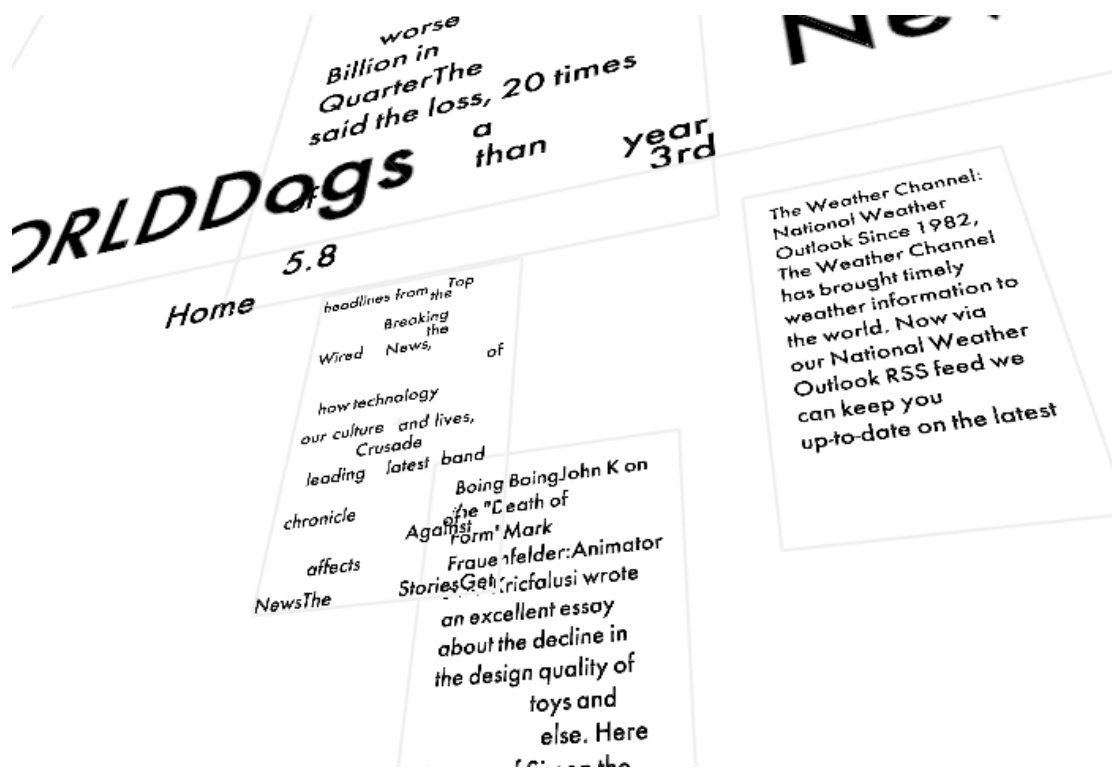


Dictria[



Arto Kellokoski

25.10.2006

Taiteen maisterin lopputyö

Medialaboratorio

Taideteollinen korkeakoulu

Tiivistelmä

Medialaboratorio, Taideteollinen korkeakoulu

Taiteen maisterin koulutusohjelma

Arto Kellokoski

arto.kellokoski@uia.fi

Dictria

Lopputyö

51 s.

Dictria on yhdessä Ida Blekelin kanssa tehty taiteen maisterin lopputyö. Projekti sai alkunsa yhteisestä motivaatiostamme etsiä keinoja ihmisten väliselle vuorovaikutukselle verkossa. Halusimme kokeilla yksinkertaisilla menetelmillä ja visuaalisilla malleilla mahdollisuuksia läsnäolon tunteen luomiseen. Työn inspiraatioina ovat toimineet muun muassa Jonathan Harrisin ja Sepandar Kamvarin We Feel Fine sekä Stefan Richterin Just Letters.

Yhdessä tekemämme konseptisuunnittelun jälkeen, tuotantovaiheeseen siirtyessämme, erotimme työtehtävät vahvuuksiemme mukaan. Oma roolini keskittyi tekniseen suunnitteluun. Tämä koostui Dictrian tietojärjestelmän ja arkkitehtuurin kehittämisestä sekä lopullisen prototyypin toteutuksesta.

Dictria on kollektiivinen kokemus, jonka tavoitteena on tutkia ilmaisukeinoja ihmisten läsnäololle ja vuorovaikutukselle verkossa. Teos sijoittuu sosiaalisten ohjelmien viitekehykseen, jossa käyttäjien toiminta on painottunut sosiaaliseen toimintaan ja verkostoitumiseen. Ratkaisu pohjautuu visuaaliseen asetelmaan, jossa perusyksikkönä toimivat sanat. Sanat muodostavat isompia kokonaisuuksia, tekstialueita, jotka on sijoitettu abstraktiin kolmiulotteiseen tilaan. Käyttäjät voivat liikutella sanoja luoden hetkittäisiä jälkiä läsnäolostaan. Lopputulosta voidaan pitää visuaalisena leikkinä sanojen muodostamissa naapurustoissa, jossa käyttäjien toiminnot muodostavat heidän läsnäolonsa.

Asiasanat: Processing, sosiaalinen ohjelmisto, vuorovaikutus, avoin lähdekoodi, XML, RSS

Sisällysluettelo

Sisällysluettelo	3
1. Johdanto	5
1.1 Motivaatio	6
1.2 Konsepti	6
1.2.1 Sana peruselementtinä.....	7
1.2.2 Konseptuaalinen kehys.....	7
1.2.3 Käyttäjä.....	7
2.1 Sosiaalinen ohjelmisto ja Web2.0.....	9
2.1.1 Avoin lähdekoodi.....	10
2.1.2 RSS ja XML	10
2.1.3 Processing.org.....	11
2.2 Vastaavia töitä	12
2.2.1 We feel Fine, Jonathan Harris ja Sepandar Kamvar	12
2.2.2 Just Letters, Stefan Richter	13
3.1 Ongelmakuvaus	15
3.2 Lähestymismetodit.....	15
3.2.1 Konseptin suunnittelu.....	16
3.2.2 Toiminnollain mallintaminen	17
3.2.3 Työkalut suunnitteluprosessissa.....	17
3.3 Flash-prototyyppi.....	18
3.3.1 Prototyypin Perustoiminnot.....	18
3.3.2 Avaruus ja liikkuminen	18
3.3.3 Vuorovaikutus.....	20
3.3.4 Datan noutaminen	21
3.3.5 Flash-prototyypin testaus ja huomiot.....	23
3.3.6 Prototyypin tekniset ongelmat	24
3.3.7 Prototyypistä tehdyt johtopäätökset	25
3.4 Processing-prototyyppi	26
3.4.1 Client-vaihtoehdot.....	26
4. Processing-prototyyppi	28
4.1 Ohjelmiston rakenne	28
4.2 Visuaaliset elementit.....	29
4.2.1 Grafiikkaprimitiivit	29
4.2.2 Sana	30
4.2.3 Tekstialue.....	31
4.3 Tilan, liikkumisen ja paikannuksen toteutus	32
4.3.1 Vektori.....	32
4.3.2 Kvaternio	32
4.3.3 Liike ja navigaatio.....	33
4.4 Käyttöliittymä.....	34
4.4.1 3D-käyttöliittymä.....	34
4.4.2 2D-käyttöliittymä.....	35
4.5 Palvelin ja tietokanta.....	37
4.5.1 Tietokantayhteys	38
4.6 Ääni.....	39
4.7 Processing prototyypin ongelmakohdat	40
4.7.1 Uuden tekstialueen lisääminen	41

4.7.2 Renderöinti	42
5. Yhteenveto	44
5.1 Dictrian nykytila	44
5.1.1 Rakenne	44
5.1.2 Toiminnallisuus	45
5.2 Projektista opitut asiat	45
5.3 Tulevaisuus	46
5.3.1 Rakenne	46
5.3.2 Lisäominaisuudet	47
Lähdeluettelo	48

1. Johdanto

Tämä lopputyö on teos joka on syntynyt Arto Kellokosken ja Ida Blekelin yhteistyönä. Olemme työskennelleet tämän projektin parissa vuoden ajan, aloittaen työn syksyllä 2005. Olemme tehneet projektia täysipäiväisesti toukokuusta 2006 lähtien viimeistellen prototyypin ja lopputyön kirjallisen osan lokakuussa 2006.

Projekti alkoi kahden ihmisen yhteistyönä. Molemmat työskentelivät samalla digitaalisen median alalla, mutta erilaisin taustoin. Ida Blekelin tausta on visuaalisessa suunnittelussa ja hänen kokemuksensa tulee graafisen ja vuorovaikutteisen median töistä. Itselläni on tekninen tausta. Olen opiskellut tietotekniikkaa Teknillisessä korkeakoulussa, jossa keskityin vuorovaikutteiseen digitaaliseen mediaan ja ohjelmistotuotantoon. Näiltä aloilta minulla on myös käytännön työkokemusta. Yhteinen mielenkiintomme ja idea sosiaalisiin ohjelmistoihin saivat kimmokkeen niiden yhteisten kurssien ja projektien aihepiireistä, joissa työskentelimme Medialaboratoriossa opiskellessamme.

Halusimme tehdä konseptisuunnittelun yhdessä ja erottaa roolimme tuotantovaiheeseen siirryttäessä. Tahdoimme luoda yhteisen konseptin ja hyödyntää molempien osaamista projektin hyväksi. Idan vastuualueeksi muodostui visuaalisten mallien hahmottelu ja toteutus. Lisäksi Ida työskenteli paljon käyttäjäskenaarioiden luomisessa ja suunnitteli sekä toteutti käyttäjähaastattelut. Oma roolini keskittyi tekniseen suunnitteluun, mihin kuului DICTRIAN tietojärjestelmien ja arkkitehtuurin kehittäminen lopullisen prototyypin toteutuksen lisäksi.

DICTRIAN toteutuksessa on alusta alkaen huomioitu äänimaailman toteuttaminen. Äänen tarkoituksena on tukea tilallista vaikutelmaa ja korostaa käyttäjien toimintoja. Äänisuunnittelijana on toiminut Taneli Bruun keväästä 2006 lähtien.

1.1 Motivaatio

Koimme sosiaalisten ohjelmistojen olevan sopiva alue kokeilulle. Projektin alussa keskustelimme mahdollisuudesta toteuttaa tarinapohjainen maailman, jossa olisi kerrostettu malli henkilörakennukselle. Tunsimme kuitenkin tarinankerronnan rajoittavan liikaa käyttäjien mahdollisuutta toimia keskinäisessä vuorovaikutuksessa ja vähentävän sosiaalisuuden merkitystä. Tämän päätöksen pohjalta päätimme lähestyä ongelmaa avoimen ja yksinkertaisen toteutuksen kautta, joka olisi mahdollisimman helposti saavutettavissa.

Projektin motivaatio syntyi ajatuksesta mahdollistaa keinoja ihmisten väliselle vuorovaikutukselle. Tämä on projektimme keskeisin tutkimusongelma. Emme halunneet tehdä foorumia, blogia tai chat-järjestelmää. Sen sijaan halusimme kokeilla yksinkertaisilla ja leikkisillä mekaniikoilla jotka tarjoaisivat käyttäjille hienovaraisia menetelmiä kommunikoida keskenään. Ideana oli toteuttaa yksinkertaistettu ympäristö joka voisi inspiroida ja luoda leikkisiä kokemuksia. Hakiessamme inspiraatiota projektillemme törmäsimme pieneen Flash-sovellukseen nimeltä Just Letters (Richter 2005) (Käsitelty tarkemmin alakohdassa 2.2.2). Just Lettersissä kirjaimia on käytetty vuorovaikutteisina elementteinä. Tässä sovelluksessa ihmiset pystyivät liikuttelemaan värikkäitä kirjaimia pienellä valkoisella alueella joka muistutti pientä osaa jääkaapin ovesta. Tapa jolla kirjaimet toimivat rakennusosina luodakseen pienen interaktiivisen kokemuksen inspiroi meitä määrittäessämme projektin suuntaa. Meitä alkoivat kiinnostaa tavat, joilla tämä yksinkertainen idea voitaisiin laajentaa rikkaammaksi kokemukseksi.

1.2 Konsepti

Dictria on kollektiivinen kokemus, jossa ihmiset voivat liikuttaa sanoja uusiksi lauseiksi ja siten uusiksi tarkoituksiksi. Dictria on sijoitettu kolmiulotteiseen avaruuteen, jossa sanaryhmistä muodostuu dynaaminen asetelma. Sanat on sijoitettu erillisiin suorakulmioihin, tekstialueisiin. Liikkumisen ja navigaation kautta tekstialueet näkyvät käyttäjille keskittyminä, kuin rakennuksina ja naapuristoina, jotka rakentuvat sanoista.

1.2.1 Sana peruselementtinä

Dictria käyttäytyy kuin useat jääkaapin ovet täynnä sanoja, joita voidaan liikutella ympäriinsä. Sanat ja niiden liikkeet kuvaavat visuaalista ja tekstuaalista tarkoitusta, jossa liikkuvista sanoista tulee hetkellisesti käyttäjän avatar. Sanojen muodostama liike tuottaa ripauksen taikaa, kun sanat lentävät uusiin paikkoihin. Lopputulosta voi ajatella visuaalisina ”jameina” joissa osallistujat käyttävät sanoja pelataksaan keskenään. Navigointi on suunniteltu visuaaliseksi havainnoinniksi tilassa jossa sanat muodostavat satunnaisia typografisia asetelmia.

1.2.2 Konseptuaalinen kehys

Viimeaikaisena web-trendinä voidaan pitää käyttäjien toiminnan painottumista puhtaasti käytännöllisiin tai viihteellisiin arvoihin luoden uusia sosiaalisia ohjelmia ja työkaluja (O'Reilly 2005). Jotkin palveluista ovat puhtaasti viihteellisiä kuten esimerkiksi youtube.com, jossa käyttäjät voivat jakaa lyhyitä videoita (You Tube 2006). Toisaalta Wikipedian kaltaiset palvelut tuottavat paljon käytännöllistä tietoa (Wikipedia 2006a). Polttopisteessä ovat olleet toteutukset, joissa melkein rajoittamaton määrä käyttäjiä voi luoda ja rakentaa verkottuneita yhteisöjä ja sisältöjä. Halusimme keskittää Dictrian toiminnot suoraan kommunikaatioon ja läsnäoloon, sekä niiden eri tasojen etsimiseen. Dictria pyrkii korostamaan hetken tärkeyttä painottamalla läsnäolon merkitystä.

1.2.3 Käyttäjä

Oletamme Dictrian käyttäjien olevan aktiivisia Internetin käyttäjiä, jotka osallistuvat jo verkoissa tapahtuvaan tiedon tuottamiseen. He saattavat käyttää Flickrä (Flickr 2006), ylläpitää blogia, pelata online-pelejä tai olla jäsenenä verkkofoorumeissa. Dictrian käyttäjät ovat aktiivisia muodostamaan ja tuottamaan sisältöä verkkoympäristöihin.

Dictria on paikka jossa voi käydä lyhyitä aikoja kerrallaan taukona muusta toiminnasta, inspiraation lähteenä. Osalle käyttäjistä tämä voi olla laajennus

työtapoihin, joissa työympäristö on jaettu tietokoneen näytöllä ikkunoihin. Toisille Dictrian käyttö voi olla osa vapaa-aikaa. Yhteistä molemmille ryhmille on, että Dictria tulee osaksi jo valmiiksi olemassa olevaa verkkoaktiivisuutta.

