

Jouni Kuusisto

Lehtiartikkelin automaattinen taitto

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ohjelmistotekniikka

Insinöörityö

12.5.2013

Tekijä(t) Otsikko Sivumäärä Aika	Jouni Kuusisto Lehtiartikkelin automaattinen taitto 33 sivua + 2 liitettä 12.5.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	ohjelmistotekniikka
Ohjaaja(t)	yliopettaja Erja Nikunen järjestelmäarkkitehti Tuomo Telkkä
<p>Insinööritöiden tavoite oli kehittää tilaajayrityksen toimitusjärjestelmään liitettävä, lehtiartikkelin taiton automatisoiva ohjelma. Graafiselle ohjeistukselle oli luotava ohjelmaa varten esitystapa ja määrittää ohjeistus siten, että lopputuloksena artikkelista saadaan valittuun julkaisukanavaan kohdennettu, julkaisuvalmis taitettu dokumentti.</p> <p>Ohjelma lukee toimitusjärjestelmästä syötetietona artikkelin tekstin, liitetyt kuvat ja graafisen ohjeistuksen. Sivutusalgorithmi luo ohjeistuksen mukaan materiaalista erilaisia, palstaruudukoihin perustuvia yksinkertaisia sommitteluvaihtoehtoja, joita vaihtelemalla artikkeli pyritään sivuttamaan tasaisesti.</p> <p>Palstaruudukkoon perustuva sommittelu tuottaa aineistosta yksinkertaisia, mutta esteettisesti miellyttäviä ja helppolukuisia kokonaisuuksia, jotka soveltuvat hyvin niin sanomalehtisisällön kuin tekstipainotteisen aikakauslehtisisällön julkaisemiseen. Vaihtoehtoisia sovel-luskohteita ovat myös erilaisten tiedotteiden automaattinen taittaminen.</p> <p>Lopputulos osoittaa automaattisen taiton konseptin toimivuuden tietyissä käyttötapauksissa. Sovelluksella saadaan vaivatta julkaistua sama sisältö erikokoisille ja muotoisille mobiililaitteille taitettuna.</p>	
Avainsanat	automaattinen taitto, dynaaminen taitto, taittoautomaatiikka

Author(s) Title	Jouni Kuusisto Automated layout of an article
Number of Pages Date	33 pages + 2 appendices 12 May 2013
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software engineering
Instructor(s)	Tuomo Telkkä, Systems Architect Erja Nikunen, Principal Lecturer
<p>The aim of this study was to develop a software to automate the layout of an article, to define encoding of graphical instructions for the algorithm and to define these instructions in such way, that the algorithm would output a layouted document, ready for publication in a selected publication channel. The software was to be integrated in the Content Management System of the customer company.</p> <p>Software takes text and photos as well as graphical instructions of an article as input from the Content Management System. Pagination algorithm produces different layout candidates for the selected material, according to given graphical instructions and tries to optimize evenly distributed material flow across pages by selecting the optimal sequence of the page layout candidates.</p> <p>Column grid based layout algorithm produces simple, yet aesthetically pleasing and easy to read layouts that are applicable for publishing both newspaper and text oriented magazine content. Alternative applications might include automated layout of printed newsletters.</p> <p>The results of the study prove the concept of an automated layout effective at least for a certain number of cases. With the help of the software, single content may be effortlessly published with custom layouts targeted to a variety of different mobile devices.</p>	
Keywords	automated layout, dynamic layout, layout synthesis

Sisällys

Sanasto

1	Johdanto	1
2	Aiemmat työt ja tutkimukset	2
3	Toteutussuunnitelma	6
3.1	Sommittelu	7
3.2	Tekstin juoksutus	9
3.3	Kvantaminen	11
3.4	Tietorakenteet	11
4	Sommittelun toteutus	13
4.1	Sijoitussäännöt	15
4.2	Arviointi ja valinta	17
5	Tekstin juoksutuksen toteutus	18
5.1	Tavutus	19
5.2	Rivirekisteri	20
5.3	Leski- ja orporivien hallinta	20
5.4	Otsikot ja väliotsikot	21
6	Sivutuksen toteutus	22
6.1	Ennakointi	24
6.2	Materiaalin jakautuminen	25
7	Kvantamisen toteutus	26
7.1	Grafiikka / Geometriset kuviot	27
7.2	Teksti	28
7.3	Kuvat	28
8	Tulokset	29
	Lähteet	32

Liitteet

Liite 1. Layout Constraints (vain työn tilaajan käyttöön, ei sisälly kirjalliseen raporttiin)

Liite 2. Layout Evaluators (vain työn tilaajan käyttöön, ei sisälly kirjalliseen raporttiin)

Sanasto

Julkaisukanava

Tapa julkaista ja toimittaa toimituksellinen sisältö asiakkaalle. Julkaisukanava voi olla perinteinen paperille painettu tai sähköinen kanava.

Leskirivi Sivun tai palstan alareunaan jäänyt kappaleen ensimmäinen rivi.

Nosto Taiton tyylikeino, jossa lyhyehkö tekstin osa tai sitaatti on haluttu korostaa muusta tekstistä huomattavasti poikkeavalla typografialla.

Orporivi Sivun tai palstan yläreunaan jäänyt kappaleen viimeinen rivi.

Peruslinja Typografinen termi, joka tarkoittaa linjaa, johon useimpien kirjainten alareuna tasataan.

Rivirekisteri Typografinen termi, jolla tarkoitetaan vierekkäisten palstojen samankorkuista, peruslinjoiltaan tasattua riviväliä.

Sähköinen paperi

Perinteistä paperia jäljittelevä näytötekniikka, tai sitä hyödyntävä lukulaitte. Tavallisista taustavalaistuista näytöistä poiketen sähköinen paperi heijastaa valoa kuten perinteinen paperi.

1 Johdanto

Graafinen ala on viime vuosina ollut kiihtyvällä tahdilla siirtymässä painetuista tuotteista erilaisiin sähköisiin medioihin. Perinteisen Internetin rinnalle ovat nousseet erilaiset tablet-tietokoneet sekä sähköinen paperi.

Tablet-tietokoneiden ja sähköisten papereiden joukosta ei kuitenkaan ole löytynyt vakiintunutta ratkaisua sanoma- tai aikakauslehtien jakeluun, eikä vakiintumista näillä näkymin ole tapahtumassa. Laitteiden valtava kirjo luo joukon haasteita julkaisijoille. Julkaisun sisältö pitää toimittaa jokaiselle alustalle eri julkaisureittejä pitkin. Toisekseen yksittäisestä artikkelista pitäisi taittaa lukuisia versioita, johtuen näyttökokojen suuresta kirjosta, unohtamatta vaaka- ja pystysuuntien erikseen taittamista. Useiden taittojen tekeminen on julkaisijalle liian työlästä ja kallista. Tästä johtuen nykyiset tablet-julkaisut joko jättävät taiton tekemättä ja tyytyvät juoksuttamaan tekstin yhteen palstaan, tai vaihtoehtoisesti ovat saatavilla ainoastaan yhteen laitteeseen ja tiettyyn kokoon taitettuna.

Julkaisijat käsittelevät sähköisiä julkaisukanavia yhä useammin omina tuotteinaan, eikä niinkään painetun tuotteen tukitoimintoina. Ilmiö kiristää myös julkaisuvaatimuksia. Skuppeja, eli julkaisijan ennen muita julkaisemia tärkeitä uutisia, ei enää välttämättä säästetä painettuun lehteen, vaan ne halutaan julkaista välittömästi. Painopisteen siirtyminen sähköiseen mediaan on lisännyt merkittävästi julkaisun ulkonäkö- ja typografiaa-vaatimuksia. Aikaisemmin saattoi riittää se, että teksti ja kuvat ylipäättään saatiin Internetiin. Taiton ja typografian huomiointi on kuitenkin perusteltavaa, koska perinteisesti taitetun ja palstoitettun tekstin lukeminen on nopeampaa ja helpompaa. [1.]

Insinööriyön tarkoitus on toteuttaa ohjelma, joka automatisoi taittoprosessin yksittäisen lehtiartikkelin osalta. Jo nyt suuri osa lehden taitosta perustuu Art Directorin määrittämiin graafisiin ohjeistuksiin sekä yleisiin käytäntöihin. Art Director on ulkoasusta vastaava henkilö, joka muun muassa suunnittelee julkaisujen ulkoasut ja tekee myös taitotyötä [2, s. 23]. Insinööriyön tavoite on kuvata graafiset ohjeet ja käytännöt tietokoneelle sillä tarkkuudella, että ohjelma pystyy niiden perusteella taittamaan artikkelin oheismateriaaleineen.

Toteutettava ohjelma on luonteeltaan hyvin kokeellinen ja siten myös löyhästi määritelty. Tarkoitus on kehittää tilaajayritys Anygraaf Oy:n Neo-toimitusjärjestelmään uusi ohjelmamoduuli, jonka avulla järjestelmässä oleva artikkeli voidaan tulevaisuudessa julkaista automaattisesti taitettuna haluttuun sähköiseen julkaisukanavaan. Taiton automatisoinnin avulla toimittajat voivat entistä paremmin keskittyä sisällön tuottamiseen.

Anygraaf Oy on suomalainen, graafisen alan ratkaisuja tuottava ohjelmistoyritys. Yrityksen asiakaskunta koostuu pääasiassa sanoma- ja aikakauslehtien julkaisijoista, mutta mukaan mahtuu myös pienempiä yrityksiä ja kirjakustantajia. Yksi tärkeimmistä tuotteista on toimitusjärjestelmä Neo. Toimitusjärjestelmä on lehdenseon työkalu, laaja kokonaisuus, jossa koko lehden sisältö, tekstit ja kuvat, tuotetaan. Monikanavajulkaisun aikakaudella toimitusjärjestelmän on hallittava perinteisen painettavan sisällön lisäksi myös videot, äänet sekä arkistoidut aineistot. Neo on alusta alkaen suunniteltu monikanavajulkaisemisen tehotyökaluksi, joten ohjelma on luonnollinen ja toivottu lisä kokonaisuuteen. [3.]

2 Aiemmat työt ja tutkimukset

Taiton automatisointia on aikaisemmin tutkittu ja lähestytty eri lähtökohdista. Suuri osa taiton automatisoinnin tutkimuksista on aiheiltaan hieman laajempia tai keskittyvät eri asioihin, kuten kuva-albumin tai kuvakollaasin taittoon. Osa ongelmista on kuitenkin luonteeltaan samankaltaisia ja joukosta löytyy myös ratkaisuja sivutetun dokumentin taittoon. Lehden tai lehtiartikkelin taittoon ei kuitenkaan toistaiseksi ole saatavilla kaupallisia ratkaisuja [5].

