3D-OPPIMISYMPÄRISTÖN TOTEUTTAMINEN ADOBE FLASH CS3:LLA

LAHDEN AMMATTIKORKEAKOULU
Mediatekniikan koulutusohjelma
Opinnäytetyö
Kevät 2008
Jarmo Iivonen

Lahden ammattikorkeakoulu
Tekniikan laitos

IIVONEN, JARMO: 3D-oppimisympäristön toteuttaminen Adobe Flash CS3:lla

Teknisen visualisoinnin opinnäytetyö, 51 sivua, 1 liitesivu

Kevät 2008

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee 3D-oppimisympäristön toteuttamista Adobe Flash CS3:lla. Toteutustavoista käsittelyssä on oppimisympäristö valmiilla kuvilla ja oppimisympäristö 3D-moottorilla toteutettuna.

Teoriaosuudessa perehdytään siihen, mitä vaatimuksia toimivalla oppimisympäristöllä on ja miten ne saavutetaan. Lisäksi teoriaosuudessa käsitellään kolmiulotteisuuden laatuvaatimuksia ja esitellään toteuttamistapojen yleiset asiat.

Hyvän oppimisympäristön vaatimuksina ovat helppokäyttöisyys, yhdenmukaisuus ja selkeys. Helppokäyttöisyydellä tarkoitetaan ohjelman helppoa opittavuutta, joka mahdollistaa käyttäjän keskittymisen ohjelman sisältöön sen sijaan, että keskittyisi itse ohjelmaan. Yhdenmukaisuus tarkoittaa ohjelman eri osien yhteneväisyyttä, joka myös omalta osaltaan edistää ohjelman oppimista. Selkeys on tärkeä osa ohjelman käytön miellyttävyyttä. Kolmiulotteisuuden vaatimuksina päällimmäisinä esiin nousivat kuvallinen laatu ja tiedostokoon alhaisuus.

Case-osuuden tarkoituksena on selvittää, miten kolmiulotteinen oppimisympäristö kuvien ja 3D-moottorin avulla toteutetaan, mitä ongelmia niiden käytössä on ja mitä mahdollisuuksia tekniikoilla on. Vastaukset on tarkoitus selvittää toteuttamalla kolmiulotteiset oppimisympäristöt niiden avulla.

Case-osuudessa saadaan selville, että kuvilla oppimisympäristön toteuttaminen on yksinkertaista, mutta myös syntyvä lopputulos on yksinkertainen. Selvitetään myös että jos yhden istunnon aikana ladataan Flashin sisälle isoja määrä kuvatiedostoja, varaavat ne muistia vaikka tiedostoja ei enää tarvittaisikaan. 3D-moottorin erinomaiseksi puoleksi nousee mahdollisuus täysin interaktiivisen ympäristön toteuttamiseen ja mahdollisuus hyödyntää Flashin omia hyviä puolia. Negatiiviseksi asiaksi ilmenee 3D-moottorin prosessorilta vaatiman tehon määrä.

Loppupäätelmänä voi todeta, että kuvilla toteutettu oppimisympäristö sopii yksinkertaiseen esittämiseen ja 3D-moottorilla toteutettu vaativampaan. 3D-moottoria voi käyttää kuitenkin vain pienikokoisten tilojen esittämiseen.

Avainsanat: Papervision3D, low poly, interaktiivisuus, 3D-oppimisympäristö

Lahti University of Applied Sciences
Faculty of Technology

IIVONEN, JARMO: Making a 3D learning environment with Adobe Flash CS3

Bachelor's thesis in visualization engineering, 51 pages, 1 appendix

Spring 2008

ABSTRACT

This Bachelor's thesis deals with the making of a 3D learning environment using Adobe Flash CS3. There are two ways for making such an environment. The methods are using existing pictures and using a third-party 3D engine like Papervision3D. The purpose of this work is to discover the way to make a 3D environment where it is possible for the user to move and be interactive with the environment. To discover the way, both methods mentioned above are tested and used for building a full 3D learning environment. The purpose is to find the good and the bad things that those different methods have.

A good learning environment is easy to use, uniform and clear. Ease of use means, that it is easy to learn the functions in the program. It helps the user to concentrate on the contents of the program instead of the program itself. Uniformity and clearness also help the user to learn to use the program. Those are the main principles that make the user feel comfortable when using the program. The demands for three-dimensionality are good picture quality and low file size when using the program via Internet.

Creating a 3D environment using existing pictures is an easy way to simulate 3D in Flash, but the pictures are also quite a clumsy way of doing it. Because pictures are static there is no other way than using many of them to simulate movement. When there are a lot of pictures, there are a lot of files and bytes to load. When using many pictures in the Flash Player, the pictures start to reserve the computer's memory even when the pictures are not in use or have already been deleted.

Usage of Papervision3D 2.0 Alpha 3D engine in making the 3D learning environment shows that there is good potential for three-dimensionality in Flash. The 3D-engine enables free movement in the environment and lets the user to immerse into the 3D. The negative side of using Papervision3D is that the 3D engine reserves power from the computer's processor because it does not utilize the graphic card's memory and potential. 1500 triangles is nearly the highest number of faces that a 3D environment can have, because the rendering slows down remarkably after that number.

As a conclusion it can be said that a 3D learning environment made with existing pictures is suitable for a simple showcase and an environment made with a 3D engine is for more demanding usage. A 3D engine can only be used in cases where the number of faces is moderate.

Keywords: Papervision3D, low poly, interactivity, 3D environment

SISÄLLYS

1 JOHDANTO 1

2 3D-OPPIMISYMPÄRISTÖN VAATIMUKSET 2

2.1 Oppimisympäristö 2

2.2 Kolmiulotteisuus 2

2.2.1 Laatu 2

2.2.2 Interaktiivisuus 4

2.2.3 Sisällön päivitettävyys 5

2.3 Käyttöliittymä 5

2.3.1 Käyttäjäystävällisyys 5

2.3.2 Käytettävyys 6

2.3.3 Vuorovaikutus ja havainnointi 9

2.4 Käyttöympäristö 10

2.4.1 Internetselain 10

2.4.2 Paikallinen versio 11

3 TYÖKALUT 12

3.1 Adobe Flash CS3 12

3.1.1 Historia 12

3.1.2 Esittely 12

3.2 Papervision3D 2.0 Alpha - 3D-moottori Adobe Flash CS3:een 13

3.2.1 Esittely 13

3.2.2 Toimintaperiaate 14

4 KOLMIULOTTEISUUS ADOBE FLASH CS3:ssa 16

4.1 Valmiit kuvat 3D:nä 16

4.1.1 Mallintaminen ja renderöinti 16

4.1.2 Oikeanlaiset tiedostotyypit 17

4.1.3 Vuorovaikutteisuuden toteuttaminen 17

4.1.4 Kuvien päivittäminen 18
4.2 3D toteutettuna kolmannen osapuolen 3D-moottorilla 18
4.2.1 3D-objektien teko 18
4.2.2 Tiedostomuodot 19
4.2.3 3D-objektien tuominen Flashiin 20
4.2.4 Vuorovaikutteisuuden toteuttaminen 20
4.2.5 3D-objektien päivittäminen 21
5 CASE: ENGLANNIN KIELEN OPETUSOHJELMAN 3D-OPPIMISYMPÄRISTÖ 22
5.1 Lahden Teho-Opetus Oy 22
5.2 Avenue – Englannin kielen opetusohjelma 22
5.2.1 Ohjelman idea 22
5.2.2 Ohjelman runko ja käyttöliittymä 23
5.2.3 Ympäristöjen lataaminen ohjelmaan 24
5.3 3D-oppimisympäristön toteuttaminen bittikarttakuvilla 25
5.3.1 Ympäristön luominen mallinnusohjelmassa ja renderöinti 25
5.3.2 Ympäristön kokoaminen Flashiin ja ohjauspainikkeet 27
5.3.3 Ympäristön painikkeet 29
5.3.4 Avatar 31
5.4 3D-oppimisympäristö kolmannen osapuolen 3D-moottorilla 33
5.4.1 Objektien ja tekstuurien luominen mallinnusohjelmassa 33
5.4.2 Ympäristön kokoaminen Flashissa 36
5.4.3 Liikkuminen ympäristössä 39
5.4.4 Tehtäväpainikkeet 42
5.5 Case-tarkastelu 45
6 YHTEENVETO 47
LÄHTEET 49
LIITTEET 52

SANASTO

3D	Three-dimensional, kolmiulotteinen
ASF	Advance Streaming Format, Windows-mediatiedosto
AVI	Audio Video Interleave, videotiedosto
BMP	Bitmap, kuvaformaatti
DV	Digital Video, videokameroiden tallennusformaatti
FLV	Flash Video, Flashin videotiedosto
JPG, JPEG	Joint Photograghic Experts Group, kuvatiedosto
low poly	3D-objekti, jossa mahdollisimman vähän pintoja
MIT-lisenssi	Intransitiivinen lisenssi, joka sallii vapaan käytön
MOV	Applen QuickTime-videotiedosto
MP4	Applen QuickTime-videotiedosto
MPEG, MPG	Moving Pictures Experts Group, videostandardi
Plug-in	Laajennos tietokoneohjelmaan
PNG	Portable Network Graphics, häviötön kuvanpakkausformaatti
SWF	Flashista julkaistu tiedostotyyppi
TGA	Targa, pikselikuvien tallennusmuoto
TIFF	Tagged Image File Format, monipuolinen kuvaformaatti
UVW	Kuvankartoitusmenetelmä
WMV	Windows Media Video, Windows-mediatiedosto

1 JOHDANTO

Tässä opinnäytetyössä keskitytään Adobe Flash CS3:lla tehtävän kolmiulotteisen oppimisympäristön toteuttamiseen. Opinnäytetyön idea on lähtöisin Lahden Teho-Opetus Oy:n Flash-alustalle tehdyn peruskoulun 3-6 luokan englannin kielen ohjelmasta, jossa 3D-oppimisympäristö on toteutettu kuvien avulla. Opinnäytetyön tavoitteena on tutkia, mitä vaatimuksia 3D-oppimisympäristöllä yleisesti on ja millä keinoin vaatimukset saavutetaan.

Teoriaosuudessa eritellään, mitä vaatimuksia on kolmiulotteisuudella ja mitä vaatimuksia on oppimisympäristöllä. Teoriaosuudessa myös esitellään oppimisympäristön toteuttamistapoihin liittyvät asiat, jotka tarkemmin käsitellään ja joita käytetään Case-osuudessa. Työssä ei puututa pedagogiseen puoleen, johtuen sen laajuudesta ja opinnäytetyön otsikon rajauksesta.

Case-osuudessa käydään lävitse oppimisympäristön toteuttaminen sekä kuvien että oikean 3D:n avulla. Flash ei perinteisesti ole toiminut suoraan reaaliaikaisen 3D:n esitysalustana, mutta sillä on kuitenkin kyetty simuloimaan 3D:tä hyödyntämällä valmiita kuvia. Kesällä 2007 Adobe Flash CS3:lle julkaistiin ensimmäinen MIT-lisenssillä toimiva 3D-moottori nimeltään Papervision3D, jolla pystytään ilman liitännäisiä esittämään Flashissa aitoa kolmiulotteisuutta. Tavoitteena on saada vastaus siihen, onko oikea 3D parempi ja enemmän mahdollisuuksia oppimisympäristön toteuttamiseen tarjoava kuin perinteiset valmiit 3D-kuvat.

2 3D-OPPIMISYMPÄRISTÖN VAATIMUKSET

2.1 Oppimisympäristö

Oppimisympäristöllä tarkoitetaan paikkaa, tilaa, yhteisöä tai toimintakäytäntöä, jonka tarkoituksena on edistää oppimista (Soila & Tervola 2003, 11). Tässä opinnäytetyössä on keskitytty pelinomaiseen ympäristöön, jossa käyttäjä voi liikkua rajoitetusti ja suorittaa tehtäviä.

Oppiminen tämäntyyppisessä ympäristössä tapahtuu autenttisesti, eli käyttäjä oppii asioita kognitiivisten prosessien kautta. Tällaisia prosesseja ovat asioiden havainnointi, tekeminen, aistiminen, tunteminen ja kokeminen. Käyttäjälle saadaan luotua olotila, jossa tärkeämpää on tehtävien suoritus ja pelissä eteneminen kuin itse oppiminen. Oppiminen on kuitenkin lähtökohtana pelin tekemiselle.

Jotta hyvä oppimisympäristö saadaan tehtyä, täytyy sen pystyä ylläpitämään käyttäjän mielenkiintoa. Tähän päästään ympäristön graafisella ilmeellä, helppokäyttöisyydellä, pelinomaisuudella, tarinalla, joka etenee, tai palkinnolla, jonka käyttäjä saa edetessään pidemmälle ja paremmin. Hyvässä oppimisympäristössä käyttäjä pystyy myös seuraamaan edistymistään reaaliaikaisesti.

Sähköinen oppimisympäristö voi olla koulussa tuntiopetuksen tukena tai ainoana opetuksen lähteenä vaikkapa kotiopetuksessa. Tämän vuoksi ohjelmassa on syytä olla mahdollisuus, jossa myös opettaja tai vanhempi pystyy seuraamaan käyttäjän etenemistä. Tämänlainen seurantaohjelma on hyvä pitää erillään itse opetusohjelmasta, sillä oppimisympäristöön ei kannata sisällyttää muuta kuin itse oppimiseen liittyvät asiat. Hyvän oppimisympäristön vaateena onkin tarjottavan informaation rajoittaminen. Pitää pystyä jättämään turhat asiat pois, jotta oppiminen on tehokasta.

2.2 Kolmiulotteisuus

2.2.1 Laatu

Kolmiulotteisuuden laatu tietokoneen näytöllä tarkoittaa kuvan perspektiivin vaikutelmaa sekä 3D-objektien fotorealistista laatua (Lehtovirta & Nuutinen 2000, 65). Puhuttaessa tietokoneen näytöstä on kyseessä aina kaksiulotteinen kuva. Jotta kaksiulotteisesta kuvasta saadaan kolmiulotteinen vaikutelma, vaatii se kuvassa olevien asioiden ”pakenemista” yhteen pakopisteeseen, joka sijaitsee kuvan kes-

kellä horisontissa, koska kappaleesta pitää pystyä erottamaan sen etu- ja takaosa suhteessa katselupisteeseen (Hearn & Baker 1994, 299). Perspektiivikään ei vielä todellista kolmiulotteisuuden illuusiota tuota, vaan tarvitaan myös varjot ja sitä kautta mielikuva valonlähteestä. Realistisen 3D:n tuottamiseen tarvitaan luonnolliset valot. Jos kuvan objektien materiaalit ja värit ovat kaikki tasavärisiä ilman varjoja, ei haluttuun lopputulokseen päästä (Hearn & Baker 1994, 495). Myös valon tyyppi on tärkeä tekijä. Luonnollinen valo leviää ympäristöön heijastumisien kautta ja samalla luo pehmeitä varjoja, jotka lisäävät realismin tuntua (Lehtovirta & Nuutinen 2000, 65).

Viimeisenä, mutta ei vähäisimpänä, kolmiulotteisen vaikutelman tuottajana toimii kamera, joka vastaa katselijan silmiä tilassa. Kamera on ensimmäinen askel kolmiulotteisen kuvan välittämiseen, ja se määrittää muiden kappaleiden asettelun tilaan (Hearn & Baker 1994, 297). Kameran käyttämiseen on eri mahdollisuuksia. Sillä voidaan ottaa kuvat eri kuvakulmista ja näyttää valmiita kuvia katselijalle. Tällainen tulee kyseeseen esimerkiksi silloin, kun tehdään yksinkertainen kameraajo, johon käyttäjä ei voi eikä hänen tarvitse vaikuttaa. Interaktiivisessa tilassa voidaan käyttää kameraa, joka käyttäjän komentojen mukaan kääntyy ja liikkuu sekä samanaikaisesti päivittää näytöllä olevaa kuvaa. Esimerkkejä erilaisista tiloissa olevista kameroista ovat panoraamakamera ja vapaa kamera. Panoraamakameraa voidaan kääntää yhdessä paikkapisteessä sen pystyakselin ympäri, jotta saadaan aikaan kuvaus ympäristöstä. Usein panoraamakameralla luodaan 360 asteinen kuvaus ympäristöstä. Vapaata kameraa voidaan liikutella ja käännellä vapaasti, mikä antaa käyttäjälle kokonaisvaltaisen tilan tunteen.

Fotorealistinen laatu on myös tärkeä tekijä, sillä se osaltaan määrittää kuinka kiinnostava näytöllä oleva kuva on. Kuvan laatuun vaikuttavat monet eri asiat, ja yksi iso tekijä on jo aikaisemminkin mainittu valojen ja varjojen käyttäminen (Lehtovirta & Nuutinen 2000, 65). Kun käytetään valmiita bittikarttakuvia, pitää pakkausjäljen olla kohtuullista, ja kun sitä pyritään vähentämään, tulee bittikarttakuvien ongelmaksi niiden suuri koko. Jos käytetään vapaata kameraa tai panoraamakameraa valmiiden kuvien tilalla, täytyy kuvan laadussa kiinnittää huomioita ympäristön objektien näkymiseen eli siihen, miten materiaalit toistuvat ja ovatko objektien reunat sahalaitaisia. Antialiasointi, eli pikselirajojen pehmennys, poistaa sahalaitaisuuden objekteista, mutta vaatii suurempaa laskentatehoa koneelta (Lehtovirta & Nuutinen 2000, 65). Jos kuva on heikkolaatuinen eikä vastaa tekniikan suomiin mahdollisuuksin, ei se myöskään ole houkutteleva eikä myyvä.

