä täysin ilman korttelin koodin muuttamista.

Korttelin alareunassa oleva moneybanner-objekti, joka pitää laskua ansaitusta rahasta, ja pelit kuitenkin keskustelevat keskenään, joten peleissä pitää olla standardisoidut funktiot rahan ansaitsemista varten. Tähän olisi tietysti paras käyttää käyttöliittymää (interface), mutta koska swf-tiedostojen actionscript-koodi muodostuu luokan sisään ilman, että luokan periytymisiä tai implementointeja voi muuttaa, ei swf-tiedostoon voi lisätä käyttöliittymää. Pelkästään erillisiin actionscript-tiedostoihin voi määrätä periytymisiä ja käyttöliittymiä, mutta actionscript-tiedostoja ei voi käyttää itsenäisinä flash-tiedostoina. Actionscript-tiedostossa sijaitsevasta luokasta pitäisi luoda flash-tiedostoon olio, jonka voisi kääntää ja tätä sitten kutsua pääohjelmalla peliä ladattaessa. Tämäkään ei siis olisi kovinkaan elegantti ratkaisu.

Tämän takia päädyttiin luomaan peleille kehys (skeleton), jonka päälle pelit tulisi luoda. Kehyksessä on valmiiksi funktiot, joita pelin tulisi käyttää kommunikoidessaan korttelin kanssa. Tämä myös helpottaa useamman pelin samanaikaista tekemistä useampien ihmisten kanssa, sekä projektin siirtämistä uudelle henkilölle, kun hänen ei välttämättä tarvitse tuntea pääohjelman koodia lisätäkseen uusia pelejä. Kehyksen koodi on nähtävillä liitteessä 1.

Pelin ovea painettaessa kortteliohjelma lataa pelin korttelin päälle. Moneybanner-objekti jää ikkunan alareunaan näkymään pelin päälle. Pelissä tienatut rahat näkyvät moneybannerissa ja moneybannerin kysymysmerkkikuvan takaa löytyvät pelin ohjeet. Pelin sijainnin palvelimella ja pelin ohjeet kortteli lataa xml-tiedostosta. Näin uusien pelien lisäämiseen tarvitsee itse pelin lisäksi kirjoittaa vain muutama rivi xml-tiedostoon. Tiedostosta voi muuttaa myös pelin tienaamisen kerrointa, mikäli pelin rahankeruu tuntuu liian helpolta tai vaikealta. Näin pelejä pystyy hieman hallitsemaan

ilman, että pelien koodeihin tarvitsee edes koskea. Xml-tiedoston syntaksi on nähtävillä liitteessä 1.

Kolmen pelin lisäämisen jälkeen ohjelma alkoi käyttäytyä oudosti ilman mitään näkyvää syytä, tai virheilmoituksia. Kun kortteliin mentiin takaisin musiikkihuoneesta, jäi koodia ja objekteja lataamatta. Jos tämän jälkeen kävi jossain pelissä ja tuli takaisin kortteliin, kaikki toimi täysin normaalisti, vaikka kohtausten vaihtamiset oli toteutettu täysin samalla tavalla. Internetin tutkimisen jälkeen kävi ilmi, että kohtauksiin liittyy hieman omituisia ongelmia ainakin Flash CS4:ssä olevalla kääntäjällä. Tämän seurauksena yleisesti ollaan sitä mieltä ettei kohtauksia kannata käyttää. Kohtausten sisäistä järjestystä muuttamalla kortteli alkoi toimia normaalisti myös musiikkihuoneessa käynnin jälkeen. Tämä lisättynä siihen, ettei kohtauksia voi lisätä dynaamisesti actionscriptillä, vaan ne on lisättävä graafisen käyttöliittymän kautta, tekee kohtauksia käyttävän toteutuksen varsin huonoksi ja kömpelöksi ja siitä tulisi siirtyä pois.

