Tietojenkäsittelytiede **28** Joulukuu 2008 sivut 15–30
Toimittaja: Antti Valmari

Mitä saavutettaisiin algoritmianimaatiokielistandardilla?

Ville Karavirta
Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan laitos
vkaravir@cs.hut.fi

Tiivistelmä

Tämä artikkeli perustuu diplomityöhöni "XAAL – Extensible Algorithm Animation Language", joka esittelee algoritmianimaatiokielten luokittelun sekä uuden algoritmianimaatiokielen.

1 Johdanto

Ohjelmistot ovat nykypäivänä kasvaneet erittäin laajoiksi ja ohjelmistokehittäjien on ymmärettävä suuria ja monimutkaisia osia niiden lähdekoodista. Tämän helpottamiseksi on ohjelmistojen havainnollistaminen (Software Visualization) aktiivinen tutkimusala. Ohjelmistojen havainnollistamisella ymmärretään usein visuaalisten elementtien käyttöä ohjelmiston ja sen kehityksen havainnollistamiseksi [5].

Ohjelmistojen havainnollistaminen voidaan karkeasti jakaa kolmeen alueeseen: ohjelmiston *rakenteen*, *toiminnan* ja *kehityksen* havainnollistamiseen. Rakenteen havainnollistaminen tuo esille ohjelmiston staattisia osia ja niiden välisiä suhteita. Toiminnan havainnollistaminen kuvaa ohjelman suorittamista oikealla tai abstraktilla syötteellä. Kehityksen havainnollistaminen pyrkii havainnollistamaan ohjelmiston kehitysprosessia. [5]

Ohjelmoinnin opetuksessa usein käytetty ohjelmistojen havainnollistamisen erikoisalue on algoritmianimaatio. Algoritmianimaation voidaan ymmärtää olevan toiminnan havainnollistamista, jonka tavoitteena on havainnollistaa algoritmin suoritusta lähdekoodia korkeammalla abstraktiotasolla [5]. Esimerkiksi verkkoalgoritmeja havainnollistettaessa ei yleensä näytetä verkon toteutustapaa vaan kuva verkon solmuista ja kaarista.

Opetuskäyttöön on kehitetty lukemattomia algoritmianimaatiojärjestelmiä (esimerkiksi [1, 4, 7, 12, 14, 16, 20, 24, 25]). Järjestelmistä jokaisella on omat vahvuutensa ja heikkoutensa, esimerkiksi toiset ovat helppokäyttöisiä mutta sovellusalueeltaan rajoittuneita, toiset taas yleiskäyttöisiä mutta animaatioiden luominen on työlästä. Eri järjestelmät tarjoavat myös hyvin erilaista vuorovaikutusta animaation ja sen käyttäjän välillä. Esimerkiksi ANIMAL [20] esittää opiskelijalle kysymyksiä animaation aikana, kun taas TRAKLA2-järjestelmässä [14] opiskelija joutuu itse luomaan animaation. Juuri vuorovaikutuksella on todettu olevan

ratkaiseva osa algoritmianimaation vaikutuksessa opiskelijoiden oppimiseen [8].

Runsaasta tutkimuksesta huolimatta algoritmianimaatio ei ole saavuttanut suurta suosiota opettajien piirissä. Suurimpana syynä tähän on ajan puute animaatioiden etsimiseen, luomiseen sekä integroimiseen muuhun opetukseen [17]. Tuoreessa tutkimuksessa myös ilmeni, että valmiita, korkealaatuisia esimerkkejä on vähän saatavilla [22].

Opetuksessa algoritmianimaatioille on useita eri käyttötarkoituksia. Yleisimmät näistä ovat luentokäyttö, itseopiskelumateriaali sekä arvosteltavat harjoitustehtävät. Nykytilanteessa jokaista käyttötarkoitusta varten on yleensä luotava uusi animaatio käyttötarkoituksen vaatiman erilaisen vuorovaikutuksen johdosta. Ideaalitilanteessa samaa animaatiota voisi käyttää kaikkiin tarkoituksiin.

Nykyisistä järjestelmistä useat sisältävät jo oman algoritmianimaatiokielensä. Kuten järjestelmillä, myös niiden kielillä on eri ominaisuuksia ja piirteitä. Viimeaikoina jotkin järjestelmät ovat alkaneet tukea esimerkiksi useita animaatiokieliä sekä erilaisia tuonti- ja vientitiedostomuotoja. Tämä on myös yksi keskeinen haaste ohjelmistojen havainnollistamisen välineissä [5]. Kuitenkaan vielä ei olla niin pitkällä, että järjestelmät voisivat vaihtaa animaatiotietoa keskenään. Järjestelmien yhteiskäytttö ja näin eri järjestelmien vahvuuksien yhdistäminen niin animaation luonnissa kuin sen tarjoamassa vuorovaikutuksessakin mahdollistaisi järjestelmän valinnan kulloisenkin käyttötapauksen mukaan ja lisäisi valmiiden esimerkkien määrää kullekin järjestelmälle. Yksi keino yhteiskäytön mahdollistamiseksi on määritellä järjestelmille yhteinen algoritmianimaatiokielistandardi. Vuonna 2005 Innovation and Technology in Computer Science Education (ITiCSE) -konferenssin yhteydessä kokoontunut työryhmä keskittyi yhteisen, vuorovaikutusta tukevan, algoritmianimaatiokielen määrittelyyn [18]. Työryhmän näkemyksen mukaan standardi poistaisi tiukan kytkennän animaatiokielen ja -järjestelmän väliltä. Lisäksi standardi tarjoaisi muun muassa parempaa tukea loppukäyttäjille kehittäjäkunnan kasvaessa. Yleisestikin standardien on todettu nostavan tuottavuutta, tehokkuutta sekä laatua.

Jotta järjestelmille yhteisen kielen määritteleminen olisi mahdollista, tulee olemassa olevien kielten ominaisuudet tuntea ja kieliä pystyä vertailemaan. Tästä tarpeesta johtuen esitetään artikkelissa algoritmianimaatiokielten luokittelu. Tämän luokittelun avulla ja edellä mainitun työryhmän määrittelyjä hyödyntäen määrittelen uuden algoritmianimaatiokielen nimeltä XAAL (eXtensible Algorithm Animation Language). XAAL soveltuu järjestelmien yhteiseksi kieleksi. Tässä artikkelissa kuvattava työ on jatkoa edellä mainitun työryhmän pyrkimykselle kehittää algoritmianimaatiokielistandardi.