2.2.2 Interaktiivisuus

Interaktiivisuus on tekijä, jonka takia 3D:tä kannattaa käyttää opetusympäristön toteuttamisessa. Käyttäjälle avataan interaktiivisuuden myötä aivan uusia mahdollisuuksia tarkastella asioita näytöllä, ja hänelle pystytään luomaan entistä tehokkaammin tunne kuin hän olisi pelissä tai sovelluksessa itse sisällä ja mukana. (Lehtovirta & Nuutinen 2000, 63.)

Interaktiivista vaikuttamista on kolmiulotteisessa ympäristössä kahta eri tyyppiä: toimintaa, jossa käyttäjä tekee jotain, johon ympäristö reagoi, sekä toimintaa, jossa ympäristössä tapahtuu jotain, johon käyttäjä reagoi. Vuorovaikutuksen voi myös jakaa alemmalle tasolle passiiviseen ja aktiiviseen toimintaan. Passiivisessa vaikuttamisessa käyttäjää ohjataan keskittymään ympäristössä johonkin tiettyyn asiaan ilman, että siitä suoraan kerrotaan. Aktiivisesti sama asia kerrottaisiin sanallisesti tai korostamalla jollakin tavalla, jotta käyttäjältä saadaan tarvittava huomio. Aktiivinen vuorovaikutus voi kuitenkin olla liian läpinäkyvää ja häiritsevää, joten sen käyttämistä erityisesti kolmiulotteisessa ympäristössä on harkittava. Kuvaava ilmaus vuorovaikutteisuuden käyttämiseen on: ”Mitä vuorovaikutteisempi jokin sovellus on, sitä vähemmän sen avulla voi kertoa ennalta suunniteltua tarinaa.” (Metsämäki 1998.)

Jos kolmiulotteisessa ympäristössä käytetään objekteja, joita käyttäjän on tarkoitus voida hiirellä painaa, on niissä syytä käyttää selvää yhdenmukaisuutta ja jotakin muusta ympäristöstä erottelevaa tekijää. Tällaisissa objekteissa joita voi koskettaa, on käyttäjälle hyvä antaa jonkinlainen palaute toiminnasta. Ilman palautetta käyttäjä voi saada käsityksen että mitään ei tapahtunut. Palautteen antamiseen on monia eri vaihtoehtoja. Se voi tulla äänimerkkinä, jolloin näytöltä ei viedä tilaa, mutta jokaisella käyttäjällä ei välttämättä ole äänikorttia. Painettu objekti voidaan korostaa toisella värillä, jolloin käyttäjälle on selvää mitä on painettu. Viesti voi tulla myös sanallisena, jolloin kannattaa käyttää erillistä viestikenttää, jotta käyttäjä tietää, mihin viesti tulee, jos viestejä annetaan. Perinteinen tapa viestin näyttämiseen on käyttää Windowsistakin tuttua pop-up -tekstiä kursorin vieressä. (Hearn & Baker 1994, 275.)

2.2.3 Sisällön päivitettävyys

Ympäristössä sijaitsevien objektien ja näkyvien asioiden päivittäminen on tärkeä asia, joka vaikuttaa kolmiulotteisen ympäristön tekemiseen ja toteuttamistapaan. Jos käytetään valmiiksi tuotettuja kuvia eri kuvakulmista, täytyy kaikki kuvat kyseisestä tilasta tehdä uudestaan päivityksen yhteydessä. Jos käytetään ympäristöä johon objektit tuodaan erikseen, riittää silloin yksinkertaisimmillaan vain yhden objektin päivittäminen. Jos on iso objektimäärä joita pitää päivittää, on helpompaa käyttää valmiiksi tehtäviä kuvia verrattuna siihen, että kaikki objektit yksitellen lisättäisiin ympäristöön.

Päivitettäviä elementtejä, joita kolmiulotteisessa oppimisympäristössä pitää pystyä muuttamaan, ovat muun muassa objektien materiaalit ja värit. Niiden vaihtaminen tulee kyseeseen ympäristön valaistuksen muuttamisen yhteydessä. Muita muutoksen kohteita ovat objektien, painikkeiden ja niiden toimintojen poistaminen ja lisääminen. Näiden lisäksi voi olla muutakin, mitä ympäristössä pitää pystyä muuttamaan, ja ne kaikki on otettava huomioon ympäristöä tehtäessä.

2.3 Käyttöliittymä

2.3.1 Käyttäjäystävällisyys

Käyttöliittymä määrittää pitkälti sen, onko ohjelma käyttäjäystävällinen vai ei. Vaikka ohjelman muut osat olisivat kunnossa, voi kaiken hyödyn tehdä olemattomaksi käyttöliittymän huono käytettävyys.

Hyvän käyttöliittymän viisi peruspilaria ovat Nielsen yleiset käytettävyystekijät; opittavuus, tehokkuus, muistettavuus, virheettömyys ja tyytyväisyys (Kalimo 1995, 22). Näiden kohtien huomioiminen käyttöliittymää tehdessä helpottaa huomattavasti käyttöliittymän tekemistä ja käyttämistä. Yhdenkin tekijän kohdalla esiintyvä puute aiheuttaa viivästystä varsinaisen ohjelman käytössä, joka osaltaan vähentää ohjelmasta saatavaa hyötyä. Opetusohjelman kohdalla puutteet tarkoittavat suoraan oppimisen laadun ja määrän heikentymistä.

Opittavuus

Opittavuus tarkoittaa sitä, kuinka nopeasti käyttäjä sisäistää käyttöliittymän käyttämisen. Jos ohjelman ja käyttöliittymän käytön oppiminen vie liian kauan, turhautuu käyttäjä helposti ohjelmaan. (Kalimo 1995, 22.)

Tehokkuus

Tehokkuudella tarkoitetaan ohjelman valikoiden ja toimintojen vaatimaa aikaa suhteessa itse ohjelman ajan käyttöön, eli mitä enemmän käyttäjällä kuluu aikaa ohjelman valikoissa olemiseen, sitä vähemmän aikaa jää itse ohjelmalle. (Kalimo 1995, 22.)

Muistettavuus

Muistettavuudella tarkoitetaan sitä, kuinka hyvin käyttöliittymän käytön muistaa vielä edellisen käyttökerran jälkeen. Peruskoulun opetusohjelman kohdalla tämä tarkoittaa sitä, että käyttäjän tulisi muistaa ohjelma vielä pitkän kesälomankin jälkeen. (Kalimo 1995, 22.)

Virheettömyys

Virheettömyys tarkoittaa palautumista tilanteesta, jossa käyttäjä on tehnyt vaikkapa näppäilyvirheen tai erehtynyt painamaan väärää painiketta. Tällaisista tilanteista tulisi olla mahdollisimman helppo palata takaisin, mutta suositeltavaa olisi, että tällaisia virheitä ei pääsisi tapahtumaan lainkaan. Virhe voi olla myös ohjelman sisäinen, joita niitä ei saisi olla yhtään. (Kalimo 1995, 22.)

Tyytyväisyys

Tyytyväisyys on käyttöliittymän helppokäyttöisyyttä. Jos käyttöliittymä on vaikea käyttää, on ohjelma käyttäjän mielestä epämiellyttävä. (Kalimo 1995, 22.)

2.3.2 Käytettävyys

Kun käyttöliittymä ja itse ohjelma ovat siinä vaiheessa, että ohjelman voisi jo teoriassa ottaa käyttöön, on sen käytettävyys syytä vielä laittaa kriittisen tutkinnan alle ja kyseenalaistaa kokonaisuudessaan. Tämäntyyppisiä tarkistuslistoja on olemassa useita erilaisia. Yhtenä tarkistuslistana toimivat Nielsenin kymmenen heuristista sääntöä, jotka hän on koonnut jo vuonna 1990. Säännöt ovat laaja-alaisia, joten niitä pitää osata soveltaa oman ohjelman ja käyttöliittymän mukaan. Ne pystyvät silti antamaan kattavan arvioinnin ohjelman käytettävyydestä. Seuraavassa on pieni esittely kustakin säännöstä, sovellettuna peruskouluun suunnatun opetusohjelman tarpeisiin. (Kalimo 1995, 38.)

Yksinkertainen ja luonnollinen dialogi

Käyttöliittymän pitäisi olla mahdollisimman yksinkertainen, eikä siinä saisi olla mitään turhaa. Opetustarkoitukseen suunnatun ohjelman kohdalla tämä pätee erityisesti, sillä oppilailta ei pitäisi vetää huomiota liiaksi pois opetuksellisista asioista. Asioiden pitäisi olla selkeästi erottuvia, mutta kuitenkin mielenkiintoisesti esitettyjä. Painikealueiden ja harjoitusalueiden paikat on hyvää pitää selkeinä ja muuttumattomina läpi ohjelman, jotta yksinkertaisuus säilyy. Eri osa-alueissa käytettäviin väreihin tulee erityisesti peruskouluun suunnatuissa ohjelmissa panostaa, jotta ohjelma pysyy houkuttelevana ja kuvaannollisesti lapsellisena, kuitenkaan aliarvioimatta oppilaiden tietoutta ja makua tyyliasioihin. (Kalimo 1995, 38.)

Käyttäjän oma kieli

Opetusohjelmassa ei kannata käyttää tietokonesanastoa, sillä suurin osa käyttäjistä ei tunne sen termejä. Sanaston kannattaa olla mahdollisimman yksinkertaista ja helposti ymmärrettävää, jotta sekaannuksille ei jää mahdollisuutta. Käyttäjän kielellä tarkoitetaan myös ohjelman painikkeita. Hyvä keino mielenkiintoisen ja näyttävän ohjelman tekoon on käyttää painikkeissa hauskoja symboleita ja houkuttelevia värejä. Pitää kuitenkin muistaa, että symbolit ovat tulkinnanvaraisia, minkä vuoksi niissä kannattaa käyttää harkiten itse keksittyjä uusia kuvioita. Useimmille tietokoneen käyttäjille on selkeämpiä jo vanhat symbolit, kuten esimerkiksi Windows-käyttöjärjestelmän sulkemis- ja paluu-painikkeet. (Kalimo 1995, 38.)

Käyttäjän muistikuorman minimointi

Muistikuorman minimoinnilla tarkoitetaan käyttäjältä kysyttävien asioiden määrän pienentämistä. Ei kannata kysyä samoja asioita uudestaan vaan tarjota vanhaa vastausta tai valintaa, jonka käyttäjä on jo kertaalleen antanut. Tämä tulee kyseeseen opetusohjelman kohdalla esimerkiksi suoritettujen tehtävien pistemäärän ilmoittamisessa tehtäväpainikkeen vieressä, jotta käyttäjän ei tarvitse avata harjoitusta nähdäkseen pisteensä tai yrittää muistaa sitä itse. (Kalimo 1995, 38.)

Yhdenmukaisuus

Yhdenmukaisuus tulisi säilyttää läpi ohjelman. Kaikkien painikealueiden ja painikkeiden pitäisi olla samassa järjestyksessä jokaisessa valikossa, ja taulukoiden ja luetteloiden pitäisi olla samalla tavalla luettavissa. Jos toisessa taulukossa informaatio on vaakatasossa, ei sen pitäisi olla seuraavassa taulukossa pystysuun-

nassa. Yhdenmukaisuuden pitää olla kaikkialla, eikä samasta painikkeesta saa tarjota eri tapahtumaa ohjelman eri osissa. Yhdenmukaisuuden puute lisää ohjelman käytön oppimiseen tarvittavaa aikaa. Jos ohjelmasta tulee jatkoversio, on sen hyvä olla edeltäjänsä kaltainen. (Kalimo 1995, 38.)

Riittävä palaute

Palautteen antaminen käyttäjälle on ensisijaisen tärkeää. Ohjelman lataamisesta ja odottelujen syistä on hyvä kertoa käyttäjälle, ettei hänelle tule tunnetta ohjelman jumiutumisesta. Palaute on hyvä antaa jokaisesta napin painalluksesta jossakin muodossa. Muoto voi olla napin painuminen pohjaan, tai vaikkapa hiiren osoittimen muuttuminen toiseen muotoon. Tärkeintä on, että käyttäjälle kerrotaan jotain. Opetusohjelmassa palautetta on hyvä antaa myös jäljellä olevien tehtävien määristä, jolloin oppilaalla pysyisi motivaatio jatkaa harjoitukset tosissaan loppuun asti. (Kalimo 1995, 38.)

Selkeä poistumistapa eri tiloista ja toiminnoista

Jokaisesta tilanteesta pitää tarjota poispääsyn mahdollisuus. Mitä yhdenmukaisempi poistumistapa on, sitä turvallisemmaksi käyttäjä kokee ohjelman käytön. Poistumistie kertoo käyttäjälle, että mahdollisesta virhepainalluksesta pääsee vielä takaisin. Opetusohjelmassa on hyvä olla harjoituksen keskeyttämismahdollisuus. Jos harjoitusta ei voisi keskeyttää, tarvitsisi koko ohjelma silloin sulkea. Tällainen tilanne voi tulla silloin kun oppitunti loppuu kesken, tai harjoitus on tekijälleen ylivoimaisen vaikea. (Kalimo 1995, 38.)

Oikopolut

Oikopolut on tarkoitettu kokeneille käyttäjille, jotka haluavat hyödyntää ohjelman pikanäppäimiä ja saada aikaan mahdollisimman vähillä toiminnoilla. Opetusohjelmissa oikopolut tulevat kyseeseen valikoissa, joissa valitaan harjoitustasoja ja tehtävätyyppejä. Nämä jo aikaisemmin valitut tiedot voidaan pitää esivalittuina, jolloin käyttäjän ei tarvitsisi tehdä samoja valintoja aina uudestaan. (Kalimo 1995, 38.)

Selkeät virheilmoitukset

Jokaisesta käyttäjän tai ohjelman tekemästä virheestä pitää kertoa, ja siitä pitää myös ilmetä mistä virhe mahdollisesti johtui. Virheilmoituksessa on syytä myös

mainita, miten virheestä pääsee eroon tai miten se voitaisiin välttää. Pääperiaate ohjelmissa kuitenkin on se, että virheitä ei saisi tulla yhtään. (Kalimo 1995, 38.)

Virheiden estäminen

Käyttäjätason virheitä voidaan estää kysymällä vahvistuksia peruuttamattomiin toimintoihin. Opetusohjelmissa peruuttamattomia toimia ovat esimerkiksi ohjelman sulkeminen kokonaan tai tehtävistä saatujen pisteiden alustaminen. (Kalimo 1995, 38.)

Riittävä ja selkeä apu

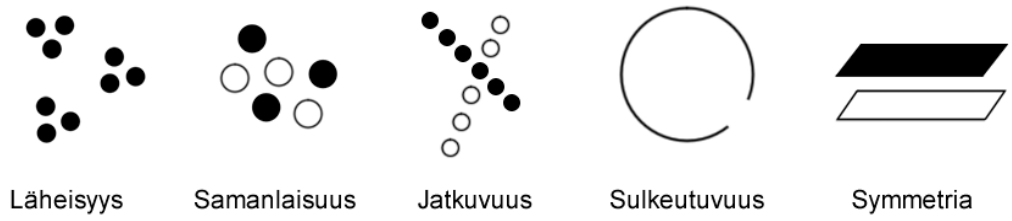
Ohjelman tulisi olla niin helppo käyttää, että siihen ei tarvittaisi erillistä ohjetta. Jokaisessa ohjelmassa on kuitenkin kohtia, jotka voivat olla käyttäjälle epäselviä. Tämän vuoksi on syytä olla olemassa erillinen info-painike. Painikkeen alta ei kannata suoraan löytyä muuta kuin kyseisen ruudulla olevan asian ratkaisemiseen ja käyttämiseen tarvittavat ohjeet. Ohjeistukseen kannattaa kuitenkin erilleen sijoittaa sisällysluettelo, josta tarvittaessa pääsee katsomaan miten ohjelman muissa kohdissa toimitaan. Sähköisen ohjeen lisäksi on ohjelman ohjeistuksesta hyvä tehdä vielä selkeä painettu versio, joka noudattaa samaa tyyliä ja sisällysluetteloa. (Kalimo 1995, 38.)

2.3.3 Vuorovaikutus ja havainnointi

Käyttöliittymässä voidaan vuorovaikutuksen luomiseksi käyttää sanallista palautteen antoa tai vaihtoehtoisesti kertoa käyttäjän toimista symbolien ja sanattomien viestien avulla. Käyttöliittymän vuorovaikutuksesta iso osa on juuri sanatonta viestintää, jossa käyttäjä ei edes tiedä että hänelle viestitään asioita. Hyvä käyttöliittymä on niin havainnollinen ja helppolukuinen, että sen olemassaoloon ei kiinnitä minkäänlaista huomiota. Käyttöliittymä ei saa asettaa käyttäjää tilanteeseen, jossa hän ei löydä vastausta.