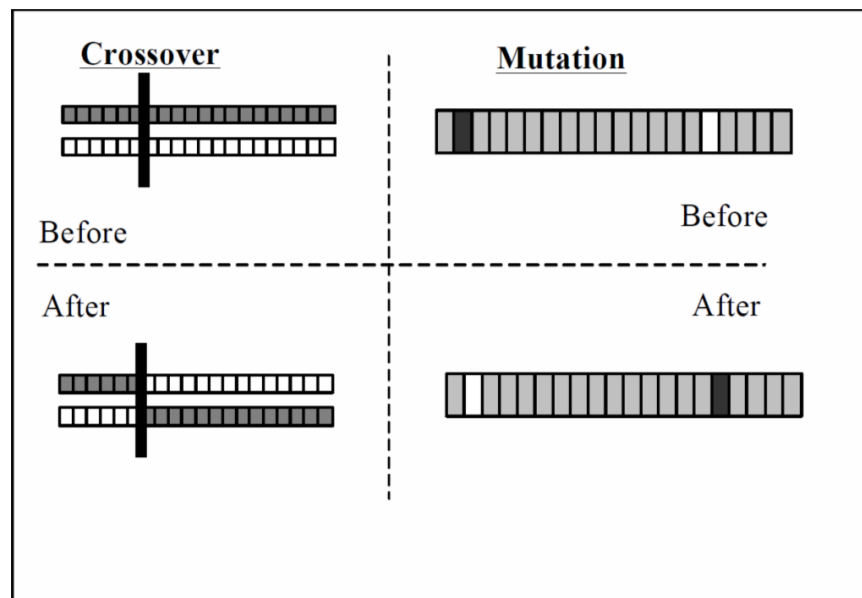
Yksi tapa sommitella sisältö on mallien (*template*) käyttö. Malleihin määritetään kiinteät paikat kuville ja tekstile. Näin ratkaistaan ongelma melko suoraviivaisesti ja tehokkaasti, mutta lopputuloksen vaihtelevuutta voidaan lisätä vain lisäämällä järjestelmään uusia malleja. [5; 6; 7.] Mallit rajoittavat myös sisällön määrää ja vaihtelevuutta, esimerkiksi kaksi ja kolme kuvaa sisältävät sivut vaativat omat mallinsa. Myös muuttuvat kuvasuhteet ja palstamäärät vaikeuttavat mallien käyttöä.

Pelkkiin malleihin perustuvien ratkaisuiden lisäksi käytännössä kaikki muut tunnetut algoritmit tukeutuvat taiton ja sommittelun määrittämisessä erilaisiin sääntöihin. Katsauksessaan Lok ja Feiner [8] luokittelevat säännöt tyypin mukaan kahteen ryhmään,

spatiaalisiin ja abstrakteihin. Spatiaaliset säännöt määrittävät tyypillisesti elementtien sijaintia, kokoa ja suhteita. Tyypillinen spatiaalinen sääntö voi olla esimerkiksi ”kuva-tekstin on oltava kuvan alla”. Abstraktit säännöt ovat korkeamman tason sääntöjä kuten ”teksti1 viittaa kuvaan1”.

Työssään Weitzman ja Wittenburg [9; 10] käyttävät relaatiokielioppia (Relational Grammar) kuvaamaan elementtien välisiä suhteita ja sääntöjä. Relaatiokieliopissa subjektin ja objektin suhde kuvataan yksittäisellä predikaatilla. Ratkaisussa säännöistä voidaan muodostaa myös kompositioita, joita voidaan käyttää uudestaan muiden sääntöjen subjekteina tai objekteina. Säännöistä muodostuu puurakenne, jonka solmut käsitellään pohjalta eli lehtisolmuista ylöspäin (*bottom-up traversal*).

Sääntöjen erilaisuuden ohella suurin ero tunnettujen algoritmien välillä on tapa, joilla ne pyrkivät luomaan sivumalleja, joita säännöillä arvioidaan. Osaltaan tavat vaikuttavat paljon myös sääntömäärittysten luonteisiin. Osa sovelluksista suorittaa elementtien sijoittamisen geneettisillä algoritmeilla. Geneettiset algoritmit ovat haku- tai optimointialgoritmeja jotka jäljittelevät luonnollista evoluutiota. Algoritmissa joukkoa ratkaisuvaihtoehtoja kehitetään satunnaisella mutaatiolla sekä risteyttämällä vaihtoehtoja. Jokaisen sukupolven vaihtoehdot arvioidaan ja parhaat sopivuusarvot saaneet valitaan tuottamaan uusi sukupolvi. [11.] Kuva 1 havainnollistaa geeninjonon risteytystä ja mutaatiota.



Kuva 1. Geneettiset operaatiot geeninjonolle [6].

Risteytyksessä kahden vaihtoehdon välillä valitaan alue, jolta yksittäiset geenit vaihdetaan vaihtoehtojen kesken. Mutaatiossa vaihdettavat geenit valitaan pistemäisesti.

Geneettisiä algoritmeja käyttävät töissään muun muassa Geigel ja Loui [6] sekä Purvis, Harrington, O'Sullivan ja Freuder [7]. Ensimmäinen on kuva-albumin taiton automatisoitu ratkaisu ja jälkimmäinen pyrkii taittamaan tekstiä ja kuvia sisältävän sivun. Jälkimmäisessä dokumentin elementtien arvot, kuten kuvien sijainnit ja koot, koodataan geeneiksi, jotka kehittyvät geneettisen algoritmin avulla. Yksilöiden sopivuutta arvioidaan kahteen ryhmään jaettujen sääntöjen perusteella. Ensimmäiseen ryhmään kuuluvat niin sanotut kovat säännöt, kuten marginaalien sisällä pysyminen tai elementtien päällekkäisyyden kieltäminen. Toiseen ryhmään kuuluvat suunnittelukriteerit kuten sijoittelu ja tasapaino. Ensimmäisen ryhmän säännöt vähentävät yksilöiden pisteytystä ja toisen ryhmän säännöt pisteyttävät yksilöä reaalilukuarvolla 0...1 riippuen siitä, kuinka hyvin sääntö toteutuu.

Moni algoritmi käyttää hyväkseen typografisia ruudukoita. Typografinen ruudukko on suosittu taiton apukeino, jossa kaksiulotteinen pinta jaetaan pienemmiksi, samankokoisiksi neliöiksi käyttäen pysty- ja vaakaviivoja. Ruudukko tarjoaa suunnittelijalle vakio-pohjan, jonka avulla hänen on helppo sijoittaa kuvat ja tekstit luovuuden kuitenkin rajoittumatta. Hyvin suunnitellulla ruudukolla saavutetaan miellyttävä yhtenäisyys julkaisun eri sivujen välillä, kuitenkin sallien joustavuus taitossa. [12; 13; 14.] Kuva 2 havainnollistaa typografisen ruudukon käyttöä tekstielementtien sijoittamiseen sivulla. Lopullisella sivulla ruudukko ei ole näkyvissä.

Vermont Symphony Orchestra		
Winter 2007 Season	Aaron Copland The Tender Land January 2007	Eric Satie Gymnopedie 1, 2 February 2007
	01/12/07 Middlebury College Center for the Arts 8:00 pm	02/03/07 Johnson State College Dibden Center for the Arts 8:00 pm
	01/19/07 Johnson State College Dibden Center for the Arts 8:00 pm	02/10/07 Castleton State College Fine Arts Center 8:00 pm
	01/26/07 Lyndon State College Alexander Twilight Theater 8:00 pm	02/17/07 Middlebury College Center for the Arts 8:00 pm

Kuva 2. Esimerkki typografisen ruudukon käytöstä [14].

Graf [12] hyödyntää työssään Boolean predikaatteja sääntöjen määrittelyssä. Säännöt ovat samankaltaisia kuin useimmissa muissa ratkaisuisa, pääasiassa spatiaalisia. Boolean operaattoreilla mukaan saadaan kuitenkin myös ehdollisuutta, joilla erilaisiin tilanteisiin voidaan reagoida. Sommittelun hakuavaruutta ratkaisussa rajataan luomalla kohdemediaan ja sisältöön sopiva ruudukko. Yksittäiset taittoelementit sijoitetaan tarkasti yhden tai useamman ruudun rajaamalle alueelle.

Jacobs, Li, Schrier, Barger ja Salesin [15; 16] käyttävät työssään sääntöpohjaisista ruudukkomalleista koostuvaa kirjastoa. Säännöt määrittävät taittoelementin koon ja sijainnin ruudukossa. Ruudukkoon perustuvat mallit ovat kuitenkin perinteisiä, tarkkaan määritettyjä malleja joustavampia ja sallivat skaalauksen tiettyyn pisteeseen asti. Ratkaisu sisältää lisäksi sivutuslogiikan, joka pyrkii tasapainottamaan artikkelin sisällön jakaantumisen eri sivuille kokeilemalla erilaisia mallisekvenssejä sekä lisäämällä valinnaisia taittoelementtejä, joilla tyhjää tilaa voidaan täyttää.

Myös O'Brien ja Liu [13] käyttävät ruudukkoita rajaamaan mahdollisten taittoratkaisujen hakuavaruutta. Toteutus sisältää kirjaston ammattilaisen suunnittelemaa ruudukkoita,

mutta elementtien sijainteja ei määritetä vaan sijoitusalgoritmi etsii mahdolliset sijoituspaikat elementeille ruudukosta. Käyttämättömiin ruudukon soluihin juoksutetaan leipäteksti palstoitettuna.

3 Toteutussuunnitelma

Toteutettavan ohjelman ei ole tarkoitus tuottaa pienimpäänkin yksityiskohtaan asti hiottua taidekokonaisuutta, vaan kohtuullisen yksinkertainen mutta kuitenkin brandin huomioiva, helposti luettava ja silmää miellyttävä taittokokonaisuus annetusta lähdemateriaalista. Lähdemateriaaliksi syötetään artikkelin teksti, siihen liitetyt kuvat sekä graafinen ohjeistus. Luettavuutta ja visualisuutta varten perinteiset typografian säännöt otetaan huomioon myös tässä sähköiseen mediaan suunnitellussa taitossa.

Julkaisun brandi huomioidaan määritettävällä automaattisen taiton ohjeistuksella, typografialla sekä vakioelementeillä kuten ylä- ja alatunnisteilla. Automaattista taittoa varten suunnitellaan sääntömääritykset, joilla taiton ilmettä voidaan ohjata haluttuun suuntaan. Sääntömäärittelyillä voidaan määrittää kuvien ja tekstilaatikoiden sijainnit karkeasti sivulla. Ylä- ja alatunnisteita varten toteutetaan yksinkertainen kuvauskieli, jolla näihin saadaan vakiotekstit, viivat ja logot.

Ohjelma lukee toimitusjärjestelmästä artikkeliin liitetyt kuvat ja käyttää ne taitossa sääntömääritysten määräämällä tavalla. Taittoalgoritmi pyrkii optimoimaan sisällön tasainen jakautuminen sivuille silloin, kun se materiaalin puitteissa on mahdollista. Liian vähäisen sisällön takia sivun täyttäminen voi olla mahdotonta, mutta sivu pyritään taittaa mahdollisimman tasapainoiseksi.

Ohjelman algoritmit ja tietorakenteet tullaan toteuttamaan alustariippumattomasti siten, että ohjelmakoodi on mahdollisimman helposti siirrettävissä eri alustoille. Taiton eri julkaisukanaviin kuvantamista varten luodaan geneerinen rajapinta, jolla mahdollisesti alustariippuvaiset kuvantamistoteutukset liitetään ohjelmaan.

Julkaisukanavaa ei ole ennalta määritetty mutta ohjelman tulisi olla laajennettavissa erilaisiin vaihtoehtoihin ja esitystapoihin. Proof of Conceptina, eli idean toimivuuden todistamiseksi, ohjelma tulee luomaan valitusta artikkelista taitetun PDF-dokumentin [4].

Artikkelin matka irrallisesta tekstistä ja kuvista julkaisuvalmiiksi taitoksi jakautuu karkeasti ainakin kahteen vaiheeseen: taiton määrittämiseen sekä taiton kuvantamiseen kohdemediää varten. Taiton määrittäminen on näistä kompleksisempi ja on edelleen jaettavissa useampaan osaan. Tässä työssä ja raportissa taitto on jaettu karkeasti kolmeen osaan, sommitteluun ja tekstin juoksutukseen sekä lopuksi näiden kahden yhdistämiseen sivutuksessa.