Artikkelin rakenne on seuraava. Luvussa 2 käydään lyhyesti läpi algoritmianimaation historiaa sekä jo olemassa olevia algoritmianimaatiokieliä. Luvussa 3 esitellään animaatiokielten luokittelu. Luvussa 4 vuorostaan käydään läpi XAAL-kielen tärkeimpiä ominaisuuksia ja luvussa 5 kuvataan kieltä tukevat työkalut. Lopuksi luvussa 6 kootaan yhteen artikkelin ajatukset ja katsotaan tulevaisuuteen.

2 Taustaa

2.1 Algoritmianimaation historiaa

Algoritmianimaatiota on tutkittu runsaasti jo vuosikymmeniä. Ensimmäinen algorit-

mianimaatio onkin jo vuodelta 1966, jolloin Ken Knowlton esitteli animaation, joka havainnollisti linkitettyjen listojen käsittelyä L^6-ohjelmointikielellä [13]. Algoritmianimaation käytön tietotekniikan opetuksen apuna katsotaan alkaneen vuodesta 1981, jolloin Ronald Baecker esitti videon, jolla havainnollistettiin yhdeksän lajittelualgoritmin toimintaa [2].

Tämän jälkeen alalla on tehty paljon tutkimusta. Esimmäinen tietokonejärjestelmä opetuskäyttöön oli BALSA (Brown ALgorithm Simulator and Animator) [3]. BALSA on vuorovaikutteinen animaatiosovelluskehys, joka tukee useita näkymiä algoritmista sekä siihen liittyvistä tietorakenteista. Toinen aikainen, runsaasti huomiota saanut järjestelmä on TANGO (Transition-based ANimation GeneratiOn) [23]. TANGO on algoritmianimaatiojärjestelmä, joka ensimmäisenä esitti muutokset vähitellen muuttuen pehmeää animaatiota käyttäen.

Tämän jälkeen on kehitetty lukuisia algoritmianimaatiojärjestelmiä (esimerkiksi [1, 4, 7, 12, 14, 16, 20, 24, 25]). Kuvassa 1 on esitetty tunnetuimpia algoritmianimaatiojärjestelmiä ja niiden julkaisuvuosia.

Koska opettajien resurssit ovat hyvin rajalliset, vaivattomuus on tärkeä ominaisuus algoritmianimaatiojärjestelmissä. [9] Jotkin järjestelmät ovat helppokäyttöisiä, mutta sovellusalueeltaan varsin rajoittuneita. Vaivattomuus liittyy usein abstraktiotasoon, jolla järjestelmä toimii. Esimerkiksi kuvassa 2 esitetty MatrixPro-järjestelmä mahdollistaa animaatioiden tekemisen hiirellä avaimia raahaamalla ja pudottamalla niitä tietorakenteisiin. Järjestelmä huolehtii operaatioiden oikeellisuudesta. Luonnollisesti näin voi toimia vain järjestelmän tukemien tietorakenteiden kanssa. Toisaalta esimerkiksi ANIMAL-järjestelmässä käyttäjä luo animaation graafisista primitiiveistä. Tämä vaatii huomattavasti enemmän työtä, mutta järjestelmällä on mahdollista kuvata mikä tahansa tietorakenne tai algoritmi.

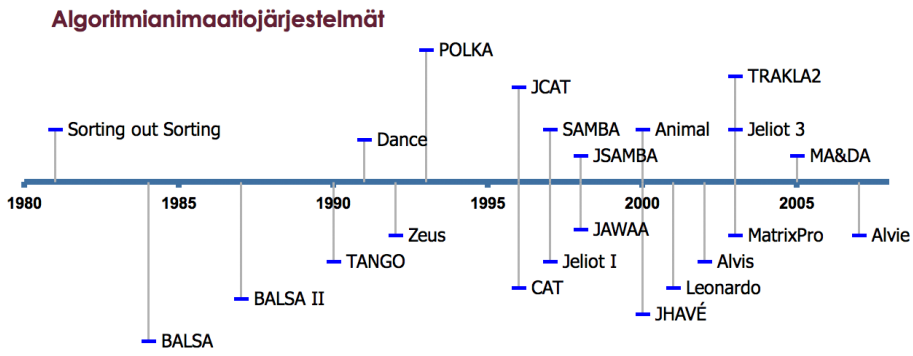
2.2 Algoritmianimaatiokielten ominaisuuksia

Järjestelmistä useat sisältävät jo oman algoritmianimaatiokielensä. Järjestelmien yhteiseksi tarkoitetulta algoritmianimaatiokieleltä vaadittavien ominaisuuksien kartoittamiseksi tehtiin laaja katsaus näihin kuvauskieliin [10]. Seuraavassa esitellään esimerkkien avulla joitakin keskeisiä kielten ominaisuuksia. Ominaisuuslista ei ole missään nimessä kattava, vaan sisältää vain kaikkein oleellisimmat ominaisuudet. Kiinnostunutta lukijaa kehoitan perehtymään viitattuihin artikkeleihin.

Kuvaustapa Varhaisimmissa algoritmianimaatiokielissä animaation kuvaus oli yleensä liitetty osaksi ohjelmakoodia esimerkiksi upottamalla kommentteihin tietoa animoinnista tai lisäämällä koodiin animaatiokirjastokutsuja. Näin toimii esimerkiksi TANGO [23]. Hieman uudemmissa kielissä animaatio kuvataan itsenäisessä dokumentissa tekstimuodossa. Listauksessa 1 liitteessä A on esimerkki ANIMALSCRIPT-kielestä. Uusimmissa kielissä kuvaukseen käytetään tyypillisesti XML-kieltä, kuten listauksen 2 GaigsXML-kielessä [15].

Abstraktiotaso Eri animaatiokielet eroavat selkeästi abstraktiotasoltaan. Yhtenä ääripäänä ovat kielet, joissa animaatio kuvataan käyttäen graafisia elementtejä, kuten viivoja ja ympyröitä. Listauksessa 1 on esimerkki tällaisesta kielestä.