Käyttöliittymän vuorovaikutteisuutta voi parantaa hyödyntämällä Preecen hahmolakeja, joita käytetään erityisesti viestinnän puolella. Hahmolait koostuvat viidestä eri osasta: läheisyys, samanlaisuus, jatkuvuus, sulkeutuvuus ja symmetria. Läheisyydellä tarkoitetaan lähekkäin sijaitsevien asioiden kuulumista yhteen. Tästä hyvänä esimerkkinä ovat erilliset linkkipalkit ja painikealueet. Samanlaisuus kuvaa asioiden samankaltaisuuden luomaa yhteenkuuluvuutta, esimerkkinä tästä ovat painikkeiden symbolit. Jos painikkeissa käytetään jossakin tekstiä ja jossakin symboleita, tulee käyttäjälle ristiriitainen tunne painikkeiden tarkoituksesta. Jat-

kuvuus kuvaa asioiden jatkuvuutta. Esimerkkinä jatkuvuudesta on rastin mieltäminen kahdeksi risteäväksi viivaksi, eikä neljäksi kohtaavaksi viivaksi. Tätä ominaisuutta voidaan hyödyntää painikkeiden sijoittelussa ja käyttöliittymän näkymän suunnittelussa. Sulkeutuvuus ja symmetria ovat ihmisen tapoja hahmottaa kuvioiden muotoja. Vaikka objekti olisikin avoin ympyrä, hahmottaa katsoja silti sen perusmuodon (Lankoski, 2001).



KUVA 1. Preecen hahmolait (Iivonen 2008).

Vaikka vuorovaikutteisuutta on hyvä luoda sanattomin keinoin, on käyttäjälle silti syytä antaa riittävä palaute ja apu myös selvästi sanallisesti. Sillä tavoin voidaan varmistaa viestin perille meno ja vältetään mahdolliset väärinymmärrykset. Hyviä keinoja siihen on jo heuristissa säännöissäkin mainitut info-painikkeet ja osoitintekstit.

2.4 Käyttöympäristö

2.4.1 Internetselain

Internet antaa opetusohjelmalle mahdollisuudet tiedon päivittämiseen ja ylläpitoon verkon välityksellä. Kun opetusohjelmaa käytetään verkon kautta palvelimelta, voivat samanaikaisesti useat käyttäjät olla kirjautuneena palveluun. Internet tarjoaa myös mahdollisuudet tietokannan käyttöön, jolloin kaikkien käyttäjien osaamiset voidaan tallentaa samaan paikkaan. Tämä antaa opettajalle mahdollisuuden seurata oppilaidensa edistystä.

Adobe Flash CS3:lla toteutettu oppimisympäristö vaatii toimiakseen selaimen, johon on asennettu Flash Player 9 -versio. Flash Playerin käyttäjä voi ladata Adoben internetsivuilta, ja selaimeksi sopivat lähes kaikki markkinoilla olevat selaimet (Adobe 2007). Jos oppimisympäristössä on käytetty Papervision3D:n, away3d:n

tai Sandy 3D:n avulla tuotettua 3D:tä, ei erillistä plug-inia tarvita. Ainoastaan Flash Playerin oikea versio riittää.

Käytettäessä 3D:tä internetselaimessa on muistettava aina tiedonsiirron rajoitukset. Kun luodaan kolmiulotteinen tila, jossa voidaan liikkua vapaasti, on syytä käyttää mahdollisimman paljon samoja objekteja ja materiaaleja. Silloin pystytään hyödyntämään internetselaimen välimuistia ja tiedonlataus saadaan minimoitua. Jos käytetään kolmiulotteisuuden luomiseen bittikarttakuvia tai videoanimaatioita, käytetään silloin mahdollisimman suurta tiedoston pakkaamista tiedostokoon rajoittamiseksi.

2.4.2 Paikallinen versio

Käytettäessä opetusohjelmasta paikallista versiota, eli ohjelmaa joka asennetaan koneelle tai käytetään joltakin ulkopuoliselta tallennevälineeltä, ei tiedostokoolla ole suurta merkitystä. Koska mitään ohjelmaan ei ladata suoraan internetin välityksellä, on myös ohjelman päivittäminen rajoittuneempaa. Tällöin luodaan päivitystapa, jolla käyttäjä saa halutessaan ohjelmasta uusimmat versiot. Koska ohjelma on käyttäjällä itsellään jollakin tallenteella, pitää ohjelman jokainen versio suojata yksilöllisesti. Silloin estetään ohjelman jakaminen muiden kuin asiakkaan käyttöön.

3 TYÖKALUT

3.1 Adobe Flash CS3

3.1.1 Historia

Adobe Flashin historia juontaa juurensa vuoteen 1995, jolloin Jonathan Gay kehitti CelAnimator animaatio-ohjelman henkilökohtaisena projektinaan. Gay kuitenkin muutti ohjelman nimen jo kehitysvaiheessa FutureSplash Animatoriksi, koska hän halusi luoda eroa sarjakuva-animaatio-ohjelmiin. Samaisena vuonna Gay yritti myydä ohjelmansa juuri Adobelle, jossa kuitenkin kieltäydyttiin tarjouksesta. Sen aikaisessa versiossa ei Adoben mielestä ollut animaatioiden ohjaaminen Java-koodin avulla tarpeeksi tehokasta. Seuraavana vuonna Macromedia osti koko yrityksen, jonka Jonathan Gay oli perustanut, ja samassa paketissa myös FutureSplash Animator siirtyi Macromedian omistukseen. Ohjelman nimeksi vaihdettiin samana vuonna Flash, josta ehti ilmestyä kahdeksan versiota, ennen kuin Adobe Systems Inc. osti Macromedian vuonna 2005. (Waldron 2006.)

Flashin ohjelmointikieli tuli mukaan ohjelmaan versiossa 5 vuonna 2000, ja kieli sai nimekseen ActionScript (1). ActionScript on luokka-pohjainen ohjelmointikieli, jonka komennot perustuvat pitkälti JavaScriptiin. ActionScriptin avulla voidaan ohjailla aikajanaa ja antaa erilaisten luokkien avulla objekteille ominaisuuksia. ActionScriptista ilmestyi kehittyneempi versio vuonna 2003 version MX 2004 mukana. ActionScript 2.0 -versiossa luokka-ominaisuudet olivat jo kehittyneempiä, mutta koodi lepäsi yhä ensimmäisen ActionScriptin varassa. Nykyisessä versiossa, eli Adoben julkaisemassa CS3-versiossa, käytetään ActionScript 3.0:aa, joka eroaa selvästi aiemmista versioista. ActionScript 3.0:n kieli on aiempaa objekti-keskittyneempi, siinä käytetään enemmän tapahtumankäsittelijöitä ja kaiken kaikkiaan koodin käsittely on tehokkaampaa kuin edeltäjiensä. (Wikipedia 2007.)

3.1.2 Esittely

”Adobe Flash CS3 on kehittynein toteutusympäristö jolla voit luoda rikasta, interaktiivista ja digitaalista sisältöä internet- ja mobiilikäyttöympäristöihin.” (Adobe 2007). Kuvaus on osuva, mutta toki markkinoiva esittely Flashin käytöstä. Flash on suosittu käyttöympäristö erilaisten sähköisten sisältöjen tuottamiseen sekä mobiilisovellusten tuotantoon. Internetissä Flash on vallannut jalansijaa niin peleissä ja mainosmateriaaleissa kuin myös itse internetsivujen toteutuksessakin. Myös mobiilipuolella Flash on suosittu pelien toteuttamisessa.

Flashin grafiikka perustuu vektoripohjaisiin objekteihin, jotka bittikarttoihin verrattuna ovat soveliaampia pientä tiedostokokoa ja skaalausta vaativaan ympäristöön. Flash toimii hyvin myös monipuolisena esittely-ympäristönä, koska se mahdollistaa liitännäisyyden videokuvaan, bittikarttakuvaan ja ääneen. Näitä kaikkia voidaan ohjata Flashin oman ActionScript-kielen avulla. Kaikki Flashissa tapahtuu sisäisellä aikajanalla, jota pyöritetään kehys kehykseltä, aivan kuten elokuvien filmirullaa. Flashissa eri objektit voidaan asettaa ja luoda aikajanan eri tasoille ja tarvittaessa myös eri kehykseen. Eri kehykset ovat omanaikaisia kuviaan, jolloin asiat eivät esiinny yhtäaikaisesti. Objekteja on myös mahdollista asettaa toistensa sisään, jonka tuloksena voidaan toteuttaa mitä monimuotoisimpia animaatioita ja objekteja.

Flashin työpöytä koostuu x- ja y-koordinaatistosta, joten varsinaista objektien syvyyssijaintia sillä ei voida suoraan toteuttaa. Flash CS3 versio, kuten edeltäjätkään, ei sisällä 3D-moottoria jolla voitaisiin simuloida oikeaa 3D:tä. Kolmannen osapuolen 3D-moottorilla se on kuitenkin mahdollista. Saatavilla on tällä hetkellä ohjelmia kuten Papervision3D, Sandy 3D ja away3d.

3.2 Papervision3D 2.0 Alpha - 3D-moottori Adobe Flash CS3:een

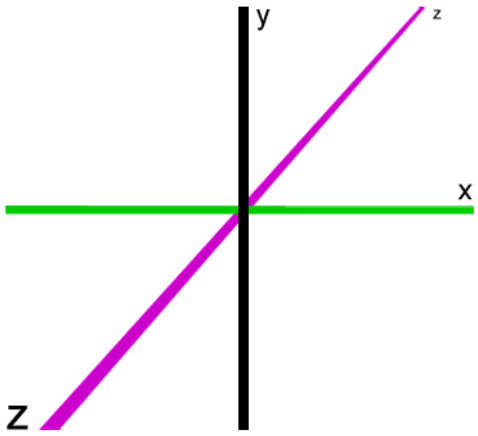
3.2.1 Esittely

Papervision3D 2.0 Alpha on avoimen lähdekoodin 3D-moottori Adobe Flash CS3:een. Papervision3D:stä julkaistiin heinäkuussa 2007 ensimmäinen julkinen beta-versio Papervision3D 1.5 Adobe Flash CS3:lle ja Macromedia Flash 8:lle. Ensimmäinen versio syntyi kuitenkin Adobe Flash 7:lle jo joulukuussa 2005. Papervision3D:n alullepanija- ja pääkehitysryhmä koostuu neljästä Flash-kehittäjästä: Carlos Ulloa, John Grden, Ralph Hauwert sekä Tim Knip. He lähtivät kehittämään ohjelmaa aluksi omiin projekteihinsa mutta huomasivat myös muiden tarpeet kolmiulotteisuuteen Flashissa. He päättivät luoda Papervision3D:n joka toimii MIT-lisenssillä, eli se on vapaa käytettäväksi sekä yksityiseen että kaupalliseen tarkoitukseen. Koska Papervision3D on avoin kaikille käyttäjille ja jokainen voi muokata alkuperäistä koodia omiin tarpeisiinsa sopiviksi, on ohjelma kehittynyt huimasti ensimmäisestä versiostaan. Siinä kolmiulotteisuus rajoittui pitkälti yksittäisten pintojen kääntelyyn akseliensa ympäri, mutta nykyinen versio 2.0 tarjoaa käytettäväksi erilaisia kameroita, valmiita perus 3D-objekteja, käyttäjän tekemien monimutkaisten objektien käsittelyä, tukea aktiivisille materiaaleille sekä paljon muuta.

3.2.2 Toimintaperiaate

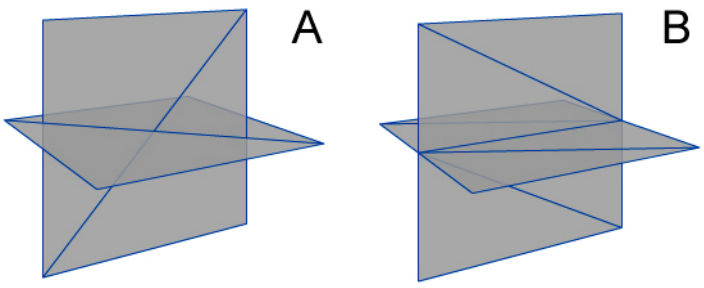
Papervision3D:n käyttöönotto tapahtuu lisäämällä sen luokat Adobe Flash CS3:ssa ActionScript-kielen luokkapolkuun. Aina kun luodaan uusi Flash-tiedosto, joka sisältää 3D:tä, tuodaan import-komennolla tarvittavat luokat tiedoston käytettäväksi. Papervision3D perustuu samaan periaatteeseen kuin Flashin ActionScript-kielikin, eli erilaisten luokkien hyväksikäyttämiseen. Luokat toimivat olioina, joilla on ja joille voidaan luoda ominaisuuksia riippuen sen tyypistä.

Papervision3D:n koordinaatisto koostuu x-, y- ja z-akseleista (KUVA 2). X-akseli on vaakasuuntainen, y-akseli on pystysuuntainen ja z-akseli on syvyyssuuntainen. Kaikki Papervision3D:llä luotava kolmiulotteisuus ladataan ActionScriptin avulla joko ulkopuolisista tiedostoista tai Papervision3D:n omasta kirjastosta joka sisältää alkeellisimmat 3D-objektit laatikosta palloon. Papervision3D:n tukema ulkopuolinen 3D-tiedostotyyppi on avoimen lähdekoodin Collada, joka perustuu XML-koodiin. Se sisältää tärkeimmät tiedot 3D-kappaleista, kuten materiaalit, koordinaatit, animaatiot, kamerat, valot ja tehosteet (Collada-blogi 2007). Papervision3D tukee myös ASE-tiedostotyyppiä, mutta Collada sisältää enemmän ominaisuuksia ja samalla myös enemmän mahdollisuuksia. (Lindquist 2008.)



KUVA 2. Papervision3D:n koordinaattiakselit (Lindquist 2008).

Papervision3D käyttää kolmiulotteisuuden piirtämiseen Painter's algorithm -laskentatekniikkaa, jossa kameraa kauimpana olevat pinnat piirretään näytölle ensimmäisinä ja lähimpänä olevat vasta lopuksi. Tämä tekniikka on valittu Papervision3D:lle koska se mahdollistaa Flashissa nopeimman mahdollisen 3D:n piirtämisen ja koska laskutekniikka on helppo (Lindquist 2008). Tekniikassa on omat heikkoutensa, jonka johdosta risteävien pintojen välillä voi esiintyä välkkymistä (KUVA 3). Ongelmat voidaan kuitenkin mallinnusvaiheessa ottaa huomioon ja vähentää välkkymisen esiintymistä. Painter's algorithm aiheuttaa myös sen, että pinnat joiden koordinaattipisteitä (x, y, z) jää kameran näkymän ulkopuolelle, jäävät kokonaan piirtämättä.



KUVA 3. Painter's algorithm. A pinnat ovat jakamatta ja B pinnat ovat jaettu (Iivonen 2008).

4 KOLMIULOTTEISUUS ADOBE FLASH CS3:SSA

4.1 Valmiit kuvat 3D:nä

4.1.1 Mallintaminen ja renderöinti

Tehtäessä kolmiulotteista vaikutelmaa Flashiin ilman 3D-moottoria täytyy sen toteuttamiseen käyttää bittikarttakuvia. Niiden hankaluutena on saada aidonoloista syvyysvaikutelmaa syntymään, eikä katselija pysty suoranaisesti vaikuttamaan valmiiseen kuvaan. Kuvia täytyy tehdä paljon että kolmiulotteisuuden illuusio saadaan luotua. Esimerkkinä on kamera-ajo, jossa kaikki kuvat ovat valmiiksi tehtyjä, mutta niitä peräkkäin esittämällä saadaan kuva elämään aivan kuin ympäristössä liikuttaisiin. Jos tehdään ympäristö, jota käyttäjä voi katsella haluamaansa suuntaan, täytyy siitä tehdä joko panoraamakuva tai ottaa kuvia katselupisteen pystyakselin ympäri sopivin välein. Jälkimmäisellä tavalla voidaan tehdä kolmiulotteinen kuvaus ympäristöstä, jonka käyttämiseen ei tarvita 3D-moottoria tai koodaustaitoa tekijältä. Riittää että kuvia otetaan sopivalla tasajaolla, esimerkiksi 35 kuvaa kymmenen asteen välein. Sen jälkeen selaamalla niitä Flashin aikajanalla eteen- ja taaksepäin voidaan simuloida kameran kääntämistä.

Kun kolmiulotteinen ympäristö tehdään mallinnusohjelmassa, pitää ottaa huomioon mistä, suunnasta kuvia otetaan. Jos otetaan yksittäinen kuva tai yhdensuuntainen kamera-ajo esimerkiksi talosta pihamaalla, riittää, että talo lähiympäristöineen ja taustoineen ovat mallinnettuna. Jos luodaan 360 asteinen kuvaus ympäristöstä, pitää jokainen kameran näköpiiriin osuva asia mallintaa. Renderöintivaiheessa täytyy huomioida myös kameran etäisyys lähellä sijaitseviin objekteihin, sillä jos kamera on liian lähellä esimerkiksi seinää, menetetään paljon asioita, joita kuvaan voisi mahduttaa. Toisaalta tällöin päästään pienemmällä mallinnustyöllä, joka taas nopeuttaa ympäristön tekoa.

Kuvia tehtäessä pitää ottaa huomioon niiden lopullinen käyttötarkoitus. Ympäristö jota katsellaan vaakatasossa eri suuntiin, vaatii kameran joka on asetettu sopivalle korkeudelle kuvitellun katselijan mukaan. Jos katselijana käytetään ympäristössä olevaa avataria, asetetaan kamera avatarin selän taakse sopivaan kulmaan vaaka-akselin suhteen. Jos riittää että saadaan katsaus ympäristöstä, jota käyttäjä voi käännellä, voidaan kuvat renderöidä vaakatasossa.

