Tietojenkäsittelytiede **25** Joulukuu 2006 sivut 38–50
Toimittaja: Antti Valmari

Sijaintialgoritmitehtävien automaattinen tarkastaminen

Jussi Nikander
Teknillinen korkeakoulu
Ohjelmistotekniikan laboratorio
jtn@cs.hut.fi

Tiivistelmä

Sijaintialgoritmit ja -tietorakenteet ovat tärkeä osa geoinformatiikan perusopetusta. Opetusta ei kuiteinkaan ole tarjolla kuin harvoissa oppilaitoksissa, eikä aiheesta ole kattavaa oppikirjaa. Hajautunutta opetusta tukemaan on TKK:lla suunniteltu verkkopohjainen oppimisympäristö, jonka tärkeä osa on sijaintialgoritmien ja -tietorakenteiden automaattinen tarkastusjärjestelmä Spatial TRAKLA2. Tässä paperissa esitellään Spatial TRAKLA2:n periaate, aiottu käyttötarkoitus sekä käsitellään toteutukseen liittyviä haasteita.

1 Johdanto

Sijainti kaksi- tai kolmiulotteisessa avaruudessa ilmoitetaan yleensä koordinaattien avulla. Tietoa, johon on liitetty sijainti, kutsutaan *sijaintitiedoksi*. Sijaintitietoalkioita ovat esimerkiksi pisteet, murtoviivat ja erilaiset alueet. Koska sijaintitietoalkiot sijaitsevat avaruudessa, voidaan niille laskea arvoja, joilla ei tyypillisesti ole muunlaisille tietoalkioille merkitystä, kuten geometrinen etäisyys, alueen koko tai sisäkkäisyys. Sijaintitiedon käsittely on tärkeä osa *geoinformatiikkaa*, joka on tieteenala, missä tietoteknisiä menetelmiä käytetään geografisen tiedon käsittelyyn.

Geoinformatiikassa käsitelty sijaintitieto tyypillisesti kuvaa karttaa tietystä alueesta. Usein tietyn ongelman ratkaisemiseksi on analysoitava useita eri karttoja samasta alueesta: yksi kartta voi kuvata alueen asutusta ja tieverkostoa, toinen maalajeja ja kolmas kasvillisuutta. Yhdistämällä tietoa näistä kartoista voidaan esimerkiksi ennustaa hirvieläinten laidunmaita ja liikkumista alueella. Tämän tiedon avulla on puolestaan mahdollista suunnitella hirviaitoja ja mahdollisia eläintunneleita tai -ylikulkuja.

Sijaintialgoritmeilla tarkoitetaan sijaintitietojen käsittelyyn tarkoitettuja algoritmeja. Ne ovat tärkeä osa geoinformatiikan tietoteknisissä sovelluksissa ja tämän takia keskeinen osa alueen perusopetusta. Sijaintialgoritmien opetusta ei kuitenkaan ole tarjolla monessakaan oppilaitoksessa, eikä kunnollista opetusmateriaalia ole olemassa. Aiheesta on toki kirjoitettu useita oppikirjoja, kuten [3, 19, 20], mutta yksikään ei kata kaikkia tarvittavia perustietoja. Koska käsillä saatavilla oleva

OHJE KIRJAPAINOLLE: B5-arkin vasen- ja yläreuna kohdistetaan A4-arkin vasempaan ja yläreunaan. Näin pitäisi marginaaliksi tulla taitteen puolella noin 33 mm ja muualla noin 22 mm.

kirjallisuus ei ole kattava, on opetuksessa käsiteltävä oppikirjojen lisäksi esimerkiksi alan tieteellisiä artikkeleita. Tämä voi kuitenkin olla liiankin haastavaa kolmannen vuoden perusopiskelijoille, jotka tutustuvat ensimmäistä kertaa sijaintialgoritmeihin. Artikkelien käyttämä terminologia voi esimerkiksi olla hyvinkin vaihtelevaa, jolloin opiskelija helposti eksyy tuntemattomien merkintätapojen viidakkoon. Useissa artikkeleissa myös oletetaan, että lukija on alueen asiantuntija, jolloin opiskelijalle tärkeät aiheen perusteet voidaan kuitata hyvinkin nopeasti. Vaihtoehtoisesti oppimateriaalia voi pyrkiä tekemään itse, mikä lisää opettajan työmäärää huomattavasti.

2 Taustaa

Geoinformatiikan opetuksen hajautuneisuus ja opetusmateriaalin vähäisyys on herättänyt Teknillisessä korkeakoulussa kiinnostuksen verkkopohjaisen oppimisympäristön rakentamiseen. Ympäristön tarkoitus on toimia keskitettynä resurssina, jota voidaan käyttää hajallaan olevien opiskelijoiden opetuksessa. Tärkeä osa järjestelmää ovat automaattisesti tarkastettavat tehtävät, jotka toteutetaan Ohjelmistotekniikan laboratoriossa kehitetyn TRAKLA2-oppimisympäristön [13] avulla. Oppimisympäristön avulla opiskelijoille jaetaan algoritmisimulaatiotehtäviä, jotka he ratkaisevat manipuloimalla annettuja tietorakenteiden graafisia esityksiä hiiren avulla. Tuotettu simulaatiosarja tarkastetaan vertaamalla sitä toteutetun algoritmin tuottamaan sarjaan. Oppimisympäristöä rakennetaan TKK:n Ohjelmistotekniikan laboratorion sekä Geoinformaatioja paikannustekniikan laboratorion yhteistyönä.