Toinen ääripää ovat kielet, joissa animaation luomisessa käytetään ainoastaan tietorakenteita. Listauksessa 2 on

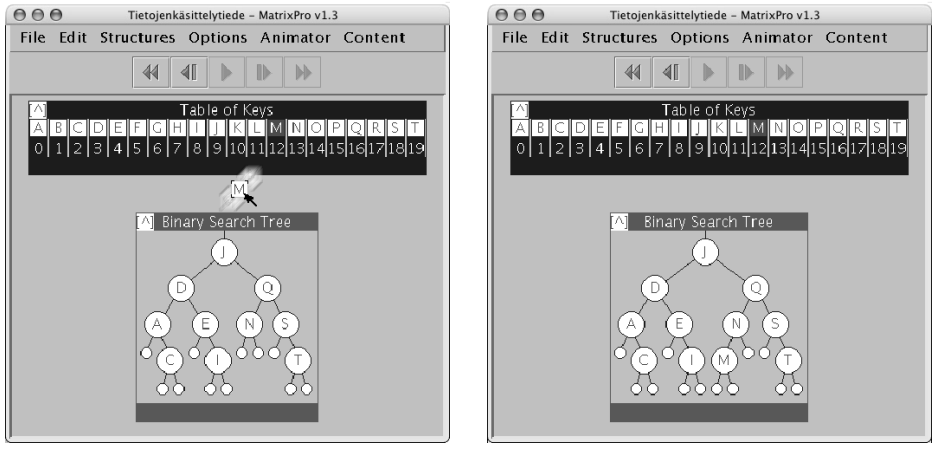


Kuva 1: Algoritmianimaatiojärjestelmiä ja niiden julkaisuvuosia. Pystysuuntainen sijoittelu on vain luettavuuden parantamiseksi.

esimerkki pinon käytöstä GaigsXML-kielessä. Tyypillisesti tietorakenteita käytettäessä animaatiot pystytään esittämään lyhyemmin. Tällöin kuitenkin menetetään graafisten primitiivien tuoma vapaus kuvata animaatiossa mitä tahansa, ei vain kielen tukemia tietorakenteita.

On kuitenkin huomattava, että useille algoritmianimaatiokielille on tyypillistä, että siinä animaatioita luotaessa käytetään sekä graafisia elementtejä että tietorakenteita.

Animaatio Algoritmianimaatiokieliä tarkasteltaessa keskeinen ominaisuus on kielen animaatiotuki. Animoinnissakin alhaisin abstraktiotaso on graafisten elementtien animointi, esimerkiksi niiden



(a) (b)

Kuva 2: Esimerkki animaation luomisesta MatrixPro-järjestelmässä. (a) Käyttäjä raahaa avaimen M taulukosta ja pudottaa sen binäärisen hakupuun päälle. (b) Järjestelmä lisää avaimen oikealle paikalle binäärisessä hakupuussa.

siirtäminen tai värin muuttaminen. Listauksessa 1 on esimerkki yksinkertaisesta operaatiosta. Tietorakenteita käyttävissä kielissä on yleensä määritelty myös operaatioita tietorakenteille. Nämä voivat olla esimerkiksi solmun/kaaren lisääminen verkkoon tai avaimen lisääminen binääriseen hakupuuhun. Listauksen 3 esimerkissä käytetään DsCats-kielen B-puun lisäys- ja poisto-operaatioita [4].

Ohjelmointiominaisuudet Jotkin algoritmianimaatiokielistä sisältävät ohjelmointikielille tyypillisiä ominaisuuksia kuten muuttujat, ehtolauseet sekä silmukat. Listauksessa 4 on esimerkki ANIMALSCRIPT2-kielen ohjelmointiominaisuuksista.

Vuorovaikutteisuus Algoritmianimaation vuorovaikutteisuudella on todettu olevan suuri merkitys opetuskäytössä. Tästä syystä jotkin animaatiokielet tukevat käyttäjän ja animaation vuorovaikutuksen määrittelemistä itse animaatiokielessä. Tämä vuorovaikutus voi olla esimerkiksi syötedatan kysymistä käyttäjältä tai kysymysten esittämistä animaation lomassa. Jälkimmäisestä tavasta on esimerkki listauksessa 5.

2.3 Algoritmianimaatiokieliä

Seuraavassa kuvaamme lyhyesti edellisien ominaisuuksien yhteydessä mainitut algoritmianimaatiokielet sekä joitakin muita tunnettuja animaatiokieliä.

ANIMALSCRIPT [19] on ANIMAL-järjestelmässä käytetty animaatiokieli. Kielessä käsitellään pääasiassa graafisia primitiivejä, mutta myös tietorakenteet taulukko ja lista ovat tuettuina. Listauksessa 1 on ANIMALSCRIPT-esimerkki. Kielen uusin versio, ANIMALSCRIPT2, tukee myös yksinkertaisia ohjelmointirakenteita, kuten silmukat, ehtolauseet sekä muuttujat [21].

DsCats -järjestelmä [4] sisältää yksinkertaisen kielen, jolla voi määritellä animaatioita käyttäen puurakenteita ja näille tehtäviä lisäys- ja poisto-operaatioita. Listauksen 3 esimerkki näyttää kuinka helppoa animaation määrittely on; B-puuhun voidaan lisätä avaimia yhdellä komennolla.

GaigsXML -kieli on käytössä JHAVÉ-animaatioympäristössä [15]. Kielessä animaatio muodostetaan kuvaamalla joukko tietorakenteiden tiloja. Järjestelmä animoi tilojen muutokset. Kieli sisältää myös tuen vuorovaikutteisten kysymysten kuvaamiselle. Esimerkkejä kielestä löytyy listauksista 2 ja 5.

JAWAA -järjestelmän [1] animaatioiden kuvauskieli käsittelee graafisia primitiivejä sekä tietorakenteita. Tuettuja tietorakenteita ovat taulukko, jono, pino, puu sekä verkko.

Samba -järjestelmä [25] sisältää kielen, jolla voi kuvata graafisia primitiivejä sekä näiden muutoksia. Mielenkiintoisena ominaisuutena kieli tukee muun muassa useita näkymiä. Useat järjestelmät osaavat näyttää kielellä kuvattuja animaatioita, esimerkiksi JHAVÉ ja JSamba.