4.1.2 Oikeanlaiset tiedostotyypit

Valmiiden 3D-kuvien käyttämiseen Adobe Flash CS3:ssa on olemassa kaksi eri vaihtoehtoa: kuva- tai videomuoto. Kuvaformaatit joita Adobe Flash CS3 tukee, ovat kaikki yleisimmät tyypit: JPG, PNG, TIFF, TGA, BMP jne. Videokuvaa voidaan parhaiten hyödyntää, kun muokataan valmis video Adobe Flash Video Encoderissa, joka pystyy pakkaamaan videot tehokkaasti On2:n VP6-videonkäsittelyllä. Encoderista ulostuleva tiedosto on muotoa FLV (Adobe Flash Video), ja ohjelma pystyy käsittelemään ASF-, AVI-, DV-, MOV-, MP4-, MPEG-, MPG- ja WMV-tiedostoja. Videoiden kokoa saadaan Encoderin avulla pudotettua reilusti alkuperäisestä, ja parhaiten se sopii kamera-ajojen ja muiden Flashissa käytettävien videoiden pakkaamiseen.

Kun käytetään bittikarttakuvia tai videokuvaa ympäristön esittämiseen, saavat mallinnukset olla vektori- ja pintamääriltään hyvin tarkkoja. Kuvien ja videon käyttö ei vaadi suuria laskentatehoja niiden pyörittämiseen, mikä on loppukäyttäjän kannalta hyvä asia.

4.1.3 Vuorovaikutteisuuden toteuttaminen

Flashiin tuodut kuvat ja videot asetellaan Flashin aikajanalle. Periaatteena on että jokainen aikajanan kehys vastaa yhtä kuvaa, eli jos kamera-ajo kestää viisi sekuntia ja kuvataajuus on 25, kestää video aikajanalla 125 kehystä. Sama pätee mahdolliseen kameran kääntämiseen, johon on käytetty esimerkiksi 35 kuvaa. Tällaisessa tapauksessa aikajanaa ohjataan kuva kerrallaan joko eteen- tai taaksepäin riippuen siitä, kääntääkö käyttäjä kameraa oikealle vai vasemmalle.

Koska kuvilla toteutettu ympäristö koostuu staattisista kuva- tai videotiedostoista, ei niistä pystytä erittelemään yksittäisiä asioita. Irrallisia objekteja voidaan luoda hyödyntämällä Flashin tukemaa PNG-kuvatiedostoa tai FLV-videota, jossa on läpinäkyvyys käytössä. Samalla tavalla kuin muutkin kuva- ja videotiedostot, asetetaan kuvat ja video jokaiseen kehykseen erikseen ja annetaan niille tapahtumankäsittelijöitä. Toinen tapa luoda interaktiivisia objekteja on käyttää Flashin omia MovieClip- ja Button-objekteja. Näiden avulla voidaan luoda objekteja, joille voidaan asettaa myös tapahtumankäsittelijöitä. Niiden paikka ympäristössä voidaan määrittää joko dynaamisesti ActionScript-koodin avulla tai asettelemalla ne käsin haluttuun kohtaan.

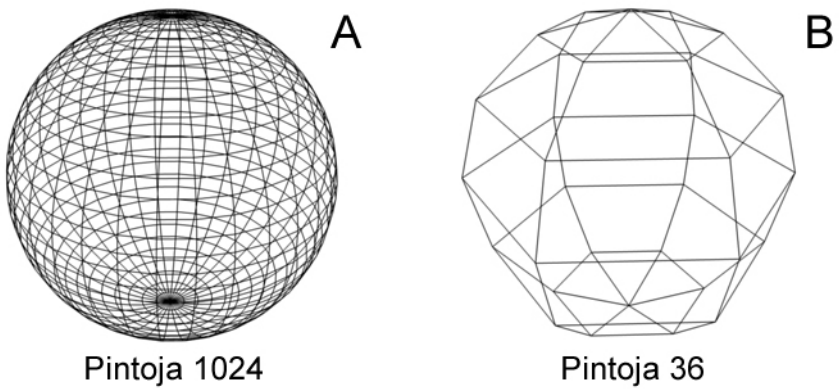
4.1.4 Kuvien päivittäminen

Kuvien päivittäminen tapahtuu tekemällä kuvat kokonaan uudestaan. Jos valaistusta muutetaan tai ympäristöön lisätään objekteja, muutokset tehdään ensin mallinnusohjelmassa, ja tämän jälkeen kuvat renderöidään uudestaan. Jos lisätään vain uusi objekti, riittää sellaisten kuvien renderöinti alueelta, jossa muutettava objekti näkyy. Jos käytetään videota, pitää koko video leikata tai tehdä uudestaan. Muutosten jälkeen kaikki kuvat ja videot pitää päivittää vielä Flashin kirjastoon, jolloin muutokset tulevat uuden julkaisun jälkeen käyttöön.

4.2 3D toteutettuna kolmannen osapuolen 3D-moottorilla

4.2.1 3D-objektien teko

Adobe Flash CS3:ssa voidaan hyödyntää oikeaa kolmiulotteisuutta, kun ohjelmaan liitetään jokin 3D-moottoria simuloiva plug-in. Tässä opinnäytetyössä on käytetty Papervision3D 2.0 Alpha plug-inia, joka mahdollistaa objektien tuomisen ja pyörittämisen Flashissa. Koska Papervision3D ei hyödynnä näytönohjainta, tapahtuu kaikkien vektoripisteiden ja pintojen paikkatietojen laskeminen tietokoneen prosessorin kustannuksella. Tästä syystä käytettävien 3D-mallien tulee olla mahdollisimman kevyitä, eli niin kutsuttuja low poly-malleja (KUVA 4). Jos mallissa on suuria tasaisia pintoja, jaetaan ne pienempiin osiin. Tällöin vältytään pintojen häviämiseltä, kun sen koordinaattipisteitä ei ole enää kameran näköpiirissä. Jokainen pinta mallissa tulee olla sellainen, että se ei suoraan leikkaa toista pintaa, vaan niiden leikkauskohdassa pinnat ovat jaettu.



KUVA 4. Low poly-mallinnus. A on alkuperäinen ja B on low poly-malli (Iivonen 2008).

Kaikki materiaalit luodaan mallinnusohjelmassa UVW-mappingilla, joko baked- tai tiled-materiaaleja käyttämällä. Baking-tekniikassa kaikki kappaleen materiaalit yhdistetään yhdeksi isoksi kuvaksi. Toinen vaihtoehto on käyttää tiled-tekniikkaa, jossa tekstuuria toistetaan toinen toisensa perään. Kuvaformaatti saa olla mikä tahansa Flashin tukema muoto, mutta usein käytetään JPG- ja PNG-tiedostoja niiden yleisyyden takia. Koska Flashissa ei ainakaan Papervision3D versio 2.0:n aikana pystytä suoraan hyödyntämään mallinnusohjelmassa luotavia heijastuksia ja kiiltoja, ei mallin tekstuureihin kannata näitä ominaisuuksia lisätä.

4.2.2 Tiedostomuodot

ASE (ASCII Scene Export)

ASE on alun perin 3ds Maxille kehitetty tiedostomuoto, jolla voidaan jakaa tekstitiedostona staattisia 3D-malleja ilman animaatioita. ASE-formaatti on tästä syystä yleistynyt, ja nykyään sitä voidaan lukea ja kirjoittaa useissa yleisimmässä mallinnusohjelmassa. ASE-tiedosto koostuu listoista, jotka sisältävät kaiken tarpeellisen tiedon yksittäisestä kappaleesta. Listoissa on eritelty objektit ja niiden vektoripisteet, materiaalit, valot ja kamerat. Kun ASE-tiedostoa käsitellään Flashissa, on siihen kuitenkin turha sisällyttää valoja tai kameraa, sillä ne eivät ole Flashin tukemia objekteja. (Unreal Wiki 2007.)

COLLADA (Collaborative Design Activity)

COLLADA-formaatti on kehittynein 3D-tiedostomuoto, jota Flashissa voidaan Papervision3D versio 2.0:llä käyttää. Se perustuu tekstimuotoiseen XML-kaavioon, joten kaikki tarvittava tieto on helposti luettavissa ja käyttäjän muokattavissa. COLLADA-tiedosto voi sisältää yksittäisen kappaleen tietojen lisäksi myös animaation vaatimat tiedot. Tiedostoon tallennettavia perustietoja ovat objektien koordinaattipisteet, materiaalilista, valot, kamerat, efektit ja fysiikkatiedot. Materiaalilista sisältää tekstuureiden lisäksi myös kaikki materiaalin ominaisuudet heijastuksista bump-karttoihin. Fysiikkaan sisältyvät mm. kappaleen massa, kitka ja jäykkyys. Koska Papervision3D 2.0 ei osaa käsitellä efektejä, erikoismateriaaleja, fysiikkaa, valoja eikä kameroita, ei niitä kannata COLLADA-tiedostoon lisätä sitä tallennettaessa. (Collada-blogi 2007.)

4.2.3 3D-objektien tuominen Flashiin

3D-objektit tuodaan Flashiin dynaamisesti, eli ne ladataan ActionScript-kielen avulla Flash Playeriin. Koska 3D-objektien tiedostot ladataan aina varsinaisen Flash-tiedoston ulkopuolelta, ei tekovaiheessa nähdä Adobe Flash CS3:ssa suoraan, miltä sisältö näyttää.

Vähimmäisvaatimukset, joita 3D:n näkyminen Flashissa tarvitsee, ovat oikeantyyppinen scene, näkymä, renderöintityökalu ja kamera. Nämä kaikki ovat Papervision3D:n luokkia, jotka ladataan ensimmäiseksi. Seuraavaksi scenen sisään ladataan 3D-objektit, jotka koodin avulla asetetaan oikeille paikoille. Jotta objektit saadaan näkymään oikein, ladataan niille materiaalit joko suoraan 3D-objektin tiedoston sisältävän tiedostopolun päästä tai Flashin omasta kirjastosta johon tekijä voi tuoda ja luoda omia objekteja. Kirjaston materiaalit voivat olla joko bittikarttakuvia tai Flashissa luotuja MovieClip-objekteja, jotka saavat sisältää kaikkea Flashin tukemaa materiaalia videosta Button-objekteihin. Scenen näkyville saaminen vaatii kameran päivittämistä, aina kun sillä tapahtuu jotain muutoksia. Papervision3D:n kirjastosta löytyy edellä mainittujen lisäksi paljon erilaisia luokkia, joiden avulla voidaan luoda esimerkiksi partikkeleita.

4.2.4 Vuorovaikutteisuuden toteuttaminen

Vuorovaikutus 3D-objektien ja käyttäjän välille saadaan luotua MovieClip-materiaalien ja interaktiivisten objektien avulla. MovieClip-materiaalit sisältävät kaikki samat ominaisuudet kuin perinteisesti käytetty MovieClip-objektikin. MovieClip-materiaalien sisälle voidaan lisätä painikkeita ja niille kuuntelijoita, sekä niiden sisällä olevia MovieClip-objekteja voidaan ohjata ulkopuolelta.

Koska 3D-objekteille voidaan luoda samanlaisia tapahtumankäsittelijöitä kuin perinteiselle MovieClip-objekille, voidaan niitäkin käännellä, liikutella ja skaalailla vapaasti. Voidaan esimerkiksi luoda näppäimistön oikea- ja vasen-painikkeille kuuntelijat, joita painamalla jokin kappale pyörii vasemmalle tai oikealle. Edellä mainitulla tavalla voidaan luoda esimerkiksi paikallaan kääntyvä kamera.

4.2.5 3D-objektien päivittäminen

3D-objektien päivittäminen tapahtuu mallinnusohjelmalla, jossa tarvittavat muutokset objekteihin tehdään. Mallinnusohjelmassa voidaan myös päivittää materiaalit, joiden päivittäminen onnistuu luonnollisesti myös kuvankäsittelyohjelmalla. Koska 3D-objektit ja kuvatiedostot ladataan käyttövaiheessa Flash Playeriin sen ulkopuolelta, ei Flash-tiedostoa tarvitse muokata lainkaan. Vanhat objektit ja kuvat ainoastaan korvataan uusilla versioilla.

5 CASE: ENGLANNIN KIELEN OPETUSOHJELMAN 3D-OPPIMISYMPÄRISTÖ

5.1 Lahden Teho-Opetus Oy

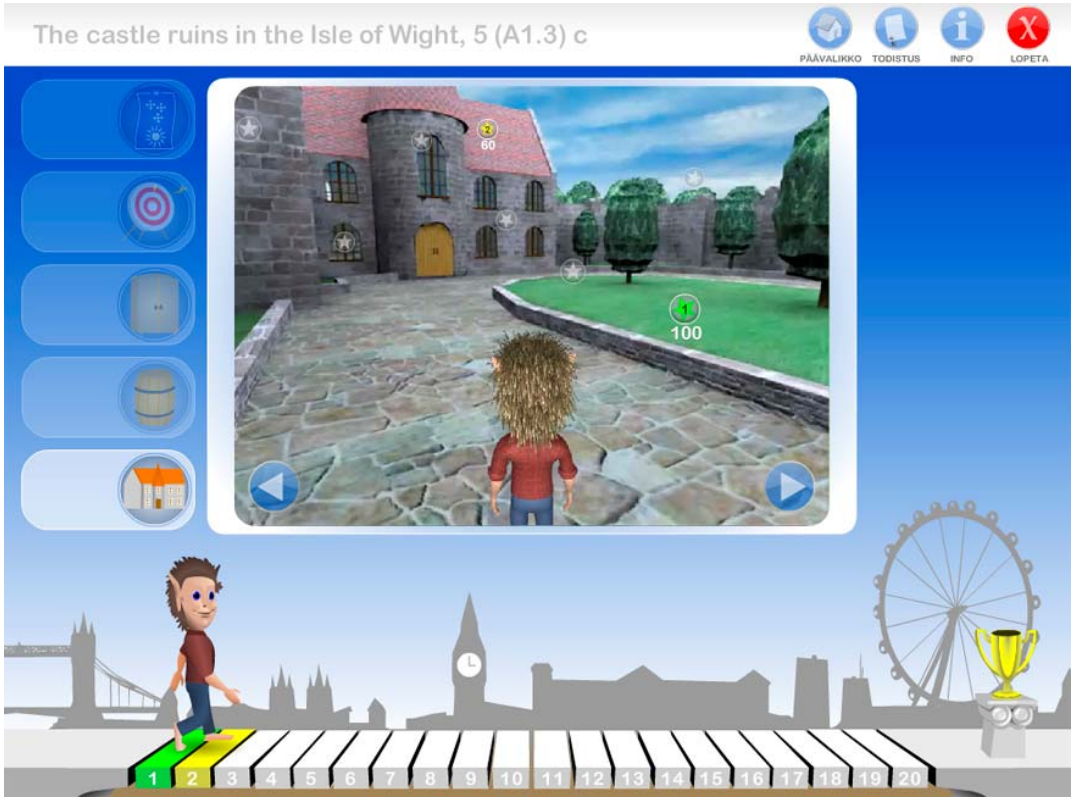
Lahden Teho-Opetus Oy:lla on perinteet opetusohjelmien tekemisessä jo vuodesta 1995 lähtien. Yritys on keskittynyt matemaattis-luonnontieteellisten ohjelmien tekemiseen, mutta uusi englannin kielen opetusohjelma Avenue laajentaa tätä kategoriaa kielten puolelle. Avenuen ensimmäinen versio on suunnattu peruskoulun 3-6 luokkien englannin kielen opetukseen. Opetusohjelma on tehty Flash-sovellukseksi, joka toimii sekä internetin kautta että paikallisena versiona Flash Playerilla. Flash on valittu toteutustekniikaksi siksi, että se on internetin yleisin multimedian esittämiseen käytetty toteutusympäristö.

Opetusympäristön esittämiseen käytetään interaktiivista 3D-ympäristöä, jota muissa vastaavissa ohjelmissa ei Suomessa ole ennen Avenueta käytetty. Avenuen kolmiulotteisuus on toteutettu käyttämällä bittikarttakuvia 3D:n esittämiseen ja tässä opinnäytetyössä vertaillaan saman asian toteuttamista Papervision3D:n avulla. Alkuperäinen opetusohjelma toteutettiin kuvilla, koska sen tekeminen aloitettiin keväällä 2007, ja Papervision3D julkaistiin vasta saman vuoden heinäkuussa. Jos opinnäytetyö antaa lupaavia tuloksia Papervision3D:n mahdollisuuksista, saattavat tulevaisuuden opetusohjelmat käyttää hyväksi sen tarjoamaa tekniikkaa.

5.2 Avenue – Englannin kielen opetusohjelma

5.2.1 Ohjelman idea

Avenue opetusohjelman ideana on, että pelaaja suorittaa englannin kielen harjoituksia ja etenee aina uuteen ympäristöön 20 suoritetun tehtävä jälkeen. Avenuessa on yhteensä neljä eri vaikeustasoa, joista jokaisessa on viisi ympäristöä liittyen johonkin Ison-Britannian oikeaan ympäristöön. Eri tasojen ympäristöinä toimivat Wightin saaren eläintarha, Shakespearen syntymäkoti, Wightin saaren linnarauniot ja Roomalainen kylpylä Bathin kaupungissa. Peruskoulun 3-6 luokalle toteutetussa ohjelmaversiossa käytetään kahta avatar-hahmoa, Peikko-tyttöä ja Peikko-poikaa, joista pelaaja valitsee toisen pelihahmokseen.



KUVA 5. Avenue opetusohjelman pelinäkymä (Iivonen 2008).