2.1 Algoritmien havainnollistaminen

Tietorakenteet ja algoritmit ovat abstrakteja entiteettejä, joiden opetteleminen ei ole triviaalia. Pelkän algoritmin koodin ja mahdollisten sanallisten selitysten avulla on usein työlästä ottaa selvää miten tietorakenne on järjestetty, tai kuinka algoritmi tarkalleen manipuloi syötetietoa. Tietorakenteita ja algoritmeja usein havainnollistetaankin oppikirjoissa ja tieteellisissä julkaisuissa kuvien tai kuvasarjojen avulla. Kuvasarjojen piirtäminen käsin voi kuitenkin olla työlästä ja niitten muokkaaminen omaan opetukseen vaikeaan, joten opetuskäyttöön tarkoitettuja algoritmien havainnollistamisohjelmistoja on tuotettu useissa yliopistoissa [12, 15, 16, 17].

Tyypillinen algoritmien havainnollistamisjärjestelmä mahdollistaa tietorakennekuvien tuottamisen valmiita näkymiä käyttäen. Näkymä on esimerkiksi puu, jossa juuri on piirretty ylimmäiseksi, sen lapset seuraavalle tasolle, niiden lapset alemmaksi, ja niin edelleen. Näkymässä on tärkeää, että jokaisen puun solmun etäisyys juuresta ja paikka puuhierarkiassa näkyy kuvasta selkeästi. Esimerkiksi MatrixPro-järjestelmässä tällaisen kuvan voi tuottaa graafisesti asettamalla puun solmuihin halutut avainarvot [10]. Kuvien lisäksi järjestelmissä voidaan tuottaa algoritmianimaatioita, joiden avulla voidaan näyttää esimerkiksi kuinka tietorakenne muuttuu algoritmin suorituksen aikana.

Järjestelmiä voidaan käyttää joko luentotyökaluina, opiskelijoiden itsenäisen työskentelyn apuvälineinä tai harjoituksissa. Itsenäisen työskentelyn apuvälineen tai harjoitusvälineenä käytettävän järjestelmän on mahdollistettava riittävän aktiivinen vuorovaikutus opiskelijan kanssa, jotta järjestelmän käytöstä olisi hyötyä. Pelkkä algoritmianimaatioiden passiivinen katselu ei edistä oppimista [7].

Motivaatio monien havainnollistamisjärjestelmien tekemiseen on helpottaa ja nopeuttaa tietorakenne- ja algoritmikuvien tuottamista sekä mahdollistaa algoritmianimaatioiden tuottaminen. Kuitenkin vaikuttaa siltä, että yleinen syy olla käyttämättä havainnollistamisjärjestelmiä on niiden käyttöön kuluva aika ja vaiva [7]. Vaikuttaa myös siltä, että nykyisissä järjestelmissä helppokäyttöisyys ja mahdollisten havainnollistamistapojen monipuolisuus ovat ristiriidassa keskenään: mitä helppokäyttöisempi järjestelmä on, sitä rajoitetumpi se myös on [9].

2.2 Automaattinen tarkastaminen

Automaattinen tarkastaminen on prosessi, jossa tietokone arvioi opiskelijan vastauksen ilman ihmisen väliintuloa. Nykyään, kun yliopistojen kursseilla voi olla satoja opiskelijoita, on automaattinen tarkastaminen tärkeä opetuksen apuväline. Tietotekniikassa automaattista tarkastusta on käytetty varsinkin ohjelmointitehtävien tarkastamiseen, ja ensimmäiset järjestelmät valmistuivat jo 60-luvulla [5]. ITiCSE-konferenssin raportti automaattisen tarkastuksen käytöstä antaa hyvän kuvan siitä, miten automaattisia järjestelmiä käytetään nykyisin [2].

Raportissa tietokoneavusteisesti tarkastettavat tehtävät jaetaan viiteen kategoriaan: monivalintatehtävät, tekstivastaukset, ohjelmointitehtävät, graafisesti ratkottavat tehtävät ja vertaisarvioitavat tehtävät [2]. Näistä kategorioista viimeisessä ei käytetä automaattista tarkastusta, vaan tietokone jakaa tehtävät vertaisryhmille.

Monivalintatehtävät ovat hyvin yksinkertaisia toteuttaa, ja siksi laajimmalle levinnein automaattisesti tarkastettavien tehtävien muoto. Niitä käytetään monissa yhteyksissä, joilla ei ole mitään tekemistä opetuksen kanssa. Monivalintatehtävät ovat teknisesti helppoja toteuttaa, mutta hyvien tehtävien tekeminen voi olla työlästä.

Tekstivastaustehtävissä opiskelijan palautus on luonnollista tekstiä. Vastaus voi olla hyvin lyhyt (sana tai pari) tai kokonainen essee. Ohjelmointitehtävissä opiskelija palauttaa ohjelmakoodia, jonka eri ominaisuuksia automaattinen järjestelmä tarkastelee.