SALSA [7] on korkean tason algoritmianimaatiokieli, joka on tarkoitettu alhaisen tarkkuuden animaatioiden kuvaamiseen. Kieli sisältää sekä asettelun että logiikan kuvaamiseen tarvittavia ominaisuuksia.

Kuten edellä olevista lyhyistä kuvauksista ja liitteen A esimerkeistä ilmenee, olemassa olevien kielten tapa kuvata animaatioita eroaa suuresti toisistaan. Jotta kielten vertailu ja niiden erojen selvittäminen onnistuisi helpommin, määrittelimme algoritmianimaatiokielten luokittelun.

3 Algoritmianimaatiokielten luokittelu

Tässä luvussa esitellään lyhyesti algoritmianimaatiokielten luokittelu. Tarkemmin luokittelusta voi lukea lähteestä [11].

Animaatio (Animation) Kategorialla vertaillaan animaatiokielen tarjoamien animaatio-ominaisuuksien laajuutta. Tarkasteltavia asioita ovat muun muassa tietorakenteiden operaatiot, graafisten primitiivien animointi sekä animaation ajoittaminen.

Vuorovaikutus (Interaction) Kategoria mittaa kielen kautta määriteltävän, animaation loppukäyttäjälle tarjottavan vuorovaikutuksen määrää. Kategoriassa tarkastellaan muun muassa vuorovaikutuksen tyyppiä, joka on esimerkiksi ANIMALSCRIPT:n kohdalla opiskelijalle esitettäviä kysymyksiä ja SALSA:n tapauksessa syötedatan pyytämistä opiskelijalta.

Kieli (Language) Kategoria mittaa kielen ominaisuuksia jotka eivät suoraan liity algoritmianimaatioon, kuten esimerkiksi metatietoa sekä erilaiset debug-tulosteet ja kommentointi.

Asettelu (Positioning) Kategoria vertaa kielen tarjoamia ominaisuuksia olioiden asetteluun animaatiossa, kuten kuinka monessa dimensiossa toimitaan ja minkälainen kielen koordinaattijärjestelmä on.

Tyyli (Style) Kategoria tarkastelee kielen ominaisuuksia olioiden tyylin määrittelemiseen, kuten käytettävissä olevien värien ja kirjasimien määrä.

Sanasto (Vocabulary) Kategoria tarkastelee, mistä rakennuspalikoista kielessä animaatiot koostetaan. Tarkasteltavia asioita ovat muun muassa tietorakenteiden ja graafisten primitiivien määrä sekä mahdolliset ohjelmointikielille tyypilliset ominaisuudet.

Edellisessä luvussa esiteltyjen algoritmianimaatiokielten arviointi em. luokittelun perusteella on esitetty lähteessä [11].

4 XAAL-kieli

4.1 XAAL-kielen synty

Kesällä 2005 ITiCSE-konferenssin yhteydessä kokoontui kansainvälinen työryhmä “Development of XML-based Tools to Support User Interaction with Algorithm Visualizations”. Ryhmä koostui algoritmianimaatiojärjestelmien kehittäjistä sekä järjestelmiä opetuksessa käyttävistä opettajista ja tutkijoista. Työryhmän tavoitteena oli luoda XML-määrittelyjä algoritmianimaation eri osille, kuten vuorovaikutukselle, graafisten primitiivien kuvaamiselle sekä tietorakenteiden kuvaamiselle ja näin määritellä algoritmianimaatiokieli, jota useat järjestelmät tukisivat. Työryhmän raportti tarjoaa esimerkkejä näistä määrittelyistä sekä ohjeita niiden käytöstä olemassa olevissa algoritmianimaatiojärjestelmissä [18]. Työryhmä teki paljon tärkeää työtä, jota tässä artikkelissa kuvattava työ jatkaa tarkentamalla ja laajentamalla määrittelyjä. Jatkossa puhuttaessa

työryhmästä viitataan tähän työryhmään ja sen tekemään raporttiin.

Luvussa 2 esitetyn katsauksen ja sen perusteella määritellyn luokittelun avulla määrittelin uuden algoritmianimaatiokielen nimeltä XAAL (Extensible Algorithm Animation Language). XAAL on määritelty XML-kieleksi kehittämällä XML-dokumentin rakenteen kuvaus. XML-dokumenttien koneellinen käsittely ja luominen on helppoa olemassa olevien työkalujen avulla. Lisäksi XML-dokumenttien kirjoittaminen tavallisella tekstinkäsittelyohjelmalla on mahdollista. Myös XML-dokumentin muuntaminen toiseen muotoon on joustavaa käyttämällä XSLT-kieltä (Extensible Stylesheet Language Transformations) ja työkaluja.

XAAL-kieleen on pyritty löytämään yhteiseltä kieleltä vaadittavat ominaisuudet. Lisäksi työryhmän määrittelyjä on pyritty käyttämään mahdollisimman paljon. Kielen ominaisuudet voidaan karkeasti jakaa kolmeen osaan: tietorakenteiden määrittelyyn, visuaalisen ulkoasun määrittelyyn ja animaation määrittelyyn. Seuraavissa aliluvuissa käsitellään lyhyesti nämä pääosat.

4.2 Visuaalinen ulkoasu

Visuaalisen ulkoasun määrittelyllä tarkoitetaan tässä yhteydessä sekä tietorakenteiden ulkoasun määrittelyä että graafisten elementtien käyttöä animaation luonnissa.

Tähän tarkoitukseen XAAL hyödyntää työryhmän määrittelyä. Tuettuja graafisia elementtejä ovat mm viiva, monikulmio, kaari, ellipsi, ympyrä, kolmio, suorakulmio sekä neliö. Näistä elementeistä voidaan myös koostaa monimutkaisempia muotoja. Liitteen B listaus 6 antaa esimerkin graafisten elementtien käytöstä ja muodon määrittelystä.