3.1 Sommittelu

Taittamiseen liittyy yleisesti paljon sääntöjä ja ohjeita, mutta toisaalta kaikki säännöt on tehty rikottavaksi. Seuraus on se, että yksittäisen kuvan tai tekstilaatikon voi sijoittaa sivulle käytännössä mihin tahansa, kunhan lopputulos näyttää johdonmukaiselta. Lisäksi kuva voi olla sovitettuna yhteen tai useampaan palstaan, palstojen väliin tai vaikka syväty siten, että teksti kiertää sitä. Erilaiset nostot kuten lainaukset on usein sijoitettu palstojen väliin.

Algoritmin ensimmäinen ongelma on taittoelementtien sijoittaminen sivulle. Aikaisemmat ratkaisut ovat lähestyneet ongelmaa malleilla, ruudukoilla ja geneettisillä algoritmeilla. Tämän työn lähestymistapa yhdistää piirteitä ruudukkoihin ja sääntöihin perustuvista ratkaisuista.

Tarkoitus on kehittää algoritmi, joka luo annetusta lähdemateriaalista tarpeeksi monta taittovaihtoehtoa kohdesivulle, jotta joukosta löydetään kontekstiin sopiva vaihtoehto. Hakuavaruutta rajoitetaan luomalla kohdemediään sopiva palstaruudukko, johon taittoelementit sijoitetaan. Aikaisemmista ruudukkoihin perustuvista algoritmeista poiketen ruudukkoa ei ole etukäteen jaettu pystysuunnassa. Sen sijaan algoritmi jakaa ruudukkoa tarpeen mukaan, aina uutta taittoelementtiä sijoittaessaan. Ruudukon solut eivät pysy tasakokoisina, mutta ruudukko ohjaa algoritmia sijoittamaan elementit tasattuina. Ruudukon toimintaa on kuvattu kuvassa 3.



Kuva 3. Sivun palstoista rakentuva ruudukko.

Kuvassa vasemmalla on ensin kahteen palstaan jaetun sivun ruudukko, jossa pystysuuntaista jakoa ei ole vielä suoritettu. Oikeanpuoleisessa mallissa sivun vasempaan yläreunaan on sijoitettu elementti ja ruudukko on jaettu pystysuunnassa siten, että sijoitettu elementti täyttää kokonaisen solun.

Satojatuhansia vaihtoehtoja tuottavasta algoritmista ei sellaisenaan ole hyötyä. Ohjelman pitää myös osata valita vaihtoehtoista parhaat. Valintaa varten toteutetaan rajapinnat, joilla taittoon voidaan dynaamisesti määrittää uusia sääntöjä sekä arviointimenetelmiä. Menetelmä on tuttu monista aikaisemmin esitetyistä töistä. Tässä työssä säännöt on jaettu ja nimetty luonteensa mukaan kahteen ryhmään, sääntöihin ja arvioitsimiin. Säännöt ovat luonnoltaan ehdottomia ja niiden tarkoitus on samalla rajata sommittelupermutaatioiden määrää. Esimerkki tyypillisestä säännöstä voisi kuulua näin: ”kuvan 1 tulee sijaita kuvan 2 yläpuolella”. Arvioitsimien on tarkoitus puolestaan arvioida ja pisteyttää taittovaihtoehtoja omien tavoitteidensa mukaisesti. Arvioitsimet voivat mitata valmiista sommitelmasta erilaisia suureita, kuten tasapainoa ja symmetrisyyttä.

Mikäli artikkeli ei mahdu yhdelle sivulle, pitää materiaalitasapainosta huolehtia yksittäisten sivujen lisäksi myös koko artikkelissa. Kuvituksen on tarkoitus elävöittää tekstiä, joten kuvat olisi hyvä pystyä jakamaan sivuille mahdollisimman tasaisesti. Tiukasti määritelty sääntösarja tosin voi määrittää myös kuvamäärän kiinteästi sivulle, jolloin valinta ei välttämättä ole ohjelman käsissä.

Vähäisen materiaalin tai aikaisempien sivujen taittovalintojen takia viimeinen sivu voi jäädä puolityhjäksi. Puolityhjäksi jääneet sivut näyttävät lähes poikkeuksetta huonolta. Puolityhjän sivun havaitessaan ohjelma pyrkii etsimään sommittelupermutaatioista vaihtoehtoa, joka täyttää sivua paremmin.

Julkaisijan brandia tukemaan sivulle voidaan lisätä ylä- ja alatunnisteet. Ylä- ja alatunnisteet ovat sommittelun kannalta kiinteitä, koko sivun levyisiä elementtejä sivun ylä- ja alareunassa. Niiden sisältö on aina vakiomuotoista, ja se sijoitetaan aina yksiselitteisten määritysten mukaan.

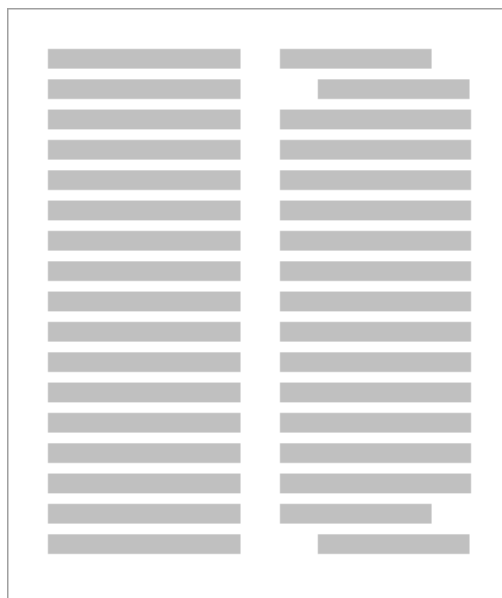
3.2 Tekstin juoksutus

Tekstin juoksutukseen liittyy paljon yleisiä sääntöjä ja typografisia ohjeita. Sana typografia tulee kreikan kielestä, *typos grafein*, kirjoittaa merkein. Nykyisin typografialla tarkoitetaan tekstin painoasua, tai kuten tässäkin, vain tekstin asua. [18, s. 221.] Typografian säännöt ja ohjeet ovat huomattavasti yleispätevämpiä kuin muut graafiset ohjeistukset. Leipätekstin osalta tärkeimpiä asioita ovat rivirekisteri sekä leski- ja orporivien hallinta. Rivirekisterillä tarkoitetaan eri palstojen tekstin peruslinjan kohdistusta läpi koko sivun. Tilanteissa, joissa palstat alkavat pystysuunnassa eri kohdista tai tekstin seassa on esimerkiksi väliotsikko, joudutaan seuraava leipätekstikappale kohdistamaan uudestaan rivirekisteriin. Kuva 4 havainnollistaa rivirekisterin merkitystä palstoitella sivulla.



Kuva 4. Palstoitettu sivu ilman rivirekisteriä ja rivirekisterin kanssa.

Kuvassa vasemmalla väliotsikon jälkeiset rivit eivät enää ole samassa tasossa viereisen palstan kanssa. Oikealla puolella väliotsikko on tasattu oikein siten, että sitä seuraavat rivit noudattavat sivulle asetettua rivirekisteriä. Kuva 5 puolestaan havainnollistaa orpo- ja leskirivejä.



Kuva 5. Orpo- ja leskirivit havainnollistettuna oikeanpuoleisella palstalla.

Orporivillä tarkoitetaan kappaleen viimeistä riviä, joka jää sivun tai palstan ensimmäiseksi. Leskirivi puolestaan tarkoittaa sivun tai palstan loppuun jäänyttä yksittäistä riviä. [17, s. 31.]

Riippumatta siitä, onko teksti juoksutettu liehureunaisena vai tasattuna, on rivin pituus pyrittävä pitämään mahdollisimman tasaisena. Liehureunassa epätasaiset rivit johtavat esteettisesti epämiellyttävään reunaan ja tasatuissa palstoissa suureen sana- tai kirjainvälin vaihteluun. Euroopassa on tapana tasata rivit säätämällä sanaväliä ja myös Yhdysvalloissa ollaan luopumassa kirjainväliharvennuksesta. [17, s. 25.] Rivien tasaus tullaankin toteuttamaan ainoastaan sanavälejä säätämällä.

Leipätekstissä rivien epätasaisuus ratkeaa tavutuksella. Sisällön kieltä ei ole kuitenkaan määritetty, joten mitään tiettyä tavutuskirjastoa ei tulla sisällyttämään ohjelmaan. Toimitusjärjestelmään on kuitenkin jo toteutettu liitäntä Lingsoftin tavutusohjelmaan. Tavutustieto sisällytetään materiaaliin samalla kun se tuodaan toimitusjärjestelmästä. Otsikoiden ja väliotsikoiden osalta ongelma on hieman monimutkaisempi, sillä tavutusta näissä ei kuulu tehdä.

Tekstin juoksutus on vahvasti sidottu kohdemediaan. Juoksutusta varten ohjelmaan toteutetaan korkean tason logiikka, joka puolestaan kutsuu kohdemedian mukaan vaihtuvan kuvantimen alemman tason rutiineja, esimerkiksi merkkijonon esityskoon mittaamista varten.

3.3 Kuvantaminen

Taittoalgoritmin luonteen takia artikkelia ei voida kuvantaa samaan aikaan, kun ratkaisuja etsitään. Kuvantaminen on yleensä laskennallisesti vaativaa ja näin ollen hidastaisi raskasta taittoratkaisun etsintää huomattavasti.

Työssä kuvantimella tarkoitetaan luokkaa joka toteuttaa yleisen rajapinnan, jolla elementit voivat piirtää konkreettiselle sivulle tekstiä, kuvia tai muuta grafiikkaa. Työn tässä vaiheessa toteutetaan PDF-kuvannin käyttäen Skia-kirjastoa. Skia on korkean tason monialustainen 2D-grafiikkakirjasto tekstin, grafiikan ja kuvien piirtämiseen [19].

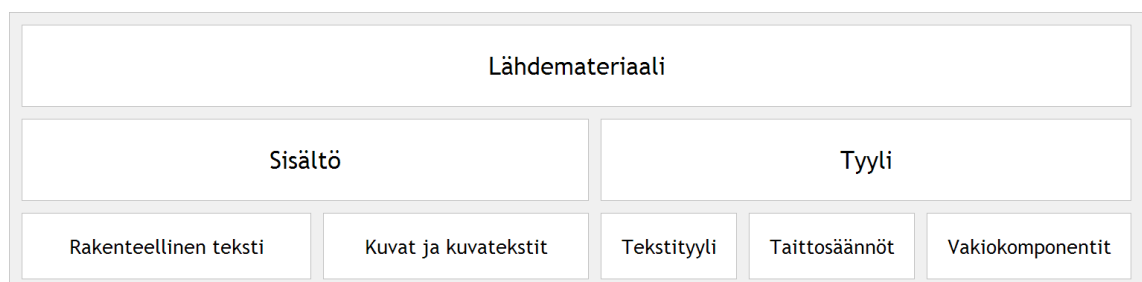
3.4 Tietorakenteet

Toimitusjärjestelmän tietorakenteet sisältävät pääasiassa toimituksellista materiaalia, eikä juurikaan taittamiseen liittyvää tai tarvittavaa tietoa. Lisäksi ne on vahvasti sidottu

tietokantoihin ja alustariippuvaiseen koodiin. Taittomoduulin on tarkoitus olla helposti siirrettävissä eri alustoille, ja raskaan algoritmin takia tietorakenteiden pitää vastaavasti olla ketteriä.