2 Konteksti

Dictria pohjautuu käyttäjien väliseen vuorovaikutukseen ja sen tutkimiseen. Vertailtuamme omaa verkkokäyttäytymistämme otimme Dictrian kontekstiksi Internetin ja siellä vaikuttavat sosiaaliset ohjelmistot. Näistä syntyneestä inspiraatiosta lähdimme kehittämään Dictrian konseptia.

2.1 Sosiaalinen ohjelmisto ja Web2.0

Dictrian merkittävin yksittäinen viitekehys on sosiaaliset ohjelmistot. Webin viimeaikainen kehitys on painottunut paljon käyttäjien sosiaaliseen toimintaan sekä verkostoitumiseen. Blogien kaltaiset ratkaisut ovat muodostaneet verkosta entistä enemmän sosiaalisen, demokraattisen rakenteen. Uudet palvelut sekä työkalut kuten Wikipedia ovat luoneet uusia tapoja käyttää ja hakea tietoa Internetistä sekä jakaa sitä. (O'Reilly 2005)

Näitä toisen sukupolven Internet-pohjaisia palveluja, jotka painottuvat sosiaalisiin verkkoihin ja sivuihin sekä erilaisiin wikeihin ja kommunikaatiotyökaluihin, on alettu kutsumaan termillä Web2.0 (O'Reilly 2005). Sillä pyritään viittaamaan webin tulevaisuuteen ja nykyhetken kehitykseen. Web2.0:n keskeisimmäksi asiakokonaisuudeksi on muodostunut niin sanottujen sosiaalisten ohjelmistojen (Social software) käyttö, jossa ihmisten välinen vuorovaikutus on avainasemassa. Web2.0:ssa sosiaalinen toiminta mahdollistaa työkalujen ja kommunikaation kasvamisen vuorovaikutuksen kautta.

Vuorovaikutus voi ulottua myös ohjelmistojen kautta tapahtuvan kommunikaation ulkopuolelle. Sosiaalisella ohjelmistolla voidaan tarkoittaa myös ohjelmistotuotantoa, joka syntyy yksilöiden yhteisestä kiinnostuksesta tiettyä aihepiiriä kohtaan kaupallisen valtavirran ulkopuolella. Yhteinen mielenkiinnon kohde, joka sitoo verkoston yhteen, voi olla esimerkiksi tekninen, sosiaalinen tai kulttuurinen tekijä. Näiden kahden määritelmän erilaisuudesta huolimatta molemmilla on vahvat siteet sosiaalisiin ryhmiin joiden jakama kieli ja toimintatavat ovat sosiaalisen verkon määrittelemiä ja hyväksymiä.

2.1.1 Avoin lähdekoodi

Avoimella lähdekoodilla voidaan tarkoittaa mitä tahansa ohjelmistoa, joka täyttää esimerkiksi Open Source Initiativen (OSI) vaatimukset (Open Source Initiative 2006). Avoimen lähdekoodin määritelmiä on useita ja OSI on yksi niistä. Se määrittää pragmaattiset menetelmät ohjelmistojen tuottamiselle ja kehittämiselle, joilla sallitaan lopputuotteen käyttämisen ja lähdekoodin jakamisen vastaaville projekteille ja muille käyttäjille. DICTRIA perustuu avoimelle kommunikaatiolle, joten avoimen lähdekoodin käyttö sekä oman lähdekoodin jakaminen luonteva osa DICTRIAN kehitystä.

Avoimen lähdekoodin on havaittu olevan tuotannollinen lähestymistapa, jossa luotetaan ohjelmistotuotantokulttuuriin ja jonka tietyt ratkaisumallit sekä toteutustavat ovat hyviksi havaittuja. Nämä menetelmät on annettu kaikkien käytettäväksi avoimen lähdekoodin sääntöjä noudattaen. Etenkin Processingin sosiaalinen ohjelmistotuotannon malli sekä sitä kehittävän yhteisön tapa jakaa koodia muiden käytettäväksi on ollut merkittävässä osassa kehittäessämme DICTRIAA (alakohta 2.1.3).

2.1.2 RSS ja XML

Vaikka informaatiota voidaan siirtää Internetissä monella tavalla, on olemassa tiedostomuotoja sekä protokollia jotka tekevät tiedon hakemisen helpommaksi. XML (Extensible Markup Language) on yksinkertainen metakieli, jolla voidaan kuvata toisia kieliä tai tiedon rakenteita (W3C 2003). RSS (Really Simple Syndicate) on XML-pohjainen tiedostoformaatti, jonka tavoitteena on luoda sisältöprotokolla tiedon jakamiseen webissä (O'Reilly 2002). Sen tavoitteena on tarjota tietoa verkosta syötteinä, joihin käyttäjät voivat kirjautua (assign). Niitä voidaan lukea ja järjestellä RSS-lukuohjelmalla.

RSS:n ideana on tarjota menetelmä reaaliaikaiseen viestintään, jossa tiedonhaun vaihe ohitetaan ja sisällöntuottaja päivittää RSS-syötettä. Syöte luetaan yleensä erillisellä lukuohjelmalla, joka hakee ne syötteet, joihin käyttäjä on kirjautunut. Kukin syöte päivittyy käyttäjälle sitä mukaa mitä itse syöte päivittyy, tehden sisällön välittämisestä reaaliaikaisempaa ja keskitetympää kuin perinteinen web-selaaminen.

RSS:llä on ollut edeltäjiä (O'Reilly 2002), mutta vasta Web2.0 mukana RSS yleistyi etenkin blogien kautta. Dictrialle se on käytännöllinen ja luontainen tapa lukea käyttäjän luomaa informaatiota. Dictria ei käytä hyväkseen syötettä jatkuvana päivittyvänä virtana, vaan Dictria lukee yhden syötteen sisältämän XML-informaation. RSS on yleensä hyvin standardoidussa formaatissa, mikä tekee siitä helpommin luettavan kuin esimerkiksi HTML. Web2.0:n kautta se on myös luonteva menetelmä datan välittämiseen, koska se on usein käytetty mekanismi erilaisten sosiaalisten verkkojen tuottaman sisällön jakamisessa.

2.1.3 Processing.org

Processing on avoimeen lähdekoodiin perustuva ohjelmointikieli ja kehitysympäristö, ja Dictria on toteutettu sitä käyttäen. Processing.org on Processingia tuottava organisaatio. Processing ei ole yhtiö, vaan se koostuu yksilöistä jotka jakavat aikaansa ja tietotaitoansa yhteisen ohjelmistotuotannon hyväksi (Processing.org 2006a).

Processingin kehittämisessä merkittävässä roolissa ovat sitä käyttävät henkilöt ja etenkin ne, jotka aktiivisesti osallistuvat Processingin käyttöön ja sen beta-vaiheessa olevien versioiden kehittämiseen. Sen foorumeilla on myös käyttäjien lähettämiä esimerkkejä ja ongelmakuvauksia. Käyttäjien keskenään jakamalla kokemuksella on suuri merkitys Processingin ja sille pohjautuvien ohjelmien kehityksessä, sillä Processing on vasta beta-asteella.

Suurin osa Processingin dokumentaatiosta on lähdekoodipohjaista. Käyttäjien jakamat esimerkit ja kokemukset laajentavat tätä pohjaa ja luovat käsitystä siitä, mihin ohjelmisto kykenee ja mitkä ovat sen asettamat rajat. Processingin sosiaalisella luonteella on Dictrialle suuri merkitys, sillä useat Dictriaan liittyvät ongelmakohtat olivat suorasti tai epäsuorasti tulleet esille jo muiden käyttäjien keskusteluissa.

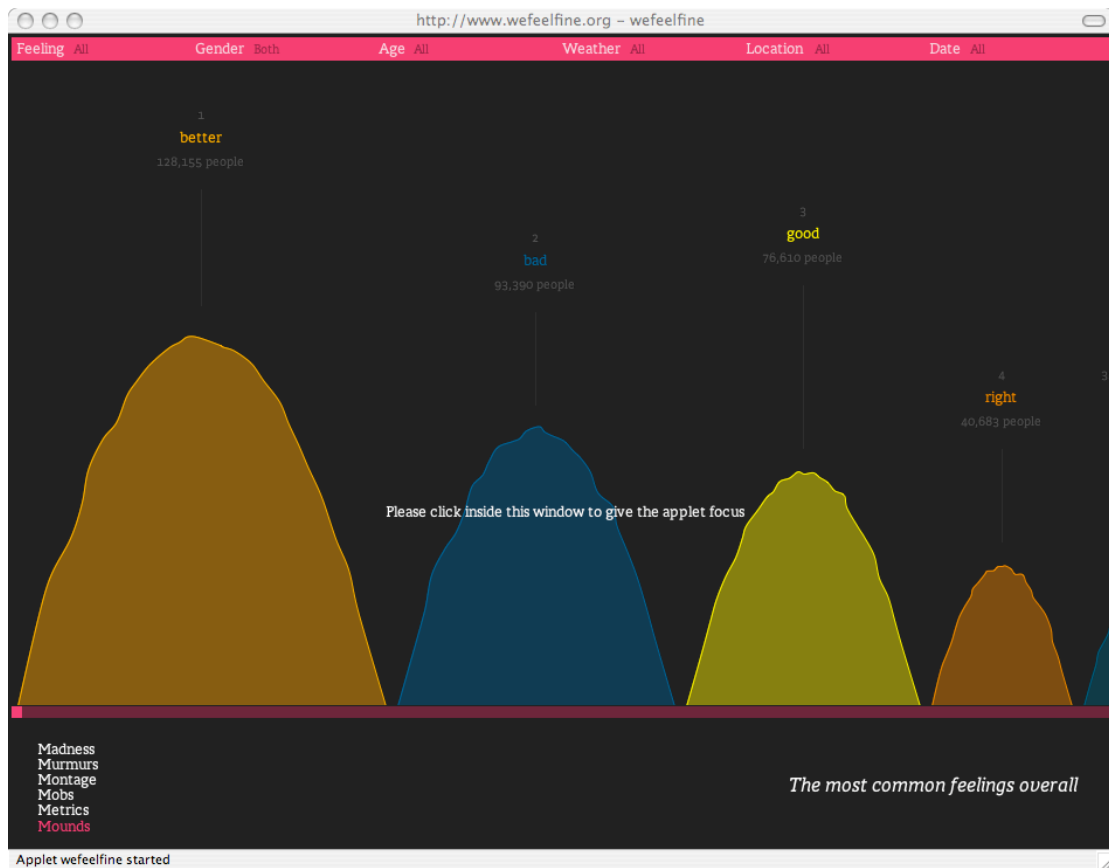
2.2 Vastaavia töitä

Läpi suunnittelu- ja tuotantoprosessin ovat muutamat työt toimineet inspiraationa Dicitrian kehitykselle kontekstin ja toteutustavan kautta. Niiden merkittävimmät ominaisuudet ovat olleet sosiaalisten näkökulmien korostamisessa (Just Letters, alakohta 2.2.2), sekä sen yhdistämisessä toimivaan ja uniikkiin visualisointiin (We Feel Fine, alakohta 2.2.1).

2.2.1 We feel Fine, Jonathan Harris ja Sepandar Kamvar

We Feel Fine (Harris, Kamvar 2005) on ensimmäinen Processingillä tuotettu teos jota tarkastelimme ennen kuin aloitimme Dicitrian toteutuksen Processing-ohjelmistolla. Sen merkittävyys projektin kannalta oli kartoittaa mahdollisuuksia, joita Processing tarjoaa.

We Feel Finen idea on kerätä suuri määrä ihmisten tunteita eri blogeista sekä visualisoida ne kysymysten, kuten ”Ovatko Eurooppalaiset muita iloisempia?”, kautta. We Feel Fine etsii muutaman minuutin välein uusimmista blogeista lauseita, jotka sisältävät lausahduksen ”I feel” tai ”I am feeling” ja etsii tunteen jota lauseessa kuvaillaan. Tämä tieto tallennetaan tietokantaan. Tietokannan kasvaessa eri tunteita voidaan hakea ja lajitella erilaisin visuaalisin ja demografisin keinoin (Kuva 1).



Kuva 1: We feel Finen Visualisointia

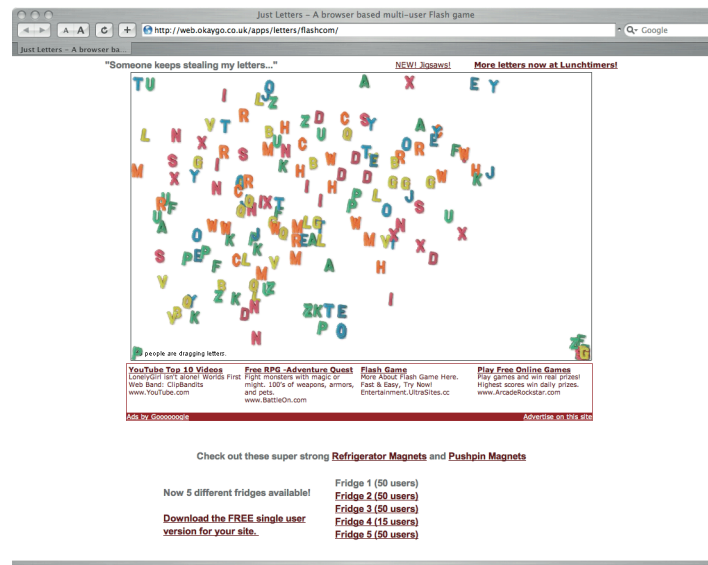
Lopputuloksena on järjestelmä, jossa kukin yksittäinen visuaalinen yksikkö edustaa yksittäistä tunnetta, joka on jonkin yksittäisen henkilön kirjoittama. Kullakin yksiköllä voi olla ominaisuuksia kuten koko tai väri, jotka korostavat kunkin tunteen yleisyyttä.

We Feel Fine on kaikkien tuottama taideteos, joka kasvaa ja muuttuu sitä mukaa kun käyttäjät ympäri maailmaa ja verkkoa päivittävät blogejaan. We Feel Finen tarkoitus on kuvata sisimpämme tunteita tietoverkkojen välityksellä. Se osoittaa kuinka sosiaalinen kommunikaatio syntyy yksiköistä, joista kasvaa suurempi kokonaisuus, joka lopulta esitetään yksinkertaisista muodoista luotuna visualisaationa.

2.2.2 Just Letters, Stefan Richter

Yksi ensimmäisistä töistä, joka liittyy Dictrian aihepiiriin, on Stefan Richterin Just Letters (Richter 2005). Työ koostuu valkoisesta ikkunasta, jossa on värikkäitä

kirjaimia joita kukin käyttäjä pystyy liikuttamaan ympäri ruutua. Kukin kirjain muodostaa eräänlaisen magneetin, jota voi liikutella pintaa pitkin (Kuva 2).



Kuva 2: Just Letters

Just Letters on selaimessa toimiva usean käyttäjän Flash-sovellus ja muistuttaa Dictriaa kokemukseltaan ja tekniikaltaan. Liikkuvista kirjaimista tulee hetkittäin muiden käyttäjien avatar, mutta muuten muiden ihmisten läsnäoloa ei mitenkään kuvata.