4.3 Tietorakenteet

XAAL tukee tietorakenteiden käyttöä ja määrittelyä animaatioiden koostamisessa. Tuetut rakenteet ovat lista, taulukko, puu sekä verkko. Rakenteiden sisältämä tieto kuvataan solmujen (taulukon tapauksessa indeksien) ja niiden välisten kaarien avulla. Rakenteet voivat muodostaa hierarkioita ja olla täten rajattoman monimutkaisia. Yksinkertaisimmillaan rakenne voi olla merkkijono tai luku. Listauksessa 7 on esimerkin vuoksi esitetty taulukon määrittely sekä yksi mahdollinen tapa visualisoida se.

4.4 Animaatio

Animaatiossa tehtäviä operaatioita on kahdenlaisia: tietorakenteille tehtäviä operaatioita sekä graafisia primitiivejä muuttavia operaatioita. Tietorakenteille tehtäviä operaatioita ovat muun muassa luominen, lisäys, poisto, korvaus, etsiminen sekä vaihtaminen. Listaus 8 antaa esimerkin poisto-operaatiosta.

Graafisia primitiivejä muuttavat operaatiot ovat XAAL-kielessä työryhmän määrittelyn mukaiset. Mahdollisia operaatioita ovat muun muassa siirtäminen, kiertäminen, koon muuttaminen, ryhmittely sekä tyylin muuttaminen. Listauksessa 9 on esimerkki kiertämisestä.

Jotta muunnokset kahden erilaisen lähestymistavan, graafisia primitiivejä käyttävän ja tietorakenteita käyttävän, välillä olisivat mahdollisia, mikä tahansa tietorakenteelle tehtävä operaatio on voitava määritellä myös graafisten primitiivien animointina (katso listaus 8). Näistä animaatiojärjestelmä voi sitten valita sen esitysmuodon, jota se tukee.

4.5 Muut ominaisuudet

Edellä mainittujen pääosien lisäksi kielessä on useita sekalaisia ominaisuuksia.

Äänen käyttö algoritmianimaatiossa on melko vähän tutkittu alue. Yleisesti on kuitenkin havaittu, että äänen käytölle on perusteita, muun muassa se, että ihminen pystyy käsittelemään kuulemaansa kuuntelematta aktiivisesti ja havaitsemaan toistuvia osia melodioissa [6]. Lisäksi ääni tarjoaa uuden "näkymän" havainnollistettavaan algoritmiin. Tästä syystä XAAL tukee myös äänen liittämistä animaatioon.

Metatieto (tietoa tiedosta) parantaa tietojen etsittävyyttä ja on nykypäivän valtavien tietomäärien johdosta entistä tärkeämpää. XAAL sisältää ominaisuuden kuvata animaation sisältöä ja näin mahdollistaa animaatiokokoelmien keräämisen ja animaatioiden etsimisen näistä kokoelmista.

5 Toteutus

Jotta kieli olisi mahdollisimman helppo ottaa käyttöön olemassa olevissa järjestelmissä, on kieltä tukemaan kehitetty työkaluja. Nämä työkalut mahdollistavat XAAL-tuen lisäämisen olemassa oleviin järjestelmiin ilman suuria muutoksia. Lisäksi työkalut mahdollistavat XAAL-dokumentin muuntamisen useisiin eri muotoihin.

5.1 Toteutusvaihtoehdot

Kielen lukemiseen ja luomiseen olemassa olevilla järjestelmillä ehdotan kahta vaihtoehtoista lähestymistapaa.

XAAL-animaatio voidaan *XSL-tyylitiedoston* avulla muuttaa järjestelmän jo ymmärtämään muotoon (katso kuva 3). XSL-tyylitiedosto on kuvaus XAAL-kielestä kohdejärjestelmän animaatiokieleen. Tämä lähestymistapa soveltuu järjestelmiin, joissa on jo olemassa algoritmianimaatiokieli. Hyvänä puolena on, että kohdejärjestelmään ei vaadita suuria muutoksia.

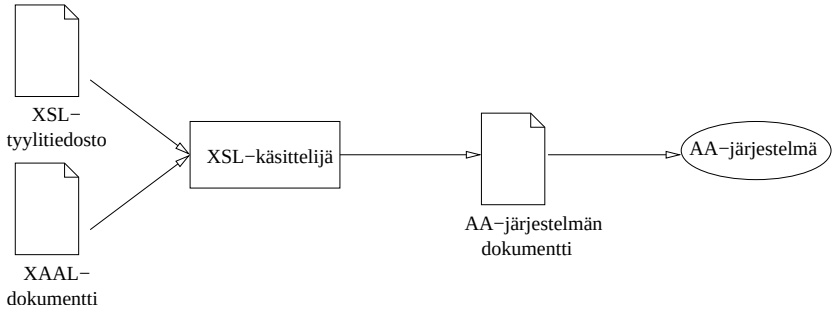
Toinen toteutusvaihtoehto on jäsentää XAAL-animaatiosta *Java-oliohierarkia* ja sovittaa tämä järjestelmän sisäiseksi esitysmuodoksi (katso kuva 4). Tätä vaihtoehtoa tukemaan on kehitetty XAAL-jäsennin joka luo Java-oliohierarkian sekä komponentti, joka sarjallistaa oliohierarkian XAAL-dokumentiksi. Animaatiojärjestelmä voi käyttää XAAL-animaatioita sovittamalla oliohierarkian järjestelmän sisäiseksi esitysmuodoksi. Toteuttamalla tuottajan, joka muuttaa sisäisen esitysmuodon XAAL-oliohierarkiaksi, järjestelmällä voi luoda XAAL-dokumentteja.

5.2 Toteutetut muunnokset

Tähän mennessä toteutetut muunnokset on esitetty kuvassa 5. Nämä muunnokset ovat:

XAAL ↔ MatrixPro MatrixPro on ainoa järjestelmä, jolla tällä hetkellä voi luoda XAAL-animaatioita. Järjestelmällä on vaivatonta luoda esimerkkejä käyttäen sen tukemia tietorakenteita. Muut toteutetut muunnokset mahdollistavat näiden esimerkkien käytön ja muokkaamisen muilla järjestelmillä. Lisäksi MatrixPro mahdollistaa animaatioiden muuntamisen LaTeX-kuviksi (esimerkiksi listausten 7 ja 8 kuvat ovat tällä ominaisuudella luotuja). XAAL-animaatiot, joissa käytetään tietorakenteita, on myös mahdollista ladata järjestelmällä. Muunnos on toteutettu Java-oliohierarkian avulla.