Sekä lähdemateriaalin että taitetun sivun kuvaamiseen luodaan omat tietorakenteet, jotka tukevat taittamista mahdollisimman vähäisellä ylimääräisellä työllä ja tilavaatimuksella.

Lähdemateriaali on jaettu osiin. Konkreettinen sisältö ja esitys on määritetty toisistaan erillään, mutta joudutaan taiton yhteydessä yhdistämään. Tämä mahdollistaa tehokkaan sisällön tuotannon ja yhtenäisemmän lopputuloksen eri artikkeleiden välillä. Kuva 6 havainnollistaa materiaalin jaottelua. Artikkelin konkreettiseen sisältöön laskeaan itse teksti sekä siihen liitetyt kuvat. Teksti on kuvattu rakenteellisesti XML-muodossa kappaleittain sisältäen nimitiedot sekä kappale- että merkkityyleistä. Taittoa ohjaaviin määrittäyksiin kuuluvat konkreettiset kappale- ja merkkityylit kuten käytetyt fontit ja pistekoot, sommittelua ohjaavat taittosäännöt ja arvioitsimet sekä vakiokomponenttien kuten ylä- ja alatunnisteiden kuvaukset.



Kuva 6. Taiton lähdemateriaali jaoteltuna ryhmiin.

Tekstisisältö ja tekstityylit on valmiiksi kuvattu järjestelmässä XML-muotoisina. Vakio-komponentteja, sääntö- ja arviointimäärittäyksiä varten suunnitellaan työn yhteydessä XML-kuvaus. Ennalta määräämättömän sivukoon takia vakiokomponenttien grafiikkasisältöä varten notaatioon tehdään tapa merkitä koordinaatteja sekä absoluuttisesti että relaatiivisesti.

Lähdetekstiä varten toteutetaan tietorakenne, johon teksti ja tavutustieto saadaan luetua XML-esityksestä ja josta se saadaan käyttöön tehokkaasti ja joustavasti. Lähdetekstiin liittyy lisäksi erilaisia kappale- ja merkkityylejä, jotka tietorakenteen pitää myös

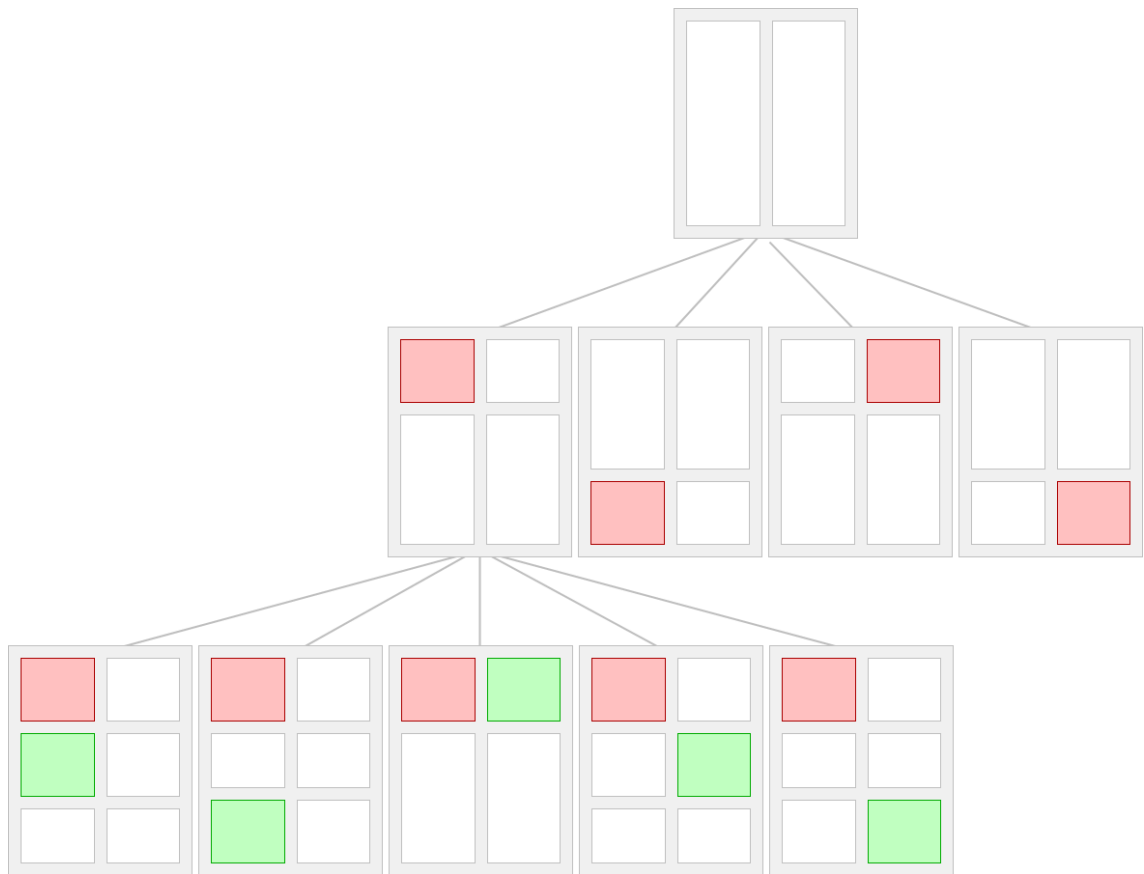
ottaa ketterästi huomioon. Teksti ja tyyli pohja luetaan toimitusjärjestelmän XML-rakenteesta.

Taitettu sivu kuvataan omilla tietorakenteilla, jotka voidaan myöhemmin kuvantaa yleisellä rajapinnalla ennalta määrittelemättömään määrään kohdemedioita. Sivuelementit kuten tekstilaatikko, kuva tai muu grafiikka toteuttavat abstraktin metodin, jolle annetaan viite kulloinkin käytettyyn kuvantimeen. Kaikille kuvantimille yhtenäistä rajapintaa hyödyntäen yksittäiset sivuelementit osaavat kuvantaa itsensä valittua julkaisukanavaa varten.

4 Sommittelun toteutus

Sommitteluprosessi alkaa palstaruudun määrittämisestä. Taittosääntöihin on määritetty ammattilaisen toimesta sivun marginaalit, palstavälit ja toivottu palstaleveys. Algoritmi luo kulloinkin valitun kohdemedian sivukoon perusteella marginaaleihin rajatun palstaruudun ja laskee määritysten mukaan optimaalisen palstamäärän. Toiminto muistuttaa hieman Grafin [12] käyttämää ratkaisua, mutta ruudukkoa ei tässä vaiheessa jaeta lainkaan pystysuunnassa, vaan jokainen palsta muodostaa yhden solun.

Rekursiivinen sijoitusalgoritmi sijoittaa yhden elementin kerrallaan, ruudun jokaiseen paikkaan, johon se mahtuu. Elementin sijoituksen yhteydessä algoritmi jakaa ruudun pystysuunnassa siten, että sijoitettu elementti täyttää luodun solun kokonaan. Näin jaettu ruudukko ohjaa algoritmia sijoittamaan elementit tasaisesti ja toisiinsa nähden tasattuina. Algoritmi sijoittaa laatikoita vain täyden palstan leveyden välein, ja pystysuunnassa saatavilla olevan tilan ylä-, ala- ja keskiosaan. Kuten kuva 7 havainnollistaa, erilaiset sommittelupermutaatiot syntyvät puumaisesti.



Kuva 7. Puumaisesti kehittyvät taittovaihtoehdot.

Puussa jokaisella tasolla uusi elementti on sijoitettu kaikin mahdollisin tavoin edellisen tason elementin jatkoksi. Mikäli viimeksi sijoitettu elementti ei täytä sijoitussolua kokonaan, ruudukko jaetaan pystysuunnassa uudestaan. Kuvaa on yksinkertaistettu jättämällä pois solun keskelle sijoitettujen sekä useampaan palstaan levitettyjen elementtien luomat haarat.

Palsta- ja materiaalmäärän kasvaessa mahdollisten sommittelupermutaatioiden määrä nousee eksponentiaalisesti. Jokainen uusi palsta luo uusien sijoituspaikkojen lisäksi myös uusia tapoja esittää yksittäinen elementti levittämällä sitä useamman palstan kokoiseksi. Jokainen elementti taas saattaa jakaa ruudukon uudestaan siten, että seuraaville elementeille löytyy yhä enemmän sijoituspaikkoja.

Taittoon käytettävissä oleva laskuaika on rajallinen. Pienillä palsta- ja elementtimäärillä ongelmaa ei synny, mutta useamman kuvan ja tekstilaatikon sijoittaminen esimerkiksi kolmeen palstaan luo satoja tuhansia vaihtoehtoja. Sommitteluvaiheessa aikaa sääste-

tään parhaiten, kun huonot permutaatiot saadaan mahdollisimman aikaisessa vaiheessa rajattua pois.

4.1 Sijoitussäännöt

Monien elementtien sijoitukseen liittyy sitovia sääntöjä. Esimerkiksi otsikkoa ei voi sijoittaa sivun oikeaan alakulmaan. Tällaiset säännöt ovat luonteeltaan spatiaalisia: niillä rajataan elementtien sijaintia ja kokoa. Säännöillä voidaan esimerkiksi pakottaa elementti ylälaitaan tai kooltaan korkeintaan yhden palstan levyiseksi.

Sääntöjen tarkoitus on rajata sommittelupermutaatioiden hakuavaruutta. Sääntöjä rikkovat permutaatiot kannattaa hylätä heti, kun se on mahdollista ja lopettaa rekursio jo ennen lopussa tapahtuvaa arviointia. Joskus säännön pitävyys voidaan kuitenkin tarkistaa vasta lopussa. Sääntöjä loukkaavien haarojen mahdollisimman nopean poissulkemisen saavuttamiseksi säännöt jaettiin kahteen eri ryhmään: yksittäisen elementin sijoituksen yhteydessä tarkastettaviin sekä valmiissa sommittelussa tarkastettaviin. Koodiesimerkki 1 esittää yksinkertaistetun idean yllä kuvatusta rekursiivisesta sijoitusalgoritmista sekä siitä, kuinka kahdentyyppiset säännöt tarkastetaan prosessin aikana.

```

1. function sijoita_seuraava(ruudukko R, elementtilista L)
2.     // Otetaan ensimmäinen elementti listasta.
3.     elementti E := pop_first(L)
4.     for each paikka P in R
5.         // Sijoitetaan elementti jokaiseen paikkaan, johon se mahtuu.
6.         if E mahtuu paikkaan P
7.             if tarkista_sijoitussäännöt(P, E) // Tarkistetaan, tyydyttääkö
8.                                                 // sijoitus elementtiin
9.                                                 // liittyvät säännöt.
10.                sijoita E paikkaan P ruudukossa R
11.            if L != tyhjä
12.                // Jatketaan rekursiota sijoittamalla seuraava elementti.
13.                sijoita_seuraava(R, L)
14.        else
15.            // Ei lisää elementtejä, sommitelma valmis.
16.            if tarkista_jälkisäännöt(R) // Tarkistetaan tyydyttääkö
17.                                       // vaihtoehto lopuksi
18.                                       // tarkastettavat säännöt.
19.                pisteytys := arvioi(R) // Suoritetaan arvioitsimet.
20.                if pisteytys >= minimi
21.                    // Pisteet ylittävät minimin,
22.                    // talletetaan vaihtoehto.
23.                    talleta vaihtoehto(R)

```

Koodiesimerkki 1. Yksinkertaistettu kuvaus sijoitusalgoritmista.