5.2.2 Ohjelman runko ja käyttöliittymä

Opetusohjelman sovelluksen ulkomitat ovat 800x640 pikseliä, minkä avulla varmistetaan, että ruudulle ei laiteta liikaa informaatiota. Mitat sallivat myös skaalauksen suurempaan kokoluokkaan, joka palvelee peruskoulun opetusohjelman kohderyhmän hahmottamis- ja keskittymiskykyä. Kuvataajuus on 20, jonka avulla animaatiot ja efektit saadaan toimimaan pehmeästi.

Ohjelman runko koostuu kolmesta eri osasta: Kirjautumissivu on ensimmäinen näkymä, jossa käyttäjiltä pyydetään henkilökohtaisia tunnuksia. Tunnusten käyttäminen mahdollistaa jokaisen käyttäjän yksilöllisten pisteiden ja tietojen tallentamisen ja hakemisen tietokannasta. Seuraava näkymä kirjautumisen jälkeen on päävalikko, josta pelaaja valitsee haluamansa moodin (peli, tehtävät tai pikakertaus), yhden neljästä tasovaihtoehdosta ja pelihahmon (tyttö tai poika). Viimeinen näkymä on suorituspuoli, jossa käyttäjä joko pelaa, tekee harjoituksia tai opiskelee sanoja. Tässä opinnäytetyössä käsitellään ainoastaan pelipuolta, jossa näkyvissä on 3D-oppimisympäristö ja käytössä olevan ympäristön suoritettujen tehtävien pisteet ja edistyminen.

Käyttöliittymä on tehty ohjelmassa mahdollisimman selkeäksi. Yläreunassa sijaitsee painikealue, jossa on kaikki tärkeät painikkeet jokaisessa ohjelman osassa. Painikkeet ovat samankaltaisia, minkä vuoksi käyttäjä osaa tunnistaa ne helposti ohjelman eri osissa. Käyttöliittymän informaation määrä on jätetty mahdollisimman pieneksi, ja siinä on otettu huomioon peruskoulun käyttäjien tietokonetaidot. Tällä perusteella on myös ohjelmassa navigoiminen tehty muutaman hiirenpainalluksen mittaiseksi. Jokaisen ohjelman käyttökerran jälkeen käyttäjän tiedot ja valinnat tallentuvat henkilökohtaiseen tietokantaan. Ne haetaan sieltä seuraavalla kirjautumiskerralla, ja käyttäjä pääsee yhdellä hiirenpainalluksella jatkamaan siitä, mihin oli viimeksi lopettanut.

5.2.3 Ympäristöjen lataaminen ohjelmaan

Opetusohjelmassa tarvitaan yhteensä 20 ympäristöä, minkä takia jokainen ympäristö on syytä tehdä omaan tiedostoonsa. Yhden bittikarttakuvilla toteutetun ympäristön koko lisäosineen on noin 1 Mt ja yhden Papervision3D:n avulla toteutetun ympäristön koko noin puolet siitä. Jotta varsinaisen opetusohjelman (swf-tiedosto) sisälle voidaan ladata yksittäinen 3D-ympäristö (swf-tiedosto), tehdään sen sisälle UILoader-lataaja. UILoader mahdollistaa ulkopuolisten swf-tiedostojen lataamisen Loader-luokan avulla, ja se käsittelee ladattua tiedostoa child-objektinaan. Ladatun tiedoston sisälle voidaan lähettää päätasolta muuttujia ja kutsua ladatun tiedoston sisältämiä funktioita. Ladatusta swf-tiedostosta voidaan vastaavasti LocalConnection-yhteyden avulla kutsua hierarkiassa ylempänä olevan opetusohjelman funktioita ja lähettää sinne ympäristön omia muuttujia.

3D-ympäristön UILoader-lataaja asetetaan opetusohjelmassa ActionScript-koodilla dynaamisesti sopivaan koordinaattipaikkaan (x, y) pois nollapisteestä. Tämän jälkeen ladattavan tiedoston leveys- ja korkeusmitat määritellään pienemmiksi kuin 800x640 pikseliä. Avatarille luodaan samanlainen UILoader, mutta sen paikkatiedoiksi ja mitoiksi määritellään arvot, joilla se asettuu samalle kohdalle ympäristön kanssa. Ympäristön ja avatarin ulkomitoiksi asetetaan pienemmät arvot kuin ne oikeasti ovat, koska silloin niiden ympärille jää tilaa painikkeille ja pistetaulukoille. Koko opetusohjelman ulkomittoja voidaan tällöin myös skaalata suuremmaksi kuin 800x640 ilman, että kuvien laatu heikkenee.

```
//Ympäristön lataaja asetaan tasolle 1
var ymparistoLaturi:UILoader = new UILoader();
ymparistoLaturi.x=110;
ymparistoLaturi.y=63;
```

```
ymparistoLaturi.setSize(580,332);
addChildAt(ymparistoLaturi,1);

//Hahmon lataaja asetaan tasolle 2 (ympäristön
etupuolelle)
var hahmoLaturi:UILoader = new UILoader();
hahmoLaturi.x=175;
hahmoLaturi.y=64;
hahmoLaturi.setSize(450,332)
addChildAt(hahmoLaturi,2);
```

5.3 3D-oppimisympäristön toteuttaminen bittikarttakuvilla

5.3.1 Ympäristön luominen mallinnusohjelmassa ja renderöinti

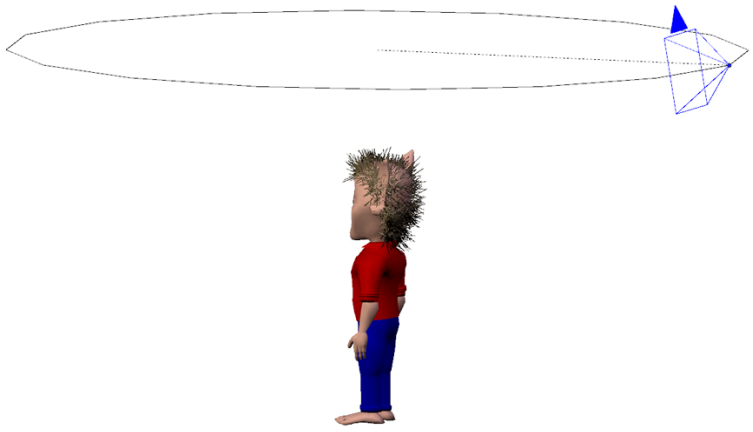
Mallinnettaessa ympäristöä mallinnusohjelmalla, tässä tapauksessa Blenderillä, ei vektoreiden ja polygonien määrällä ole merkitystä. Bittikarttakuvat, joita lopuksi renderöidään, eivät kasvata tiedostokokoaan pintojen määrän lisääntyessä, ja ne vaikuttavat ainoastaan koneen vaatimaan laskentatehoon. Koska lopputuloksena on tarkoitus saada mahdollisimman hyvää mallinnusjälkeä, ei pintojen ja vektoripisteiden määrään kannata kiinnittää erityistä huomiota.

Ympäristön suunnittelu lähtee siitä, mitä kameran läpi näkyy. Kaikki asiat, kuten seinät tai puut, täytyy asemoida ympäristöön kameran sijoittelun mukaan. Jos jokin asia jää toisen taakse, eikä sitä näy kamerasta, on se järkevää jättää mallintamatta kokonaan. Mallinnettaessa pitää ottaa huomioon myös, että missään sellaisessa paikassa ei näy tyhjää, jossa oikeassa elämässä pitäisi jotain näkyä. Esimerkiksi maanpinta ei saa loppua ennen horisonttia kesken. Kun ympäristöön lisätään vielä Flashissa 20 tehtäväpainiketta, pitää niidenkin sijoittelu ottaa huomioon. Tässä tapauksessa painikkeet tulevat tyhjiin kohtiin seinille ja ilmaan, joten ympäristöön on tarkoituksella jätetty tyhjää tilaa.

Kamera, jonka linssin kautta kuvat renderöidään, on syytä sijoittaa mahdollisimman keskelle tilaa ja etäälle suurista vertikaalisesti peittävistä pinnoista. Tällöin ympäristöstä saadaan kameran näköpiiriin mahdollisimman paljon avointa syvyyssuuntaista tilaa ja saavutetaan kolmiulotteisuuden tunne. Myös vertikaaliset suorat jotka ovat kamerasta kauempana, eivät vääristy silloin perspektiivin vaikutuksesta aivan yhtä paljoa, kuin jos ne olisivat lähellä kameraa. Kameran asettamisessa pitää ottaa huomioon myös, mihin ympäristöstä seuraavaksi mennään.

Kun tarkoitus on päästä seuraavaan ympäristöön esimerkiksi oven kautta, täytyy varmistaa, että sen edessä ei ole peittäviä esteitä silloin, kun ovi on kameran näköpiirissä. Sisätilojen kohdalla edellä mainitut asiat ovat erityisen haastavia, koska kaikki asiat ovat lähellä kameraa. Opetusohjelman sisäkuvia renderöidessä joutui useaan otteeseen tarkistamaan kameran sijainnin, jotta seinät eivät tuntuisi kaatuvan päälle ja että kulkuväylät olisivat viisaasti näkyvillä.

Kamera, jota englannin kielen ohjelmassa käytetään, tarkastelee ympäristöä avatarin selän takaa pään yläpuolelta. Kamera asetetaan ympäristöön siten, että ensin luodaan Circle-objekti jonka keskipiste simuloi avatarin paikkaa ja sen säde 200 cm edustaa etäisyyttä jonka kamera on avatarin takana. Koska avatarin korkeus on 150 cm, on sopiva korkeus Circle-objektille 200 cm. Tämän jälkeen kamera linkitetään Circle-objektiin katselusuunta sen keskustaa kohti ja kameraa kallistetaan horisontaalisessa suunnassa -10° siten, että katselulinja menee avatarin oletetun päälaen yli (KUVA 6). Kameran linssiksi sopii parhaiten mallinnusohjelma Blenderin linssiasetus 20, joka vastaa noin 77° katselukulmaa. Jotta kaikki on valmista renderöintiä varten, tehdään Circle-objektille vielä animaatio, jossa se kääntyy tasaisin 15° välein täyden ympyrän kamera mukanaan.



KUVA 6. Kamera asetetaan oletetun hahmon suhteen (Iivonen 2008).

Kun kameran kääntäminen renderöidään kuviksi, on tiedostomuotona mahdollista käyttää joko video- tai kuvaformaattia. Englannin kielen ohjelmassa ympäristön kuvien tekemiseen on käytetty AVI Jpeg -videoformaattia, jolla ei menetetä laatua merkittävästi. Videon renderöinnissä jätetään sen viimeinen kuva tekemättä, koska alku- ja loppukuva ovat samat. Yhteensä kuvia tulee siis 23. Videotiedoston

kuvataajuudeksi asetetaan sama kuin Flashin, eli 20. Videon kooksi on taas valittu 550x400 pikseliä, jota käyttämällä käsittelemättömän AVI-videotiedoston kooksi tulee noin viisi megatavua. Videotiedoston käyttäminen on perusteltua, sillä 23 kuvatiedoston asettelu Flashin aikajanalle on huomattavasti työläämpää kuin videon, eikä valmiin ympäristön tiedostokoossa tule eroa.

KUVA 7. Renderöidyn ympäristön ensimmäinen, toinen ja viimeinen kuva (Iivonen 2008).

5.3.2 Ympäristön kokoaminen Flashiin ja ohjauspainikkeet

Flash-tiedoston valmistelu

Ensimmäiseksi luodaan uusi tyhjä Flash-tiedosto, jonka kehystaajuudeksi asetetaan 20 ja tiedoston mitoiksi 950x554 pikseliä. Videotiedoston tuomista varten tehdään tyhjä MovieClip-objekti, jonka paikaksi asetetaan x=100 ja y=0 ja instanssinimeksi jokin kuvaava nimi, tässä tapauksessa TILA. Tämän jälkeen tehdään maski-taso aikajanalle ja sille luodaan 750x554 pikselin kokoinen musta suorakaide, joka peittää tulevan ympäristön kokoisen alueen. Lopuksi MovieClip-objekti avataan, ja sen ensimmäinen kehys aktivoidaan videotiedoston tuontia varten valmiiksi.

Import Video

Valmiin videotiedoston tuonti Flashiin tapahtuu Import Video -toiminnon kautta. Videotiedostoa tuotaessa pitää ottaa huomioon, että se upotetaan Flashin kirjastoon, jolla vältetään loppukäytössä ylimääräinen lataaminen ulkopuolelta. Importvaiheessa valitaan vielä tuontimäärityksiksi sen asettaminen suoraan stagelle ja soittaminen aikajanalla siten, että jokainen videotiedoston kuva vastaa yhtä kehystä aikajanalla.

Seuraava askel videotiedoston tuontivaiheessa on sen muuntaminen Flash Video -muotoon Flash Video Encoderilla. Video Encoderissa tiedosto pakataan On2 VP6 -pakkausformaatilla, joka on kehittyneempi ja paremmin laadun säilyttävä kuin Encoderin toinen pakkausformaatti Sorenson Spark. Pakkausvaiheessa on otettava huomioon kuvalaadun säilyminen hyvänä, ja tiedostokoon pysyminen mahdollisimman alhaisena. Sopivan pakkauslaadun löytäminen on hankalaa, mutta hyvä pakkauslaatu suhteessa tiedostokokoon on tässä tapauksessa 275 kt/s, jolla lopullisen pakatun videotiedoston kooksi tulee noin 600 kilotavua. Pakkauksessa on otettava huomioon että videon kuvataajuus on sama kuin lähteellä, ja että jokainen kuva on avainkehys, eli jokainen kuva pakataan samalla 275 kt/s laadulla. Lopuksi vielä varmistetaan, ettei ääniraita tule tiedostoon mukaan ja tehdään kuvan rajaus ja skaalaus siten, että videosta syntyy 750x554 pikselin kokoinen. Toimenpiteiden jälkeen video on valmis tuotavaksi kirjastoon.

Asettelu aikajanalle ja ohjausnapit

Stagella oleva videotiedosto asetetaan TILA-MovieClipin nollakohtaan. Sen jälkeen palataan päätasolle jonka ensimmäiseen kehykseen lisätään koodi, jolla pysäytetään video haluttuun kohtaan. Lisäksi tehdään kaksi Button-objektia (oikeaNappi ja vasenNappi), joille lisätään tapahtumankäsittelijät. Näiden avulla käännetään ympäristöä vasemmalle ja oikealle.

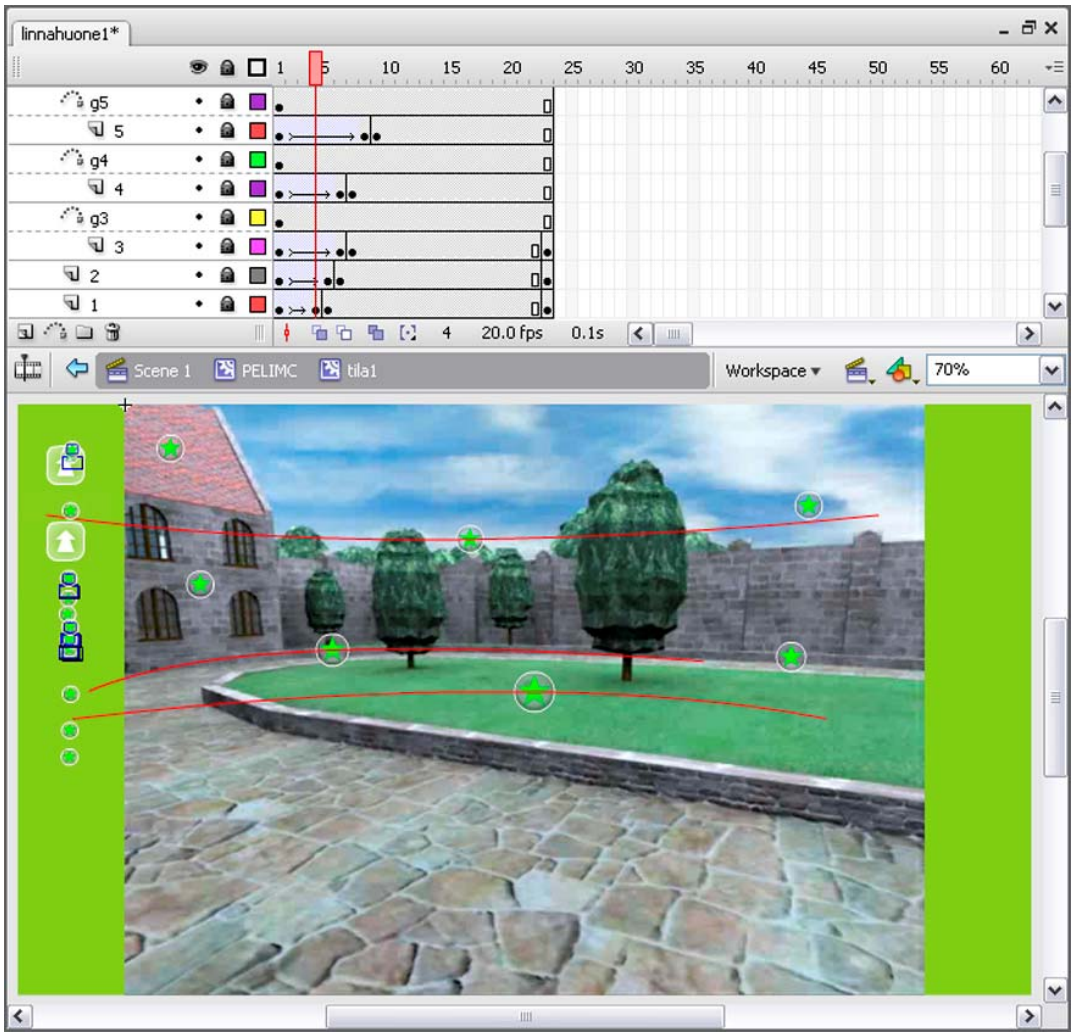
```
//Pysäytetään video
TILA.gotoAndStop(1);
//Tehdään painikkeille kuuntelijat
oikeaNappi.addEventListener(MouseEvent.MOUSE_DOWN,
oikealle);
vasenNappi.addEventListener(MouseEvent.MOUSE_DOWN,
vasemmalle);


function oikealle(e:MouseEvent):void{
    if (TILA.currentFrame==23)
        TILA.gotoAndStop(1);
    else
        TILA.nextFrame();
}
function vasemmalle(e:MouseEvent):void{
    if (TILA.currentFrame==1)
        TILA.gotoAndStop(23);
```

```
        else
            TILA.prevFrame();
    }
```

5.3.3 Ympäristön painikkeet

Kaikki painikkeet tehdään MovieClip-objekteista, jotka asetellaan ympäristöön TILA-MovieClipin sisään. Koska painikkeet ovat MovieClip-objekteja, voidaan niille määrittää kuuntelijoita ja niiden läpinäkyvyyttä ja kokoa voidaan säätää sekä dynaamisesti että käsin. Painikkeet reagoivat hiiren kursorin päälle ja poisvientiin suurentumalla ja pienentymällä MOUSE_OVER- ja MOUSE_OUT-kuuntelijoiden avulla. Jokaiselle painikkeelle luodaan myös oma MOUSE_DOWN-kuuntelija sen painamista varten.