Just Lettersin vahvuus on siinä, ettei siinä yksittäisten kirjainten liikuttelun lisäksi ole mitään muita keinoja kommunikoida käyttäjien kesken. Silti hetken sitä seurattuaan voi havaita kuinka käyttäjät toimivat yhdessä toistensa kanssa, muodostaen sanoja ja varastaen toistensa kirjaimia.

Just Letters ja Dictria jakavat tiettyjä ydinideoita, mutta ne eroavat toisistaan kahden oleellisen suunnitteluratkaisun osalta. Ensinnäkin Dictria käyttää kokonaisia sanoja vuorovaikutuksen peruselementteinä. Sanoja käyttämällä pyrkimyksemme oli painottaa Dictrian toimintaa tekstuaaliseen käyttöön ja ilmaisuun. Suurimpana erona pidämme kuitenkin kolmannen ulottuvuuden käyttöä Dictrian ratkaisussa sekä vuorovaikutuksen visualisointia kolmiulotteisessa tilassa. Tällöin yksittäisten sanojen lisäksi visualisointiin liittyvät myös niiden muodostamat tekstialueet.

3. Ongelmat ja ratkaisut suunnitteluprosessissa

Ongelman ratkaisua varten tarvitsimme menetelmän konseptin muodostukseen ja sen siirtämiseksi prototyypiksi. Pyrimme löytämään tapoja, joilla voisimme testata konseptin oleellisemmat toiminnot ja löytää sen mahdolliset ongelmakohdat.

3.1 Ongelmakuvaus

Dictrian suunnittelussa suurimmaksi ongelmaksi nousi niiden olennaisten toimintojen määrittely, jotka täyttävät konseptin ja motivaation tavoitteet. Useissa sosiaalisissa ohjelmistoissa keskitytään tietoon ja kommunikointiin, kun taas Dictriassa keskitytään siihen, miten vuorovaikutuksesta voisi johtaa myös läsnäolon tunteen. Haasteeksi muodostui se, kuinka tämä saataisiin toteutettua audiovisuaalisin menetelmin, ja mitä toiminnallisia sekä teknisiä ominaisuuksia niiden luomiseksi tarvittaisiin.

Internet on ollut alusta alkaen haluamamme kanava Dictrian käytölle. Web-ympäristöön on pitkään ollut monia, laajalti käytettyjä kehitysympäristöjä. Dictrialle tuli löytää kehitysympäristö, jonka lopputuotteena olisi helposti saavutettava sovellus ja joka mahdollistaisi dynaamisen 3D-ympäristön luomisen.

Toteutuksen monimutkaisuutta on lisännyt valinta toteuttaa Dictria kolmiulotteisena tilana. Koska yksi projektin haasteista oli tunnistaa tarvittavat mekanismit konseptin toteutukseen, emme kyenneet tunnistamaan kaikkia mahdollisia projektin ongelma-kohtia. Tämän vuoksi pyrimme valitsemaan ohjelmistotuotantoprosessin tavalla, joka jättäisi tuotantovaiheessa varaa rakenteen ja ongelma-kohtien havainnoinnille ja muutoksille itse konseptia rikkomatta.

3.2 Lähestymismetodit

Lähdimme ratkaisemaan konseptin ongelmaa ensin suunnittelemalla konseptia ja sen jälkeen kehittämällä sitä edelleen luomalla prototyyppejä. Kävimme läpi käyttäjän roolia ja tämän käyttäytymisen luomaa kokemusta. Pääideana oli toteuttaa

vuorovaikutus visuaalisin keinoin, jossa ympäristön komponentit määrittelisivät käyttäjän identiteetin tekstuaalisen sisällön kautta.

3.2.1 Konseptin suunnittelu

Konseptia suunniteltaessa keskustelimme paljon käyttäjän läsnäolosta ja siitä miten se saataisiin yhdistettyä visuaaliseen maailmaan. Kontekstin muodostamisen jälkeen haasteeksi tuli luoda konsepti ja kehittää sitä eteenpäin ohjelmistoprototyypiksi.

Aloitimme konseptisuunnittelun ideoimalla vapaasti eri mahdollisuuksia käyttäjäkokemuksen ja ympäristön toteuttamiseksi. Pidimme useita tapaamisia, joiden tarkoituksena oli kehittää konseptin ydinajatus keskustelumme kautta. Piirsimme hahmotelmia tilan rakenteesta ja tutkimme mahdollisia vastaavia ratkaisuja. Olimme alusta alkaen kiinnostuneet tilan visuaalisen toteutuksen vaikutuksesta itse käyttäjäkokemukseen. Idea tekstin käytöstä keskeisimpänä visuaalisena elementtinä muodosti konseptin suuremman kehyksen.

Ensimmäinen idea oli käyttää tekstiä tarinankerronnallisena elementtinä, josta käyttäjän hahmoa rakennettaisiin. Mielestämme tämä ei kuitenkaan sopinut vuorovaikutteisen tilan kontekstiin, koska tarina olisi tällöin keskeisimpänä elementtinä vuorovaikutuksen sijaan, ja siten mielestämme liian rajoittava. Tässä vaiheessa saimme ajatuksen kehittää tekstuaalista sisältöä enemmän graffitien kaltaisina graafisina elementteinä.

Konseptin ydinelementeistä päätimme toteuttaa abstraktin kolmiulotteisen maailman, joka kasvaa sisällön mukaan käyttäjän perspektiivistä. Kaikilla käyttäjillä on käytettävissä sama sisältö, jota he voivat lisätä.

Teimme paperiluonnoksia ideoista, mutta halusimme löytää interaktiivisemmän menetelmän konseptin jatkokehitykselle. Valitsimme roolimme jatkokehityksessä ja aloitimme tutkia konseptia tässä muodossa. Ida jatkoi työstämällä konseptia paperiprototyyppien kautta (Blekeli 2006), ja minun rooliksi muodostui interaktiivisten prototyyppien luominen ohjelmoimalla. Päätimme jatkaa konseptin

kehitystä, mutta alkaa toteuttaa siitä teknisesti toimivia demoja ydintoimintojen tarkentamiseksi.

3.2.2 Toiminnoittain mallintaminen

Kun aloimme rakentaa Dictrian konseptia kokeelliseksi malliksi, päätimme ottaa käyttöön ohjelmistotuotannollisen lähestymistavan, jonka avulla voisimme viedä kehitysvaihetta eteenpäin toiminnoittain mallintamalla. Metodien tarkoitus oli luoda testiympäristö, johon sisällytettäisiin kaikki peruselementit ja tietotyypit sisältävät objektit. Malli kattaisi demovaiheesta riippuen eri toimintoja. Kutakin toimintoa varten rakensimme erillisiä prototyyppejä, joiden avulla erillisiä toimintoja kyettiin testaamaan. Jokaista demovaihetta varten mallia laajennettiin aina niillä ominaisuuksilla joita kukin demovaihe vaatii.

Tätä metodia kutsutaan ohjelmistotuotannossa nimellä ”Rapid Prototyping Model”. Menetelmän pääideana on tuottaa ripeässä tahdissa lopputuotteesta prototyyppejä eri toimintoja mallintamalla, sekä asteittain tarkentaa lopullisen tuotteen vaatimuksia prototyypeistä opittujen kokemusten pohjalta. Metodi toimii erityisesti niissä tilanteissa, joissa lopullisen ohjelmiston kaikkia vaatimuksia ja ominaisuuksia ei tunneta. Niitä voidaan saada esille prototyypimalleilla. (Schach 2002)

3.2.3 Työkalut suunnitteluprosessissa

Suunnitteluprosessi ja demovaihe toteutettiin Adoben Illustrator- ja Flash-ohjelmistoilla. Flash tarjosi alustan toimivien ja interaktiivisten demojen nopealle tuottamiselle. Illustrator mahdollisti visuaalisten mallien hahmottamisen ja tuottamisen prototyyppejä varten.

Tavoitteena oli toteuttaa Dictria niin alustariippumattomasti kuin mahdollista, jotta se tulisi vaatimaan mahdollisimman harvojen asennettavien lisäkomponenttien (plugin) käyttöä. Dictrian tuli siten toimia joko pelkästään selaimessa tai sen tuli käyttää ohjelmistoa, jonka laajennukset (plugin) olisivat yleisesti käytössä. Varalle jätimme mahdollisuuden Dictriasta itsenäisesti toimivana sovelluksena.

Demovaihetta aloittaessamme oletimme, että Flash sopisi parhaiten projektillämme. Flash on suunniteltu multimediatuotantoon ja se toimii integroituna suunnitteluympäristönä. Flashiin valmiiksi rakennetut objektit ja graafinen käyttöliittymä nopeuttavat prototyyppien luomista. Meillä molemmilla oli entuudestaan paljon kokemusta sekä Flashillä että Illustratorilla työskentelystä, mikä antoi meille luottamusta niiden käyttämisestä.

3.3 Flash-prototyyppi

Päätimme toteuttaa prototyypit vaiheittain konseptin toiminnallisuuksien perusteella. Ensimmäinen vaihe sisältäisi avaruudellisen rakenteen sekä itse kolmiulotteisessa tilassa navigoimisen. Samalla toteuttaisimme objektit sanoille sekä tekstialueille. Toisessa vaiheessa lisäisimme prototyyppiin mahdollisuuden tekstin syöttöön ja lisäämiseen sekä mahdollistimme käyttäjän vuorovaikutuksen itse avaruudessa olevien objektien kanssa. Kolmannessa vaiheessa toteuttaisimme datan hakemisen RSS-syötteestä sekä sen käsittelyn.

3.3.1 Prototyypin Perustoiminnot

Prototyypissä tuli mallintaa kolmiulotteista ympäristöä, jossa käyttäjän olisi mahdollista liikkua eteen- ja taaksepäin sekä sivuttain. Käyttäjän tuli voida myös katsoa vapaasti ympärilleen. Ensimmäinen prototyyppi sisälsi toiminnot liikkumiseen ja navigoimiseen, sekä objektit tekstialueiden sekä sanojen luomiseksi. Näiden toimintojen lisäksi käyttäjän tuli voida vuorovaikuttaa tilassa olevien tekstialueiden kanssa. Tämä edellytti tekstialueiden lisäämistä ja valitsemista ympäristöstä, sekä sanojen liikuttamista tekstialueen sisällä.

3.3.2 Avaruus ja liikkuminen

Toteutimme ensimmäisen prototyyppiversion varsin nopealla tahdilla ja pidimme ratkaisun yksinkertaisena. Flash on suunniteltu webtuottamista varten ja se painottuu suurimmalta osalta kaksiulotteisen grafiikan käyttöön. Flash ei tue aitoa 3D-

mallinnusta vaan sen menetelmänä voi pikemminkin pitää kolmannen ulottuvuuden luomista erilaisten pseudomenetelmien kautta.

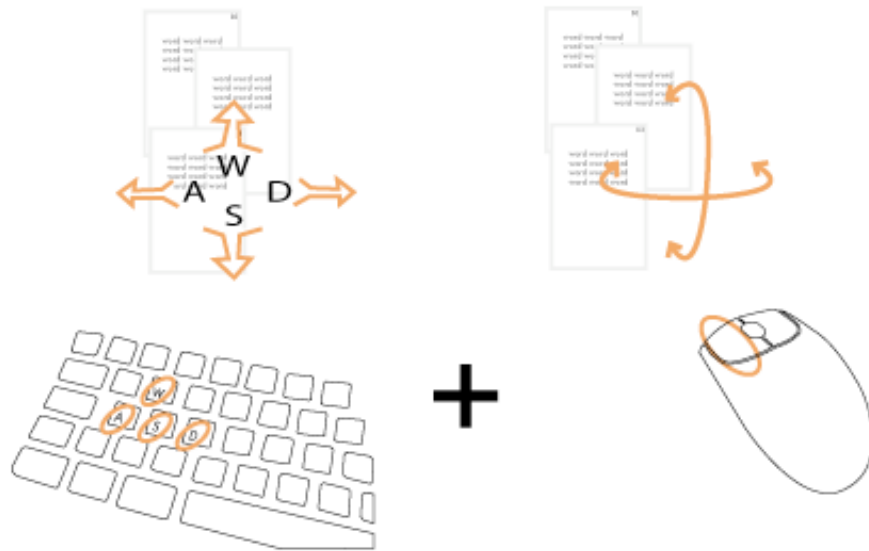
Jotta saimme mallinnettua kolmiulotteisen maailman, loimme ympäristön objektit siten, että ylimmillä tietorakenteilla kuten käyttäjällä ja tekstialueilla on määritelty koordinaatit kolmessa ulottuvuudessa (x,y,z). Näiden yläluokkien sisään tulevat komponentit, kuten sanat tekstialueiden sisälle, perivät arvot objektilta jonka sisään ne on luotu.

Tärkein ominaisuus avaruudellisen vaikutelman luomisessa oli käyttäjän sijaintia ja katselusuuntaa edustava kameraobjekti. Rakensimme sen niin, että se sisältää käyttäjän sijainnin x,y,z - koordinaatistossa. Koordinaattien lisäksi kameraobjekti pitää sisällään arvot jotka määrittävät katselusuunnan. Katselusuunta määritettiin kahdesta kulmasuureesta, jotka olivat rotaatioita x - ja y -akselin suhteen.

Käyttäjän toiminnot kuten liikkuminen ja katseleminen voitiin lukea käyttäjän syötteistä skalaarisuureina ja muuntaa liike vektorisuureiksi trigonometrian avulla. Jotta vaikutelma olisi kolmiulotteinen, lisäsimme kameraobjektiin polttoväliä muistuttavan arvon, jonka avulla perspektiivin skaalaus x -, y - ja z - akseleiden sekä katselualan suhteen saatiin luonnolliseksi. Piirsimme kaukana olevat objektit pienemmiksi ja skaalasimme ne vaaka ja pystysuunnassa polttovälin avulla. Piirsimme kaikki objektit kohtisuorassa käyttäjää kohti.

Lopputuloksena oli eräänlainen pseudo-3D-ratkaisu, jonka kappaleiden mallinnus oli kaksiulotteista. Käyttäjän sijainnin ja rotaatioiden suuruuden avulla voitiin laskea ruudulle piirrettävät objektit ja kolmiulotteinen vaikutelma pystyttiin toteuttamaan skaalaamalla kaksiulotteisia objekteja.

Toteutimme käyttäjän liikkumisen prototyypissä näppäimistön ja hiiren yhteistoiminnalla (Kuva 3). Hiiren liikkeistä laskettiin ylös- ja sivuille suuntautuva kiertoliike ja näppäimistöstä kyettiin lukemaan eteen, taakse ja sivuille suuntautuva liikkuminen. Yhdistämällä nämä kaksi, katsetta voitiin kiertää kaikissa suunnissa ja liikkua katseen määrittämän suunnan suhteen. Tämä ratkaisu käyttää tietokonepeleissä usein käytettyä menetelmää (Bowman et al. 2004).



Kuva 3: Dictrian navigaation toteutus

3.3.3 Vuorovaikutus

Liikkeen ja vapaan katselunäkymän lisäksi prototyypiin tuli toteuttaa menetelmät joilla käyttäjä kykeni vuorovaikutukseen sisällön kanssa. Prototyypin ensimmäisessä versiossa kukin toteutettiin kaksi toimintoa. Ensimmäinen on yksittäisen tekstialueen valitseminen editoitavaksi ja toinen on uuden tekstialueen lisääminen.