Graafisesti ratkottavissa tehtävissä tyypillisesti käytetään ohjelmistojen havainnollistamiskeinoja apuna tehtävän ratkaisuprosessissa. Kategoria eroaa muista siinä, että se kattaa kaikki tehtävät jotka ratkotaan graafisen työkalun avulla riippumatta aihealueesta tai vastauksen tyypistä. TRAKLA2-tehtävät ovat esimerkki graafisesti ratkottavista tehtävistä. Tietorakenteiden ja algoritmien lisäksi visuaalisesti ratkottavia tehtäviä on olemassa muun muassa formaaleista järjestelmistä [18] sekä UML-diagrammeista [6].

3 TRAKLA2-järjestelmä

TRAKLA2 on tietorakenteiden ja algoritmien automaattinen tarkastusjärjestelmä, joka otettiin TKK:lla ensimmäisen kerran testikäyttöön kevään 2003 tietorakenteiden ja algoritmien peruskurssilla. Se korvasi kokonaan kurssilla käytetyn vanhan TRAKLA-järjestelmän [8] keväällä 2004. Järjestelmä on tämän jälkeen otettu käyttöön muun muassa Turun yliopistossa, Tampereen teknillisessä yliopistossa ja Åbo akademissa. Opiskelijat ovat olleet tyytyväisiä järjestelmään ja kokevat sen auttavan opiskelua [11].

TRAKLA2-järjestelmä koostuu kahdesta osasta: verkkoympäristöstä ja Java-sovelmasta. Verkkoympäristö huolehtii opiskelijoiden tunnistamisesta, pistekirjanpidosta ja muista kurssin ylläpitoon

tarvittavista toiminnoista. Verkkoympäristö on geneerinen ja sitä voi käyttää minkä tahansa kurssin yhteydessä. Java-sovelma on upotettu verkkoympäristöön. Sovelman avulla opiskelijat ratkovat TRAKLA2-tehtäviä.

TRAKLA2-tehtävät ovat visuaalisia algoritmisimulaatiotehtäviä [13], joita ratkotaan WWW-sivuun upotetun sovelman avulla. Visuaalisissa algoritmisimulaatiotehtävissä sovelma antaa käyttäjälle joukon tietorakenteita, joihin hänen on tehtävä muutoksia tietyn algoritmin mukaisesti. Tietorakenteet havainnollistetaan graafisina visualisaatioina ja käyttäjä muokkaa niitä manipuloimalla annettuja symboleja esimerkiksi napsauttamalla tai raahamalla niitä hiiren avulla. Järjestelmä peilaa visualisaatioihin tehdyt muutokset vastaaviin tietorakenteisiin, joten käyttäjä muokkaa oikeita tietorakenteita visualisaation kautta.

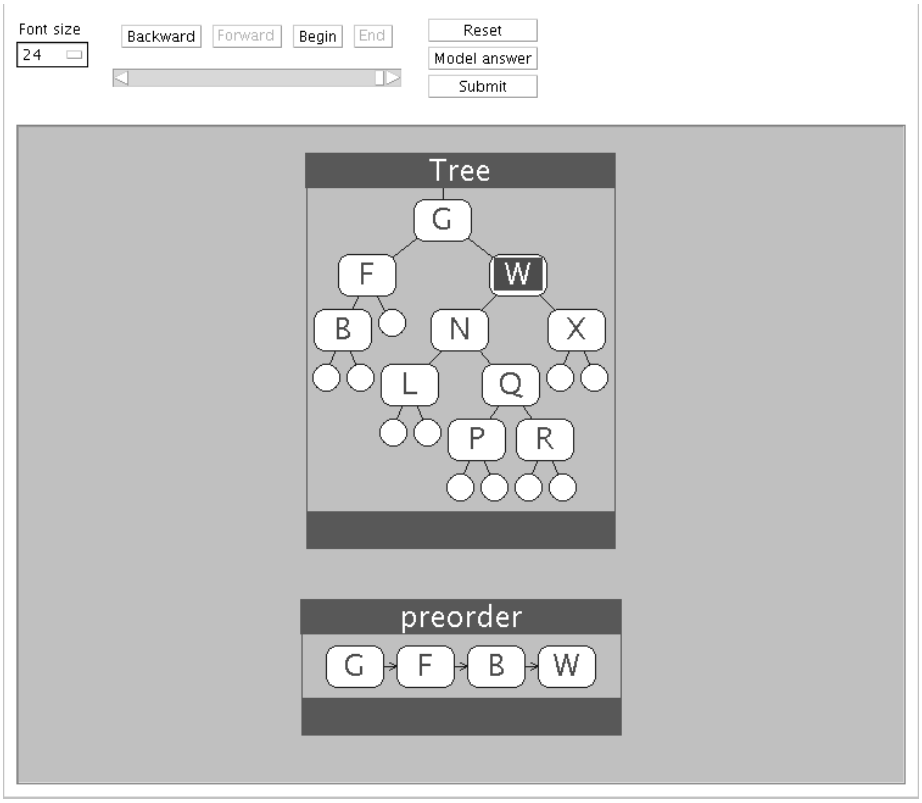
Kuvassa 1 näkyy tyypillinen TRAKLA2-tehtävä. Ylhäällä on ohjauspaneeli, jolla käyttäjä voi siirtyä tuottamassaan toimintosarjassa eteen- ja taaksepäin, alustaa tehtävän uudelleen, pyytää arviointia tai katsoa tehtävän mallivastauksen. Ohjauspaneelin alla on tietorakenteiden visualisaatiot. Tässä tehtävässä syöterakenteena on binäärinen hakupuu, joka on kuvassa näkyvistä kahdesta tietorakenteesta ylempi. Opiskelijan tehtävänä on käydä annettu binäärinen hakupuu läpi esijärjestyksessä. Läpikäynti suoritetaan raahamalla puun solmut oikeassa järjestyksessä sen alapuolella olevaan linkitettyyn listaan. Raahatun solmun avain kopioidaan listaan. Mikäli opiskelija ratkaisee tehtävän oikein, ovat puun alkiot tehtävän lopuksi listassa oikeassa järjestyksessä.