Sijoituksen yhteydessä tarkistettavat säännöt tarkistetaan rivillä 7, välittömästi elementille tarpeeksi tilavan paikan löydyttyä. Mikäli ehdot täyttyvät, jatketaan rekursiota kunnes elementit loppuvat, jonka jälkeen tarkistetaan, täyttääkö valmis sommitelma lopuksi tarkistettavat ehdot.

Sääntöjen on tarkoitus olla yksinkertaisia, joten niiden tarkastaminen lopussakin nopeuttaa algoritmia, sillä silloin laskennallisesti vaativampaa arviointia ei jouduta suorittamaan.

Sääntöjä voidaan sitoa yhteen tai useampaan taittoelementtiin. Kun taittoelementti tuntee kaikki siihen liitetyt säännöt ja sääntö siihen liittyvät elementit, ei ylimääräisiä hakuja jouduta tekemään ja tarkistukset saadaan ajettua mahdollisimman nopeasti. Elementin sijoituksen yhteydessä saadaan automaattisesti lista siihen liitetyistä säännöistä, jotka voidaan tarkastaa heti. Lopuksi tarkastettavat säännöt tuntevat niihin liittyvät elementit ja saavat suoraan niiden sijoitustiedot ilman hakua.

Suurin osa käytännöllisistä säännöistä liittyy yhden tai useamman elementin sijoitukseen. Tässä vaiheessa toteutettuihin sääntöihin kuuluu esimerkiksi sääntöjä, jotka määräävät kahden elementin esiintymisjärjestyksen tai pakottavat tietyt elementit yhteen. Sääntömääritysten luontia helpottamaan toteutettiin lisäksi säännön negaatio, jolla minkä tahansa säännön merkitys voidaan kääntää päinvastaiseksi. Näin esimerkiksi sääntö ”kuva 1 tulee olla kuvan 2 vieressä” saadaan muotoon ”kuvan 1 ei tule olla kuvan 2 vieressä”. Tarkemmat kuvaukset toteutetuista säännöistä on esitetty vain työn tilaajalle toimitetussa liitteessä 1.

Ehdottomat säännöt käsitellään samanarvoisesti, eikä niistä voi muodostaa monimutkaisempia tai ehdollisia rakenteita. Sovellukseen ei toteutettu sääntöjen loogisuutta tarkastavia piirteitä. Sääntöjä määritellessä pitää siis olla tarkkana. Liian suuri sääntömäärä tai huonosti laaditut säännöt voivat johtaa siihen, että vaihtoehtoja ei synny. Liian väljät säännöt taas saattavat antaa sijoitusalgoritmille liian vapaat kädet. Huonoilla sääntömäärittelyillä saatetaan toisaalta muodostaa epäjohdonmukaisia silmukoita, jolloin ehdot toteuttavaa ratkaisua ei ole olemassakaan. [8; 12.]

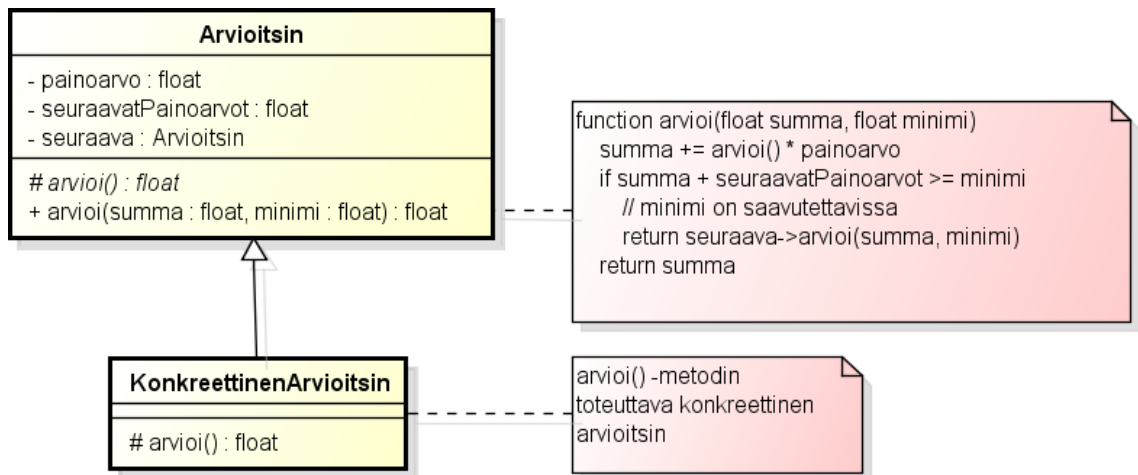
4.2 Arviointi ja valinta

Lopputuloksen kannalta tärkeintä on valita tuhansien tai satojen tuhansien vaihtoehtojen joukosta paras. Siinä missä säännöt rajaavat hakuavaruutta, ne eivät ota lainkaan kantaa siihen, mikä vaihtoehtoista on hyvä. Arvioitsimien on tarkoitus pisteyttää jotain valmiista sommittelusta mitattavaa suuretta. Suureet voivat olla erilaisia estetiikkaan liittyviä ja mitata esimerkiksi elementtien pysty- tai vaakasijoitusta, tasausta, ryhmitteilyä, symmetrisyyttä tai tasapainoa. Toisaalta suureet voivat liittyä esimerkiksi elementin tai leipätekstialueen varaamaan pinta-alaan sivulla, jolloin sitä verrataan määritettyyn toivearvoon.

Sommitelman arviointia varten luotiin rajapinta, jolla uusia arviointimenetelmiä voidaan lisätä helposti. Yksittäistä mallinnettua arviointimenetelmää kutsutaan arvioitsimeksi. Arvioitsimia varten suunniteltiin abstrakti luokkakuvaukseen, joka hyödyntää *template method* -suunnittelumallia. Mallissa luokkaan määritetään abstrakti metodirajapinta, joka lapsiluokan tulee toteuttaa. Kaikille periytetyille luokille yhteinen isäluokan algoritmi hyödyntää rajapintaa kutsuessaan lapsiluokan toteutusta. [20, s. 325.] Konkreettiset arvioitsijat toteuttavat abstraktiksi määritellyn arviointimetodin. Arviointimetodin tulee palauttaa mitatun suureen pisteytys reaalitylukuna nollasta yhteen, nollan tarkoittaen huonointa ja yhden parasta tulosta. Arvioitsimen perusluokka toteuttaa arviointifunktion, joka suhteuttaa arviointimenetelmän arviointimetodin laskeman pisteytyksen määritetyn painoarvon mukaisesti.

Arvioitsimet ovat laskennallisesti usein vaativampia kuin yksinkertaiset säännöt. Siinä, missä säännöt keskittyvät lähinnä yksittäisen elementin sijaintiin sivulla, arvioitsimet saattavat esimerkiksi etsiä sivulta kaikki tietyntyyppiset elementit ja laskea näiden pinta-alan.

Useammat arvioitsimet kerätään linkitettyinä listana ja arviointia kutsutaan rekursiivisesti hieman kuten *chain of responsibility* -suunnittelumallissa. Mallissa käsittelijät saavat viestin parametrikseen ja käsittelevät sen, mikäli se kuuluu käsittelijän vastuualueelle. Muussa tapauksessa käsittelijä antaa viestin eteenpäin seuraavan tason käsittelijälle. [20, s. 223.] Kuva 8 havainnollistaa arvioitsimen rakennetta.



Kuva 8. Yksinkertaistettu arvioitsimen luokkakaavio ja arvioitsinketjun läpikäyvän algoritmin kuvaus.

Erona *chain of responsibility* -malliin toteutuksessa on se, että vastuu alkaa käänteisesti heti ja jatkuu, kunnes minimipistemäärää ei voida enää saavuttaa. Arviota lasketaan jokaisella tasolla, kunnes huomataan, että alempien tasojen painokertoimilla tulos ei voi enää ylittää aiempaa maksimia. Tällaisessa tilanteessa arviointi lopetetaan kesken ajan säästämiseksi. Tehokkuuden varmistamiseksi kaikki arvioitsimet tuntevat seuraavien tasojen yhteenlasketut painokertoimet.

Tarkemmat kuvaukset työn ohessa toteutetuista arvioitsimista on esitetty vain työn ti-laajalle toimitetussa liitteessä 2. Joukosta löytyy esimerkiksi leipätekstialueen yhtenäisyyttä ja pinta-alaa mittaavat arvioitsimet.

5 Tekstin juoksutuksen toteutus

Tässä luvussa kuvataan tekstin juoksutuksen ongelmia sekä käytettyjä ratkaisumalleja. Mahdollisimman suuri osa typografisista asetuksista ja päätöksenteosta jätettiin käyttäjän määritettäväksi parametreiksi kappale- ja merkkityyleihin. Moniin typografisiin seikkoihin, kuten kirjasintyylien tai pistekokojen valintaan, kursivien tai lihavointien käyttöön toteutus ei ota lainkaan kantaa. Tällaiset asiat on edelleen ammattilaisen määritettävä järjestelmään. Merkkityyleihin määritettäviä parametreja ovat kirjasintyyli, pistekoko, väri, lihavointi sekä kursivi. Kappaletyyliin määritetään lisäksi tasaus, riviväli, sisennykset sekä kappaleväli ennen ja jälkeen.

Sommitteluratkaisun luonteesta johtuen tekstiä joudutaan juoksuttamaan useita kertoja hyvää lopputulosta etsiessä. Tästä syystä tekstin juoksutuksen tulee olla nopeaa, jottei se hidasta merkittävästi eri taittovaihtoehtojen läpikäyntiä.

5.1 Tavutus

Tekstin tavutustieto tuodaan tekstin mukana toimitusjärjestelmästä. Koska tekstiä juoksutetaan satoja kertoja taittoprosessin aikana, on tekstin oltava tehokkaasti luettavissa mistä tahansa kohtaa siten, että mikä tahansa kuvannin ymmärtää tekstipätkän. Näin ollen tavutustietoa ei voi upottaa tekstin sekaan edes merkkijonossa näkymättömällä unicode soft hyphen -merkeillä [21].

Tavutustietoa varten luotiin tekstin rinnalle tekstin pituinen bittikartta. Bittikartassa on bitti jokaista tekstin merkkiä kohden. Esimerkki tavutusbittikartasta on esitetty kuvassa 9.

E	s	i	m	e	r	k	k	i		t	e	k	s	t	i
0	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0

Kuva 9. Esimerkki tekstiin liitetystä tavutusbittikartasta.

Kuvan ensimmäinen rivi esittää tekstiä ja toinen rivi tähän liitettyä bittikarttaa. Tosi-arvo bittikartassa tietyn merkin kohdalla tarkoittaa, että teksti voidaan tavuttaa kyseistä merkkiä edeltävästä kirjainvälistä.

Käytännössä toteutettu rivitysalgoritmi ottaa tekstistä aina yksittäisen sanan kerrallaan ja mittaa sen leveyden käyttäen kuvantamisrajapintaa sekä etenee, kunnes rivin leveys ylitetään tai jäljellä on alle tavun verran leveyttä. Mikäli rivileveys ylitettiin, etsitään ylitäneen sanan tavukohdat bittikartasta ja lyhennetään sanaa lopusta lähtien, kunnes sanan osa mahtuu tavuviivan kanssa riville.