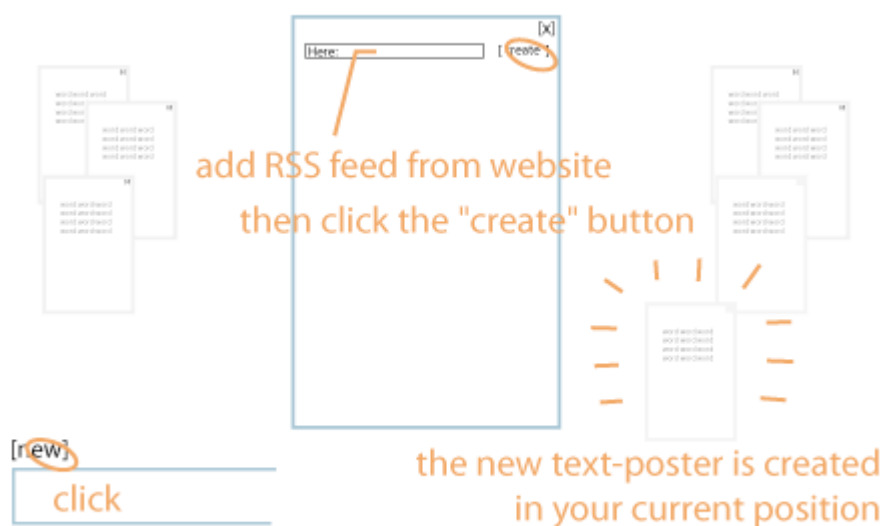
Kukin tekstialue on valittavissa Dictrian kolmiulotteisesta tilasta. Ympäristö pyrkii antamaan visuaalisia vihjeitä objektien syvyysvaikutelmista sekä antamaan palautetta käyttäjän valinnoista visuaalisina korostuksina. Visuaaliset vihjeet muodostuvat tekstialueiden suhteellisesta koosta ja niiden sijainneista toisiinsa nähden. Kun yksi tekstialue valitaan, avautuu se käyttäjän muokattavaksi täyttäen noin kolmanneksen Dictrian käyttöliittymästä. Uuden tekstialueen lisääminen tapahtuu samaa metodologia käyttäen: käyttäjä valitsee uuden tekstialueen lisäämisen, jonka jälkeen avautuu tekstialueen näköinen ikkuna, jonne käyttäjä voi syöttää RSS-syötteen osoitteen.

Prototyypin toisessa versiossa käyttäjä kykenee lisäksi liikuttelemaan yksittäisiä sanoja avatun, ja siten muokattavaksi valitun, tekstialueen sisällä. Sanaa liikutellaan hiirtä raahaamalla. Nämä vuorovaikutusmekanismit kuvaavat niitä toimintoja, joihin

Dictria purettiin demovaiheessa sekä ensimmäistä ja toista prototyyppiä rakennettaessa.

3.3.4 Datan noutaminen

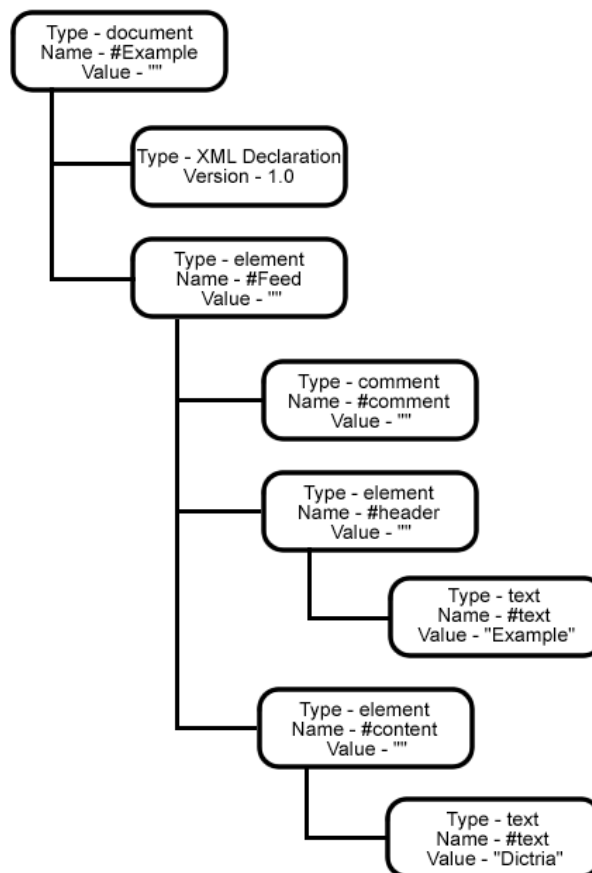
Suunnittelimme sisällön toimimaan Dictriassa RSS-syötteiden kautta. Lisätäkseen uuden syötteen käyttäjän tulee prototyypissä klikata [new]-painiketta, jonka jälkeen käyttäjälle avautuu ikkuna syötteen lisäämiseen. Käyttäjän tulee kirjoittaa oikeellinen RSS-syötteen osoite kenttään ja painaa nappia [create]. Uusi syöte suunniteltiin lisättäväksi käyttäjän eteen. Menetelmä on kuvattu kuvassa 4.



Kuva 4: Uuden tekstialueen lisääminen

Datan noutamiseksi ja esittämiseksi tuli Dictriaan luoda toiminnot sen käsittelemiseen. RSS-syöte on XML-muotoista dataa, joka noudetaan käyttäjän antamasta url:sta (W3C 2006). Dataa noudettaessa tuli huomioida se, etteivät Flashin tietoturvaominaisuudet salli lainkaan palvelinkutsuja sen toimialueen (domain) ulkopuolelle jossa Flash-sovellus itse toimii (Macromedia 2002). Tiedon noutaminen ja jäsentäminen toteutettiin siten, että prototyyppiin luotiin XML-jäsentäjä ja datahaku reititettiin samalla palvelimella sijaitsevan PHP-sivun (PHP 2006) kautta.

PHP-sivu sisältää yksinkertaisen skriptin, joka lukee XML-tiedoston sille lähetetystä url:sta. PHP-skripti lähettää XML-merkkijonon (string) Flashille. Jotta XML-merkkijonon sisältämä tieto saataisiin haettua, luotiin prototyyppiin yksinkertainen XML-jäsenttjä.



Kuva 5: Yksinkertaistettu kaavio XML:n rakenteesta

Jäsenttjä käyttää tietoa hakiessaan hyväksi XML:n rekursiivista rakennetta (Kuva 5), jossa se käy läpi koko merkkijonon seuraavalla periaatteella: Jäsenttjä ottaa argumentikseen yhden XML-solmun (node), joka tässä tapauksessa on XML-merkkijonon ensimmäinen solmu. Jäsenttjä tarkistaa onko solmulla sisar tai lapsisolmuja. Jos on, kutsuu jäsenttjä itseään kullekin sisar- ja lapsisolmulle niin kauan, että kohdataan solmu joka on tyypiltään tekstielementti (TEXT_NODE) ja jolla ei ole lapsisolmuja. XML-jäsenttjä kerää kaikkien tällaisten solmujen sisällön ja palauttaa ne kun koko merkkijono on käyty läpi.

3.3.5 Flash-prototyypin testaus ja huomiot

Testasimme Flash-prototyyppejä kahdessa eri tilanteessa. Ensimmäistä prototyyppiä testasimme Medialaboratorion järjestämässä Demo Dayssä 18.5.2006.

Yksityiskohtainen prototyypin testaus toteutettiin käyttäjähaastatteluina, jotka pidettiin kolmena eri tilaisuutena Medialaboratorion tiloissa 15.6.2006 – 20.6.2006. Käyttäjähaastatteluissa käytimme Dictrian toista prototyyppiä.

Kumpaankin prototyypin versioon oli toteutettu vapaa tilassa liikkuminen ja navigoiminen. Kummassakin prototyypissä käyttäjä kykeni valitsemaan ja avaamaan yksittäisen tekstialueen. Prototyypit erosivat toisistaan kahden eri toiminnon osalta: Demo dayn prototyyppiin oli toteutettu dynaaminen RSS-syötteen hallinta ja tämän avulla uusia tekstialueita kyettiin lisäämään ympäristöön. Käyttäjätesteissä taas oli lisätty mahdollisuus liikutella sanoja, mutta kaikki sisältö oli staattista. Kumpikaan versio ei kyennyt tallentamaan omaa tilaansa sovelluksen sulkeutuessa.

Prototyyppien avulla huomasimme, että navigaation ja liikkumisen helppous on avainasemassa käytettävyyden kannalta. Tässä vaiheessa vapaa liike oli toteutettu hiirellä toimivaksi, mutta sen aktivointi tapahtui painamalla välilyöntiä näppäimistöllä. Samoin hiiren liikkeen määrää mitattiin hiiren cursorin etäisyytenä sovellusikkunan keskikohtaan, eikä esimerkiksi hiiren liikkeen määränä sitä liikutettaessa. Tämä aiheutti prototyypissä turhaa monimutkaisuutta ja käyttäjän muistikuorman lisäämistä. Vaikka toteutus toi halutunlaisen kelluvan liiketilan Dictriassa liikkumiseen, oli se liian vaikea opittava.

Myös dynaamisen tai staattisen sisällön käyttäminen vaikutti käyttäjien motivaatioon ottaa osaa Dictriaan. Vaikka sisällöllä ei ole vaikutusta visuaaliseen lopputulokseen, käyttäjien mielipiteistä huomasi varsin selvästi sen, kuinka sisällöllä on merkitystä sille miten käyttäjä tuntee olevansa osa yhteisöä. Dynaaminen sisällön hallinta tuntui käyttäjistä mielekkäältä ja motivaatiota ylläpitävältä.

3.3.6 Prototyypin tekniset ongelmat

Prototyypissä ja etenkin käyttäjähaastatteluissa esille tulleet ongelmat liittyivät toiminnallisuuden rajaamiseen. Saimme prototyypistä käyttäjäpalautetta, jota oli vaikea jäsentää, koska se käsitteli osittain ominaisuuksia, jotka olivat vielä toteuttamatta. Tämän lisäksi joidenkin ominaisuuksien osalta ei oltu vielä päätetty, sisällytetäänkö niitä lopulliseen ratkaisuun.

Teknisesti prototyyppi toimi, joskin alakohdassa 3.3.5 esitetyt ongelmat oli otettava huomioon tuotannon seuraavassa vaiheessa. Yhtenä vaikeimpana prototyyppiä koskevana ongelmana pidettiin mekaniikkaa, joka mahdollisti sisällön dynaamisen tuottamisen.

Sanojen toiminnallisen mallin kykenimme prototyypissä simuloimaan manuaalisesti tuottamalla, mutta emme kyenneet toteuttamaan sitä dynaamisesti tuotetusta sisällöstä. Flash tekee helpoksi käsitellä ja luoda tekstialueita, mutta se ei tarjoa mekanismeja syötteen jakamista pienempiin yksiköihin ja niiden dynaamiseen hallintaan. Tämä oli suuri puute prototyypissä ja konseptin toteutuksessa. Prototyypissä oli siten mahdollista lukea dynaamisesti RSS-syötteitä ja esittää ne tekstialueina, mutta tätä sisältöä ei kyetty jakamaan dynaamisiksi objekteiksi joita voitaisiin liikutella.

Dynaamisen objektihallinnan ongelmien lisäksi toinen este tuli ilmi Flashin suorituskyvyssä. Flash käyttää graafisten objektien piirtämiseen vektorigrafiikkaa, joka on visuaalisesti hyvin korkealaatuista, mutta Flashin tuottamana raskasta piirtää. Pieniä määriä graafisia objekteja, kuten muutamia kymmeniä sanoja, käytettäessä tämä ei ole ongelma. Flash kuitenkin hidastui kun sisällön määrää nostettiin tasolle, jolle Dictria oli suunniteltu. Tähän vaikuttaa myös se, ettei Flash käytä grafiikan tuottamiseen grafiikkakiihdytintä, vaan kaikki laskenta tapahtuu tietokoneen keskusyksiköllä.

Sovellusta rasiettiin kokeissa, joissa tekstialueita lisättiin yksi kerrallaan. Testasimme käytettävyyttä jokaisen vaiheen jälkeen. Testeistä oli havaittavissa se, kuinka sovellus

muuttui äärimmäisen raskaaksi jo silloin kun ympäristössä oli noin 1000 sanaa. Tämä määrä riitti noin 5-7 tekstialueen tuottamiseen ennen kuin suorituskyky kärsi niin kovasti, ettei sisällön lisääminen ollut enää mielekäästä. Sovellus hidastui edelleen sanoja interaktiivisiksi objekteiksi muuttaessa. Tämä lisäsi laskennallisen tehon tarvetta, jolloin esitettävän sisällön määrää tuli pudottaa käytettävyyden parantamiseksi.

3.3.7 Prototyypistä tehdyt johtopäätökset

Flash tarjosi alustan joka mahdollisti nopean prototyyppien tekemisen. Sen sisäänrakennetut toiminnot tarjosivat mahdollisuuden nopeaan sisällön tuottamiseen ja graafisten objektien hallintaan, mutta se ei antanut työkaluja niiden tehokkaaseen dynaamisen luontiin, hallintaan eikä visuaaliseen esitykseen.

Prototyyppiä kehitettiin ”Rapid Prototyping”-menetelmällä, minkä avulla kykenimme kehittämään konseptia demovaiheessa yksittäisiä ominaisuuksia vaiheittain luomalla. Käyttjähaastatteluiden perusteella totesimme toiminnallisuuksien rajaamisen tärkeäksi. Prototyypin eri versioilla ja käyttjähaastatteluilla kykenimme muodostamaan toiminnallisen ytimen, jonka pohjalle Dictrian tuli kehittyä. Tämä oli yksi prototyypin parhaiten esille tuomista konkreettisista tuloksista.

Prototyyppien perusteella totesimme Flashin olevan Dictrialle sopimaton kehitysalusta. Skriptikieleen pohjautuva Flash ei ole ilmaisuvoimaltaan tarpeeksi tehokasta ja toiminnallista, jotta Dictrian toiminnot olisi sillä kyetty toteuttamaan. Flash osoittautui myös liian tehottomaksi tuottamaan ja animoimaan sitä määrää objekteja, joita Dictrian oli suunniteltu visualisoivan. Nämä kaksi ominaisuutta olivat kuitenkin oleellisia Dictrian konseptin kannalta.

Prototyypit toimivat hyvinä alustoina toiminnallisuuden luomisessa ja edelleen kehittämisessä. Useat ratkaisumallit, joilla ongelmia oli käyty läpi, olivat hyväksyttäviä kehitysympäristöstä huolimatta. Näiden kokemusten avulla voitiin määrittää toiminnallinen ydin, jonka pohjalle Dictrian seuraava version kyettäisiin toteuttamaan.

3.4 Processing-prototyyppi

Prototyypeistä keräämämme kokemuksen perusteella tulimme siihen johtopäätökseen, ettei Flash ole sopiva kehitysympäristö Dicitrian rakentamiselle. Täten projektin seuraava vaihe oli löytää Flashin korvaava alustariippumaton vaihtoehto, joka mahdollistaisi vuorovaikutteisen, kolmiulotteisen ympäristön luomisen Web-ympäristössä toimivaksi.