Opiskelija voi pyytää tehtävän arviointia tai katsoa sen mallivastausta milloin tahansa. Arvioinnista annetaan palautteena muun muassa oikeiden operaatioiden lukumäärä sekä tehtävästä annettu pistemäärä. Mallivastaus on simulaatiotehtävissä graafinen algoritmianimaatio, joka näyttää miten oikea algoritmi muokkaa annettuja tietorakenteita. Muissa tehtävissä mallivastaus voi esimerkiksi kertoa monivalinnan oikeat vaihtoehdot. Mallivastauksen katsomisen jälkeen ei tehtävään voi enää pyytää arviointia, vaan se täytyy alustaa uudelleen. Myös arvioinnin pyytämisen jälkeen on tehtävä alustettava uudelleen. Alustettaessa tehtävään arvotaan uudet alkuarvot.

Simulaatiotehtävien tarkastaminen perustuu mallivastauksen ja opiskelijan tuottaman toimintosarjan vertailuun ja tapahtuu kaikissa tehtävissä samalla algoritmilla. Tarkastettaessa tehtävää opiskelijan toimintosarjan tiloja verrataan mallivastauksen tiloihin. Koska käyttäjän tekemät muutokset visualisaatioon heijastuvat vastaaviin tietorakenteisiin, voidaan käyttäjän tekemän toimintosarjan oikeellisuus tarkastaa vertaamalla sitä algoritmin tuottamiin tiloihin. Koska järjestelmä pystyy tallentamaan sekä käyttäjän että algoritmin tuottaman toimintosarjan, voidaan molemmista tehdä algoritmianimaatio, jossa tehdyt muutokset näytetään askel askeleelta.

4 Työn päämäärä ja haasteet

TRAKLA2-järjestelmää on käytetty jo useita vuosia menestyksekkäästi tietorakenteiden ja algoritmien peruskursseilla. Alusta lähtien on kuitenkin ollut tarkoitus laajentaa järjestelmää kattamaan myös muita aihealueita. Sijaintialgoritmien laajennus, työnimeltään Spatial TRAKLA2, on ensimmäinen algoritmiikan erikoisalue, joka järjestelmään lisätään.



Kuva 1: TRAKLA2-sovelma, jossa puun läpikäyntitehtävä

4.1 opetuksellinen haaste

Spatial TRAKLA2 on tarkoitus ottaa keväällä 2007 käyttöön Geoinformaatio- ja paikannustekniikan laboratorion sijaintitietorakennekurssilla[1]. Kurssi on suunnattu geoinformatiikan kolmannen vuoden opiskelijoille ja sen esitietona on geoinformatiikan, tietorakenteiden ja algoritmien sekä ohjelmoinnin perusteiden tuntemus. Kurssin opiskelijoilta vaaditaan vain kaksi ohjelmointiin ja algoritmiikkaan liittyvää opintojaksoa, joten heiltä ei voi olettaa samanlaista tietoteknistä osaamista kuin kolmannen vuoden tietotekniikan opiskelijoilta. Esimerkiksi opiskelijoiden valmiudet seurata ja tulkita pseudokoodia eivät välttämättä ole samalla tasolla kuin tietotekniikan opiskelijoilla.

Opetuksessa on tärkeää ottaa opiskelijoiden tausta ja tarpeet huomioon. Koska Spatial TRAKLA2 on suunnattu ensisijaisesti geoinformatiikan opiskelijoille, ei esimerkiksi ole järkevää syventyä käsiteltävien tietorakenteiden toteutuksen yksityiskohtiin. Työelämässä suurin osa opiskelijoista ei joudu toteuttamaan kurssilla käsiteltäviä tietorakenteita itse. Todennäköisempää on, että he käyttävät työssään joko ohjelmistoja tai ohjelmakirjastoja, joissa tarvittavat rakenteet on toteutettu valmiiksi. Täten tärkeämpää kuin toteutuksen yksityskohdat, ovat algoritmin yleinen toimintaperiaate, sen soveltuvuus tiettyjen ongelmatapausten ratkaisemiseen sekä tehokkuus verrattuna muihin samaan ongelmaan kehitettyihin algoritmeihin.