Musiikkihuone

Musiikkihuone toteutettiin tekemällä jokaiselle soittimelle oma äänikanava ja lataamalla siihen soittimelle tehty mp3-tiedosto. Kaikki kanavat laitettiin toistamaan musiikkia samanaikaisesti play-nappia painettaessa, jolloin soittimien äänet kuuluvat päällekäin ja kappale kuulostaa kokonaiselta.

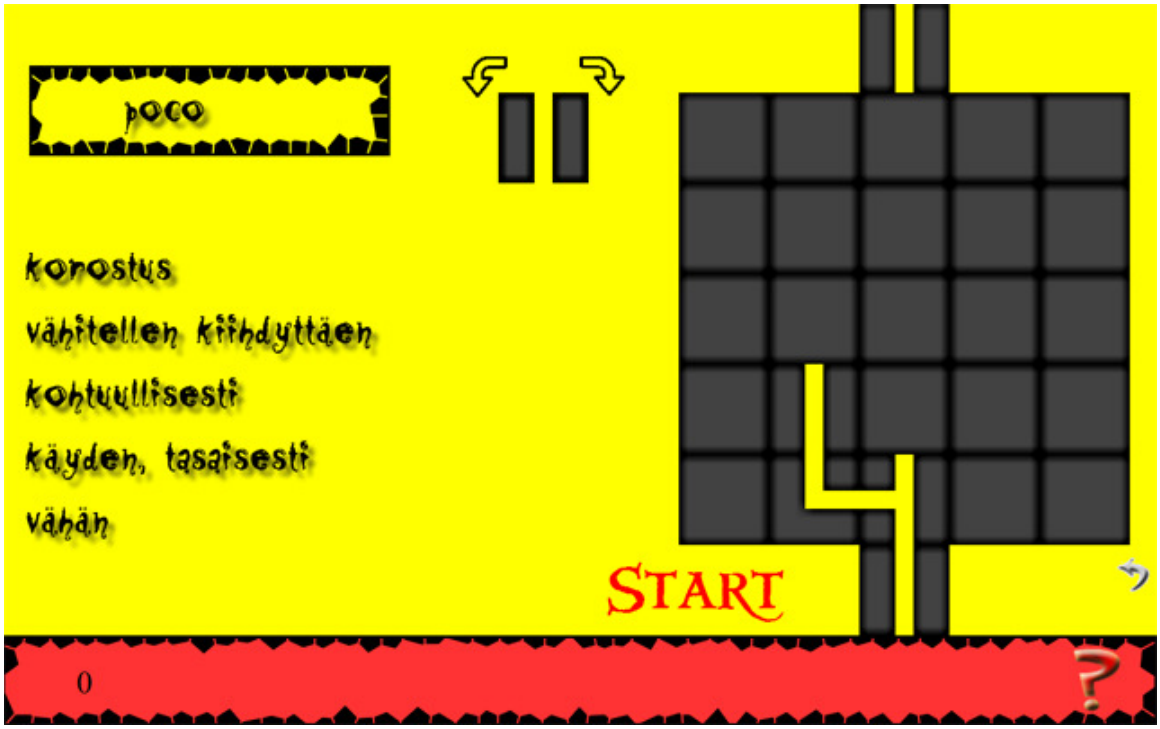
Musiikkihuoneeseen lisättiin napit, joilla yksittäiset soittimet saa pois päältä. Tämä tehtiin vähentämällä kyseisen äänikanavan äänenvoimakkuus nollaan nappia painettaessa. Tällöin soitin pysyy rytmissä muiden kanssa, vaikkei ääni kuulu. Jokainen soitin täytyy ensin ostaa, jotta äänikanavan voi kytkeä päälle. Myös eri kappaleet täytyy ostaa.

Putkipeli

Jostain syystä ensimmäisenä pelinä lisätyssä putkipelissä MovieClip- ja Sprite-objektien painiketila-ominaisuus (buttonmode) lakkasi toimimasta, kun peli upotettiin korttelin päälle. Pienen tutkimisen jälkeen vaikutti siltä, että hiirikursorin muuttamiseen käytettävät funktiot eivät toimi, koska pelin tiedoston yläpuolella hierarkiassa ei olekaan flash-virtuaalimoottori, vaan toinen flash-tiedosto ja sen seurauksena

funktiokutsut menevät väärään paikkaan. Hiirikursorin muuttaminen kädeksi nappien päällä toteutettiin siis käsin tapahtumakäsittelijän (actionlistener) kautta.