3.4.1 Client-vaihtoehdot

Projektin seuraavassa vaiheessa käsitelimme vaihtoehtoja Dicitrian uudeksi kehitysalustaksi. Katsoimme, että ensimmäinen vaihtoehtomme olisi käyttää puhtaasti ohjelmointipohjaista ratkaisua ja lähestyä toteuttamista esimerkiksi Javan tai C++:n kaltaisia ohjelmointikieliä käyttäen. Toinen vaihtoehtomme olisi käyttää jotain jo olemassa olevaa integroitua kehitysympäristöä, jossa osa tarvittavista primitiiveistä ja toiminnoista olisi jo toteutettu.

Asiaan perehdyttyämme totesimme, että projektin toteuttaminen Javan tai C++:n kaltaisilla kielillä olisi projektin aikataululle liian raskas vaihtoehto. Koimme, että parempi vaihtoehto olisi käyttää hieman keveämpää kehitysympäristöä, joka olisi tarkoitettu vuorovaikutteisten visuaalisten mallien tuottamiseen ja webissä jakamiseen.

Edellä mainittujen kriteereiden kautta päädyimme kahteen vaihtoehtoon: Adobe (ent. Macromedia) Directoriin ja beta-tuotantoasteella olevaan Processing-kehitysympäristöön. Director perustuu metaforaan, jossa käyttäjä toimii ohjaajana, joka rakentaa eri medioista kokonaisuuksia.

Kuten Flash, myös Director käyttää skriptikieltä dynaamisen vuorovaikutuksen luomisessa. Se tarjoaa kuitenkin Flashia kehittyneemmät mahdollisuudet aitojen 3D-mallien ja grafiikan tuottamiseksi. Tämän lisäksi se myös käyttää hyväkseen laitteistopohjaista grafiikkakiihdytystä, jos sellainen optio on käytettävissä.

Olimme skeptisiä Directorin suhteen, koska sitä ei ole pitkään aikaan kehitetty ja tuotteen tulevaisuuden epävarmuudesta on pitkään keskusteltu useassa käyttäjäyhteisössä (Sawyer McFarland 2005). Lisäksi Director on menettänyt paljon rooliaan esimerkiksi Flashille, johon on toteutettu aiemmin ainoastaan Directorissa nähtyjä ominaisuuksia ja toimintoja. Täten pelkäsimme, että Director saattaisi olla vanhentunut tuote. Meitä myös epäilytti lähteä käyttämään toista skriptipohjaista ohjelmointikieltä Dictrian toteutuksessa.

Vahvimmaksi vaihtoehdoksi nousi Processing (Processing 2006a). Processing on tarkoitettu kehitysympäristöksi digitaalisten luonnosten tuottamiseen ja sen tavoitteena on olla tehokas, mutta helposti ymmärrettävä ohjelmointikieli. Processing on Javasta johdettu ohjelmointikieli, joka yksinkertaistaa osaa Javan rakenteita luomalla omia tietorakenteita ja luokkia erityisesti hahmotelmien ja visuaalisten tuotantojen tarpeisiin.

Processingin vahvuutena voidaan pitää sitä, että se perustuu Javaan, jonka kautta se kykenee käyttämään osaa Javan tarjoamista luokista, kirjastoista ja rakenteista. Java on korkean tason ohjelmointikieli, jonka ilmaisuvoima on skriptikieliä suurempi. Nämä seikat vahvistivat kuvaa Processingin mahdollisuuksista projektin toteuttamiseen, sillä Processing tukee suoraan kolmiulotteisen grafiikan tekemistä perusrakenteissaan.

Processing tuottaa lopputuloksenaan joko itsenäisiä sovelluksia tai appletteja (Sun Microsystems 2006a). Appletti on Javalla toteutettu sovellus, joka toimii toisen ohjelman sisällä. Tässä tapauksessa appletti toimisi www-selaimessa, mikä tekisi siitä helposti saavutettavan, kunhan käyttäjän koneeseen vain on asennettu Java-laajennus (plugin).

Lopputuloksemme oli siirtää Dictrian tuotanto Processingille. Pitkän harkinnan jälkeen tulimme siihen päätökseen, että Processing antaa laajemmat mahdollisuudet Dictrian kaltaisen kokonaisuuden luomiseksi. Processingin kehittämiseen osallistuu hyvin aktiivinen yhteisö, joten sen ohjelmistotuotannollinen malli sopii hyvin myös Dictrian kontekstiin.

4. Processing-prototyyppi

Kykenimme määrittämään Dictrian ydintoiminnot Flash-prototyypin ja käyttäjähaastattelujen analyyseistä. Processing-vaiheen aloittamisen kannalta tämä oli merkittävää, koska kaikki toiminnot, jotka tuli toteuttaa, olivat tiedossa. Näiden pohjalta pystyimme suunnittelemaan seuraavan prototyypin rakenteen.

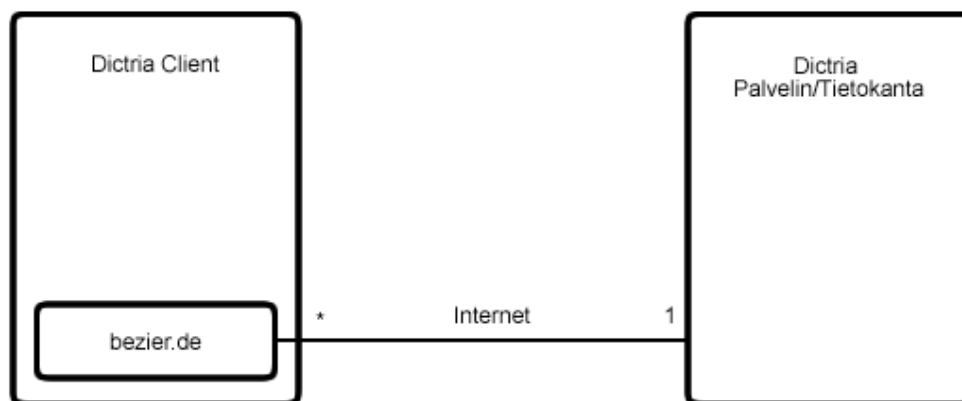
Aloittaessamme Processing-prototyypin toteuttamisen otimme käyttöön niin sanotun lisäyksellisen ohjelmistotuotannon mallin (Incremental Model) (Schach 2002). Lisäyksellisen mallin ideana on viedä ohjelmistokehitystä versioittain eteenpäin. Tämä tarkoittaa sitä, että jokainen versio tuo jonkin uuden ominaisuuden prototyyppiin. Tämä ohjelmistotuotannon malli sopi mielestämme Processing-prototyypin tuotantoon, koska Dictrian vaatimukset ja määritelmät oli jo luotu. Tästä eteenpäin pystyimme jatkamaan kehitysyhteistyötä niin, että toteutimme yhden toiminnon kerrallaan.

4.1 Ohjelmiston rakenne

Dictrian nykyinen rakenne luottaa asiakas-palvelinarkkitehtuuriin (client-server) (Kuva 6). Dictria jakautuu niin sanottuun hybridiasiakasohjelmistoon ja yhteiseen tietokantaan, johon järjestelmän tila ja muutokset tallennetaan (Wikipedia 2006b). Dictria luottaa käyttäjän koneella paikallisesti tapahtuvaan laskentaan ilman, että käyttäjän koneelle tallennetaan mitään tietoa. Tietokanta toimii erillisellä palvelimella ja jakaa asiakkaiden tuottamaa tietoa.

Arkkitehtuurin ideana on, että Dictrian tiedot ovat tallennettuna yhteen yhteiseen tietokantaan, jonka kaikki Dictrian käyttäjät jakavat. Tämän rakenteen kautta voidaan luoda yksinkertainen, mutta toimiva, monen käyttäjän sovellus. Kukin asiakasohjelmisto toimii itsenäisesti, mutta ne hakevat muiden käyttäjien tekemät muutokset tietokannasta ja tallentavat sinne itsenäisesti tekemänsä muutokset. Dictria pitää huolta siitä, että muiden käyttäjien toiminnot esitetään kullekin käyttäjälle. Tämä tapahtuu siten, että ympäristöä päivitetään tasaisin väliajoin. Tällä ratkaisumallilla pyrimme mahdollisimman yksinkertaistettuun asiakaspalvelinarkkitehtuuriin

toteutukseen. Dictriassa asiakasohjelmat toimivat aktiivisina osina ja palvelimen tietokanta vain tallentaa tilan toimien tietovarastona.



Kuva 6: Yksinkertaistettu asiakas-palvelin-malli.

4.2 Visuaaliset elementit

Dictrian keskeisinä elementteinä toimivat sanat ja lauseet, sekä niiden muodostamat visuaaliset kokonaisuudet. Näiden visuaalisten peruselementtien rakentamiseksi oli ensimmäiseksi selvitettävä Processingin tarjoamat ominaisuudet niiden toteuttamiseksi.

4.2.1 Grafiikkaprimitiivit

Työkalut sanojen esittämiseen löytyivät suoraan Processingin ohjelmointirajapinnasta (Processing 2006b). Sanan peruselementti on text-objekti joka ottaa argumenteikseen joko kaksi tai kolmiulotteiset sekä esitettävän sanan. Kullekin tekstielementille voidaan määrittää käytettäväksi mikä tahansa fonttikirjasto, joka on luotu Processingin tarjoamilla työkaluilla.

Sanojen lisäksi peruselementteinä toimivat 3D-maailman ja -käyttöliittymän tekstialueet sekä 2D-käyttöliittymän tekstialueet. Tekstialueet koostuvat suorakulmion muotoon asetetuista viivoista sekä taustasta. Suorakulmio voitiin määrittellä helpoiten käyttämällä QUADS-tyyppistä shape-objektia. Quads on nelikko, nelisivuinen polygoni ja se eroaa suorakulmiosta siten, etteivät sen kärkien kulmat ole sidottu 90 asteeseen (Processing 2006c). Se tarvitsee argumentteinaan polygonin kärkien koordinaatit.

Tekstialueet toteutetaan shape-objekteina, jotka mahdollistavat monimutkaistenkin kappaleiden mallintamisen (Processing 2006d). Kaikki Processingin piirtämät kappaleet ovat shape-objekteja. Nelikoiden käyttö oli projektin kannalta tarpeellista, sillä shape-objektit voidaan lisäksi teksturoida. Tällä tavoin tehtiin esimerkiksi Dictriaan suunniteltujen maamerkkien toteutus, joiden tarkoituksena on helpottaa käyttäjän navigaatiota kolmiulotteisessa ympäristössä.

Muita käytännöllisiä primitiivejä olivat shape-objekteihin vaikuttavat toiminnot kuten fill ja strokeWeight. StrokeWeight määrittelee shape-objektien viivojen paksuudet ja fill shape-objektien värejä. Näitä toimintoja oikeassa piirtojärjestyksessä käyttämällä kyettiin muuntelemaan eri objektien visuaalista ilmettä. Yhdessä muotojen kanssa saatiin toteutettua kaikki Dictrian visuaaliset elementit.

4.2.2 Sana

Loimme sanasta oman luokan, sillä niistä tuli tehdä dynaamisia objekteja, joita voitaisiin liikutella. Sanaluokka pitää sisällään merkkijonon jota se edustaa, sekä vektorin, joka määrittää sanan sijainnin. Kukin sana pitää sisällään yksilöllisen tunnuksen, id:n, jotta ne kyetään erottamaan toisistaan sekä tietokannassa että ympäristössä.

Sanaluokan toiminnallisuudet ovat yksinkertaiset. Luokalle muodostettiin alustaja, joka alustaa sanan sen koordinaateilla ja merkkijonolla jota objekti kuvaa. Lisäksi luokan sisään on luotu animaatiotoiminnot. Animaatiotoiminnot mahdollistavat sanojen animaation, kun jokin käyttäjä on siirtänyt sanaa.

4.2.3 Tekstialue

Tekstialue luodaan visuaalisesti nelikkona (QUADS) ja sen tausta täytetään yksinkertaisella valkoisella taustavärillä fill-objektia käyttäen. Visuaalisen rakenteen lisäksi tekstialueeseen on toteutettu vektori määrittämään tekstialueen sijaintia, sekä yksinkertainen taulukko sana-muotoisille objekteille. Kukin tekstialueen sana-taulukko pitää sisällään kaikki ne sanaobjektit, jotka kuuluvat yksittäiseen tekstialueeseen. Tekstialuetta piirrettäessä piirretään myös kaikki siihen kuuluvat sanat.

Kunkin tekstialueen piirtäminen tapahtuu Processingin tarjoamaa matriisipinoa (Hearn 1994) hyväksi käyttämällä. Aina uutta tekstialuetta piirrettäessä kutsutaan pushMatrix-operaattoria, joka tallentaa senhetkisen transformaatiomatriisin matriisipinon. Tämä mahdollistaa sen hetkisen koordinaatiston tallentamisen siksi aikaa, kun yksittäistä tekstialuetta piirretään. Kun koordinaatiston tila on tallennettu matriisipinon, suoritetaan translaatio (siirto koordinaatiston akseleiden suhteen) tekstialueen koordinaatteihin. Tämä mahdollistaa sanojen ja muiden tekstialueiden piirtämisen paikallisia koordinaatteja käyttäen. Kun kaikki objektit on piirretty, kutsutaan popMatrix-toimintoa joka palauttaa koordinaatiston edelliseen tallennettuun tilaan.

Tämä mekaniikka oli toteutettava, koska 3D-tilassa toimivat tekstialueet jakavat samat sana-objektit, joita käytetään myös 2D-käyttöliittymässä. Saadaksemme sanat toimimaan dynaamisesti molemmissa koordinaatistoissa, luodaan kunkin objektin piirtämisessä translaatio tekstialueen koordinaatteihin. Tällä tavalla saadaan sanoille paikalliset koordinaatistot. Vastaava toiminto suoritetaan myös 2D-käyttöliittymää piirrettäessä. Tämä takaa sen, että kolmi- ja kaksiulotteisia tiloja jakavat objektit piirtyvät aina oikein. Ympäristön tila tallennetaan suorittamalla kutsu matriisipinon. Translaatiot vaikuttavat tämän jälkeen ainoastaan sillä hetkellä piirrettävään tekstialueeseen. Sanojen koordinaatit ovat siis määriteltyjä translaatiolla nollattuun tilaan, missä ne voidaan isäntäobjektista riippumattomasti piirtää aina samaa koordinaatistoa käyttäen.

4.3 Tilan, liikkumisen ja paikannuksen toteutus

Käyttäjän ja objektien sijoitus kolmiulotteiseen ympäristöön toteutettiin samoja Flash-prototyypin yksiköitä käyttäen (alakohta 3.3.2). Kappaleiden ja käyttäjän sijainti tallennetaan x-, y- ja z-koordinaatein. Kolmiulotteisia koordinaatteja käsittelevät mekanismit oli toteutettava eri tavalla, koska Processing-prototyyppi on aidosti kolmiulotteinen. Tilan rakenne toteutettiin vektoriavaruutena, jossa objektien sijainti ja niiden muutokset toteutettiin vektorilaskutoimituksina.