4.2 Visualisointihaaste

Geoinformatiikassa tiedon graafisella esittämisellä on tärkeä osa, joten esimerkiksi sijaintitietoa käsiteltäessä on syytä käyttää opiskelijoille tuttuja havainnollistamiskeinoja, kuten erilaisia karttanäkymiä. Alueen algoritmit käsittelevät (vähintään) kaksiulotteisessa avaruudessa sijaitsevaa sijaintitietoa, joten data-alkiot ovat joko pisteitä, murtoviivoja tai polygoneja. Näiden esittämiseen luontevin tapa on piirtää käytetty avaruuden osa ja siinä olevat dataelementit. Tällaiset näkymät ja kuvat ovat geoinformatiikan opiskelijoille keskeinen ymmärtämisen väline. He ovat tottuneet käyttämään karttaesityksiä ja menetelmien havainnollistaminen kuvia piirtämällä on heille luonnollinen oppimismuoto. Lisäksi useat sijaintitietorakenteet sisältävät tietoa, jolle on olemassa luonnollinen visualisaatio kaavakuvana tai karttana. Esimerkiksi korkeustieto voidaan näyttää korkeuskäyrien tai värikoodauksen avulla, kaupunkien sijainnit karttanäkymässä ja niin edelleen.

Monien sijaintitietorakenteiden vakiintunut visualisaatio ei suoraan ota kantaa siihen, miten rakenne on toteutettu. Esimerkiksi Voronoi-diagrammi [1] on tiettyyn pistejoukkoon S perustuva kaksiulotteisen alueen jako pienempiin osiin. Alue jaetaan polygoneihin siten, että polygoniin p kuuluvat kaikki alueen pisteet, joita lähinnä on tietty pistejoukon piste $s \in S$. Voronoi-diagrammi visualisoidaan yleensä polygoniverkkoina riippumatta siitä, mitä tietorakennetta käyttäen diagrammi on tallennettu. Monet oppikirjoissa ja tieteellisissä artikkeleissa esitellyistä sijaintiongelmien ratkaisuista sisältävät useita kuvia, joiden avulla selitetään esiteltävän algoritmin toimintaa. Näitä kirjallisuudesta tuttuja visualisaatioita tulisi käyttää myös TRAKLA2-tehtävissä, koska niiden avulla käyttäjän on helppo yhdistää tehtävä johonkin tiettyyn aihealueeseen.

Valmiit visualisaatiot voivat joissain tapauksissa myös helpottaa TRAKLA2-

[1] http://www.tkk.fi/Units/Cartography/courses/maa-123340/

tehtävien suunnittelua. Tällöin tehtävää varten on jo valmiiksi olemassa selkeä malli siitä, millaisia visualisaatioita sen esittämiseen voitaisiin käyttää. Usein tehtäviä toteutettaessa ongelmaksi muodostuukin tarkan algoritmin toteuttaminen. Geoinformatiikan oppikirjoissa ja tieteellisissä artikkeleissa ei välttämättä ole esitetty algoritmin koodia laisinkaan. Usein ongelman ratkaisumenetelmä kerrotaan muun tekstin lomassa, eikä siihen ole sisällytetty minkäänlaista pseudokoodia. Koska TRAKLA2-tehtävät kuitenkin tarkastetaan vertaamalla opiskelijan tuottamaa algoritmisimulaatiosarjaa oikean algoritmin tuottamaan toimintosarjaan, on algoritmi toteutettava, jotta tehtävä voidaan tarkastaa.

Sijaintialgoritmitehtävien ratkaisemiseen ei kuitenkaan tulisi tarvita algoritmin yksityiskohtien tuntemusta, vaan tehtävien tulisi kuvata algoritmit korkealla käsitetasolla käyttäen hyväksi geoinformaatikoille jo valmiiksi tuttuja visualisaatioita. Tällöin tehtävät palvelevat kurssin opetustarkoitusta, eli tiettyjen sijaintialgoritmien toiminnan yleisperiaatteiden opettelua. Korkeasta käsitetasosta huolimatta tehtävien tulisi testata algoritmin keskeisen toiminnallisuuden sekä perusidean hallintaa. Ratkaistakseen tehtävän opiskelijan tulisi pystyä osoittamaan, että hän tietää mitä ongelmaa algoritmi ratkaisee, ja miten algoritmin suoritus etenee.