Putkistossa olevat palikat reagoivat hiiripainalluksiin ja painettaessa tiettyä palikkaa, siirtyy kysymyksen vieressä sijaitseva putkenpätkä putkistoon siinä asennossa mihin se on käännetty. Jottei putkistoon voi listätä kuin yhden pätkän jokaista oikeaa vastausta kohden, lisättiin putkiston päälle koko putkiston kokoinen lähes näkymätön objekti. Tämän tarkoitus on estää hiiripainallukset vääriin aikoihin putkiston päällä. Objekti päästää painalluksen läpi vain jokaisen oikean vastauksen jälkeen. Tällä ratkaisulla ei tarvitse poistaa käytöstä putkiston tapahtumakäsittelijöitä aina kun putkien lisääminen pitää estää, eikä lisätä tapahtumakäsittelijöiden funktioon minkäänlaista tarkistusta. Kuvassa 11 näkyy putkipeli oikean vastauksen jälkeen.



Kuva 11. Putkipelin näkymä.

Putkipeliin pyydettiin kaareutuvaa tekstiä kysymyksiin. Tämän toteuttaminen ei kuitenkaan onnistu flashilla kovinkaan helposti. Yksittäistä tekstikenttää ei voi taivuttaa, joten kaaret täytyisi toteuttaa jakamalla tekstikenttä osiin ja kääntää tekstin osat sisältävät tekstikentät kaaren muotoon. Ongelman selittämisen jälkeen asiakas suostui jättämään kaaret suunnitelmasta.

Intervallipeli

Putkipelin jälkeen toteutettiin intervallien harjoitteluun suunnattu peli. Peli toimii hyvin samalla tavalla kuin musiikkiopistolle tehty kalastuspeli. Ruutuun tulee kuvia intervalleista, ja pelaajan täytyy valita minkälainen intervalli on kyseessä. Oikeasta vastauksesta saa tietyn määrän rahaa, ja seuraava intervallikuva tulee ruutuun.

Kalastuspelin ongelmista viisastuneena intervallipelin arvottavat kuvat toteutettiin tekemällä kuville sprite-luokkaan pohjautuva luokka. Luokan olioita luodessa luokkamuodostimelle annetaan ladattavan kuvan suhteellinen sijainti sekä intervalliluokan nimi. Vastauksia tarkistettaessa funktio kysyy oliolta luokan nimeä ja vertaa sitä vastattuun arvoon. Näin tehtynä toteutus on huomattavasti selkeämpi, eikä sisäisten nimien kanssa tule ongelmia. Intervallikuville luotu luokka on nähtävillä liitteessä 1.