4.3.1 Vektori

Vektorilla tarkoitetaan elementtiä vektoriavaruudessa (Weisstein 2006). Dictriassa Vektori toteutetaan reaalilukujen kolmikkona, joka määrittelee objektien sijainnin avaruudessa. Vektoreiden kolmikot luodaan x-, y- ja z-koordinaateista ja ne rakennettiin omina vektoriluokan objekteina, jotka sisältävät myös toiminnot niiden laskemiseksi ja muuntelemiseksi

Vektori-luokka on tärkeä rakenne lineaarisen, kolmiulotteisen tilan rakentamisessa sekä etenkin objektien mallintamisessa avaruudessa. Dictrian kannalta vektori-luokka on eräs perusluokista, sillä vektorin avulla on mahdollista piirtää jokainen visuaalinen objekti oikealle paikalleen kolmiulotteisessa tilassa.

4.3.2 Kvaternio

Kvaterniot ovat laajennuksia kompleksiluvuille ja Dictriassa niitä käytetään vektorien laajennuksena. Ne muodostavat normalisoidun vektoriavaruuden ja ovat käytännöllisiä mallinnettaessa kolmiulotteista orientaatiota (kappaleen kallistuskulma koordinaattiakseleiden suhteen), tai kolmiulotteisten kappaleiden kiertymistä (kiertymistä koordinaattiakseleiden suhteen). Dictriassa kvaternioita käytetään rotaation kolmessa ulottuvuudessa laskemiseen. Niitä tarvitaan käyttäjän vapaan katseen luomisessa ja liikkumisessa.

Kvaterniot luotiin nelikkoina, jossa kukin kvaternio-objekti sisältää neljä skalaarisuuretta ($w = (x,y,z,w)$). Näitä suureita voidaan lisätä ja kertoa yksiköinä kuten normaaleja algebrallisia yksiköitä. Dictriassa rotaatio tapahtuu katselukulmaa pyöritettäessä, mikä tarkoittaa katselusuunnan kiertämistä käyttäjän sijainnin ympäri. Tämä ei onnistu ainoastaan vektoreita käyttämällä, vaan se on suoritettava kvaternioiden avulla.

4.3.3 Liike ja navigaatio

Jotta käyttäjän navigaatio- ja liiketoiminnot saatiin toteutettua yhtenäisesti, keskitettiin niiden toiminnot yhdeksi kokonaisuudeksi, kameraksi (Camera) (vrt. alakohta 3.3.2). Kameran tulee yksinkertaisimmillaan muistaa käyttäjän sijainti ja orientaatio, sekä sisältää toiminnot näiden muuttamiseksi. Näiden toimintojen lisäksi kameran tulee luoda toiminnot liikkeen ja vapaan katseen toteuttamiseksi.

Käyttäjän sijainti luodaan vektorina. Lisäksi orientaatiota varten kameraan luotiin katselusuunta ja ylös-vektori (up-vector). Ylös-vektori kuvaa suuntaa suoraan ylöspäin käyttäjän katselusuuntaan nähden. Näitä vektoreita tarvitaan liikkeen ja vapaan katseen luomiseksi ja orientaation määrittämiseksi.

Kamera luodaan antamalla sille käyttäjän sijainti, katselusuunta ja ylösvektori. Käyttäjän liikkuminen toteutetaan vektorilaskutoimituksina muuttelemalla käyttäjän sijaintia ja katselusuuntaa. Idea on kaikissa sama: käyttäjän sijainti ja katselusuunta ovat pisteitä vektoriaruudessa. Näiden kahden pisteen avulla saadaan aikaan vektori, joka osoittaa käyttäjän sijainnista suoraan hänen katselusuuntaansa. Liikkeen toteutus on näiden vektoreiden muokkaamista yksinkertaisia vektorilaskutoimituksia käyttäen.

Liikkeet oli yksinkertaista toteuttaa normaaleina vektorilaskutoimituksina, mutta rotaation toteuttaminen hiiren avulla oli monimutkaisempaa. Kvaternioiden käyttö on yksi tapa laskea kyseisiä kolmiulotteisia rotaatioita. Se on lähestymistapa jota Dictriassa käytetään. Neliulotteinen kvaternio kykenee itse esittämään kameran orientaation kolmiulotteisessa avaruudessa. Dictriassa käytetään kvaternioita

ainoastaan rotaatioiden laskemisessa, jolloin translaatiot suoritetaan vektoreille erikseen.

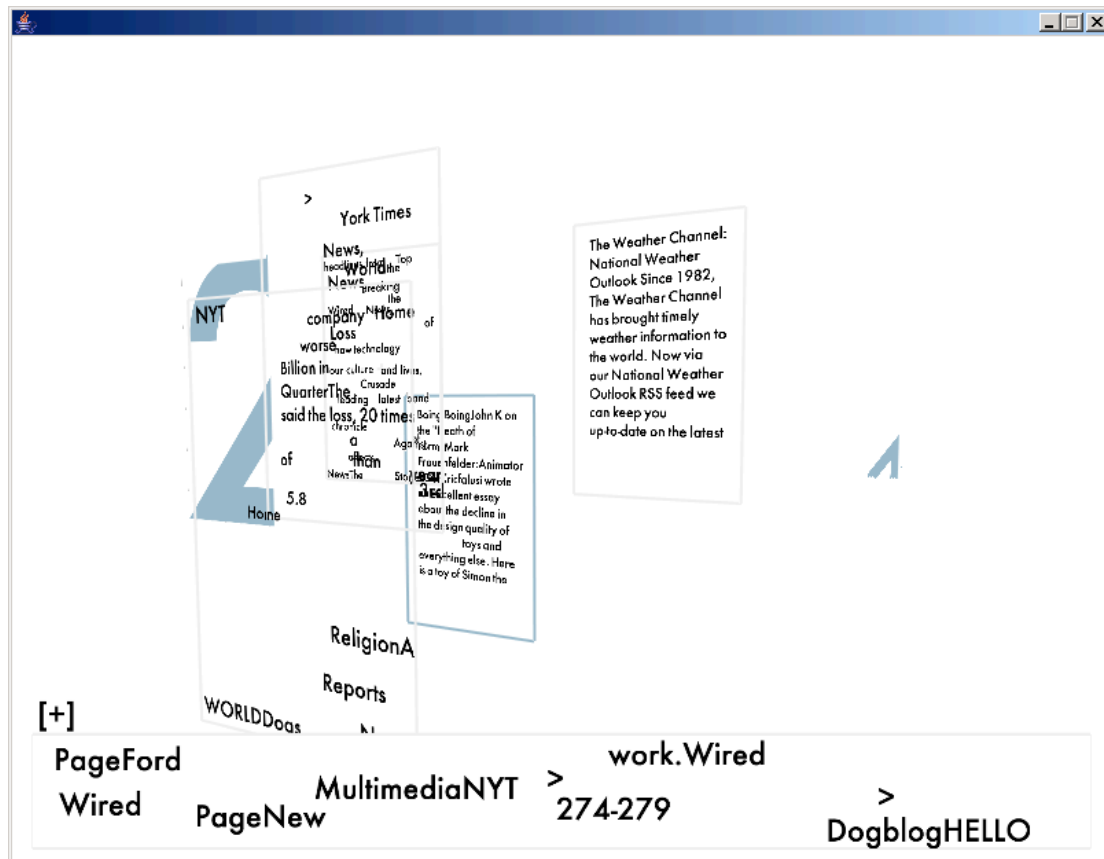
4.4 Käyttöliittymä

Kuten Flash-prototyypissäkin, on myös Processingilla tuotettu Dictria jaettu toiminnallisesti kaksi- ja kolmiulotteiseen käyttöliittymään. Koska Processingilla tuotettu Dictria on aidosti kolmiulotteinen, osa vuorovaikutusmekaniikoista on jouduttu toteuttamaan eri tavalla.

4.4.1 3D-käyttöliittymä

Kun 3D-vuorovaikutusta rakennettiin, ydintoiminto oli tunnistaa milloin hiiri on jonkin tekstialueen päällä. Tämän lisäksi tuli erottaa yksittäinen, lähimpänä käyttäjää oleva tekstialue hiiren ollessa usean tekstialueen päällä yhtä aikaa.

3D-käyttöliittymään rakennettiin ensimmäiseksi tunnistus tilanteelle jossa hiiri on tekstialueen päällä. Tämä toteutettiin projisoimalla tekstialueen reunat näytön muodostamaan kaksiulotteiseen tasoon. Jos hiiren kursori on projisoidun alueen sisällä, on hiiri yksittäisen tekstialueen yllä. Lisäksi tuli toteuttaa yksinkertainen syvyyslajittelu, joka tunnisti käyttäjää lähinnä olevan tekstialueen niitä piirrettäessä. Saadun tiedon avulla voidaan korostaa ja valita yksittäisiä tekstialueita, jotka ovat 3D-käyttöliittymän ydintoiminnot (Kuva 7). Tekstialueitten valitsemisen lisäksi kolmiulotteiseen käyttöliittymään kuuluu myös oleellisesti navigoinnin ja liikkumisen toteutus, joka on selitetty alakohdassa 4.3.3.

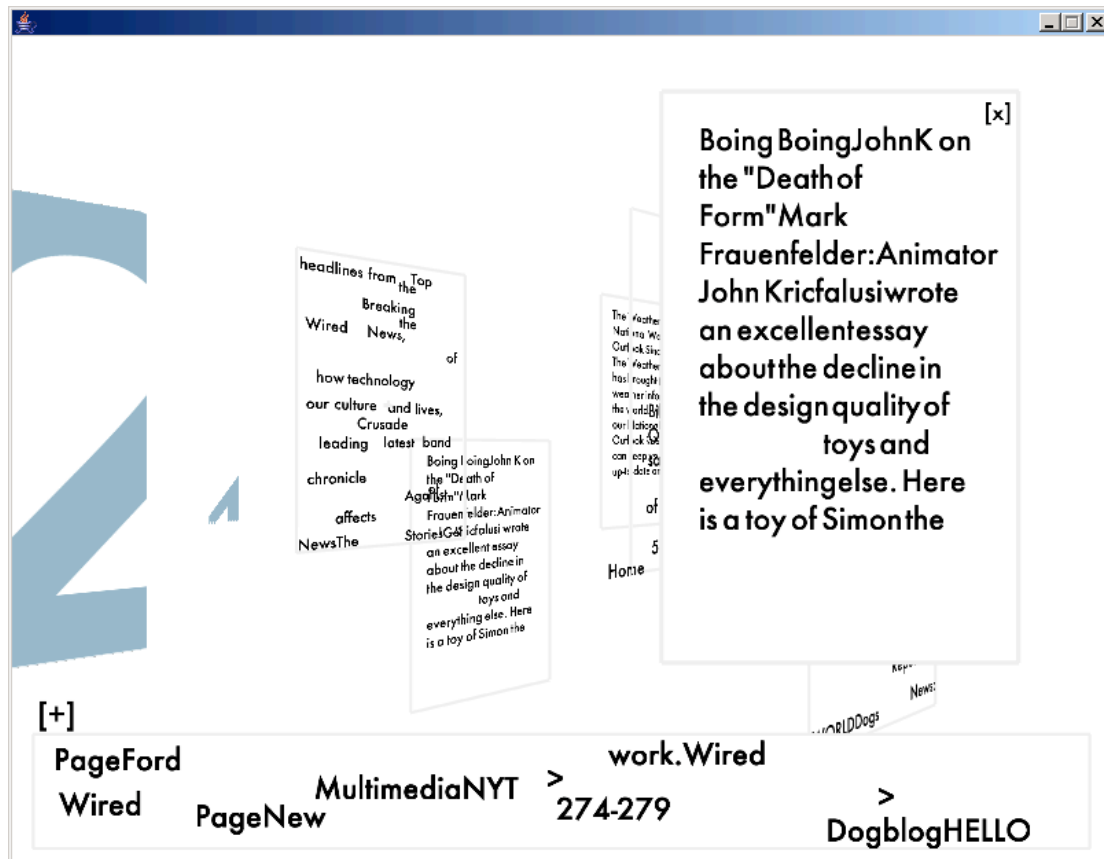


Kuva 7: 3D-käyttöliittymä jossa yksi tekstialue on korostettu.

4.4.2 2D-käyttöliittymä

Siinä missä 3D-käyttöliittymä keskittyy tilassa navigointiin ja tekstialueiden käyttöön, on sanojen kanssa vuorovaikuttaminen toteutettu 2D-käyttöliittymänä.

Kaksiulotteinen käyttöliittymätila on jaettu kahteen osaan: ikkunan alareunassa olevaan yhteiseen sana-alueeseen, ja avattuun tekstialueeseen (Kuva 8). Yhteinen tekstialue on aina näkyvässä. Tämä esittää aluetta johon voi kerätä yksittäisiä sanoja tekstialueista ja siirtää sanoja sen kautta toisiin tekstialueisiin. Yhteinen tekstialue muuttuu aktiiviseksi kun jokin tekstialue avataan muokattavaksi.



Kuva 8: 2D-käyttöliittymä

Sanojen liikuttelussa käytetään samaa mekanismia kuin tunnistettaessa hiiren koordinaatteja tekstialueisiin nähden (alakohta 4.4.1). Jos tekstialue on avoinna muokkaamista varten ja käyttäjä klikkaa hiirtä, on tunnistettava onko kursori yhdenkään sanan yllä. Mekaniikka on sama kuin tekstialueen tunnistamisessa, mutta tällä kertaa projektio tehdään sanojen ääriiviivojen muodostamista suorakulmioista. Jos hiiri on jonkun sanan päällä ja käyttäjä alkaa raahaamaan sanaa hiirellä muutetaan sanan koordinaatteja vastaavan muutoksen verran kunnes käyttäjä vapauttaa hiiren painikkeen.

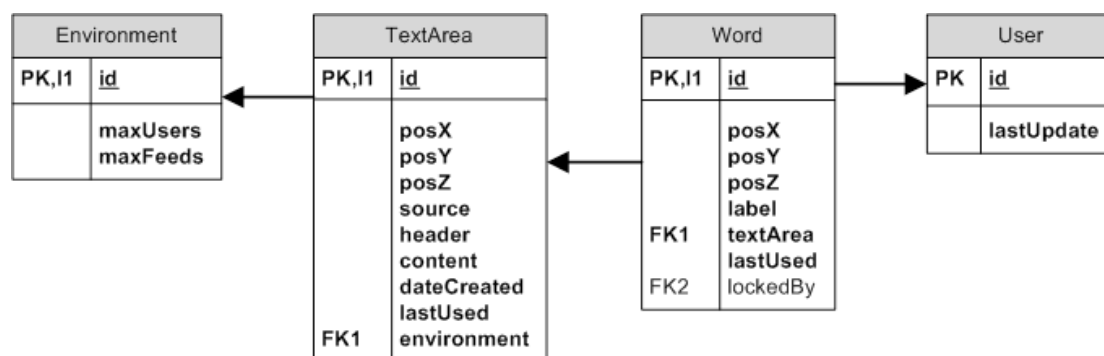
2D-käyttöliittymä on toteutukseltaan 3D-käyttöliittymän alaluokka. Se luodaan yhteisestä tekstialueesta ja yhdestä editoitavaksi avatusta tekstialueesta, joka piirretään aivan kuten muutkin tekstialueet. Ideana on vain kiertää nämä komponentit kolmiulotteisessa tilassa siten, että ne ovat aina kohtisuorassa käyttäjän katselusuuntaan nähden.