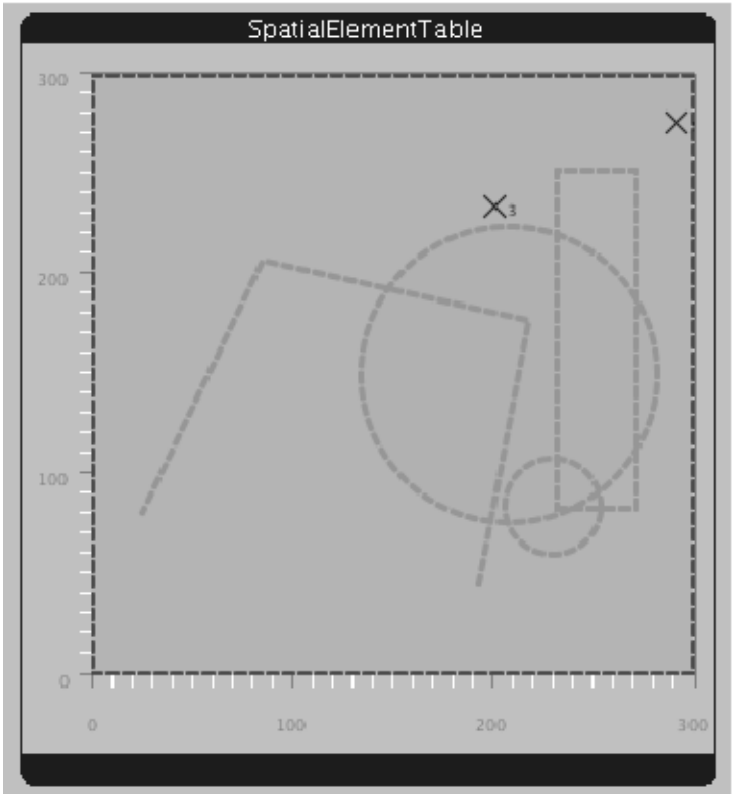
Korkean käsitetason visualisaatioita käytettäessä on pidettävä huolta siitä, etteivät ne trivialisoi tehtävää. Visualisaatioon on helppo sisällyttää suuria määriä tietoa, mitä itse algoritmilla ei ole käytettävissään. Tällöin opiskelija esimerkiksi pystyy tekemään algoritmille mahdottomia päättelyjä, jolloin tehtävän ratkaiseminen helpottuu, eikä se enään testaa simuloitavan algoritmin toiminnan osaamista.

4.3 Tekninen haaste

Kun sijaintitietorakenntehtäviä ryhdyttiin suunnittelemaan, ei TRAKLA2-järjestelmässä ollut ainuttakaan tietorakennetta tai visualisaatiota, joka olisi millään tavalla ottanut huomioon moniulotteiseen avaruuteen sijoittuvaa dataa. Kaikki järjestelmän visualisaatiot keskittyivät tietorakenteen sisäisen hierarkian selkeään näyttämiseen, jolloin moniulotteisten dataelementtien keskinäisten suhteiden näyttäminen ei yleensä ole mahdollista.

Ennen projektin alkua järjestelmässä oli neljä eri graafista metaforaa, joiden avulla tietorakenteita voitiin siinä visualisoida: taulukko, lista, puu ja graafi. Kaikki visualisaatiot käyttivät oppikirjoistakin tuttuja kanonisia näkymiä, jotka jokainen tietorakenteita tunteva assosioi tietyntyyppiseen rakenteeseen. Esimerkiksi puu esitettiin joukkona solmuja, jotka on järjestetty eri tasoille ylhäältä alaspäin siten, että jokaisen solmun lapset ovat solmua seuraavalla tasolla, ja solmusta on kaaret lapsisolmuihin.

Kanonisista näkymistä ainoastaan graafivisualisaatiossa olisi edes mahdollista ottaa huomioon moniulotteinen data. Kanoninen graafinäkymä on joukko vapaasti kaksiulotteiseen avaruuteen sijoitettuja solmuja, sekä näiden välisiä kaaria. Useimmissa graafien asetteluissa solmut ovat ruudulla siten, että niiden väliset kaaret on mahdollisimman helppo erottaa toisistaan. Solmut on myös mahdollista asettaa tiettyihin paikkoihin esimerkiksi niiden sisältämän koordinaattitiedon perusteella. Tällöin solmujen paikat ruudulla voisivat esimerkiksi vastata kaupunkien paikkoja maastossa ja niiden väliset kaaret kaupunkien välisiä yhteyksiä. Tällaista graafiasettelua järjestelmässä ei kuitenkaan ollut.



Kuva 2: TRAKLA2-järjestelmän aluevisualisaatio, jossa useita tietoalkioita

Ennen kuin uusia tehtäviä oli mahdollista toteuttaa, oli järjestelmää siis laajennettava ja siihen tuotava uusia visualisaatiota. Tehtäviä varten järjestelmään lisättiin uusi asettelu graafeille sekä yksi uusi visualisaatio. Uudessa graafiasettelussa graafin solmuille lisättiin koordinaatit, joiden mukaan sen paikka visualisaatiossa määräytyi.

Kokonaan uusi visualisaatio oli *aluevisualisaatio*, joka kuvaa tietyn kaksiulotteisen avaruuden osan sekä sen sisältämät dataelementit. Esimerkki aluevisualisaatiosta näkyy kuvassa 2. Alueessa olevien dataelementtien oletetaan olevan kaksiulotteisen avaruuden entiteettejä. Visualisaatio tukee neljänlaisia tietoalkioita: pisteitä, murtoviivoja, ellipsejä ja polygoneja. Lisäksi alue on mahdollista jakaa pienemmiksi osa-alueiksi mielivaltaisella tavalla. Alueen dataelementit visualisoidaan vihreällä, ja alueen jaot osiksi punaisella. Esimerkiksi kuvassa 2 olevassa aluevisualisaatiossa on kuusi dataelementtiä, eikä sitä ole jaettu pienempiin osa-alueisiin.