5.2 Rivirekisteri

Estetiikan takia leipätekstin peruslinja on läpi sivun pidettävä samassa tasossa rinnakkaisten palstojen kesken. Toimintoa kutsutaan yleisesti rivirekisteriksi. Rivirekisteriä sekoittavat pystysuunnassa eri tasoista alkavat palstat, tekstin väliin tulevat väliotsikot sekä muut nostot.

Rivirekisteri on leipätekstiksi määritetyn kappaletyylin yhteenlaskettu rivikorkeus ja rivi-väli. Lisäksi rivirekisterin laskentaa varten lisätään sivukohtainen, marginaalien mukaan määräytyvä siirtymä siten, että ylämarginaali osuu rivirekisteriin. Toteutetussa sovel-luksessa rivirekisterin hallinta jätettiin kokonaan tekstinjuoksutusalgoritmin hallittavaksi, lukuun ottamatta sivukohtaisen siirtymän asetusta, joka annetaan algoritmille paramet-rina.

5.3 Leski- ja orporivien hallinta

Rivirekisterin jälkeen ehkä silmiinpistävin typografinen ongelma ovat orporivit. Orpori-veillä tarkoitetaan kappaleen viimeistä riviä joka siirtyy uuden palstan tai sivun alkuun. Orporivi aiheuttaa lukiessa katkon väärään kohtaan. Leskirivillä viitataan taas palstan tai sivun loppuun jääneeseen uuden kappaleen ensimmäiseen riviin. Toisinaan leskellä tarkoitetaan myös yhden tavun mittaiseksi jäänyttä kappaleen viimeistä riviä. [17, s. 31.]

Yleisin keino orpo- ja leskirivien välttämiseen on säätää merkkiväliä hieman. Merkkivä-liä kaventamalla yhdeltä tai useammalta edeltävältä riviltä voidaan orvoksi tai leskeksi jäänyt saada mahtumaan edelliselle riville. Toisaalta levittämällä merkkiväliä voidaan saada uusi rivi aikaan. [17, s. 32.]

Merkkiväliä ei voida kuitenkaan säätää mielivaltaisesti. Painomediassa pistetarkkuus on täysin eri luokkaa kuin sähköisessä mediassa. Toteutettu rivitettyä tekstiä kuvaava tietorakenne käsittelee tekstiä pikselikoordinaatistossa ja usein yhdenkin pikselin muu-tos tekstin merkkivälissä voi olla liikaa. Tästä syystä merkkivälin muuttaminen jätettiin työn tässä vaiheessa toteuttamatta.

Vaihtoehtoisena ratkaisuna päädyttiin katkaisemaan tekstikappaleet siten, että orpo- ja leskirivejä ei synny. Mikäli tekstiä saadaan juoksutettua palstan loppuun alle vaaditun rivimäärän, perutaan juoksutus ja siirrytään seuraavaan palstaan. Vastaavasti, jos palstan alussa huomataan, että kappaleesta ei synny tarpeeksi rivejä, käydään poistamassa edellisestä palstasta tarvittava määrä rivejä. Poistettavien rivien määrä ei aina ole pelkästään riippuvainen rivien tarpeesta orvon välttämiseksi. Samalla pitää myös tarkistaa, ettei edelliseen palstaan luoda uutta leskeä. Tällaisessa tapauksessa koko kappale siirretään seuraavalle palstalle tai sivulle.

Leski- ja orporivien hallintaan käyttäjä voi vaikuttaa kappaletyyliin asetuksilla. Kappaletyyleihin voidaan määrittää sallittujen leski- ja orporivien määrä, molemmat erikseen. Lisäksi kappaletyyliin voidaan määrittää kappale liitettäväksi sitä seuraavaan kappaleeseen, jolloin sen ei sallita jäävän palstan tai sivun viimeiseksi.

1. `<flow keep-with-next="yes" orphan-control="0" widow-control="0" hyphenate="no"/>`
2. `<flow orphan-control="2" widow-control="2"/>`

Koodiesimerkki 2. Kaksi esimerkkiä tekstin juoksutusasetusten XML-määrittämisestä.

Koodiesimerkin 2 ensimmäisellä rivillä on esimerkiksi väliotsikolle sopivat juoksutusasetukset. Kappale on määritetty pysymään kiinni seuraavassa kappaleessa ja asettamalla sallittujen orpo- ja leskirivien määrä nolnaan kielletään kappaleen jakaminen kokonaan. Lisäksi esimerkissä on kytketty kappaleen tavutus pois. Esimerkin toisella rivillä on leipätekstille sopiva määrittäminen, jossa vaaditaan orpo- ja leskirivien vähimmäismääräksi kahta riviä.

5.4 Otsikot ja väliotsikot

Otsikoihin ja väliotsikoihin liittyy niin ikään useita typografisia sääntöjä. Monirivisiin otsikoihin pätevät samat säännöt kuin liehupalstaan. Rivipituudet tulisi pitää mahdollisimman tasaisina, kuitenkin siten, että ensimmäinen ja viimeinen rivi ovat muita lyhyempiä. Tasauksesta tekee kuitenkin haastavaa se, että otsikoita ei kuulu tavuttaa. Lisäksi suuremman pistekoon takia suhteellinen palstaleveys pienenee, varsinkin väliotsikoiden tapauksessa, kun otsikkoa ei voida levittää useamman palstan alueelle. Esimerkiksi suomenkielessä tavutussääntöön tekee poikkeuksen yhdyssanat, jotka voidaan tavuttaa yhdysvälistä. Lisäksi rivityksessä tulisi huomioida otsikon asiasisältö. [17, s. 28.]

Yhdyssanaongelman pystyisi selvittämään ohjelmallisesti, mutta se vaatisi hyvin kehittyneen algoritmin kattavalla sanastolla. Sisällön analysointi rivitystä varten on käytännössä mahdoton toteuttaa ohjelmallisesti. Kummatkin ongelmat jätettiin toteutuksessa huomiotta. Sen sijaan rivitykseen toteutettiin tuki pakotettuihin rivinvaihtoihin. Näin vaikeat ja pitkät otsikot voidaan tarvittaessa rivittää valmiiksi jo toimitusjärjestelmässä.

Leipätekstin väliin sijoittuvat väliotsikot aiheuttavat hyppäyksiä rivirekisterissä, kun seuraava leipätekstirivi joudutaan taas kohdistamaan rivirekisteriin. Hallitsematta tätä tilannetta oikein päädytään herkästi tilanteeseen, jossa väliotsikon jälkeen jää huomattavasti tyhjää tilaa. Koska väliotsikko liittyy nimenomaan seuraavaan kappaleeseen, kuuluu sen olla lähempänä seuraavaa kuin edellistä riviä. Yksi ohje tyhjän tilan suhteeseen väliotsikon ylä- ja alapuolella on 2:1, mutta yhtä ainoata sääntöä ei ole. [17, s. 33.]

Toteutuksessa väliotsikon etäisyys jätettiin osittain kappaletyyliin määritettäväksi parametriksi siten, että tyhjän tilan kappaleen ylä- ja alapuolelle voi määrittää erikseen. Lisäksi rivirekisterin muuttuessa väliotsikkoa siirretään alaspäin siten, että sen ja seuraavan kappaleen väliin jää määritetyn verran tyhjää tilaa.

6 Sivutuksen toteutus

Aikaisemmin käsiteltiin lähinnä yksittäisen sivun sisällä tehtäviä ratkaisuja. Onnistunutta kokonaisuutta varten koko taittokokonaisuuden tulee olla tasapainoinen. Sivut saavat toki olla keskenään erilaisia, mutta esimerkiksi puolityhjä sivu näyttää käytännössä poikkeuksetta huonolta, varsinkin jos sisältö on sijoitettu epätasapainoisesti.

Kuvituksen on tarkoitus elävöittää tekstiä. Tilanne, jossa artikkelin kaikki kuvat on sijoitettu ensimmäiselle sivulle, ei ole käytännössä koskaan toivottava, mikäli teksti jatkuu useammalla sivulla. Kuvamateriaali olisikin hyvä saada jaettua mahdollisimman hyvin tarvittavalle sivumäärälle. Tavoitetta vaikeuttaa kuitenkin mahdollisten määritettyjen taittosääntöjen rajoitukset sekä artikkelin toimituksellinen rakenne ja kuvamateriaali.

Aikaisemmissa ratkaisuista ainakin Jacobs, Li, Schrier, Barger ja Salesin [15] ottivat kantaan myös sivutukseen. Heidän mallissaan kiinnitettiin huomiota lähinnä tekstin jakaantumiseen, sillä kuvien paikat ja koot määritettiin ruudukkomalleissa. Ratkaisussa

tekstin jakautumista optimoitiin kokeilemalla erilaisia mallisekvenssejä dokumentin sivutuksessa ja taitossa.

Toteutettu ohjelma rakentaa sivukokonaisuuden yrityksen ja erehdyksen kautta, taittamalla rekursiivisesti sivun kerrallaan. Mikäli tasapainossa huomataan ongelma, jota ei voida sen hetkellä sivulla korjata, algoritmi palaa edellisen sivun taittoon ja pyrkii tekemään helpottavat muutokset täällä. Koodiesimerkki 3 havainnollistaa idean yksinkertaistettuna. Esimerkissä sivun taittoa ohjataan ohjausparametreilla, joita päivitetään jokaisen taittoyrityksen jälkeen, kunnes kyseisellä tasolla muutoksia ei voida enää tehdä.

```

1. function taita(tekstivirta T, kuvavirta K)
2.     // Aloitetaan laskemalla ohjausparametrit
3.     // jäljellä olevalle materiaalille.
4.     ohjausparametrit P := laske_ennakointiparametrit(T, K)
5.     do
6.         // Luodaan sommitelma ohjausparametreilla P.
7.         sommitelma := luo_sommitelma(P, T, K)
8.         // Juoksutetaan tekstiä sommitelmaan.
9.         juoksuta_teksti(sommitelma, T)
10.    if T == tyhjä
11.        // Viimeinen sivu, palautetaan sommitelma ja ohjetiedot
12.        // kuvien ja pinta-alan käytöstä takaisin ylemmälle tasolle.
13.        return sommitelma, jäljellä(K), käytetty pinta-ala(sommitelma)
14.    else
15.        // Jatketaan rekursiota seuraavalle sivulle.
16.        sommitelma, kuvapalaute, alapalaute := taita(T, K)
17.        // Lasketaan uudet ohjausparametrit saadun palautteen perusteella
18.        P := päivitä_ennakointiparametrit(T, K, kuvapalaute, alapalaute)
19.    while ohjausparametrit P muuttuneet // Jatketaan sommittelun optimointia,
20.                                         // kunnes tilannetta ei voida korjata.
21.    // Palautetaan paras sommitelma kuva- ja alapalautteineen ylemmälle tasolle.
22.    return sommitelma, kuvapalaute, alapalaute

```

Koodiesimerkki 3. Rekursiivisen sivutusalgoritmin yksinkertaistettu kuvaus.

Kuten koodiesimerkin 3 kuvauksesta käy ilmi, jokainen kokonaisuus taitetaan aina ensin materiaalin loppuun asti, huolimatta materiaalin epätasaisesta jakautumisesta. Rekursiivinen algoritmi ei pohjatasolla tiedä, onko parempaan lopputulokseen edes mahdollista päästä. Jokainen loppuun laskettu taittokokonaisuus analysoidaan ja pisteytetään. Pisteytyksessä otetaan huomioon kaikkien sivujen sommittelupisteytys sekä painotetusti viimeisen sivun tilanne. Tyhjä tila ja kuvien puute viimeisellä sivulla laskee pisteytystä. Aina paremman kokonaisuuden löydyttyä aiempi hylätään, eikä erilaisia vaihtoehtoja oteta talteen. Näin ollen myöskään eri vaihtoehtoja ei näytetä käyttäjälle.