XAAL → ANIMALSCRIPT Muunnos ANIMAL-järjestelmän käyttämään



Kuva 3: XAAL-tuen lisääminen algoritmianimaatiojärjestelmään XSL-tyylitiedostolla.

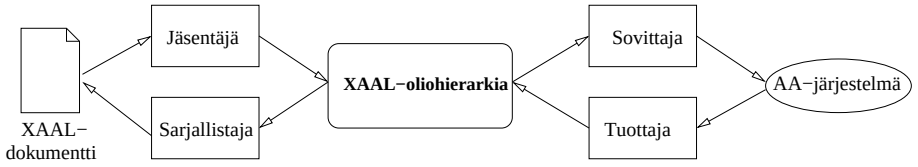
animaatiokieleen toteutettiin, sillä järjestelmällä pystyy luomaan graafisista elementeistä mitä tahansa animaatioita. Näin esimerkiksi MatrixPro:lla luotuja XAAL-animaatioita voidaan muokata eteenpäin. Lisäksi ANIMAL soveltuu luentokäyttöön korvaten esimerkiksi Microsoft PowerPointin. Toteutus käyttää XSL-tyylitiedostoa muunnoksen tekemiseen.

XAAL → JHAVÉ JHAVÉ-ympäristön kantavana ajatuksena on mahdollistaa eri animaatiojärjestelmien käyttö saman, yhtenäisen käyttöliittymän kautta (katso kuva 6). Java-oliohierarkiaa käyttäen toteutettu JHAVÉ-lisäosa mahdollistaa XAAL-animaatioiden katselemisen ympäristössä.

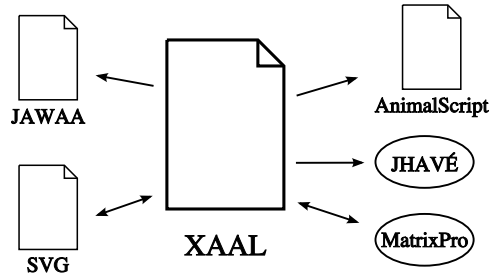
XAAL ↔ SVG SVG-animaatioita on helppo lisätä verkkomateriaaliin. Muunnos tukee ainoastaan XAAL-kielen graafisia elementtejä ja on toteutettu XSL-tyylitiedostolla.

XAAL → JAWAA Koska SVG:n tuki selaimissa on tällä hetkellä hataralla pohjalla ja sen tulevaisuus epäselvä, toteutettiin muunnos JAWAA-järjestelmän animaatiokieleen. JAWAA mahdollistaa animaatioiden lisäämisen verkkomateriaaliin Java-sovelmana toteutetun järjestelmän avulla. Tämäkin muunnos on toteutettu XSL-tyylitiedostolla.

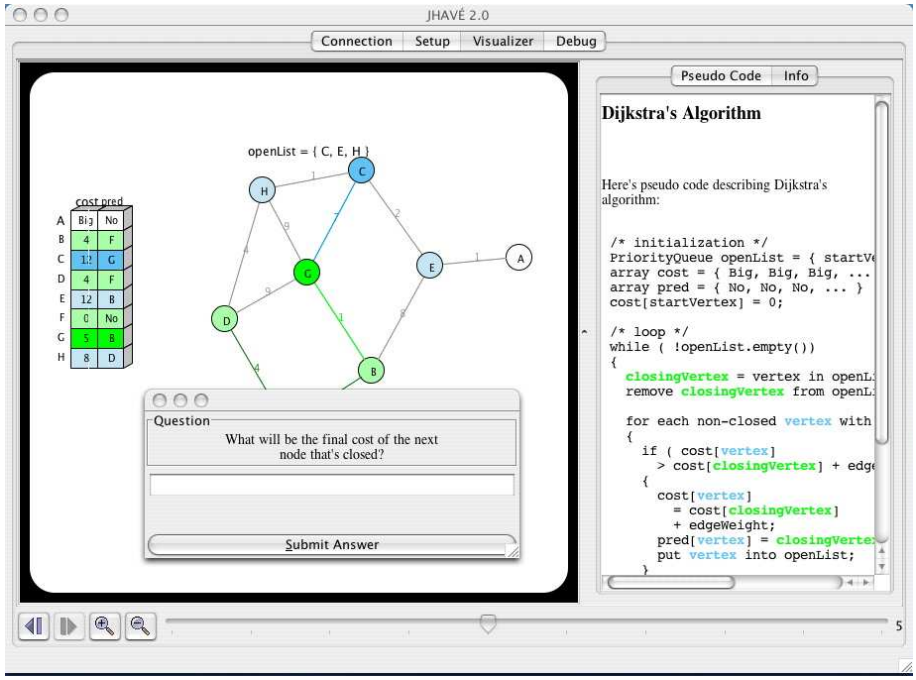
Näiden prototyyppitoteutusten avulla voidaan esimerkiksi luoda animaatio vaivattomasti MatrixPro:lla ja muokata sitä sen jälkeen ANIMAL:lla. ANIMAL soveltuu hyvin käytettäväksi luentotilanteissa. Lisäksi XAAL-animaatioita voidaan lisätä verkkomateriaaliin useilla tavoilla.



Kuva 4: XAAL-tuen lisääminen algoritmianimaatiojärjestelmään Java-oliohierarkian avulla.



Kuva 5: Toteutetut muunnokset järjestelmien/animaatiokielien välillä. Nuolet kuvaavat muunnoksen suuntaa.



Kuva 6: Esimerkki vuorovaikutteisesta algoritmianimaatiosta JHAVÉ-ympäristössä.

6 Yhteenveto

Tässä artikkelissa on esitetty XML-kieli algoritmianimaatioiden kuvaukseen sekä kieltä tukevia työkaluja. Tavoitteena työssä on saada aikaan useiden järjestelmien ja kehittäjien tukema algoritmianimaatiokielistandardi. Kieli ja työkalut mahdollistavat kielen käytön olemassa olevissa järjestelmissä sekä tiedonvaihdon eri algoritmianimaatiojärjestelmien välillä tukemalla luokittelun perusteella määriteltyjä ominaisuuksia. Tämä tiedonvaihto lisää tietorakenteiden ja algoritmien opettajien valinnanvaraa animaatiojärjestelmien ja valmiiden animaatioiden käytössä. Näin

se toivottavasti lisää järjestelmien käytettävyyttä ja käytön yleisyyttä opetuksessa.