Jokainen painike asetetaan omalle tasolleen aikajanalle, jolloin niiden ohjailu ympäristön kääntämisen mukaan on helpompaa. Painikkeille voidaan luoda liikuttamisen helpottamiseksi yksilöllisiä Motion Tweenejä ja Guide-tasoja Ne ohjailevat painikkeiden liikkumista alkupisteestä loppupisteeseen ilman, että paikkaa tarvitsee määrittää jokaisessa kehyksessä erikseen. Painikkeet sijoitellaan ympäristön kuvien päälle tasaisesti sellaisiin kohtiin, joissa on niille jo mallinnusvaiheessa varattuja paikkoja (KUVA 8). Painikkeet skaalataan käsin sopivan kokoisiksi ympäristöön suhteutettuna kuvissa olevaa perspektiiviä hyväksi käyttäen.



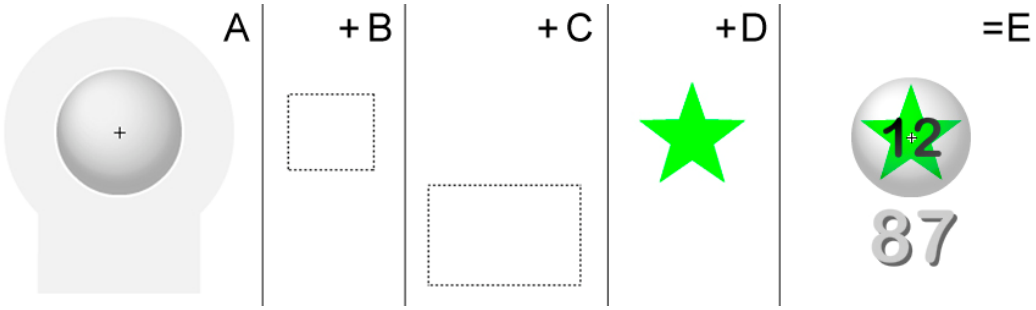
KUVA 8. Painikkeiden asettelu ympäristöön (Iivonen 2008).

Jotta painikkeet toimivat oikein, täytyy jokainen niistä lisätä jo ensimmäiseen kehykseen. Kun ActionScriptin kautta kutsutaan jotakin objektia, jota ei ole olemassa, syntyy virhetilanne, joka keskeyttää koko Flash-sovelluksen pyörittämisen. Koska jokainen painike ei kuitenkaan ole ympäristössä näkyvissä ensimmäisestä kuvasta asti, laitetaan tällaiset painikkeet joko ympäristön vasemmalle tai oikealle puolelle sivuun (KUVA 8). Tämän takia alussa lisättiin päätasolle maski-taso, joka näyttää ainoastaan ympäristön alueen eikä mitään sen ulkopuolelta. Näin tehdään jokaisessa kehyksessä jokaiselle painikkeelle, jonka ei kuulu olla näkyvissä.

Tehtäväpainikkeet

Tehtäväpainikkeiden avulla avataan harjoituksia, joiden pisteet ja tehtävätyypit sidotaan jokaiseen painikkeeseen yksilöidysti. Ympäristössä on yhteensä 20 teh-

täväpainiketta, jotka kaikki ovat yhden kirjastossa olevan MovieClip-objektin kopioita. Painike koostuu tekstikentästä, johon painikkeesta suoritetun tehtävän pistemäärä laitetaan; tekstikentästä, johon tehtävänumero laitetaan; läpinäkyvästä painikealueesta sekä MovieClip-objektista, jonka väriä muutetaan portaallisesti vertaamalla saavutettua pistemäärää harjoituksessa saatavilla olevaan sataan kokonaispisteeseen (KUVA 9). Värit paremmuusjärjestyksessä ovat vihreä, keltainen ja punainen, sekä valkoinen kuvastaa suorittamatonta tehtävää. Kun tehtäväpainiketta painetaan, haetaan olemassa olevista taulukoista kyseisen painikkeen kohdalta harjoituksen käynnistämiseen tarvittavat tiedot ja lähetetään päätasolle tieto harjoituksen avaamisesta.



KUVA 9. Laukaisupainikkeen rakenne (A=painikealue, B=tehtävänumero, C=pistekenttä, D=väriä vaihtava MovieClip-objekti, E=valmis painike) (Iivonen 2008).

Ovipainikkeet

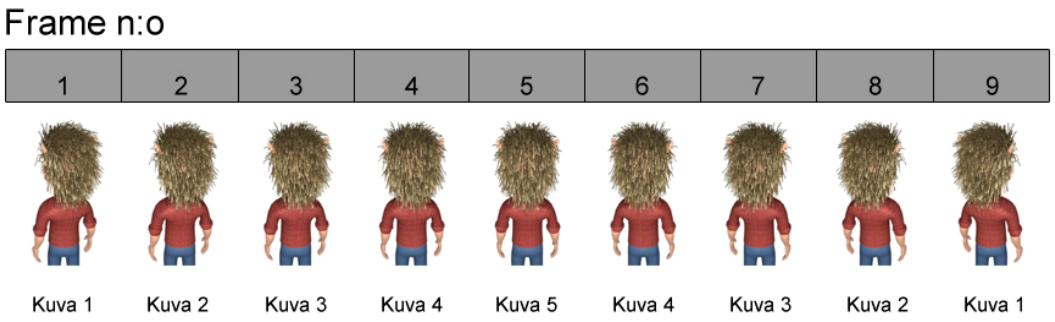
Ovipainikkeet ovat painikkeita, joita painamalla vaihdetaan ympäristöä. Ne eivät sisällä mitään tietoa vaan toimivat ainoastaan painikkeina. Niille luodaan samanlaiset kuuntelijat kuin tehtäväpainikkeille, mutta ovipainikkeista vaihdetaan ympäristöä joko seuraavaan tai edelliseen, riippuen siitä kumpaa painiketta on painettu. Kun ovipainiketta on painettu, pelihahmo kävelee painikkeen suuntaan ja samalla kutsutaan opetusohjelman päätasolla olevaa kuuntelijaa, joka lataa uuden ympäristön vanhan tilalle.

5.3.4 Avatar

Avatar on käyttäjän ohjaama hahmo, jolla simuloidaan kulkijaa oppimisympäristössä. Se tehdään mallinnusohjelmassa luomalla 150 cm pituinen hahmo joka koostuu luurangosta ja siihen linkitetyistä ruumiinosista. Luurankoa liikuttamalla luodaan avatarin liikkeet. Liikkeet saadaan bittikarttakuviksi renderöimällä jokai-

nen asento omaksi kuvakseen. Tässä englannin kielen ohjelmassa on avatarin vartalon kääntämiset ja kävely renderöity PNG-kuviksi, joista kävely on myöhemmin muunnettu Flashin avulla läpinäkyvyyden sisältämäksi MOV-videotiedostoksi. Kamera, jolla hahmon liikkeet renderöidään, asetetaan samoille asetuksille ja sijainnille, kuin ympäristöäkin renderöidessä (KUVA 6). Kamera on siis 200 cm korkeudella ja 200 cm etäisyydellä avatarin selästä, ja sen horisontaalinen kulma on -10° ja linssi 20.

Avatarin kokoaminen Flashissa aloitetaan luomalla uusi tyhjä tiedosto, jonka kehystaajuudeksi asetetaan 20 ja mitoiksi 750x554 pikseliä. Tämän jälkeen luodaan tyhjä MovieClip-objekti, jolle annetaan kuvaava instanssinimi, tässä tapauksessa HAHMO. Objektin sisälle tuodaan hahmon kääntämiseen tarvittavat viisi kuvaa, jotka sijoitellaan jokainen omaan kehykseen aikajanalle (KUVA 10). Neljä ensimmäistä kuvaa käännetään peilikuviksi viimeisiin kehyksiin, jolloin säästetään lataamismäärässä neljän kuvan verran. Viimeisen kääntökuvan jälkeiseen kehykseen numero 10 lisätään vielä kävelyvideo.



KUVA 10. Hahmon kuvien asettelu aikajanalle (Iivonen 2008).

Jotta avatar saadaan kääntymään hiiren kursorin liikuttamisen mukaan, lisätään Flash-tiedoston päätasolle kääntymiseen tarvittavat kuuntelijat. Hahmon sisältämä MovieClip-objekti vaihtaa kehystä kursorin paikkatiedon mukaan, jolloin hahmo kääntyy oikealle tai vasemmalle. Lopuksi luodaan vielä funktio, jolla saadaan hahmo kävelemään kun siihen tulee tarvetta.

```
stage.addEventListener(MouseEvent.MOUSE_MOVE, hahmoSeuraa);
function hahmoSeuraa(e:MouseEvent):void{
        if (hahmo.currentFrame<10){
                if (e.stageX >= 175 && e.stageX <= 625){
```

```
                    if(e.stageX < 626)
                        HAHMO.gotoAndStop(9);
                    if(e.stageX < 575)
                        HAHMO.gotoAndStop(8);
                    if(e.stageX < 525)
                        HAHMO.gotoAndStop(7);
                    if(e.stageX < 475)
                        HAHMO.gotoAndStop(6);
                    if(e.stageX < 425)
                        HAHMO.gotoAndStop(5);
                    if(e.stageX < 375)
                        HAHMO.gotoAndStop(4);
                    if(e.stageX < 325)
                        HAHMO.gotoAndStop(3);
                    if(e.stageX < 275)
                        HAHMO.gotoAndStop(2);
                    if(e.stageX < 225)
                        HAHMO.gotoAndStop(1);
                }
            }
        }

        function hahmoKavele():void{
            HAHMO.gotoAndStop(10);
        }
```

5.4 3D-oppimisympäristö kolmannen osapuolen 3D-moottorilla

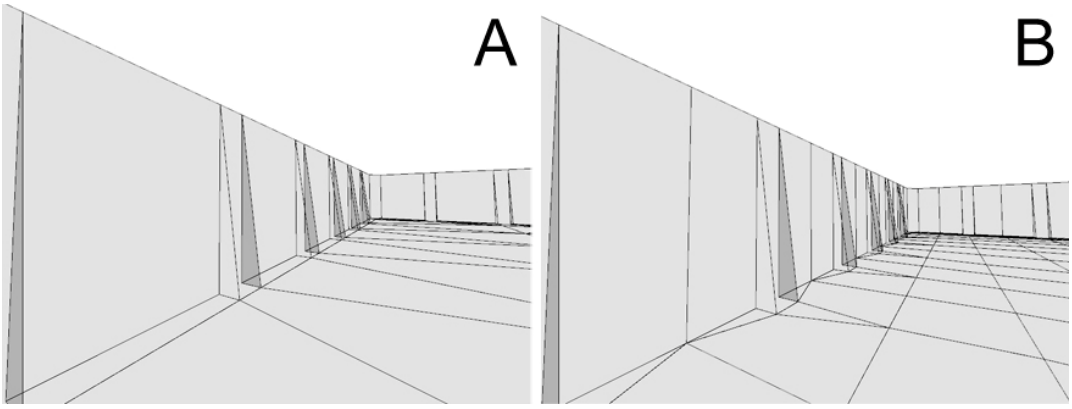
5.4.1 Objektien ja tekstuurien luominen mallinnusohjelmassa

Mallinnus

Kun 3D-ympäristö toteutetaan Flashissa Papervision3D:n avulla, on kaikki monimutkaiset 3D-mallit toteutettava erillisessä mallinnusohjelmassa. Tässä opinnäytetyössä on käytetty Blender-mallinnusohjelmaa, jolla 3D-objektit tallennetaan Papervision3D:n tukemaan Collada-formaattiin. Ympäristön kolmiopintojen ko-

konaismäärä pitäisi saada pysymään alle 1500, sillä sen ylittävä osa hidastaa normaalitehoisen kotitietokoneen toimintaa jo huomattavasti. Paras tapa on jakaa pinnat ympäristöön tasaisesti, jolloin Flashissa käytettäessä ei yhdellä kerralla ole kameran näköpiirissä kuin osa niistä.

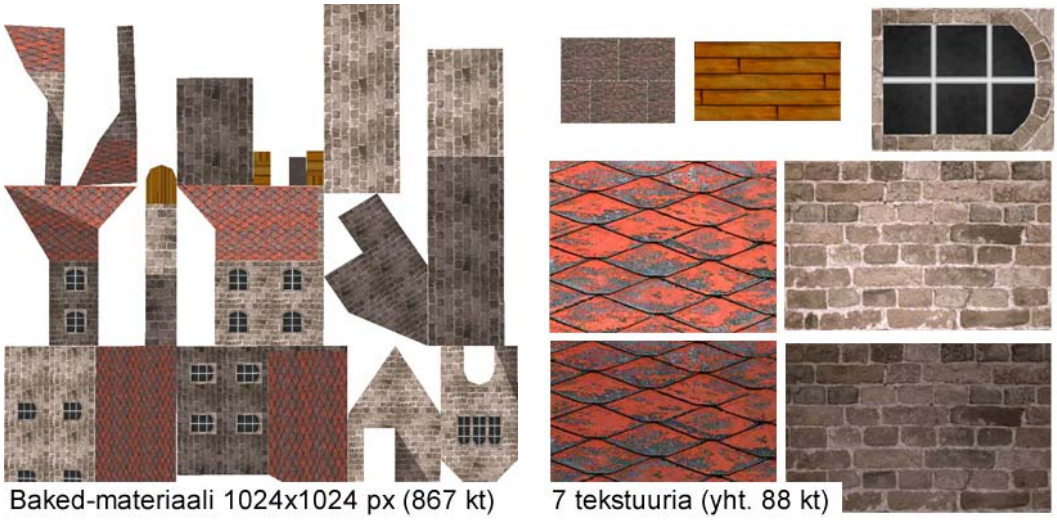
Mallinnettaessa ympäristöä mallinnusohjelmalla, tehdään kaikista yksittäisistä kappaleista, kuten tuoleista, taloista ja ovista, erillisiä skaalaamattomia low poly -objekteja. Ne ovat skaalaamattomia siksi, että animoitaessa kappaletta Flashissa saattaa sen koko joissakin tapauksissa vaihdella, jos kappale ei ole alkuperäisessä koossaan. Jos skaalaamiseen on tarvetta, pystyy sen suorittamaan Flashissa ActionScriptin avulla. Koska kappaleet ovat low poly -malleja, on niitä kevyempi käsitellä Flashissa ja tarpeen tullen niitä voidaan monistaa useampaan kertaan. Monistamisella saadaan vähennettyä Papervision3D:n vaatimia laskentatehoja, koska jo olemassa olevan kappaleen kopion pintojen laskemiseen ei kulu yhtä paljon tehoja kuin uuden kappaleen laskemiseen. Jokaisen risteävän pinnan leikkauskohtaan tehdään pintojen jako Papervision3D:n käyttämän Painter’s algorithm -piirtotekniikan takia. Jokaisesta pinnasta tehdään myös kolmen vektoripisteen kautta kulkeva kolmiopinta, koska neljän vektoripisteen kautta kulkeva pinta jää Flashissa piirtämättä. Ympäristön kolmiopinnat eivät saa olla liian suuria, sillä Papervision3D:n 3D-moottori ei laske eikä piirrä pintoja, joista yksikin vektoripiste jää kameran näkymän ulkopuolelle. Tätä ilmiötä vähennetään jakamalla pystysuorat pinnat vertikaalisesti pienempiin osiin ja vaakatasossa olevat pinnat pienempiin pintoihin (KUVA 11).