6.1 Ennakointi

Ylimääräisten erehdysten välttämiseksi algoritmiin toteutettiin piirteitä, joilla pyritään etukäteen arvioimaan artikkelin sivumäärää. Kun suuntaa antava sivumäärä on tiedossa, voidaan kuva- ja muu materiaali jakaa alustavasti jo ennen ensimmäistä yritystä ja erehdys -kierrosta.

Tekstin juoksutuksen havaittiin olevan kuitenkin niin nopeaa, että perinteiset arviointimenetelmät sivuutettiin kokonaan. Ennen ensimmäistäkään taittoyritystä on sivun palstaleveys tiedossa. Tietoa käytetään tässä kohtaa hyväksi siten, että koko artikkelin teksti juoksutetaan yhteen palstaan. Koska artikkelin teksti ingressiä ja otsikkoa lukuun ottamatta juoksutetaan vakiolevyiseen palstaan, tuloksena saadaan suoraan mahdollisimman tarkka arvio todellisesti tarvittavasta tekstialasta.

Kuville varattava pinta-ala on puolestaan hieman vaikeampi arvioida. Käytännössä se joudutaan arvaamaan. Mikäli taittosääntöihin on määritetty suhteellinen kuvien toivepinta-ala, voidaan tätä tietoa käyttää arviona. Jos toivepinta-alaa ei ole määritetty, pitää tyytyä kiinteästi määritettyyn vakioarvoon. Toivepinta-alan lisäksi kuvan kuvasuhteesta voidaan päätellä, onko toive edes saavutettavissa. Leveän vaakakuvan tapauksessa pinta-ala voi jäädä pieneksi, vaikka kuva levitettäisiin kaikkiin saatavilla oleviin palstoihin. Pystykuvaa taas ei välttämättä mahduta jakamaan useammalle palstalle, jos sivun korkeus tulee ensin vastaan. Lisäksi erikseen määritettävissä taittosäännöissä voidaan rajata tai jopa määrätä, kuinka monelle palstalle kuva levitetään. Tällaisessa tapauksessa pinta-alasta saadaan kohtuullisen tarkka arvio.

Kuvien pinta-ala-arvion epätarkkuuden takia kokonaissivumäärän arvioinnista tulee melko hankalaa. Lisäksi typografisten sääntöjen johdosta sama teksti voi käyttää hyvin erilaisen pinta-alan palstakorkeuksien vaihtuessa. Suurin syy tähän on leski- ja orpovien hallinta sekä väliotsikon kaltaiset kappaleet, joita ei saa jakaa lainkaan. Mahdollisimman tarkkaa arviota varten arviointi suoritetaan uudestaan rekursion jokaisella tasolla, aina jäljellä olevan materiaalin mukaan. Materiaalimäärän pienentyessä arvio tarkentuu koko ajan.

6.2 Materiaalin jakautuminen

Rekursion jokaisella tasolla kuvia otetaan aluksi käyttöön ensin lasketun arvion mukaan, jos mahdollista. Lisäksi arvioidun kokonaispinta-alan mukaan annetaan sommitelualgoritmilte sivun leipätekstialueen arvioija, joka suosii lähimmäksi tavoitepinta-alaa pääseviä taittovaihtoehtoja.

Viimeisen sivun analyysissä lasketaan ohjeparametrit korjausyritystä varten. Esimerkiksi kuvattomuus luo ohjeen vähentää kuvia aikaisemmilla sivuilla. Tyhjäksi jääneen pinta-alan määrä ohjaa edellisiä tasoja valitsemaan paremmin sivut täyttävän sommitelman. Aiemmin esitetty Koodiesimerkki 3 havainnollistaa ohjeparametrien välitystä sivutusprosessissa. Kun rekursio palaa ylöspäin, luetaan pisteet ja ohjeparametrit ja taittokokonaisuus otetaan talteen. Ohjeparametrit tulkitaan ja nykyisen sivun taittoa mukautetaan niiden mukaan, mikäli mahdollista. Uudelleentaiton jälkeen jatketaan taas rekursiivisesti, kunnes palataan samaan tasoon ja saadaan paluuarvona uusi taittokokonaisuus. Kokonaisuutta verrataan aikaisemmin valittuun ja näistä paremmin pisteytetty jatkaa. Mikäli uusi vaihtoehto oli vanhaa parempi, jatketaan lukemalla ohjeparametrit ja mukauttamalla uudestaan niin pitkään kuin kokonaisuuksien pisteet kasvavat.

Kun mukautus ei enää tuota paremmin pisteytettyä kokonaisuutta tai nykyisen tason taittoa ei voida muuttaa ohjeparametrien mukaan, palautetaan valittu kokonaisuus pisteineen ja ohjeparametreineen taas ylemmälle tasolle.

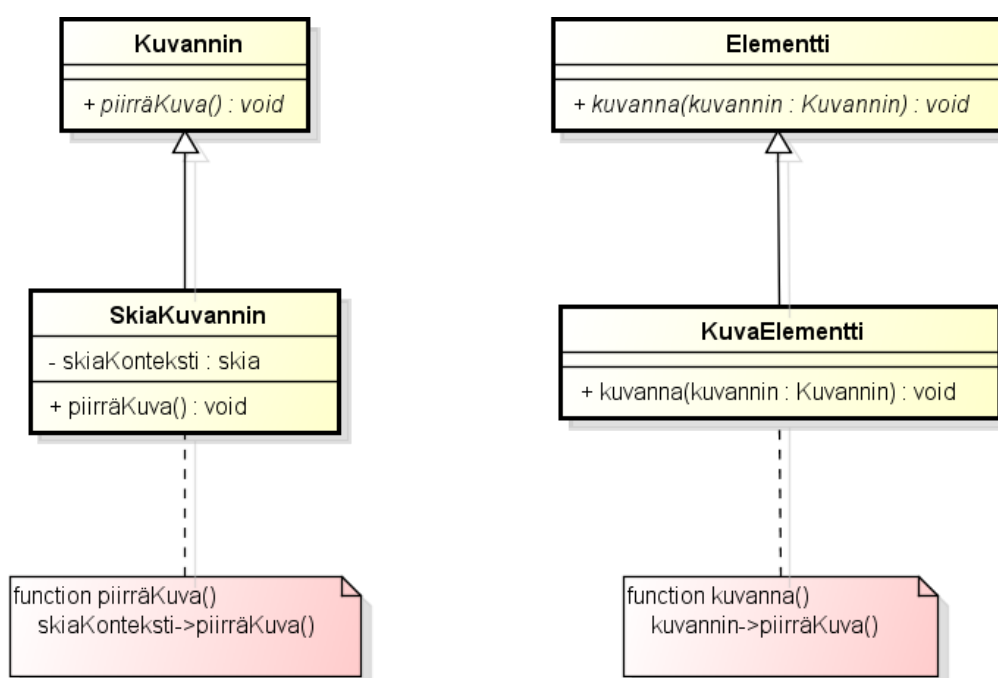
Mikäli ajaudutaan tilanteeseen, jossa kuvat ovat loppuneet mutta tekstiä riittää uudelle sivulle, algoritmi palaa takaisin ja pyrkii löytämään mahdollisia paikkoja jakaa kuvia tasaisemmin. Aikaisemmille sivuille mahdollisesti määritettyjen taittomääritysten takia kuvia ei voida ottaa miltä tahansa sivulta, jos määritys määrää tarkasti sivulla käytettävien kuvien määrän. Mikäli joukosta kuitenkin löytyy sivu jolta kuvamäärää voidaan vähentää, aloitetaan taitto uudestaan tästä sivusta, mikäli kuvia jää sivulle vähintään yksi.

7 Kuvantamisen toteutus

Taittokokonaisuuden löydyttyä se pitää saada kuvannettua graafiseen muotoon esitystä varten. Kuvantaminen on prosessin suoraviivaisin osa, sillä tässä vaiheessa kaikki on laskettu valmiiksi, eikä mitään tarvitse enää muuttaa.

Työn ohessa toteutettiin PDF- ja PNG-kuvantimet käyttäen Skia-kirjastoa. Skia oli alun perin Skia Inc-yrityksen kehittämä kirjasto, jonka Google osti 2005 [22]. Nykyisin Skia-kirjastoa käytetään muun muassa Mozilla Firefoxissa, Google Chromessa, Chrome OS:ssä, Chromium OS:ssä sekä Androidissa [23]. Kirjasto osaa tällä hetkellä käyttää kuvantamislaitteena rasteroitua kuvaa, OpenGL:ää, PDF:ää ja XPS:ää [19].

Kuvannintoteutus vastaa *adapter*-suunnittelumallia. Mallissa luokan rajapinta muutetaan sellaiseen muotoon, jossa asiakas osaa sitä hyödyntää. Adapter toteuttaa määritellyn kaltaisen rajapinnan ja muuntaa kutsut taustalla olevan konkreettisen luokan ymmärtäviksi. [20, s. 139.] Kuva 10 esittää yksinkertaistetun idean kuvannin- ja sivuelementtitoteutuksesta.



Kuva 10. Kuvannin- ja elementtitoteutuksen idea.

Kuvan yläosassa on esitelty rajapinnat määrittävät abstraktit luokkakuvaukset. SkiaKuvannin esittää toteutettua konkreettista kuvanninta, joka tuntee Skia-kirjaston rajapinnat

ja toteuttaa abstraktit *adapter*-metodit. Taittovaiheessa sivun sisältö kuvattiin listana sivuelementtejä. Kaikki sivuelementit toteuttavat piirtorutiinin, jolle viite kulloinkin käytettyyn kuvantimeen annetaan parametreina. Sivuelementti tietää sijaintinsa sekä sisällönsä ja kutsuu kuvantimen piirtorutiineja kuvantaakseen itsensä. Kuvassa esitetty KuvaElementti on esimerkki konkreettisesta sivuelementistä, joka toteuttaa rajapinnan määrittämisen kuvanna-metodin ja käyttää kuvantimen rajapintaa kuvantaakseen itsensä.

Kuvantimen rajapintaan määritettiin vain tarvittavat piirtorutiinit, mahdollisimman yksinkertaisesti siten, että ne ovat toteutettavissa lähes millä tahansa grafiikkakirjastolla.

```

1. function kuvanna(sivu S, kuvantaja K)
2.     // alustetaan käyttöön sivun kokoinen lakana
3.     K->alusta_lakana(S->geometria)
4.     for each elementti in S
5.         // kuvannetaan jokainen sivuelementti järjestyksessä
6.         elementti->kuvanna(K)

```

Koodiesimerkki 4. Kuvannussilmukan kuvaus.

Koodiesimerkki 4 havainnollistaa sivun kuvantamisen suoraviivaisuutta. Jokaista taitettua sivua varten alustetaan määritetyn kokoinen lakana, jota kuvantaja käyttää taustalla kuvantamispohjana. Seuraavaksi käydään sivun elementit läpi yksitellen, kutsutaan niiden kuvantamismetodia antaen kuvantimen viite parametrina. Koska konkreettisen, kuvannetun sivun esitystapaa ei etukäteen tiedetä, rajapinnat eivät määritä sitä, kuinka kuvannettu sivu luetaan kuvantimesta. Käytännössä kuvanninilmentymät luodaan taittoalgoritmin ulkopuolella kontekstissa, jossa julkaisukanava ja siihen liittyvä konkreettinen kuvantaja tunnetaan. Vastuu kuvannetun sivun talletuksesta tai julkaisusta jää siis algoritmin ulkopuolelle.