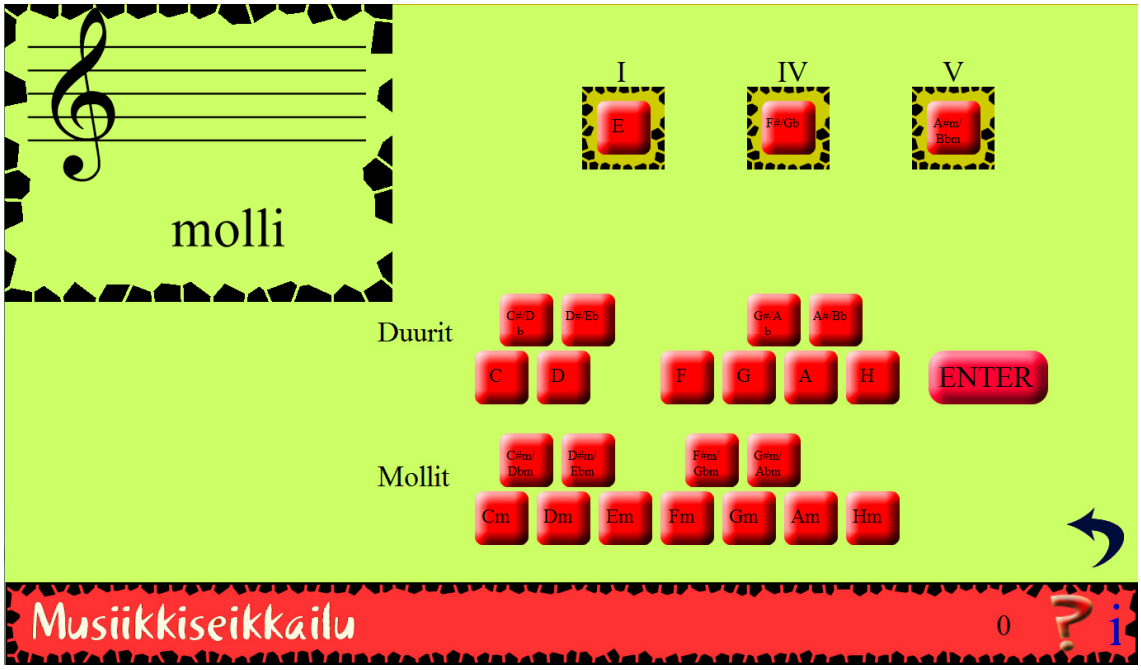
Intervallipeliin päätettiin käyttää teemana samantyylistä putkistoa kuin putkipelissäkin, koska putkiston tekemiseen käytettiin sen verran paljon vaivaa. Putkistoa kuitenkin hieman muutettiin lisäämällä putkiston sisään kohtia, joihin putkenpätkiä ei voi lisätä. Tämä pakottaa pelaajan suunnittelemaan putkistoaan hieman enemmän.

Sointupeli

Kolmas peli on suunnattu sointujen harjoitteluun. Pelissä tulee ruudulle kuva soinnusta, jonka jälkeen pelaajan täytyy siirtää kolmeen laatikkoon sointua vastaavat duurit tai mollit. Tämän jälkeen painetaan vastaamisnappia ja oikean vastauksen saadessaan arpoo uuden sointukuvan.

Putkistoa ei sointupelissä enää käytetty. Sointujen siirtämisessä laatikoihin on jo riittävästi tekemistä, eikä peliä tarvinnut monimutkistaa turhaan. Vastaamisnapin yhteyteen oli myös tarkoitus lisätä valittujen duurien tai mollien kuuleminen, josta pelaaja voisi hieman saada virhetilanteissa apuja. Tähän tarvittavia ääniä ei kuitenkaan ajoissa saatu.

Pelin oikean vastauksen tarkistus liittää kolmen valitun nuotin tekstit yhteen ja vertaa lopputulosta sointukuvan näyttävän olion sisältämään tekstiin. Jos nämä täsmäävät, vastaus on oikein. Kuvassa 12 näkyy sointupeli kolmella valitulla nuotilla.



Kuva 12. Sointupeli.

Sointujen siirtämisessä tuli vastaan ongelma flashin kerroksellisuuden ja toimintakuuntelijoiden käyttäytymisen kanssa. Nuotit ovat ruudulla lataamisjärjestyksessä päällekäin. Jos ensiksi ladattua siirtää myöhemmin ladatun päälle, menee siirrettävä nuotti sen alle. Jos tällöin päästää hiirestä irti, toimintakuuntelija pitää toiminnan aiheuttajana päälimmäistä nuottia. Tämä oli kuitenkin yksinkertainen ratkaista vaihtamalla siirrettävä nuotti aina päällekkäisyydessä päälimmäiseksi.