4.5 Palvelin ja tietokanta

Dictrian prototyypivaiheessa yksinkertainen ratkaisu usean käyttäjän -toiminnon luomiseksi oli tärkeää. Yksinkertaisin ratkaisu on käyttää tietokantaa palvelimena datan säilömiseen ja muutoksien tallentamiseen. Tämän ratkaisumallin käyttäminen painottaa asiakasohjelmistojen aktiivisuutta, jolloin ne kommunikoivat keskenään epäsuorasti tietokannan kautta.

Tietokanta toteutettiin MySQL:lla, joka on avoimeen lähdekoodiin perustuva tietokantaohjelmisto ja siten ilmainen käyttää ei-kaupallisissa projekteissa (Mysql 2006). Processingille on toteutettu bezier.de-kirjasto (Florien Jenett 2005), joka oli helppo integroida Dictriaan Mysql-tietokannan kanssa kommunikoimiseksi. Tällä tavoin kaikki tietokantakutsut kyettiin tekemään integroituilla toiminnoilla, mikä helpotti usean käyttäjän toiminnon luomista ja sisällyttämistä järjestelmään.

Tietokanta (Kuva 9) rakentuu neljästä taulusta: *Environment*, *TextArea*, *Word* ja *User*. *Environment* pitää yllä eri Dictrian instansseja. Tämä taulu on toteutettu siltä varalta että tiedon määrä kasvaa niin isoksi että ympäristöjä tulee jakaa itsenäisiksi kokonaisuuksiksi.



Kuva 9: Tietokannan rakenne

TextArea pitää sisällään kaikkien tekstialueiden tiedot. Kukaan tekstialue kuuluu yhteen ympäristöön kerrallaan ja taulu pitää sisällään tiedot kaikki tiedot yksittäisistä tekstialueista. Tauluun tallennetaan kunkin tekstialueen sijainti kolmiulotteisessa

avaruudessa, sen alkuperäinen teksti, url josta se on haettu sekä aikaleimat jolloin se on luotu ja milloin sitä on viimeksi käytetty. Osa *TextArea*n kentistä on suunniteltu tulevaisuuden toimintoja varten (source, header ja content), mutta rakenteet eivät haittaa lainkaan nykyisen prototyypin käyttöä. Aikaleimoja käytetään yksittäisten asiakkaiden päivittäessä tietoaan. Tietokanta päivittää aikaleimoja automaattisesti tietokannan joka kerta kun kenttään tehdään muutoksia, kuten lisäyksiä tai päivityksiä.

Word toimii identtisesti *TextArea*:n kanssa ja kuten itse *Dictri*assakin kukin sana kuuluu yhteen tekstialueeseen. Muuten taulu pitää sisällään tiedot sanan sijainnista, merkkijonosta jota se esittää, sekä milloin sitä on viimeksi käytetty (aikaleima). Lisäksi jokainen sana voidaan lukita yhden käyttäjälle. Tämä toiminnallisuus tehtiin estämään yhden sanan liikuttamista yhtä aikaa kahden käyttäjän toimesta. *Dictria* pitää huolta siitä, että sanaan tarttuessaan järjestelmä tarkistaa onko sana lukittu toiselle käyttäjälle. Jos sana on vapaa voi käyttäjä liikuttaa sitä, muuten ei.

User-taulua käytetään yksilöimään käyttäjä tietokantatoimintoja varten. *Dictrian* appletti saa käynnistyessään parametrina php-skriptin taltioiman sessio-id:n. Tämä arvo tallennetaan tietokantaan, jotta yksittäiset käyttäjät voidaan erotella, mutta muuten järjestelmä ei yksilöi käyttäjiä.

4.5.1 Tietokantayhteys

DBConnection on luokkarakenne, joka hoitaa sisällön synkronoimisen tietokannan kanssa, sekä alustaa ja ylläpitää asiakasohjelman tietoja. Sovellusta käynnistettäessä *DBConnection* alustaa ympäristön tilan. *DBConnection* hakee kaikki tekstialueet, joita ympäristössä on ja luo uudet tekstialue-objektit kullekin tietokannassa olevalle tekstialueelle. Tämän jälkeen jokaiselle luodulle tekstialueelle haetaan kunkin tekstialueen sisältämät sanat, joista luodut objektit asetetaan tekstialueiden sisään.

Alustettuaan ympäristön *DBConnection* pitää huolta käytönaikaisista tietokantakutsuista sekä tiedon päivittämisestä. *Dictrian* usean käyttäjän ominaisuus perustuu siihen, että kaikki muutokset tallennetaan tietokantaan, josta muut käyttäjät

voivat havaita muutoksen. Tämän tapahtuessa kukin asiakasohjelma tallentaa muutoksen tietorakenteisiinsa ja animoi muutoksen käyttäjälle.

Yksittäiset asiakkaat päivittävät tietoaan ylläpitämällä itsessään aikaleimaa (timestamp) siitä hetkestä, jolloin ne ovat viimeksi suorittaneet tietokantahaun päivittääkseen tietoaan. Tietoa ylläpidetään hakemalla tietokannasta kaikkia niitä sanoja ja tekstialueita, joita on muutetta aikaleiman jälkeen. Tätä toimintoa ajetaan kerran sekunnissa varmistaen että kunkin käyttäjän tila on ajan tasalla.

Bezier.de-kirjasto käyttää http-kutsuja tietokantakyselyissään. Koska kirjasto perustuu Javan omaan JDBC-komponentin käyttöön, varmistaa se hakujen lopputuloksen eikä tietoa pitäisi jäädä välittämättä asiakasohjelman ja tietokannan välillä. Dictrian alustamisen jälkeen ei myöskään siirry niin isoa määrää tietoa, että tiheästä päivittämisestä olisi ongelmia.

DBConnection sisältää myös mekanismit tiedon päivittämiseen Dictrian sisällä, mikä tarkoittaa päivitettyjen sanojen etsimistä tekstialueiden sisältä ja niiden siirtämistä toisiin tekstialueisiin tai sanojen arvojen muuttamista. Näitä toimintoja tarvitaan jos tieto on muuttunut. DBConnection-luokka toimii rajapintana asiakasohjelman muiden luokkien ja tietokannan välillä, reitittäen muiden toimintojen välittämät tilamuutokset tietokantaan.

4.6 Ääni

Dictriaan on alusta alkaen suunniteltu käytettävän ääntä, mutta tämä toiminto toteutettiin vasta Processing-prototyypin. Dictrian ääniympäristö koostuu äänitekstuurista ja tiettyihin toimintoihin sijoitetuista pisteäänistä. Äänisuunnittelun idea oli luoda äänistä kokonaisuutta tukeva ja sen tilallista vaikutelmaa vahvistava tekijä. Lisäksi äänet suunniteltiin tukemaan ja havainnollistamaan toimintoja, joita käyttäjät tilassa tekevät.

Pisteäänit aktivoituvat tietyistä käyttäjien toiminnoista, kuten uuden tekstialueen lisäämisestä tai sanan siirtämisestä. Samoin käyttäjien omia toimintoja, kuten sanojen

siirtelyä voidaan korostaa pisteäänillä. Pisteäänien tarkoituksena on tukea itse äänitekstuuria, mutta tehdä se hienovaraisella tavalla, jotta ne eivät veisi liiallista huomiota itse visuaalisilta tapahtumilta. Tavoitteenamme oli luoda ääniavaruudesta tilallista tunnetta vahvistava tekijä, joka ei olisi pääroolissa, vaan tukemassa visuaalisia tapahtumia.

Äänitoiminnot on koottu yhteen Sound-luokkaan ja ne käyttävät hyväkseen Krister Olssonin luomaa Ess-kirjastoa (Olsson 2006). Toteutus on jaettu tekstuuriäänille ja pisteäänille jaettuun kokonaisuuteen. Tekstuuriäänille on varattu yksi äänikanava joka luodaan sovellusta käynnistettäessä. Tekstuuriääni koostuu erillisistä äänitiedostoista, jotka on suunniteltu toimimaan yhteen. Niiden toisto päätettiin toteuttaa siten, että erilliset äänet soivat päättymättömänä silmukkana. Yhden äänen loppuessa väliin jää aina pieni tauko, joka vaihtelee 10-15 sekunnin välillä. Tämän jälkeen valitaan jälleen yksi äänitiedosto soitettavaksi.

Myös pisteäänillä on oma kanavansa. Kullekin pisteääntä luovalle toiminnolle on määritelty prioriteetti, joka tarkastetaan ääntä soittaessa. Tällä haluttiin korostaa eri toimintojen tärkeyttä pisteääniä kutsuttaessa. Korkeamman prioriteetin pisteääntä ei keskeytetä, mutta se saa pisteäänelle varatun kanavan käyttöönsä aina kun sitä kutsutaan. Tämän ratkaisun taustalla on myös tavoite olla tukkimatta äänimaisemaa liiallisilla pisteäänien soitolla ja pitää ne eräänlaisina houkuttimina, jotka aktivoituvat muiden käyttäjien toiminnoista.

4.7 Processing prototyypin ongelmakohdat

Processing-prototyyppiä rakentaessa oli kaksi kohtaa jotka tuottivat ongelmia: uuden tekstialueen lisääminen ja tilan renderöinti. Uuden tekstialueen lisäämisessä ongelma kiteytyi yksittäisen komponentin puuttumiseen, kun taas renderöinnin ongelmia ei kyetty kokonaan poistamaan.

4.7.1 Uuden tekstialueen lisääminen

Dictrian ydintoimintoihin kuului uuden syötteen tuominen ympäristöön. Processing-demon toiminto päätettiin toteuttaa saman mekaniikan mukaan kuin Flash-prototyypissä, joka on selitetty alakohdassa 3.3.4. RSS-syötteen lisääminen ympäristöön on suunniteltu toimimaan tekstialueen kokoisessa tilassa, joka toimisi osana kaksiulotteista käyttöliittymää. Komponentti koostuu tekstialueen kaltaisesta graafisesta osasta, johon lisätään yksinkertainen tekstikenttä käyttäjän syötettä varten sekä nappi, joka aloittaa tekstin hakemisen.

Tässä vaiheessa kohtasimme ongelman Processingin toteutuksessa. Processing on suunniteltu ensisijaisesti visuaalisten tuotosten ja hahmotelmien luomiseen, eikä siihen toistaiseksi ole olemassa minkäänlaisia kirjastoja käyttöliittymäkomponenttien tekemiseen. Processing ei käytännössä tarjoa mitään menetelmiä luoda komponenttia, johon käyttäjä voisi kirjoittaa RSS-syötteen osoitteen. Yritimme ratkaista ongelman kahdella tavalla.

Ensimmäiseksi yritimme toteuttaa tekstisyötön käyttämällä Javan omia Swing- ja AWT-kirjastoja puuttuvien käyttöliittymäkomponenttien toteuttamiseksi (Sun Microsystems 2006b, Sun Microsystems 2006c). Näiden kirjastojen textinput-komponentin käyttö toimi 2D- muttei 3D-tilassa. Jos Dictriaa renderöitiin mitenkään muuten kuin kaksiulotteisessa tilassa aiheuttivat Javan käyttöliittymäkomponentit ruudun välkkymistä, sekä haittasivat muiden objektien piirtämistä. Syyksi löytyi ongelma, jota kutsutaan raskaiden ja kevyiden komponenttien sekoittamiseksi (Sun Microsystems 2006).

Javan käyttöliittymäkomponentit toimivat joko toteuttamalla natiivit menetelmät objektien luomiseen sovelluksen ikkunassa tai lainaamalla hallintamenetelmät objekteja kutsuvilta komponenteilta. Tämä on jako raskaiden ja kevyiden komponenttien välillä. Sekoittamalla raskaita ja kevyitä komponentteja voi syntyä ristiriitoja, jotka aiheuttavat virheitä sovelluksen toiminnassa ja piirtämisessä. Sekä AWT että Swing aiheuttavat näitä ristiriitoja Processingin kanssa (Processing.org 2006e), joten niiden käyttäminen oli sovelluksessamme mahdotonta.

Toinen vaihtoehto oli käyttää jonkin kolmannen osapuolen kehittämää käyttöliittymäkomponenttikirjastoa, mutta epäonneksemme toistaiseksi ei ole kehitetty yhtään toimivaa käyttöliittymäkirjastoa joka olisi toiminut kolmiulotteisessa tilassa. Tekstisyötteen lukemiseksi Processingin 3D-ympäristössä ei toistaiseksi ole yhtään menetelmää.

Ratkaisimme ongelman tässä vaiheessa luomalla tekstialueiden luomiselle täysin erillisen ikkunan johon kykenimme asettamaan yhden AWT-komponentin RSS-osoitteen syöttämiseksi. Tämä sovellus toimii itsenäisenä applettina ja sitä kutsutaan Dictriasta url:n kautta uuteen selainikkunaan. Tähän luokkaan luotiin yksinkertainen tietokantayhteys sekä XML-jäsentäjä, joka toimii samanlaisella toteutuksella kuin Flashissa (alakohta 3.3.4). XML-toimintojen toteuttamista varten käytettiin Christian Riekoff:n kehittämää proXML-kirjastoa (Riekoff 2006), joka mahdollisti XML-syötteen lukemisen ja käsittelyn Processingissa.

4.7.2 Renderöinti

Sovelluskehityksessä oli päätettävä myös mitä piirrottilaa käyttämällä Dictria toteutettaisiin. Processing mahdollistaa kolmiulotteisten sovellusten piirtämisen kahden eri ohjelmistorajapinnan kautta: Processingin oman P3D- tai OpenGL-tilan.

P3D on Processingin oma kevyt ja tehokas ohjelmointirajapinta, jonka ensisijaisena tarkoituksena on tuottaa nopeasti piirrettävää 3D-grafiikkaa web-ympäristöön. Se luottaa tehokkuuteen, uhraaten hieman kuvanlaatua. Kuten Processing itsekin, on P3D:n kehitys edelleen beta-vaiheessa, mikä näkyy paikoitellen vääristyneinä tekstuureina ja virheinä muotoja (shapes) piirrettäessä (Processing.org 2003). OpenGL on Silicon Graphicsin luoma standardoitu ohjelmointirajapinta kolmiulotteisen grafiikan tuottamiseen (OpenGL 2006). Se on P3D:tä hieman raskaampi tuottaa, mutta se tarjoaa paremman kuvanlaadun objekteja piirrettäessä.

Päädymme käyttämään P3D:tä, koska sen luoma grafiikka on kevyempi tuottaa ja se toimii vakaammin eri laitteistoalustoilla. Vaikka OpenGL on standardi, on sen

toiminta web-ympäristössä hyvin paljon riippuvaista laitteistosta ja ajuriversioista (Processing.org 2006). OpenGL on virallisesti tuettu Processingin tuottamissa appleteissa vasta 1.10.2006 julkaistusta 0116-versiosta lähtien ja testiversioiden perusteella sen toiminta on epävakaampaa eri alustoilla. OpenGL tarjoaa visuaalisesti paremman lopputuloksen, mutta katsoimme, että vakaus ja kevyempi sovellus ovat parempia vaihtoehtoja tämän prototyypin kannalta.