7.1 Grafiikka / Geometriset kuvat

Geometrisiä kuvioita varten toteutettiin ainoastaan yksi piirtorutiini, suorakulmion piirtäminen. Piirtorutiini on kuitenkin monikäyttöinen, sillä suorakulmion lisäksi sillä voidaan piirtää myös viiva, kun toinen sivu määritetään viivaleveyden levyiseksi. Toki rajoituksena on se, että viivan tulee olla joko täysin pysty- tai vaakasuora.

Tarvetta monimutkaisemmille geometrisille kuvoille ei todennäköisesti kuitenkaan ole. Sisältöön liittyvä grafiikka on käytännössä aina valmiiksi rasteroitu kuvaksi. Ylä- ja alitunnisteissa harvemmin tarvitaan muuta, kuin viivaa tai suorakulmiota, monimutkaisempi grafiikka voidaan tuoda rasteroituna kuvana.

7.2 Teksti

Tekstiä varten kuvantimeen toteutettiin kolme rutiinia. Ensimmäinen lataa annetun fontin kokotiedot taittoa varten. Toinen mittaa parametrina annetun tekstin leveyden. Kolmas piirtää annetun tekstin annettuun koordinaattiin. Mittaus ja piirtorutiinit ottavat lisäksi viitteen käytettävään fonttiin parametrina.

Juoksuttaessa teksti kuvataan sivuelementtiin, jonka datapuskuriin talletetaan kompakti lista sanojen x-koordinaateista ja alkuperäiseen tekstiin viittaavista osoitteista. Rivin ja merkkityylin vaihtoja varten listassa on oma koodauksensa, samoin rivin loppuun osuneita osittaisia sanoja, joiden perään lisätään tavuviiva.

Sivuelementin piirtorutiini käy datapuskurin läpi vaihe vaiheelta, pitäen samalla kirjaa kulloinkin voimassa olevasta merkkityylistä. Piirto alkaa aina elementin määritetystä vasemmasta yläkulmasta, vasempaan reunaan lisätään aina sanan x-koordinaatti ja rivinvaihtojen yhteydessä siirrytään aina rivikorkeuden verran alaspäin.

7.3 Kuvat

Kuvien sivuelementit tallettavat sivukoordinaattien lisäksi tiedostopolun itse kuvatiedostoon sekä lähdekoordinaatit mahdollista rajausta varten.

Kuten useimmat 2D-grafiikkakirjastot, myös Skia toteuttaa valmiiksi rajapinnan, jolla kuvan tai rajatun osan kuvasta saa piirrettyä määrittämällä lähde- ja kohdekoordinaatit.

Kuvantamisrajapintaan toteutettiin siis vain yksi rutiini kuvan piirtämistä varten. Rutiini saa parametrikseen kuvan tiedostopolun, lähdekoordinaatit sekä kohdekoordinaatit. Kuvannintoteutuksen vastuulle jää ladata varsinainen kuva tiedostosta.

8 Tulokset

Tavoitteena oli kuvata taiton graafinen ohjeistus ja toteuttaa ohjelma, joka sen perusteella taittaa lehtiartikkelista yksinkertaisen, mutta helppolukuisen ja silmää miellyttävän, sivutetun kokonaisuuden. Palstaleveyteen sidottujen elementtien sijoituksen johdosta taittoratkaisut ovat melko yksinkertaisia ja muistuttavat sanomalehdistä tuttua tyyliä. Palstaruudukkoon perustuva taittoalgoritmi tuottaa useimmiten tasapainoisen ja tasatun lopputuloksen, joka on helposti hahmotettavissa. Esimerkki algoritmin automaattisesti taittamasta sivusta on esitetty kuvassa 11.



Kuva 11. Algoritmin automaattisesti iPad-tablet tietokoneen näytölle taittaman artikkelin ensimmäinen sivu pysty- ja vaakatasossa.

Kuva havainnollistaa, kuinka taitto mukautuu käytettävissä olevaan alueeseen. Konkreettisia esimerkkejä eri tablet-tietokoneille mukautetuista taitoista on esitetty kuvassa 12.



Kuva 12. Algoritmin automaattisesti taittamaa sisältöä Apple iPad 3-, Samsung Galaxy Tab 7.7- ja Apple iPod Touch 4 -laitteissa.

Yksinkertaiset taitot ovat palstoituksen johdosta helppolukuisia ja soveltuvat hyvin uutis- ja asiasisällön esittämiseen. Tällaisenaan ohjelma soveltuu hyvin sanomalehtisisällön tai tekstipainotteisen aikakauslehtiartikkeleiden julkaisemiseen. Sovelluksessa ei ole mahdollista tehdä ristiviitettä tekstin ja kuvan välillä, joten usein kuviin viittaavan tekstin julkaisemiseen algoritmi ei tällaisenaan sovellu. Algoritmia kokeiltiin myös painetun materiaalin taittamiseen. Mahdollisia sovelluskohteita ovat myös esimerkiksi erilaiset tiedotteet, jotka saadaan kerran tehtyjen määritysten jälkeen julkaistua ammattimaisesti taitettuna, ilman lisätyötä.

Samanlaisella materiaalilla ja sääntömäärittelyillä algoritmi tuottaa helposti toinen toistaan muistuttavia taittoja, mutta materiaalimäärän vaihtuessa ja esimerkiksi erimuotoisten kuvien kanssa taitot saavat kuitenkin herkästi erilaisen ilmeen.

Algoritmi täyttää nykyisellään kaikki sivut niin tiivisti kuin mahdollista. Toisinaan tämä johtaa silmää epämiellyttävään tilanteeseen, jossa sivu on täynnä ja tukkoinen. Jatkokehityksenä sovellukseen voisi uusien sääntöjen ja arvioitsimien lisäksi suunnitella tavon, jolla halutut paikat saadaan jätettyä ilmavammiksi. Yksi mahdollisista toteutustavoista on lisätä elementeille toisiinsa nähden sallittujen välien ylä- ja alarajat. Toinen

mahdollinen lähestymistapa on määrittää taittoon tyhjä elementti, joka varaa ruudukosta halutun verran tilaa.

Jatkossa monipuolisempaan taittoon voidaan pyrkiä jakamalla ruudukkoa enemmän siten, että kuvia ja tekstilaatikoita voitaisiin sijoittaa esimerkiksi puolen palstan väleihin. Uudet elementtikoot antaisivat sivutusalgoritmile lisäksi paremmat mahdollisuudet ohjata tekstin jakautumista. Toisaalta ruudukon jakaminen kasvattaa sommitteluiden haakuvaruutta entisestään moninkertaisesti.

Hakua kasvavassa avaruudessa voidaan yrittää optimoida muuntamalla sijoitusalgoritmi rekursiivisen toteutuksen sijasta geneettiseksi, hieman kuten Purvis, Harrington, O'Sullivan ja Freuder [7] ovat työssään tehneet. Pistekoordinaatiston sijasta geeneiksi voitaisiin koodata elementtien ruudukkokoordinaatit.

Oman haasteensa, mutta myös mahdollisuutensa, tuo sommitteluun kuvien rajaaminen. Rajaamisessa tulee kuitenkin huomioida kuvan sisältö, jotta esimerkiksi henkilöiden kasvojen pois rajaamiselta vältyttäisiin. Kuvien automaattinen rajaaminen vaatii kuvan analysointia, mutta jos tämä tieto on saatavilla, voidaan automaattista taittoa kehittää huomattavasti.

Nähtäväksi jää, onko perinteisen typografian sääntöjen mukaan taitetulla ja sivutetulla aineistolla kysyntää sähköisen julkaisun markkinoita. Helposti luettavan ja selkeän taiton ystävät varmasti arvostavat sääntöjen kunnioittamista myös sähköisissä julkaisuissa. Jos markkinoille vakiintuu sopiva laitekanta ja kuluttajat haluavat julkaisijalta työssä kuvattujen sääntöjen noudattamista, on tilaajayrityksellä vahva ohjelmisto ja reilu etumatka kilpailijoihin.

Lähteet

- 1 Tuomo Telkkä. 2013. Järjestelmäarkkitehti, Anygraaf Oy, Helsinki. Haastattelu 22.4.2013.
- 2 Milla Toro. 1999. DTP & painotyö - Käytännön opas painotuotteiden tekijöille ja tilaajille. Helsinki: Inforviestintä Oy.
- 3 Anygraaf. 2012. Neo Publishing Solution by Anygraaf. Verkkodokumentti. <http://www.anygraaf.fi/main/rightcol_fin/neo_publishing_solution_68.html>. Luettu 27.4.2013.
- 4 Verkkotietojärjestelmä. <http://en.wikipedia.org/wiki/Proof_of_concept>. Luettu 2.5.2013.
- 5 Ida-Maria Kivelä, Raisa Halonen, Mikko Kuhna & Pirkko Oittinen. 2011. Current state-of-the-art of automatic layout tools – Literature review.
- 6 Joe Geigel, Alexander Loui. 2001. Automatic Page Layout Using Genetic Algorithms for Electronic Albuming.
- 7 Lisa Purvis, Steven Harrington, Barry O’Sullivan, Eugene C. Freuder. 2003. Creating Personalized Documents: An Optimization Approach.
- 8 Simon Lok, Steven Feiner. 2001. A survey of automated layout techniques for information presentations.
- 9 Louis Weitzman, Kent Wittenburg. 1993. Relational grammars for interactive design.
- 10 Louis Weitzman, Kent Wittenburg. 1994. Automatic presentation of multimedia documents using relational grammars.
- 11 Thomas Weise. 2009. Global Optimization Algorithms – Theory and Application. E-kirja. <<http://www.it-weise.de/projects/book.pdf>>. Luettu 2.5.2013.
- 12 Winfried Graf. 1992. Constraint-Based Graphical Layout of Multimodal Presentations.
- 13 Eamonn O’Brien-Strain, Jerry Liu. 2010. Learning from graphic designers: using grids as a scaffolding for automatic print layout.
- 14 Verkkotietojärjestelmä. <[http://en.wikipedia.org/wiki/Grid_\(graphic_design\)](http://en.wikipedia.org/wiki/Grid_(graphic_design))>. Luettu 8.5.2013.

- 15 Charles Jacobs, Wil Li, Evan Schrier, David Pargeron, David Salesin. 2003. Adaptive grid-based document layout.
- 16 Charles Jacobs, Wil Li, Evan Schrier, David Pargeron, David Salesin. 2004. Adaptive document layout.
- 17 Markus Itkonen. 1999. Typoteesejä Tarkan typografian opas. Helsinki: RPS-yhtiöt
- 18 Markus Itkonen. 2004. Kirjatypeografia. Kustannustoimittajan käsikirja. Tampere: Suomen Kustannusyhdistys.
- 19 Verkkodokumentti. <<https://sites.google.com/site/skiadocs/>>. Luettu 22.4.2013.
- 20 Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. 1995. Design Patterns, Elements of Reusable Object-Oriented Software. USA: Addison Wesley Longman, Inc.
- 21 Verkkotietojärjestelmä. <http://en.wikipedia.org/wiki/Soft_hyphen>. Luettu 1.5.2013.
- 22 Om Malik. 2008. Verkkodokumentti. GigaOm. <<http://gigaom.com/2008/09/02/google-open-sources-skia-graphics-engine/>>. Luettu 22.4.2013.
- 23 Verkkotietojärjestelmä. <http://en.wikipedia.org/wiki/Skia_Graphics_Engine>. Luettu 22.4.2013.