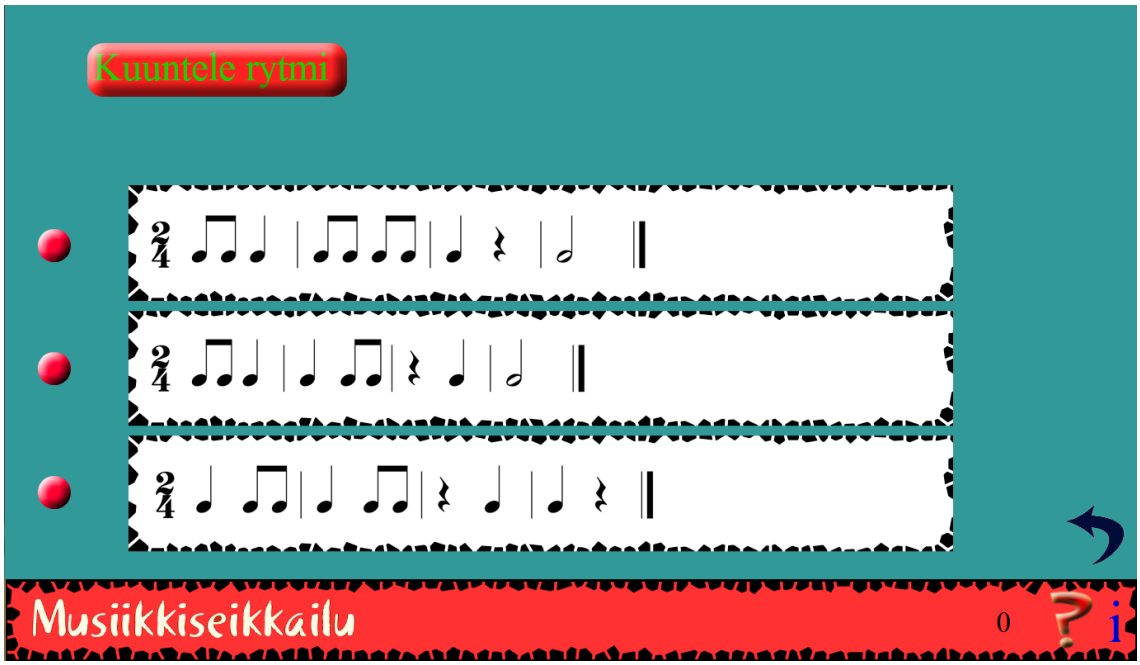
Rytmipeli

Neljäntenä ja viimeisenä pelinä tehtiin rytmien harjoitteluun suunnattu peli. Pelissä kuunnellaan rytmi ja sen jälkeen valitaan rytmiä vastaava kuva painamalla kuvan vieressä olevaa nappia tai kuvaa itseään.

Peli toteutettiin tekemällä kaksi taulukkoa, joista toisessa on rytmiäänet sisältävät oliot ja toisessa vastauskuvat. Ensin arvotaan indeksinumero, jonka avulla ladataan

molemmista taulukoista toisiaan vastaavat oliot ja tämän jälkeen kuvien sijainnit arvotaan, etteivät ne olisit joka kerta samassa kohdassa. Tämä aiheutti pienoisen ongelman vastausnappien kanssa, sillä ne eivät tienneet mikä kuva nappia vastaavassa kohdassa on eivätkä sen myötä pystyneet välittämään oikean vastausvaihtoehdon arvoa. Tämä ratkaistiin tekemällä taulukko, joka pitää kirjaa siitä, mikä vastausolio on missäkin vastauskohdassa. Taulukon kautta vastauksen tarkistava funktio pääsee käsiksi napinpainallusta vastaavaan olioon.

Pelissä oikeasta vastauksesta saa 20 pistettä ja väärästä menettää 5. Kuvassa 13 näkyy yksi kysymys rytmipelissä.



Kuva 13. Rytmipeli.

8 Yhteenveto ja jatkosuunnitelmat

Flash ja actionscript 3 huomattiin paljon monipuolisemmiksi kuin ennen projekteja oletettiin. Flashin huonohko maine johtunee lähinnä yli-innokkaista mainostajista, jotka laittavat mainoksiinsa ärsykevärejä ja ääniä huomiota kiinnittämään ja samalla huonontavat sivuston selauskokemusta. Syy ei siis niinkään ole flashissa itsessään, vaan sen käyttäjissä.