Tulevaisuudessa XAAL-kieltä ja sitä tukevia työkaluja on tarkoitus kehittää eteenpäin. Ensimmäinen jatkokehitystavoite työkaluille on lisätä tuki useammille animaatioiden käyttötapauksille, esimerkiksi käyttö arvosteltavina tehtävinä lisäämällä tuki vuorovaikutteisille kysymyksille. Luentokäyttöön soveltuvuuttakin voi kehittää toteuttamalla esimerkiksi muunnos XAAL → Open Document Format. Kielen mielenkiintoisimpia kehityssuuntia taas ovat muun muassa ohjelmakoodin tukeminen sekä omien tietorakenneoperaatioiden määrittelemisen mahdollistaminen koodin avulla.

Kiitokset

Kiitokset kollegoille kommenteista ja ideoista. Erityisesti haluan kiittää tutkimustyöni valvojaa professori Lauri Malmia sekä ohjaajaa dosentti Ari Korhosta. Lisäksi kiitos kuuluu artikkelissa mainitun työryhmän jäsenille, etenkin ryhmän vetäjille Thomas L. Napsille ja Guido Rößlingille.

Viitteet

[1] Ayonike Akingbade, Thomas Finley, Diana Jackson, Pretesh Patel ja Susan H. Rodger. JAWAA: easy web-based animation from CS0 to advanced CS courses. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE'03*, ss. 162–166, Reno, Nevada, USA, 2003. ACM Press.

[2] Ronald M. Baecker. Sorting out sorting. Selostettu värivideo, 30 minuuttia, 1981.

[3] Marc H. Brown ja Robert Sedgewick. A system for algorithm animation. *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'84*, ss. 177–186. ACM Press, 1984.

[4] Justin Cappos ja Patrick Homer. DsCats: Animating data structures for CS2 and CS3 courses. Verkossa julkaistu tekninen dokumentti, 2002. Saatavilla verkossa `http://www.cs.arizona.edu/dscats/dscatstechnical.pdf` [10. joulukuuta 2008].

[5] S. Diehl. *Software visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer New York, 2007.

[6] Joan M. Francioni, Larry Albright ja Jay Alan Jackson. Debugging parallel programs using sound. *SIGPLAN Notices*, 26(12):68–75, Joulukuu 1991.

[7] Christopher D. Hundhausen ja Sarah A. Douglas. Low-fidelity algorithm visualization. *Journal of Visual Languages and Computing*, 13(5):449–470, Lokakuu 2002.

[8] Christopher D. Hundhausen, Sarah A. Douglas ja John T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3):259–290, Kesäkuu 2002.

[9] Petri Ihantola, Ville Karavirta, Ari Korhonen ja Jussi Nikander. Taxonomy of effortless creation of algorithm visualizations. *Proceedings of the International Workshop on Computing Education Research*, ss. 123–133, New York, NY, USA, 2005. ACM Press.

[10] Ville Karavirta. XAAL - extensible algorithm animation language. Diplomityö, Tietotekniikan osasto, Teknillinen korkeakoulu, Joulukuu 2005. Saatavilla verkossa `http://www.cs.hut.fi/Research/SVG/publications/karavirta-masters.pdf` [10. joulukuuta 2008].

[11] Ville Karavirta, Ari Korhonen ja Lauri Malmi. Taxonomy of algorithm animation languages. *Proceedings of the 2006*

ACM Symposium on Software Visualization, SoftVis'06, ss. 77–85, New York, NY, USA, Syyskuu 2006. ACM Press.

[12] Ville Karavirta, Ari Korhonen, Lauri Malmi ja Kimmo Stålnacke. MatrixPro – A tool for on-the-fly demonstration of data structures and algorithms. *Proceedings of the Third Program Visualization Workshop*, ss. 26–33, The University of Warwick, UK, Heinäkuu 2004.

[13] K. C. Knowlton. L^6: Bell telephone laboratories low-level linked list language. 16mm mustavalkoinen video, 16 minuuttia, 1966.

[14] Lauri Malmi, Ville Karavirta, Ari Korhonen, Jussi Nikander, Otto Seppälä ja Panu Silvasti. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education*, 3(2):267–288, 2004.

[15] Thomas L. Naps, Myles McNally ja Scott Grissom. Realizing XML driven algorithm visualization. *Proceedings of the Fourth Program Visualization Workshop*, 2006.

[16] Thomas L. Naps. JHAVÉ: Supporting Algorithm Visualization. *Computer Graphics and Applications, IEEE*, 25(5):49–55, 2005.

[17] Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodgers ja J. Ángel Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin*, 35(2):131–152, Kesäkuu 2003.

[18] Thomas L. Naps, Guido Rößling, Peter Brusilovsky, John English, Duane Jarc, Ville Karavirta, Charles Leska, Myles McNally, Andrés Moreno, Rockford J. Ross ja Jaime Urquiza-Fuentes. Development of XML-based tools to support user interaction with algorithm visualization. *SIGCSE Bulletin*, 37(4):123–138, Joulukuu 2005.

[19] Guido Rößling ja Bernd Freisleben. Program visualization using ANIMALSCRIPT. *Proceedings of the First Program Visualization Workshop*, ss. 41–52, University of Joensuu, Finland, 2000.

[20] Guido Rößling ja Bernd Freisleben. ANIMAL: A system for supporting multiple roles in algorithm animation. *Journal of Visual Languages and Computing*, 13(3):341–354, 2002.

[21] Guido Rößling, Felix Gliesche, Thomas Jajeh ja Thomas Widjaja. Enhanced expressiveness in scripting using AnimalScript 2. *Proceedings of the Third Program Visualization Workshop*, ss. 10–17, The University of Warwick, UK, Heinäkuu 2004.

[22] Clifford A. Shaffer, Matthew Cooper ja Stephen H. Edwards. Algorithm visualization: a report on the state of the field. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE'07*, ss. 150–154, New York, NY, USA, 2007. ACM Press.