5. Yhteenveto

Olemme yhdessä vuodessa muodostaneet hahmotelmien kaltaisista lähtökohdista toimivan prototyypin ja konseptin. Tähän pisteeseen olemme päässeet hyödyntämällä erilaisia taustojamme ja työstämällä ideoita aihealueista, joista olemme alusta alkaen olleet kiinnostuneita. Projektin aikana olen oppinut ymmärtämään kontekstia, josta työmme hakee inspiraatiota. Näiden seikkojen perusteella olemme päässeet tavoitteeseemme ja saaneet aikaiseksi teoksen, joka täyttää alkuperäiset vaatimukset, joka on toimiva ja sopii aihepiiriin johon sen suunnittelimme.

5.1 Dictrian nykytila

Dictria on tällä hetkellä toimiva prototyyppi. Siihen on toteutettu kaikki konseptin kannalta oleelliset toiminnallisuudet ja visuaaliset kokonaisuudet. Ainoa kokonaisuudesta erotettu toiminto on uuden tekstialueen lisääminen, joka teknisten rajoitusten vuoksi on jouduttu luomaan omaksi sovellukseksi (appletiksi). Nykyinen versio on mahdollista laittaa verkossa julkaistavaksi ja sen toimintoja on testattu myös useiden yhtäaikaisten käyttäjien tapauksessa.

5.1.1 Rakenne

Dictrian rakenne kiteytyy sen asiakasohjelmaan, johon on toteutettu mahdollisimman suuri osa kaikista ominaisuuksista. Prototyypin toiminnot on rakennettu keskitetysti Processing-kehitysympäristössä ja sisällytetty lopputuotteeseen. Olen pyrkinyt hyödyntämään mahdollisimman paljon Processingin tarjoamia kirjastoja ja käyttämään Processing-yhteisön tarjoamia avoimen lähdekoodin ratkaisumalleja toteutuksessa.

Olemme pyrkineet toteuttamaan sen ohjelmistorakenteet mahdollisimman yksinkertaisesti sekä jakamaan sen toiminnot mahdollisimman pieniin osiin, alirakenteisiin ja -toimintoihin. Samalla tavoin olemme yrittäneet välttää turhia toistolauseita. Dictrian ohjelmakoodi ei ole täysin optimoitu, koska projekti on

edelleen prototyypivaiheessa, mutta toteutusvaiheessa on pyritty noudattamaan näitä menetelmiä.

5.1.2 Toiminnallisuus

Nykyisessä prototyypin versiossa käyttäjä pystyy liikkumaan ja navigoimaan kolmiulotteisessa ympäristössä sekä olemaan vuorovaikutuksessa ympäristön, ja siten muiden käyttäjien, kanssa. Kaikki tilan elementit ovat aktiivisia ja toimivia. Käyttäjä voi valita tekstialueita editoitavaksi ja siirrellä sanoja sekä tekstialueiden sisällä että yhteisen tekstialueen välillä. Lisäksi käyttäjä voi tuoda tilaan uutta sisältöä uusia syötteitä tekstialueiksi nimeämällä.

5.2 Projektista opitut asiat

Projekti piti mielestäni sisällään useita suuria kokonaisuuksia joista jo itsessään olisi saanut kokonaisen projektin. Olemme työstäneet konseptin lähtökohdasta, jossa meillä oli ainoastaan kahden henkilön yhteinen mielenkiinto lopputyön aihetta kohtaan. Tämän lisäksi olemme tehneet tästä konseptista toimivan prototyypin, joka todentaa konseptin toimivaksi kahdessa erilaisessa kehitysympäristössä. Lisäksi olemme käyttäneet paljon resursseja oikean alustan löytämiselle. Projektin rajaaminen yhteen näistä kokonaisuuksista olisi ollut yhteiseksi kokonaisuudeksi liian suppea, mutta lopputuloksen vuoksi olemme vieneet projektin läpi isoja asiakokonaisuuksia ratkoen.

Vaikka projektin luonteelle ja etenkin työskentelytavoillemme sopikin konseptin kehitys toiminnoittain mallintamalla, syntyi tästä työskentelymallista lisäkuormaa itse projektin läpiviemiselle. Toisaalta konseptia olisi ollut mahdotonta kehittää ilman näiden kokonaisuuksien kuljettamista rinnakkain. Kehitimme konseptia ja DICTRIAN toiminnallisuuksia aina viimeiseen Flash-prototyyppiin saakka. Olemme vieneet projektin läpi kuljettamalla rinnakkain sovelluskehitystä ja konseptisuunnittelua, samalla tehden taustatyötä. Näitä työvaiheita olisi voitu porrastaa tehokkaammin, mutta kokemuksemme vastaavista toimintamalleista oli rajoittunutta. Emme olisi välttämättä päässeet nykyiseen lopputulokseen ilman näitä työvaiheita. Ohjauksen

hakeminen heti projektin alkuvaiheessa olisi voinut ratkaista tämän ongelman.

Ehkä selkein asia, jonka projektista opin, oli tavoitteiden merkitys sekä niiden asettamisen ja kirjaamisen tärkeys jo projektin alussa. Usea työvaihe olisi voinut sujua tehokkaammin, jos olisin voinut verrata jokaista työversiota projektin vaatimuksiin. Ohjelmistotuotannossa on helppo kasvattaa projektia liiaksi, jos sen toimintoja ei ole heti alussa määritelty. Näin meille kävi etenkin Flash-prototyyppien loppuvaiheessa. Ainoastaan toimintojen rajaaminen nykyisiksi Processing-prototyypin alkuvaiheessa mahdollisti projektin valmistumisen.

5.3 Tulevaisuus

Tulemme kehittämään Dictriaa tulevaisuudessa. Konsepti on tässä muodossa toimiva, joten voimme keskittyä itse tuotteen kehittämiseen sekä sen toiminnallisuuksien parantamiseen ja testaamiseen. Dictrian nykyinen versio on valmis julkaistavaksi verkossa, jota aiomme hyödyntää projektin seuraavia vaiheita tuottaessamme. Mahdollisimman iso käyttäjäkunta on oleellista Dictrian konseptille.

5.3.1 Rakenne

Tärkein rakenteellinen muutos tulevaisuudessa on erottaa asiakkaan ja palvelimen rajapinta asiakasohjelmistosta itsenäiseksi kokonaisuudeksi. Toinen merkittävä muutos on integroida tekstialueen lisäys kokonaisuudeksi pääsovellukseen. Olemme suunnitelleet, että prototyypin seuraava vaihe tulee sisältämään erillisen PHP-rajapinnan appletin ja tietokannan välille. Tällä ratkaisulla pyrimme tekemään rakenteesta elegantimman ja luotettavamman, koska PHP sopisi hyvin rajapinnan toteuttamiseen. Tämä selkeyttäisi myös kokonaisuuksien rooleja.

Olemme myös kiinnostuneita Dictria piirtotilan valinnasta lähitulevaisuudessa. Processingin uusien versioiden myötä on OpenGL:n käyttö helpottunut Processing-ympäristössä, ja etenkin Processingin tuottamissa appleteissa. Visuaalista kuvanlaatua ajatellen tämä on yksi tärkeimmistä aiheista, jota aiomme tulevaisuudessa kokeilla.

5.3.2 Lisäominaisuudet

Prototyypin nykyiseen versioon on toteutettu konseptin kannalta keskeisimmät toiminnot ja visuaaliset mallit. Ennen lopullista konseptirajausta Dictriaan oli suunnitteilla enemmän ominaisuuksia, mutta päätimme jättää niiden tekemisen myöhempää toteutusta varten erillisinä laajennuksina.

Mielenkiintomme kohdistuu alakohdan 5.3.1 lisäksi toimintoihin, jotka helpottavat käyttäjän navigaatiota ja liikkumista komiulotteisessa tilassa. Reitinvalintaan ja paikannukseen kohdistuvien ratkaisujen toiminta voisi luoda käyttäjäkokemuksesta helpomman ja nopeamman oppia.

Yksi käyttäjähaastatteluissa esiin tullut toiminto on tiedon suodattaminen tilan sanoista. Vaikka tämä idea poikkeaa osittain Dictrian konseptista, tarjoaa se mielenkiintoisia vaihtoehtoja tiedon ja vuorovaikutuksen visualisointiin. Samoin olemme tutkineet erilaisten historiatointojen toteuttamista sanoille, jotka mahdollistaisivat syötteen muutoksien seuraamisen, ja omien valintojen muistamisen.

Näitä toimintoja ei ole testattu nykyisessä konseptissa tai toteutuksessa. Osa toteutuksista on mahdollista tehdä nykyisten tietorakenteiden pohjalta ja niitä on huomioitu teknisessä suunnittelussa. Ne jätettiin pois nykyisestä prototyypistä, sillä ne eivät ole välttämättömiä Dictrian nykyiselle toiminnalle, mutta ne voivat toimia suuntaa antavina konseptilaajennuksina tulevaisuuden versioita kehitettäessä.

Lähdeluettelo

Adobe (Macromedia). 2002. *Flash Tech Note – Loading data across domains*.
[online]. Saavutettavissa:
http://www.adobe.com/cfusion/knowledgebase/index.cfm?id=tn_16520 [viitattu 11.
Lokakuuta 2006].

Baker, Martin John. 2006. *Maths - Rotations Using Quaternions*. [online].
Saavutettavissa:
<http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/transforms/index.htm> [viitattu 19. Lokakuuta 2006].

Blekeli, Ida. 2006. *Dictria - Designing a multiuser experience*. Helsinki:
Medialaboratorio, Taideteollinen korkeakoulu.

Bowman, Doug A. et al. 2005. *3D User Interfaces – Theory and Practise*. 1. Painos.
Boston: Addison Wesley.

Flickr. 2006. *Welcome to Flickr - Photo Sharing*. (Main Page) [online].
Saavutettavissa: <http://www.flickr.com/> [viitattu 14 Syyskuuta 2006].

Harris, Jonathan ja Kamvar, Sepandar. 2005. *We Feel Fine*. [online] Saatavissa:
<http://www.wefeelfine.org> [viitattu 4 Lokakuuta 2006].

Hearn, Donald ja Baker, M. Pauline. 1994. *Computer Graphics*. Toinen painos. New
Jersey: Prentice Hall.

Jenett, Florian. 2006. *bezier.de / mysql*. [online] Saavutettavissa:
<http://www.bezier.de/mysql/> [viitattu 12. Lokakuuta 2006].

McFarland, David Sawyer. 2005. *Analysis: Considering Adobe's future*. [online].
Saavutettavissa:
<http://www.macworld.com/news/2005/04/19/adobeanalysis/index.php> [viitattu 5.
Lokakuuta 2006].

Mysql. 2006. Mysql (Etusivu). [online]. Saavutettavissa: <http://www.mysql.com/> [viitattu 12. Lokakuuta 2006].

Olsson, Krister. 2006. *Ess*. [online]. Saavutettavissa: <http://www.tree.axis.com/Ess/> [viitattu 18. Lokakuuta 2006].

O'Reilly, Tim. O'Reilly Media. 2005. *What is Web 2.0*. [online]. (Päivitetty 30. syyskuuta 2005. Saavutettavissa: <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html> [viitattu 3. Lokakuuta 2006].

O'Reilly Media. 2002. *What is RSS?*. [online]. Saavutettavissa: <http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html> [Viitattu 16. lokakuuta 2006].

Open source Initiative. 2006 (Pääsivu). [online]. Saavutettavissa: <http://www.opensource.org/> [viitattu 4. Lokakuuta 2006].

OpenGL. 2006. *OpenGL - The Industry Standard for High Performance Graphics*. [online] Saavutettavissa: <http://www.opengl.org/> [viitattu 19. Lokakuuta 2006].

PHP. 2006 (Pääsivu). *PHP – Hypertext Preprocessor*. [online]. Saavutettavissa: <http://www.php.net/> [viitattu: 10. Lokakuuta 2006].

Processing. 2006a. *Processing [beta]*. [online]. Saatavissa: <http://processing.org/contribute/index.html>. [viitattu: 6. Lokakuuta 2006].

Processing.org. 2006b. *Processing API*. [online]. Saavutettavissa: <http://processing.org/reference/index.html> [viitattu 17. Lokakuuta 2006].

Processing.org. 2006c. *Processing Quad()*. [online]. Savutettavissa:
http://processing.org/reference/quad_.html [viitattu 17. Lokakuuta 2006].

Processing.org. 2006d. *Processing beginShape()*. [online]. Savutettavissa:
http://processing.org/reference/beginShape_.html [viitattu 17. Lokakuuta 2006].

Processing.org. 2006e. *GUI for Development*. [online]. Saavutettavissa:
http://processing.org/discourse/yabb_beta/YaBB.cgi?board=os_libraries_tools;action=display;num=1160029880 [viitattu: 19. Lokakuuta 2006].

Processing.org. 2003. *distorted texture mapping?*. [online]. Saavutettavissa:
<http://processing.org/discourse/yabb/YaBB.cgi?board=Programs;action=display;num=1056367221> [viitattu 19. Lokakuuta 2006].

Processing.org. 2006. *OpenGL*. [online]. Saatavissa:
<http://processing.org/reference/libraries/opengl/> [viitattu 19. Lokakuuta 2006].

Richter, Stefan. 2006. *Just Letters*. [online] Saavutettavissa:
<http://web.okaygo.co.uk/apps/letters/flashcom/> [viitattu 4. Lokakuuta 2006].

Riekoff, Christian. 2006. *proXML*. [online]. Saavutettavissa:
<http://www.texone.org/proxml/> [viitattu 19. Lokakuuta 2006]

Schach, Stephen R. 2002. *Object-Oriented and Classical Software Engineering*. 5. painos. New York: McGraw-Hill.

Sun Microsystems, Inc. 2006a. *Applets*. [online]. Saavutettavissa:
<http://java.sun.com/applets/> [Viitattu 8. Lokakuuta 2006].

Sun Microsystems. 2006b. *Java.awt*. [online]. Saavutettavissa:
<http://java.sun.com/j2se/1.4.2/docs/api/java/awt/package-summary.html> [viitattu 19. Lokakuuta 2006].

Sun Microsystems. 2006c. *javax.swing*. [online]. Saavutettavissa: <http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html> [viitattu 19. Lokakuuta 2006].

Sun Microsystems. 2006d. *Mixing heavy and light components*. [online] Saavutettavissa: <http://java.sun.com/products/jfc/tsc/articles/mixing/> [viitattu 19. Lokakuuta 2006].

Weisstein, Eric W. 2006. *Vector*. [online]. Saavutettavissa: <http://mathworld.wolfram.com/Vector.html> [viitattu 13. Lokakuuta 2006].

Wikipedia - The free encyclopedia. 2006a. (Main Page) [online]. (Updated 13 Sep 2006) Saavutettavissa: http://en.wikipedia.org/wiki/Main_Page [viitattu 17 Syyskuuta 2006].

Wikipedia. 2006b. *Hybrid Client*. [online]. Saavutettavissa: http://en.wikipedia.org/wiki/Hybrid_client [viitattu 12. Lokakuuta 2006].

World Wide Web consortium. 2006. *Extensible Markup Language (XML)*. [online]. Saavutettavissa: <http://www.w3.org/XML/> [viitattu 9. Lokakuuta 2006].

You Tube. 2006. *You Tube - Broadcast Yourself*. (Pääsivu) [online]. Available on: <http://www.youtube.com/> [viitattu 17. Syyskuuta 2006].