Flash tuskin koskaan tulee kunnolla haastamaan javaa monialustaisien ohjelmien kehityksessä, mutta aloittelevalle kehittäjälle se sopii erityisen hyvin WYSIWYG-ympäristönsä takia. Internet-sivustojen puolella flashin tulevaisuus riippuu suurelta osin HTML5:n menestyksestä.

Flashin kehittäminen oli kokemuksena keskimäärin mielyttävää. Flashissa kuitenkin esiintyi ainakin käytetyn kääntäjän ja kehitysympäristön kanssa hieman bugeja, joista osa oli melko vakaviakin. Uusimmassa versiossa ne lienevät kuitenkin korjattu.

Musiikkiopiston musiikkipeliin olisi jatkotoiveita, mutta ainakaan toistaiseksi rahoitusta niille ei ole. Musiikkiseikkailu kirjan korttelialustaan sen sijaan lisätään vielä tulevaisuudessa muutama peli eri musiikin teorian osa-alueista. Tämän jälkeen pelien kehitystä mahdollisesti jatkaa joku muu.

Korttelipelistä on julkaisemisen jälkeen tullut erittäin paljon hyvää palautetta. Monen koulun rehtorit ovat olleet ihmeissään kuinka hyviä pelejä kohtuullisellakin vaivalla voi saada aikaan. Korttelin ja sen sisältämien pelien kääntämisestä ruotsiksi on saatu tarjouspyyntö. Käännösprojektin toteutumisesta ei vielä kirjoitusvaiheessa ole tietoa.

Projektien kuluessa opittiin flashista ja actionscriptistä huomattava määrä uusia asioita. Jos ensimmäisen projektin alun asiat tekisi näin jälkikäteen uudelleen, tulisi koodista paljon elegantimpaa ja selkeämpää.

Kirjoittamishetkellä musiikkiopistolle tehdyt ohjelmat löytyvät osoitteesta http://www.soitinpolku.fi/ ja WSOY:n apurahoilla tehty korttelipeli osoitteesta http://www.musiikkiseikkailu.com/musiikkiseikkailu/korttelipeli/musapeli.swf.

Lähteet

1 Joey Lott, Danny Patterson. 2006 Advanced ActionScript 3 with design patterns. Kalifornia: Peachpit Press.

2 Grandmasters of Flash: An Interview With the Creators of Flash. 2008. Cold Hard Flash. Verkkodokumentti. <http://coldhardflash.com/2008/02/grandmasters-of-flash-an-interview-with-the-creators-of-flash.html> 12.2.2008. Luettu: 8.1.2011

3 The Flash history. 2000. Verkkodokumentti. Rick Waldron. <http://www.flashmagazine.com/news/detail/the_flash_history/ > 20.11.2000. Luettu: 9.1.2011

4 AS3 Programming 101 for C/C++ coders. 2006. Verkkodokumentti. Darrick Brown. <http://blogs.adobe.com/kiwi/2006/05/as3_programming_101_for_cc_cod_1.html > 27.5.2006. Luettu 9.1.2011

5 ActionScript 3: Dynamic Classes. 2006. Verkkodokumentti. Fain. http://flexblog.faratasystems.com/2006/10/02/actionscript-3-dynamic-classes>. 2.10.2006. Luettu 18.11.2010

6 ECMAScript Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/ECMAScript> Luettu 1.12.2010

7 ActionScript 3 for Java Programmers. 2008. Verkkodokumentti. InfoQ <http://www.infoq.com/articles/actionscript-java> Luettu 10.1.2011

8 Comparing the syntax of Java 5 and ActionScript 3. 2006. Verkkodokumentti. Farata Systems. <http://flexblog.faratasystems.com/2006/11/12/comparing-the-syntax-of-java-5-and-actionscript-3> 12.11.2006. Luettu 10.1.2011.

9 HMTL5 Video Verkkodokumentti. Wikipedia. <http://en.wikipedia.org/wiki/HTML5_video> Luettu 15.1.2011.

10 HTML5 Audio Verkkodokumentti. W3Schools. <http://www.w3schools.com/html5/html5_audio.asp> Luettu 21.1.2011.

11 Enumerated Types in Actionscript.2004 Verkkodokumentti. Darron Schall. <http://www.darronschall.com/weblog/2004/03/enumerated-types-in-actionscript.cfm> 4.3.2004. Luettu 2.12.2010.