5 Toteutettavat tehtävät

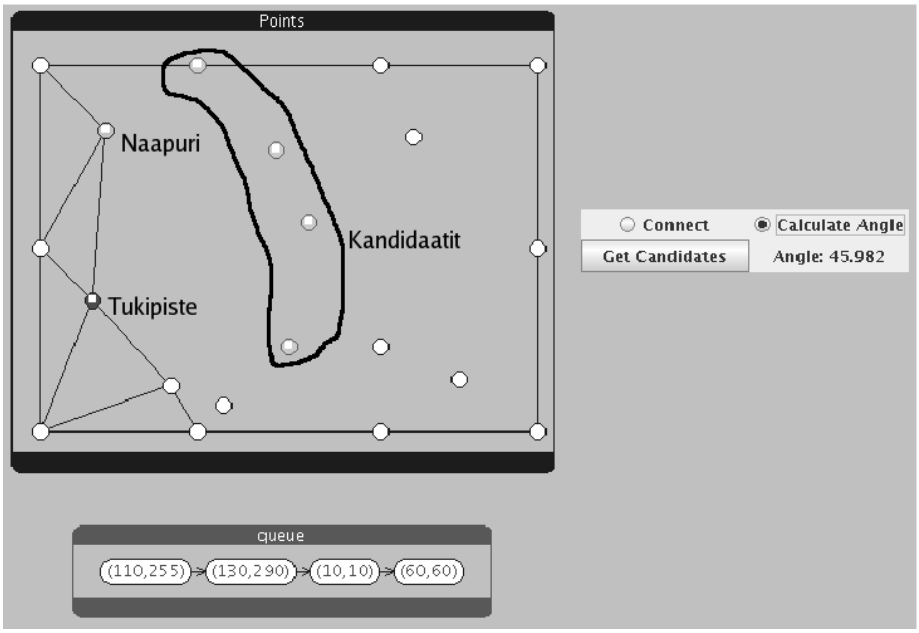
Tehtävien toteuttaminen on projektissa kolmivaiheinen operaatio. Ensin toteutettavasta tehtävästä tehdään käsikirjoitus, jossa määritellään tehtävän tarkoitus, tarkka tehtävänanto, tehtävässä käytettävät tietorakenteet, visualisaatiot, sekä toiminnot, joiden avulla opiskelija voi ratkaista tehtävän. Käsikirjoituksen avulla tehtävä toteutetaan. Valmis tehtävä on Java-luokka, jossa on tarvittava toiminnallisuus tehtäväinstanssien luomiseen, tietorakenteiden visualisointiin sekä mallivastausten tuottamiseen. Valmis tehtävä testataan järjestelmän testiympäristössä ennen käyttöönottoa.

Projektin käsikirjoitusvaihe valmistui keväällä 2006. Kaiken kaikkiaan seitsemästätoista sijaintitietorakennetehtävästä tehtiin käsikirjoitus.Tehtävät eivät kata kaikkia TKK:n kurssilla käsiteltäviä tietorakenteita ja algoritmeja. Sijaintitiedon esitystavat, ja siten myös sijaintitietoa käsittelevät algoritmit, voidaan jakaa kahteen luokkaan: vektorimalliin ja rasterimalliin. Vektorimallissa sijaintitieto tallennetaan käyttämällä pisteitä, murtoviivoja ja polygoneja. Jokainen tietoalkio sisältää koordinaattinsa, kattamansa alueen sekä tiedot kattamansa alueen sisällöstä. Rasterimallissa maasto jaetaan säännöllisen ruudukon avulla pieniin osiin, joista jokainen osanen sisältää kuvaamansa maaston osan tiedot.

Käsikirjoitusvaiheessa tultiin siihen tulokseen, että rasterimalleja käsitteleviä algoritmeja ei kannata toteuttaa TRAKLA2:n visuaalisten algoritmisimulaatiotehtävien avulla. Tehtäviä suunniteltaessa huomattiin, että rasterimalleja käyttävät algoritmit tekevät lähinnä matemaattisia operaatioita käytettävälle datalle, eivätkä kovinkaan paljoa muokkaa itse tietorakenteiden sisältöä esimerkiksi siirtämällä dataelementtejä paikasta toiseen. Puhtaan matematiikan tekeminen visuaalisen algoritmisimulaation avulla on rasittavaa tehtävän ratkaisijalle, joten rasterimallit jätettiin pois käsiteltävien algoritmien joukosta.

5.1 Esimerkki: Delaunay-kolmionti

Dalaunay-kolmiointi [4] on eräs sijaintitiedon analysoinnissa käytetty matemaattinen malli. Delaunay-kolmionti on pistejoukon kolmionti, jossa minkään kolmion ympärille piirretyn ympyrän sisäpuolelle ei jää yhtään pistettä. Kolmionnissa pistejoukosta muodostuu yhtenäinen verkko. Delaunay-kolmiointi on Voronoi-diagrammin duaali: kolmioin-



Kuva 3: Delaunay-kolmiointitehtävän sovelma

nin naapuripisteet muodostavat Voronoi-diagrammissa vierekkäiset polygonit.