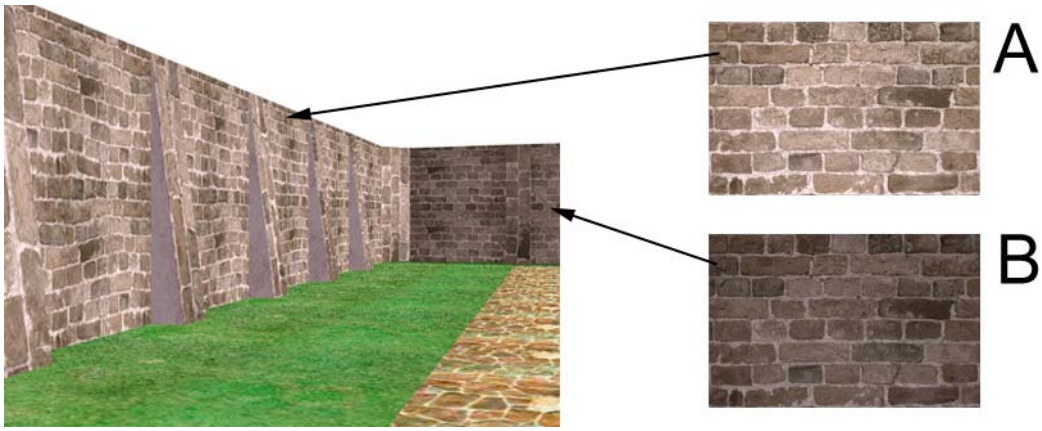
KUVA 11. Kuvassa A pinnat ovat jakamatta ja kuvassa B ne ovat jaettu (Iivonen 2008).

Tekstuurit

Ympäristön mallinnetuille objekteille lisätään sopivat tekstuurit. Tekstuurien tekemiseen käytetään UVW-mappingia, jolloin jokaiselle pinnalle saadaan luotua sopivan kokoinen ja näköinen materiaali. Jos mallissa on monimutkaisia materiaaleja tai valon luomia varjoja, on silloin helppo tapa käyttää mallinnusohjelman baking-ominaisuutta. Sen lopputuloksena on kuitenkin suuria kuvatiedostoja, joiden lataaminen ja pyörittäminen on raskasta. Ongelman ratkaisu on käyttää tiled-materiaaleja, joissa samaa materiaalia toistetaan uudestaan ja uudestaan. Kuvatiedostojen kokonaistiedostokoossa päästään tällä tavoin murto-osaan baked-materiaalin koosta ja samoja materiaaleja voidaan hyödyntää myös muille kappaleille, jolloin lataamista vähennetään vielä entisestään (KUVA 12). Kun tiled-materiaaleilla luodaan yksinkertaisia varjoja, tehdään samasta materiaalista kuvankäsittelyohjelmalla uusi tummempi versio ja kartoitetaan se varjokohtaan (KUVA 13).



KUVA 12. Baked-materiaali vs. tiled-materiaalit (Iivonen 2008).



KUVA 13. Materiaalien kartoittaminen valoisalla ja varjoisalle alueelle (Iivonen 2008).

Collada-tiedoston tallennus

Kun ympäristön malli on valmis, tallennetaan siitä Collada-tiedosto. Tallentamiseen on kaksi mahdollisuutta: Jokainen objekti tallennetaan erikseen, tai kaikki objektit sisällytetään yhteen Collada-tiedostoon. Erikseen tallentaessa on yksittäisiä kappaleita helpompi animoida Flashissa, mutta tällöin jokaisen tiedoston lataaminen määritellään erikseen. Ison ympäristön tapauksessa on yhden Collada-tiedoston tallennus parempi valinta, koska kaikki kiinteät objektit sisällytetään yhteen pakettiin, ja ne voidaan myös yhdessä paketissa ladata Flashin sisälle. Erikseen tehdään Collada-tiedostot tosin sellaisille objekteille, joita on tarkoitus monistaa Flashissa ja joille on tarkoitus tehdä animaatioita. Collada-tiedostoa tehdessä sisällytetään sen XML-koodiin tekstuurien UVW-kartat. Kaikki käytettävät tekstuurit tallennetaan oikeaan tiedostokansioon joko PNG- tai JPG-formaatissa ja tarkistetaan, että Colladan sisällä on sama polku.

5.4.2 Ympäristön kokoaminen Flashissa

Flash-tiedoston valmistelu

Aluksi luodaan uusi tyhjä Flash-tiedosto, joka kooltaan on 800x600 pikseliä, ja jonka kehystaajuus on 20. Tämän jälkeen Flashin ActionScript 3.0 asetuksiin ja Flash-tiedoston ActionScript 3.0 julkaisuasetuksiin lisätään Papervision3D:n luokkien tiedostopolku. Kun tämä on tehty, ovat kaikki Papervision3D:n luokat ActionScript-kielen käytettävissä.

3D-objektien lataaminen Flashin sisälle

Ensimmäiseksi tuodaan ActionScriptilla import-komentojen avulla tarvittavat luokkapolut tiedoston käyttöön. Tarvittavat luokat ovat Papervision3D:n tapahtumankäsittelijät, Scene3D, renderöintityökalu, Viewport3D, kamera, Collada-luokka sekä materiaalit.

```
import org.papervision3d.events.*;
import org.papervision3d.scenes.Scene3D;
import org.papervision3d.render.BasicRenderEngine;
import org.papervision3d.view.Viewport3D;
import org.papervision3d.cameras.FreeCamera3D;
import org.papervision3d.objects.parsers.DAE;
import org.papervision3d.materials.*;
import org.papervision3d.materials.utils.MaterialsList;
```

Tämän jälkeen luodaan ja määritellään jokainen edellä mainituista luokista, jotta niitä voidaan käyttää kun ympäristöä pyöritetään. Tärkeimmät perusasiat luodaan ensimmäiseksi, joita ilman mitään ei saada näkyviin.

```
function perusobjektit():void{
    // Luodaan näkymä (leveys, korkeus, skaalautuva,
    interaktiivinen)
    viewport = new Viewport3D(800,600,false,true);
    addChild(viewport);

    // Luodaan renderöintityökalu
    renderer = new BasicRenderEngine();

    // Luodaan scene (animoitu)
    scene3d = new Scene3D(true);

    // Luodaan kamera (zoom, focus) ja asetetaan
    paikalleen
    kamera = new FreeCamera3D(30,30);
    kamera.x=0;
    kamera.z=-100;
    kamera.y=20;
}
```

Seuraavaksi määritellään materiaalit ja lisätään ne materiaalilistalle. Koska materiaalit toistuvat oletuksena melko epäselvinä, laitetaan jokaiselle käytettävälle materiaalille pehmennys sahalaitaisuuden pienentämiseksi. Sellaisille materiaaleille, jotka sisältävät jonkin selvän tekstuurin, on syytä laittaa precise-ominaisuus käyttöön. Se skaalaa materiaalin perspektiivin mukaisesti kappaleen pinnoille. Ominaisuus kuitenkin kasvattaa tietokoneen tehonkäyttöä, joten jos materiaali on yksivärinen tai muuten epäselvä, ei ominaisuutta kannata käyttää. Tarvittaessa materiaaleille voidaan asettaa myös kaksipuolisuus. Jos käytettävä materiaali on tyyppiä tiled, määritetään silloin tiled-ominaisuus käyttöön. Ilman tätä ominaisuutta ei tiled-materiaali toistuisi oikein. Materiaalilista lisätään Collada-tiedostolle sen latausvaiheessa.

```
function lisaaMateriaalit ():void{
    //Luodaan jokaiselle materiaalille oma
    materiaaliobjekti
    var kiveysMat = new BitmapFileMaterial("kiveys.jpg");
    …

    //Pehmennetään materiaaleja, tehdään niistä toistuvia
    ja perspektiiviin skaalautuvia
    kiveysMat.smooth=true;
    kiveysMat.tiled=true;
    kiveysMat.precise=true;
    …
    //Lisätään materiaalit materiaalilistaan
    ymparistoMatList.addMaterial(kiveysMat,"kiveys_jpg");
    …
}
```

Lopuksi ladataan itse ympäristö ja luodaan tapahtumankäsittelijä, joka päivittää ympäristöä niin monta kertaa sekunnissa kuin kehystaajuudeksi on asetettu, tässä tapauksessa 20 kertaa. Kehystaajuuden avulla pystytään säätelemään ympäristön pyörittämisen raskautta. Tässä opinnäytetyössä käytetyn ympäristön pyörittäminen vaatii kehystaajuudella 20 neljäsosan vähemmän laskentatehoa kuin taajuudella 25.

```
//Ladataan ympäristö (tiedosto, materiaalilista) ja
asetetaan paikalleen
function lisaaObjektit():void{
    ymparisto=new DAE();
```

```
        ymparisto.load("ymparisto.dae", ymparistoMatList);
        ymparisto.x=0;
        ymparisto.z=0;
        ymparisto.y=0;
        scene3d.addChild(ymparisto);
    }

    //Lisätään ympäristön päivittäminen
    addEventListener(Event.ENTER_FRAME, paivita);

    //Renderöinti (scene, kamera, näkymä)
    function paivita(e:Event):void{
           renderer.renderScene(scene3d, kamera, viewport);
    }
```

5.4.3 Liikkuminen ympäristössä

Kameran ohjaaminen

Kameran ohjaaminen tapahtuu Papervision3D:n moveForward()-luokan ja rotationY toiminnon avulla. Luokka moveForward() liikuttaa kameraa positiivisilla arvoilla sen katselusuunnan mukaisesti eteenpäin ja negatiivisilla arvoilla katselusuunnasta poispäin. RotationY mahdollistaa kameran oikealle kääntämisen lisäämällä arvoja ja vasemmalle vähentämällä arvoja. Luokkaa ja toimintoa kutsutaan tekemällä näppäimistön nuolipainikkeille tapahtumankäsittelijä, joka tarkastelee, mitä painiketta on painettu. Lopuksi luodaan vielä kitka-muuttuja, joka vaimentaa ja pehmentää kameran liikkeen nollaan, kun näppäimen painaminen loppuu.

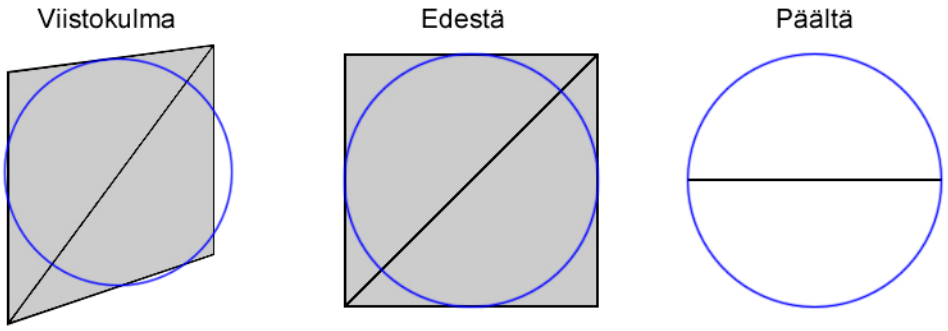
```
    //Luodaan näppäimistön painikkeille kuuntelijat
    stage.addEventListener(KeyboardEvent.KEY_DOWN,
    painikeDown);
    stage.addEventListener(KeyboardEvent.KEY_UP, painikeUp);

    //Mitä nuolipainiketta painetaan
    function painikeDown(e:KeyboardEvent):void{
        switch(e.keyCode){
            case Keyboard.UP:
                eteen = true;
                taakse = false;
```

```
            break;
            case Keyboard.DOWN:
                taakse = true;
                eteen = false;
            break;
            case Keyboard.LEFT:
                vasemmalle = true;
                oikealle = false;
            break;
            case Keyboard.RIGHT:
                oikealle = true;
                vasemmalle = false;
            break;
        }
    }
    //Mitä nuolipainiketta oli painettu
    function painikeUp(e:KeyboardEvent):void{
        switch(e.keyCode){
            case Keyboard.UP:
                eteen = false;
            break;
            case Keyboard.DOWN:
                taakse = false;
            break;
            case Keyboard.LEFT:
                vasemmalle = false;
            break;
            case Keyboard.RIGHT:
                oikealle = false;
            break;
        }
    }

    //Seuraavat koodit lisätään paivita-funktion alkuun
    kameran liikuttamiseksi
        if(eteen)
            liiku+=2;
```

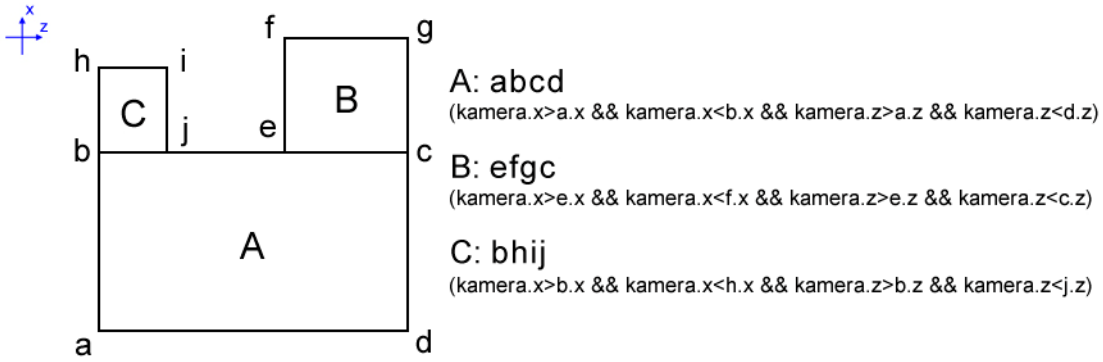
```
if(taakse)
    liiku+=-2;
if(vasemmalle)
    kaanny+=-1.5;
if(oikealle)
    kaanny+=1.5;

//Kitka pienentää liikkeen nollaan kun painaminen on loppunut
kitka=3;
liiku+=(0-liiku) / kitka;
kaanny+=(0-kaanny) / kitka;
kamera.moveForward(liiku);
kamera.rotationY+=kaanny;
```

Kameran törmäystarkastelu

Kameran törmäystarkastelun toteuttamiseen on kaksi eri vaihtoehtoa: Voidaan käyttää hitTestObject()-luokkaa, jolla tarkastellaan kameran ja toisen objektin välistä yhteentörmäystä (KUVA 14). Toinen tapa on käyttää koordinaatiston pisteitä kuljettavan alueen määrittämiseen. HitTestObject()-luokan heikkous on sen epätarkkuudessa. Luokan avulla tarkastellaan ainoastaan kappaleen koordinaatiston (x, y, z) ääripisteitä, joiden mukaan objektin sisälle luodaan pisteiden kautta kulkeva ympyrä. Tällä menetelmällä kamera törmää kappaleseen sellaisissakin kohdissa, joissa ei todellisuudessa mitään estettä ole. Parempi tapa on käyttää itse määritettyjä koordinaattipisteitä. Tämä tapa toimii kuljettaessa ympäristössä vaakatasossa, eli x- ja z-koordinaatistossa. Koordinaatit määritellään itse jakamalla kuljettava alue suorakaiteen muotoisiin alueisiin ja vertailemalla kameran x- ja z-koordinaattipisteitä saataviin arvoihin (KUVA 15). Kun kamera törmää rajaarvoon, pysäytetään sen liike ja kameran uusiksi koordinaattipisteiksi asetetaan viimeksi alueen sisällä olleet arvot.



KUVA 14. Plane-objektin törmäysalue ympyrän sisällä (Iivonen 2008).



KUVA 15. Suorakaiteen muotoiset alueet ja niiden koordinaattipisteet (Iivonen 2008).

5.4.4 Tehtäväpainikkeet

Tehtäväpainikkeet toteutetaan luomalla 20 Plane-objektia, jotka asetellaan ympäristöön halutuille paikoille. Jokainen objekti tulee samalla korkeudelle y-akslille, mutta niiden yksilölliset x- ja z-koordinaatit asetetaan kahteen taulukkoon. Koska Plane-objektit luodaan dynaamisesti, on niiden asettelu helpointa toteuttaa hakemalla paikkatiedot taulukoista ja suorittamalla for-luuppi.

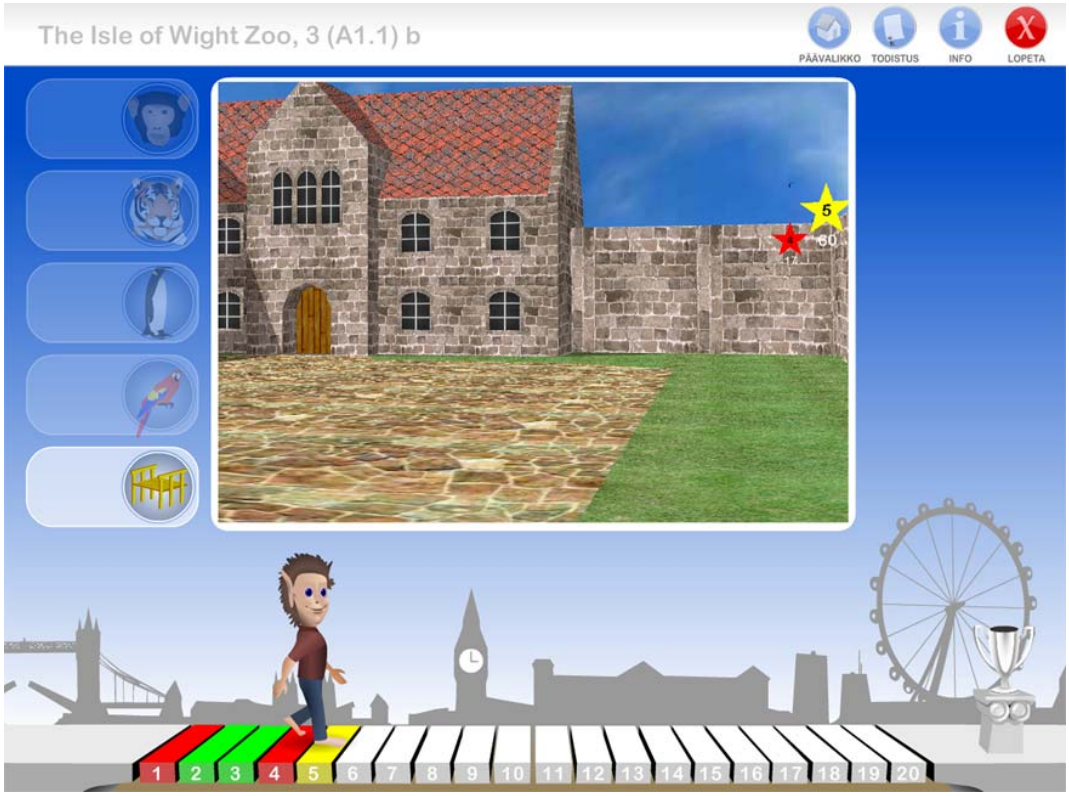
Plane-objektien materiaaliksi asetetaan Flashin kirjastosta MovieClip-objekti (KUVA 9), joka sisältää painikealueen, numerokentän, tehtäväkentän ja väriä muuttavan MovieClip-objektin. Neliön muotoiselle Plane-objektillle asetettavan materiaalin täytyy vääristymisen estämiseksi olla neliön muotoinen, joten MovieClip-materiaalin sisällä olevasta painikealueesta tehdään neliö. Koska Papervision3D:n Material-luokka käsittelee materiaaleja nollakohdan mukaan, joka on vasemmassa yläkulmassa, täytyy myös MovieClip-materiaalin nollakohdan olla siinä paikassa.