12 Flash ActionScript -Objects. 2004. Verkkodokumentti. Mitch Allen. <http://www.flashmagazine.com/tutorials/detail/flash_actionscript_objects/> 20.5.2004. Luettu 2.12.2010.

13 Game-Based Learning: What it is, Why it Works, and Where it's Going. 2009. Verkkodokumentti. New Media Institute. < http://www.newmedia.org/game-based-learning--what-it-is-why-it-works-and-where-its-going.html> Luettu 2.12.2010.

14 Public Pedagogy Through Video Games.2008. Verkkodokumentti. Game Based Learning. <http://www.gamebasedlearning.org.uk/content/view/59/> Luettu: 3.12.2010.

15 Enter the Game Factor: Putting Theory into Practice in the Design of Peer Factor 2008 Verkkodokumentti. Moeller, Ryan, White, Kim. <http://www.bgsu.edu/cconline/gaming_issue_2008/Moeller_White_Enter_the_game/pedagogy.html> Luettu: 3.12.2010.

Lähdekoodit

Pelien lataamiseen käytettävä xml-tiedosto

```
<?xml version="1.0" encoding="utf-8"?>
<?xml version="1.0" encoding="utf-8"?>
<gamelist>
<game>
<name> sanaputki2 </name>
<info> Valitse oikea käännös. Oikean vastauksen jälkeen voit lisätä ruudulla näkyvän putkenpätkän putkistoon. </info>
<multiplier> 1 </multiplier>
</game>
<game>
<name> intervalli </name>
<info> Valitse oikea intervalliluokka painamalla intervalliluokan nappia </info>
<multiplier> 1 </multiplier>
</game>
</gamelist>
```

Uusien pelien luuranko (skeleton.fla)

```
var parObj:Object;
//onLoaded();
function onLoaded(difficulty:Number) { /*Suoritetaan, kun ohjelma on ladattu. "Konstruktori", kortteli kutsuu. */

		if (this.parent != null) {
			parObj = this.parent;
		}

}
```

```
function handCursor(e:MouseEvent) { // Käytä näitä funktioita käsikursorin luomiseen buttonmoden sijaan
		parObj.Mouse.cursor = "button";
}
function arrowCursor(e:MouseEvent) {
		parObj.Mouse.cursor = "arrow";
}


function addMoney(i:uint) { // Rahaa lisäävä funktio.
		parObj.addMoney(i);
}
```

Intervallipelin nuottiluokka (Interval.as)

```
package {
		import flash.events.*;
		import flash.net.URLLoader;
		import flash.net.URLRequest;
		import flash.display.Loader;
		import flash.display.Sprite;
		import flash.display.Bitmap;

		public class Interval extends Sprite {
				private var type:noteType;
				private var loader:Loader = new Loader();
				public function Interval(u:String, t:String) { //argumentteina kuvan tiedostonimi ja intervallityyppi
						type = new noteType(t);
						var req:URLRequest = new URLRequest("intervallikuvat/"+ u);
						loader.load(req);
```

```
		loader.contentLoaderInfo.addEventListener(Event.COMPLETE, drawNote);

		loader.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR,
ioErrorHandler);
			}
			private function ioErrorHandler(e:Event) {
				throw new Error("Virheellinen tiedostonimi \n"
+ e.toString);
			}

			private function drawNote(e:Event):void {
				addChild(e.currentTarget.content);
			}
			public function getType():String {
				return type.nType;
			}
		}

}
// Intervalin apuluokka
class noteType {
		public var nType:String;
		public function noteType(t:String) {
			t = t.toLowerCase();
			if (t == "pieni" || t == "suuri" || t == "puhdas") {
				nType = t;
			}
			else throw new Error("Virheellinen intervallityyppi. (pieni,
suuri, puhdas)");
		}
}
```