[23] J. T. Stasko. TANGO: A framework and system for algorithm animation. *IEEE Computer*, 23(9):27–39, 1990.

[24] John T. Stasko. Jsamba – java version of the SAMBA animation program. Saatavilla verkossa `http://www.cc.gatech.edu/gvu/softviz/algoanim/jsamba/` [10. joulukuuta 2008].

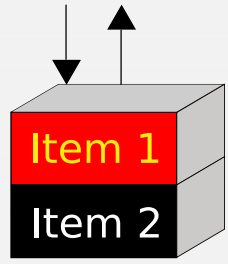
[25] John T. Stasko. Using student-built algorithm animations as learning aids. *The Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education, SIGCSE'97*, ss. 25–29, San Jose, CA, USA, 1997. ACM Press, New York.

A Esimerkkejä algoritmianimaatiokielistä

```
circle "C" (150, 100) radius 30 color black filled
    fillColor red depth 3
move "C" along line (130, 80) (130, 170) within 200 ms
```

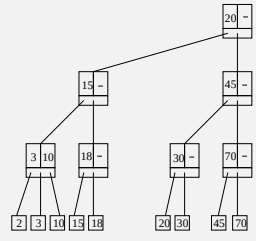
Listaus 1: Esimerkki graafisista elementeistä ja niiden animoinnista ANIMALSCRIPT-kielessä.

```
<snap>
  <title>Stack example</title>
  <stack>
    <list_item color="red">
      <label>Item 1</label>
    </list_item>
    <list_item color="black">
      <label>Item 2</label>
    </list_item>
  </stack>
</snap>
```



Listaus 2: Esimerkki GaigsXML-kielen tavasta kuvata yksi askel animaatiosta, jossa on tietorakenne pino. Huomattavaa on, että vaikka pinon määrittely koostuu listan elementeistä, pino havainnollistetaan enemmän taulokkoa muistuttavalla kuvalla.

```
OPTION DS B-TREE
INSERT 20 15 30 2 18 24 70 3 45
INSERT 10
PAUSE -- End of inserts
DELETE 24
```



Listaus 3: Esimerkki tietorakenteiden operaatioista DsCats-kielessä. Kuvassa tietorakenne kaikkien operaatioiden suorittamisen jälkeen.

```
array "values" (10, 10) length 5 int {3, 2, 4, 1, 7}
int pos = 1
int minIndex = 0
arrayMarker "pos" on "values" at Index pos label "pos"
arrayMarker "minIndex" on "values" at Index minIndex
    label "minIndex"
while ( pos < 5 ) {
  if ( values[pos] < values[minIndex] ) {
    minIndex = pos ;
    moveMarker "minIndex" to position pos within 5 ticks
  }
  pos = pos + 1
  moveMarker "pos" to position pos within 5 ticks
}
arraySwap on "values" position 0 with minIndex
    within 10 ticks
```

3	2	4	1	7

1	2	4	3	7

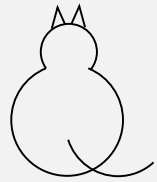
Listaus 4: Esimerkki ANIMALSCRIPT2-kielen ohjelmointiominaisuuksista. Kuvassa on taulukko elementtien vaihtamista ennen (yllä) ja sen jälkeen (alla).

```
<show>
  <snap>
    ...
    <question_ref ref="0"/>
  </snap>
  ...
  <questions>
    <question type="MCQUESTION" id="0">
      <question_text>What will the value of node A be in
           the next step?</question_text>
      <answer_option>3</answer_option>
      <answer_option is_correct="yes">8</answer_option>
      <answer_option>5</answer_option>
    </question>
  </questions>
</show>
```

Listaus 5: Esimerkki vuorovaikutteisista kysymyksistä GaigsXML-kielessä.

B XAAL-esimerkkejä

```
<define-shape name="kissa">
  <circle-segment>
    <center x="10" y="5"/>
    <radius length="4"/>
    <angle total="310" start="295"/>
  </circle-segment>
  <circle-segment>
    <center x="10" y="15"/>
    <radius length="7"/>
    <angle total="255" start="105"/>
  </circle-segment>
  <!-- Korvien määrittely viivoina, häntä kaarena -->
</define-shape>
```

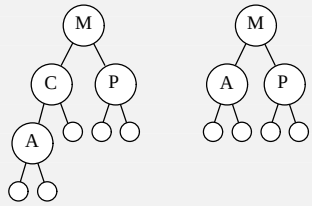


Listaus 6: Esimerkki jossa määritellään muoto (tässä tapauksessa kissa) graafisten primitiivien avulla.

```
<array indexed="false" size="7">
  <index index="0"><key value="A"/></index>
  <index index="1"><key value="B"/></index>
  <index index="4"><key value="C"/></index>
</array>
```

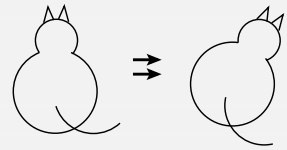
Listaus 7: Esimerkki taulukon määrittelystä.

```
<delete target="BST">
  <key value="C"/>
  <elementary>
    <remove target="nodeC"/>
    <remove target="edgeCA"/>
    <replace target="edgeMC">
      <edge from="nodeM" to="nodeA"/>
    </replace>
  </elementary>
  <graphical>
    <!-- operaation kuvaus graafisina operaatioina -->
  </graphical>
</delete>
```



Listaus 8: Esimerkki tietorakenneoperaatiosta. Kuvissa on rakenne ennen poistoa (vasemmalla) sekä poista-operaation jälkeen (oikealla). Esimerkissä on kuvattu sama operaatio myös yksinkertaisempina operaatioina sellaisia järjestelmiä varten, jotka eivät ymmärrä binäärisen hakupuun poisto-operaatiota.

```
<rotate degree="30" type="simple">
  <object-ref id="catObj1"/>
  <timing>
    <duration ms="2000"/>
  </timing>
  <coordinate x="10" y="10"/>
</rotate>
```



Listaus 9: Esimerkki XAAL-kielen kierrä-operaatiosta. Esimerkissä oletetaan, että `catObj1` on listauksessa 6 esitetyn muodon instanssi.