Tehtäväpainikkeille lisätään kuuntelijat jotka reagoivat kursorin päälle liikuttamiseen ja hiiren painallukseen. Koska objektit ovat Papervision3D:n omasta kirjastosta, on niille määritettävä sen mukaiset kuuntelijat. Papervision3D käyttää kappaleiden tapahtumankäsittelijänä InteractiveScene3DEvent-luokkaa. OBJECT_OVER kuuntelee kursorin päälle vientiä ja kursorin poisvientiä OBJECT_OUT. Objektin painamista taas kuuntelee OBJECT_PRESS. Kun Plane-objektia painetaan, haetaan olemassa olevista taulukoista kyseisen painikkeen kohdalta harjoituksen käynnistämiseen tarvittavat tiedot ja lähetetään päätasolle tieto harjoituksen avaamisesta.

Viimeinen asia joka painikkeille pitää tehdä, on kääntää Plane-objektit aina kameraa kohti kun sitä liikutetaan. Jos näin ei tehtäisi, painike vääristyisi kameraan nähden, eikä Plane-objektin toiselta puolelta näkyisi mitään. Objektit saadaan kääntymään kameraa kohti Papervision3D:n lookAt()-funktion avulla. Plane-objektin kohdalla ongelmaksi muodostuu sen väärän puolen katsominen kameraan, jolloin mitään ei näy materiaalin yksipuolisuudesta johtuen. Ongelma korjataan kääntämällä objektia vielä 180° pystyakselin suhteen.

```
//Painikkeiden koordinaatit taulukoissa
var taulukkoX = new Array(undefined,165,165,…,-220);
var taulukkoZ = new Array(undefined,-5,-60,…,-165);
var painikeTaulukko =new Array(undefined);

//Luodaan 20 painiketta
for(i=1;i<21;i++){
    //Luodaan painikkeelle oma materiaali
    var painikeMat = new
    MovieAssetMaterial("nappiTex",true,true,true);
    painikeMat.interactive=true;
    painikeMat.smooth=true;

    //Uusi painike joka laitetaan painiketaulukkoon
    painikeTaulukko = new Plane(painikeMat,10,10);
    painikeTaulukko.name = "laukaisupaikka" + i;
    painikeTaulukko.push(laukaisupaikka);
    scene3d.addChild(painikeTaulukko [i]);

    //Asetetaan painikkeille yksilölliset koordinaatit
    taulukoista
```

```
        painikeTaulukko [i].x=taulukkoX[i];
        painikeTaulukko [i].z=taulukkoZ[i];
        painikeTaulukko [i].y=30;

        //Luodaan painikkeelle kuuntelijat
        painikeTaulukko[i].addEventListener(InteractiveScene3
        DEvent.OBJECT_OVER, paalla);
        painikeTaulukko[i].addEventListener(InteractiveScene3
        DEvent.OBJECT_OUT, paalta);
        painikeTaulukko[i].addEventListener(InteractiveScene3
        DEvent.OBJECT_PRESS, hiiriPainettu);
    }
    //Avataan harjoitus
    function hiiriPainettu(e:InteractiveScene3DEvent):void{
        avataanHarjoitus(painikeTaulukko.indexOf(e.currentTar
        get));
    }

    //Värin muunnos kun hiiren kursori viedään päälle ja pois
    function paalla(e:InteractiveScene3DEvent):void{
        e.currentTarget.material.movie.gotoAndStop(2);
    }

    function paalta(e:InteractiveScene3DEvent):void{
        e.currentTarget.material.movie.gotoAndStop(1);
    }

    //Seuraavat koodit lisätään paivita-funktion loppuun
    for(i=1;i<21;i++){
        //Käännetään painike katsomaan kameraan
        painikeTaulukko [i].lookAt(kamera);

        //Käännetään sitä 180 astetta, jotta painikkeen oikea
        puoli katsoo kameraan
        painikeTaulukko [i].yaw(180);
    }
```



KUVA 16. 3D-oppimisympäristö Papervision3D:llä toteutettuna (Iivonen 2008).

5.5 Case-tarkastelu

Kuvilla toteutettu 3D on helppo tapa toteuttaa 3D-ympäristön tekeminen Flashiin. Se vaatii tekijältään ainoastaan Flashin ja jonkin mallinnusohjelman perusosaamisen. Kuvat 3D:nä ajavat asiansa kohderyhmälle peruskoulun 3-6 luokille, mutta koska ne eivät salli käyttäjän liikkua ympäristössä vapaasti, on toteutustapa vaativimmille ja kokeneimmille käyttäjille melko yksinkertainen. Kuvien käytössä on myös muita ongelmia. Koska jokainen katselusuunta vaatii oman kuvansa, kasvaa tiedostokoko melko suureksi. Ratkaisu tähän ongelmaan on kuvien ja videon pakkaaminen, mutta se tiputtaa kuvallista laatua nopeasti. Viimeinen kuvien käytön ongelma on Flashista johtuva. Flash lataa tietokoneen välimuistiin tiedostoja sitä mukaan kuin Flashin sisälle niitä ladataan, mutta se ei kuitenkaan osaa muistia välittömästi tyhjentämään. Flashissa on kyllä olemassa niin kutsuttu roskien kerääjä, joka tyhjentää välimuistista käytöstä poistettuja asioita, mutta sen tehokkuus on kovin vajavainen ja toimii ainoastaan satunnaisesti. Kuvilla toteutettu ympäristö varaakin välimuistia pienen pyörittelyn jälkeen helposti sata megatavua.

Papervision3D:n käyttäminen 3D-ympäristön toteuttamisessa antaa aidon tunteen ympäristössä liikkumisessa, onhan se oikeaa 3D:tä. Ympäristöt toimivat hyvin low poly-objekteilla, joiden kolmiopintamäärä jää yhteensä alle 1500:n. Ympäristöt itsessään on helppo toteuttaa, jos tekijällä on perustaidot mallinnusohjelman käytöstä, mutta Flashin puolella vaaditaan kuitenkin jo enemmän koodaustaitoa, jos halutaan hyödyntää kaikkia Papervision3D:n tarjoamia mahdollisuuksia. Papervision3D:n avulla toteutetun ympäristön tiedostokoko jää melko alhaiseksi erityisesti jos käytetään tiled-materiaaleja, joita toistetaan useilla kappaleilla. Suurin Papervision3D:n rajoite on kuitenkin sen laskentaan käyttämä teho tietokoneen prosessorilta. Näytönohjaimen hyödyntämättä jättäminen on selvästi suurin este Papervision3D:n tehokkaammalle käytölle, mutta ilmaiseksi työkaluksi se on hyvin toimiva.

Verrattaessa näitä kahta menetelmää 3D-oppimisympäristön toteuttamisessa saavutetaan Papervision3D:llä huomattavasti mielenkiintoisempi ja nykyaikaisempi lopputulos. Siinä missä valmiit kuvat ovat yksipuolisia ja staattisia, antaa oikea 3D mahdollisuuden animoida ympäristöä ja mahdollisuuden liikkua sen sisällä. Flash toteutusalustana tarjoaa hyvät puitteet monipuolisen median esittämiseen, ja reaaliaikainen 3D on siihen erinomainen lisä.

6 YHTEENVETO

Työn päätarkoitus oli löytää vastaus kysymykseen, miten toteuttaa 3D-oppimisympäristö Adobe Flash CS3:lla ja mitä vaatimuksia oppimisympäristöllä on. Teoriaosuudessa etsittiin vastauksia siihen mitä tällaisessa ympäristössä kolmiulotteisuus vaatii ja mitä itse oppimisympäristö vaatii. Case-osuudessa taas tutkittiin, kuinka 3D-oppimisympäristö toteutetaan Flashissa kuvien ja 3D-moottorin avulla ja mitä eroja niiden välille syntyy.

Pääasioiksi teoriapuolella nousivat kolmiulotteisuuden kohdalla kuvallisen laadun merkitys ja tiedostokoon pysyminen kohtuullisena. Itse oppimisympäristön puolella taas helppokäyttöisyys, yhdenmukaisuus ja yleensäkin selkeys olivat ensiarvoisia asioita. Erityisesti käyttöliittymän ja käytettävyyden kohdilla tulee jo tekovaiheessa painottaa niiden kyseenalaistamista ja kriittistä tarkastelua, jotta tekijä ei tee ohjelmaa itselleen vaan kohderyhmälle.

Case-osuudessa saavutettiin mielenkiintoisia tuloksia. Kuvien avulla toteutetun oppimisympäristön mahdollisiksi kompastuskiviksi muodostuvat sen yksinkertaisuus ja yksipuolisuus sekä myös Flashin muistin rajoittuminen, kun esitetään paljon kuva- ja videotiedostoja yhden istunnon aikana. Hyvä puoli oli toteutustavan yksinkertaisuus. Myös 3D-moottori Papervision3D 2.0 Alphan kohdalla saavutettiin hyviä mutta myös huonoja tuloksia. Hyvänä puolena esiin nousi ohjelman tarjoama mahdollisuus kolmiulotteisen interaktiivisen oppimisympäristön toteuttamiseen, jossa käyttäjä pääsee vapaasti kulkemaan. Ison miinuksen Papervision3D saa sen prosessorilta syömästä tehosta, joka rajoittaa merkittävästi oppimisympäristön kokoa. Vertailtaessa kahta toteutustapaa toisiinsa on 3D-moottorilla toteutettu oppimisympäristö kuitenkin mielenkiintoisempi, nykyaikaisempi ja tarjoaa enemmän mahdollisuuksia interaktiivisuuden toteuttamiseen. Sen suomia mahdollisuuksia kannattaa kyllä 3D-ympäristön toteuttamisessa hyödyntää, jos kooksi riittää pieni, alle 1500 kolmiopinnan ympäristö.

Opinnäytetyö oli kokonaisuudessaan onnistunut. Sen avulla selvisi hyvin pitkälti mitä mahdollisuuksia Papervision3D:llä on kolmiulotteisen oppimisympäristön toteuttamiseen ja teoriaosuus tarjosi vastauksia käytettävyyteen liittyviin kysymyksiin. Opinnäytetyön aikana hankalaksi asiaksi nousi lähdemateriaalien puute, koska asia on uusi. Erityisesti toteutuspuolelle ei ollut ennalta olemassa tieteellistä materiaalia, johon olisi voinut turvautua. Sen vuoksi lähes kaikki siihen liittyvä

materiaali on keskustelupalstoilta ja yksittäisiltä blogeilta sekä tietysti myös oman tekemisen ja kokeilun tulosta.

Tätä opinnäytetyötä on mahdollista hyödyntää myös muiden kuin oppimisympäristöjen toteuttamisen tutkimiseen. Työn voi yleistää sellaisiin 3D-ympäristöihin, joissa käytetään Flash-alustaa. Jatkotutkimukselle olisi tämän työn jälkeen aihetta Papervision3D 2.0 Alphan jälkeisten versioiden kohdalla.

LÄHTEET

Painetut lähteet

Hearn, D. & Baker, M. P. 1994. Computer Graphics. A Paramount Communications Company

Kalimo, A. (toim.) 1995. Graafisen käyttöliittymän suunnittelu. Tietotekniikan kehittämiskeskus, TIEKE Ry

Lankoski, P. (toim.) 2001. Ihminen, paikka ja aika. Tampere: Yliopistopaino

Lehtovirta, P. & Nuutinen, K. 2000. 3D-Sisältötuotannon peruskirja, Jyväskylä: Docendo Finland Oy

Matikainen, J. 2002. Vuorovaikutus verkossa. 2. painos. Helsinki: Palmenia-kustannus

Soila, S. & Teuvola, T. (toim.) 2003. Tieto- ja viestintätekniikan opetuskäytön väyliä ja karikoita. Hämeenlinna: Hämeen ammattikorkeakoulu

Sähköiset lähteet

Adobe 2007. What is Flash CS3 Professional?. Julkaisussa Flash [online]. Adobe Systems Incorporated. [viitattu 28.10.2007]. Saatavissa: http://www.adobe.com/products/flash/

Collada-blogi 2007. Collada FAQ [verkkojulkaisu]. Collada mediawiki [viitattu 11.11.2007]. Saatavissa Collada-blogi: http://www.collada.org/mediawiki/index.php/COLLADA_FAQ

Lindquist, J. 2008. Papervision3D Part 1 – Foundation and 3D Object Basics [verkkojulkaisu]. O'REILLY InsideRIA. [viitattu 28.3.2008]. Saatavissa: http://www.insideria.com/2008/02/papervision3d-part-1-foundatio.html

Metsämäki, M. 1998. Graafisen käyttöliittymän suunnittelu [verkkojulkaisu]. Oy Edita Ab. Helsinki. [viitattu 14.10.2007]. Saatavissa Vaasan yliopiston Editan tiedostokirjat: http://www.tritonia.fi/vanha/ov/edita/kayttoliittyma.pdf

Papervision3D-blogi 2007. Papervision3D [verkkojulkaisu]. Papervision3D mediawiki [viitattu 28.10.2007]. Saatavissa Papervision3D-blogi: http://wiki.papervision3d.org/index.php?title=Main_Page

Unreal Wiki 2007. ASE File Format [verkkojulkaisu]. Unreal Wiki [viitattu 23.1.2008]. Saatavissa Unreal Wiki hakemistosta: http://www.unrealwiki.com/wiki/ASE_File_Format

Waldron, R. 2006. How it began. Julkaisussa The Flash history [verkkolehti]. Flashmagazine. [viitattu 28.10.2007]. Saatavissa: http://www.flashmagazine.com/news/detail/the_flash_history/

Wikipedia 2007. ActionScript [verkkojulkaisu]. Wikipedia [viitattu 28.10.2007]. Saatavissa Wikipedian englanninkielinen hakemisto: http://en.wikipedia.org/wiki/ActionScript

Kuvalähteet

KUVA 1. Preecen hahmolait. Iivonen, J. 2008

KUVA 2. Papervision3D:n koordinaattiakselit. Lindquist, J. 2008. Saatavissa: http://www.insideria.com/2008/02/papervision3d-part-1-foundatio.html

KUVA 3. Painter's algorithm. A pinnat ovat jakamatta ja B pinnat ovat jaettu. Iivonen, J. 2008

KUVA 4. Low poly-mallinnus. A on alkuperäinen ja B on low poly-malli. Iivonen, J. 2008

KUVA 5. Avenue opetusohjelman pelinäkymä. Iivonen, J. 2008

KUVA 6. Kamera asetetaan oletetun hahmon suhteen. Iivonen, J. 2008

KUVA 7. Renderöidyn ympäristön ensimmäinen, toinen ja viimeinen kuva. Iivonen, J. 2008

KUVA 8. Painikkeiden asettelu ympäristöön. Iivonen, J. 2008

KUVA 9. Laukaisupainikkeen rakenne (A=painikealue, B=tehtävänumero, C=pistekenttä, D=väriä vaihtava MovieClip-objekti, E=valmis painike). Iivonen, J 2008

KUVA 10. Hahmon kuvien asettelu aikajanalle. Iivonen, J. 2008

KUVA 11. Kuvassa A pinnat ovat jakamatta ja kuvassa B ne ovat jaettu. Iivonen, J. 2008

KUVA 12. Baked-materiaali vs. tiled-materiaalit. Iivonen, J. 2008

KUVA 13. Materiaalien kartoittaminen valoisalla ja varjoisalle alueelle. Iivonen, J. 2008

KUVA 14. Plane-objektin törmäysalue ympyrän sisällä. Iivonen, J. 2008

KUVA 15. Suorakaiteen muotoiset alueet ja niiden koordinaattipisteet. Iivonen, J. 2008

KUVA 16. 3D-oppimisympäristö Papervision3D:llä toteutettuna. Iivonen, J. 2008

LIITTEET

Liite-cd, joka sisältää:

- Opinnäytetyö pdf-tiedostona
- Suomenkielinen tiivistelmä rtf-tiedostona
- Englanninkielinen abstrakti rtf-tiedostona
- Bittikarttakuvilla toteutettu 3D-oppimisympäristö swf-tiedostona
- Papervision3D 2.0:n avulla toteutettu 3D-oppimisympäristö swf-tiedostona