Kuvassa 3 näkyy kuvakaappaus tehtävästä, jossa käyttäjän täytyy muodostaa pistejoukolle Delaunay-kolmionti käyttäen expanding wave -tekniikkaa [14]. Expanding wave-tekniikassa kolmioitavan pistejoukon ympärille muodostetaan suorakaiteen muotoinen apupisteiden joukko. Apupisteiden lisäksi algoritmi käyttää jonoa, jonka ensimmäinen alkio otetaan algoritmin jokaisella iteraatiolla tukipisteeksi.

5.1.1 Expanding wave-tekniikka

Ensimmäiseksi tukipisteeksi valitaan mielivaltainen suorakaiteen kulmapiste. Jokaisella iteraatiolla algoritmi valitsee ensin tukipisteen jonkin naapurpisteen (verkossa) tunnetuksi naapuriksi. Tämän jälkeen algoritmi etsii tuki- ja naapuripisteelle uuden naapurin. Uudelle naapuripisteelle ja tukipisteelle etsitään seuraava naapuri, kunnes uusia naapureita ei enään löydy, jolloin jonosta otetaan seuraava tukipiste. Aina kun löydetään uusi naapuri, muodostetaan verkkoon uusi tukipisteen, vanhan naapurin ja uuden naapurin määrittämä kolmio. Naapuripisteet, joita ei ole aikaisemmin lisätty jonoon lisätään sinne löydettäessä. Uudet naapurit etsitään tukipisteestä katsoen joko myötä- tai vastapäivään, riippuen siitä mistä suorakulmion kulmasta algoritmin toiminta alkaa, ja miten uuden tukipisteen ensimmäinen naapuripiste etsitään.

Löytääkseen uuden naapurin algoritmi määrittää ympyrän, jonka kaarta tuki- ja naapurpiste leikkaavat. Aluksi tämän ympyrän halkaisija on tuki- ja naapurpisteen välinen etäisyys, joten ympyrän keskipiste on pisteiden muodostaman janan puolivälissä. Algoritmi etsii kaikki ympyrän sisäl-

lä olevat pisteet, jotka sillä puolella tukija naapuripisteen muodostamaa janaa, mihin algoritmi pyörähtää.

Mikäli ympyrän sisällä ei ole ainuttakaan kandidaattipistettä, määrittää algoritmi uuden, suuremman ympyrän, jonka kaarella tuki- ja naapuripiste ovat. Uuden ympyrän keskipiste ei enään ole pisteiden muodostamalla janalla, vaan jälleen janasta pyörähdyssuuntaan.

Mikäli ympyrän sisällä on vain yksi piste, valitaan se uudeksi naapuriksi. Mikäli ympyrän sisällä on useita pisteitä, valitaan uudeksi naapuriksi se, jonka avulla kulma (tukipiste — uusi piste — naapuripiste) saadaan mahdollisimman suureksi. Mikäli useampi piste antaa yhtäsuuren kulman, valitaan se piste, joka on lähinnä tukipistettä.

5.1.2 Tehtävän ratkaiseminen

Kuvasta 3 on jätetty pois TRAKLA2-sovelman ohjauspaneeli, joka näkyy esimerkiksi kuvassa 1. Ylimpänä kuvassa on kolmioitava pistejoukko. Pistejoukkoa ympäröi suorakaide. Suorakaiteella olevat pisteet ovat algoritmin tarvitsemia apupisteitä ja sen sisällä olevat pisteet ovat pisteitä, joille kolmiointia ollaan tekemässä. Kuvaan on merkitty tukipiste, sen tunnettu naapuri ja kandidaattipisteet uudeksi naapuripisteeksi. Tehtäväsovelmassa tukipiste on piirretty punaisella, tunnettu naapuri keltaisella ja kandidaattipisteet vihreällä. Tehtävässä pyörähdys tukipisteen ympäri tulee suorittaa myötäpäivään.

Pistejoukon oikealla puolella on tehtävässä tarvittavat toimintonapit, joilla käyttäjä voi valita haluamansa toimitilan. Ylemmillä valintanapeilla kerrotaan mitä tapahtuu kun käyttäjä napsauttaa jotain pistejoukon pistettä. Mikäli “connect” -toiminto on valittu, pisteen napsauttaminen muodostaa uuden kolmion napsautetun pisteen, naapuripisteen ja tukipisteen ympärille, ja vaihtaa napsautetun pisteen valituksi naapuriksi. Mikäli “calculate angle” -toiminto on valittu, pisteen napsauttaminen laskee tukipisteen, valitun pisteen ja sen hetkisen naapuripisteen muodostaman kulman suuruuden. “Get candidates” -nappia painamalla etsii järjestelmä kandidaatit seuraavaksi naapuripisteeksi. Painamalla nappia uudestaan saa suuremman ympyrän tuottaman kandidaattijoukon, kunnes alue jolta kandidaatteja etsitään saavuttaa suurimman mahdollisen koon. Napin vieressä olevaan tekstikenttään merkitään kulman suuruus calculate angle -toimintoa käytettäessä.

Pistejoukon alla on jono, johon käyttäjä lisää algoritmin löytämät uudet pisteet ja josta hän ottaa seuraavan käsiteltävän pisteen.

Pistejoukon vasemmassa reunassa on joukko pisteitä, joiden välille on piirretty viivoja. Nämä pisteet on jo käsitelty ja järjestelmä on piirtänyt vastaavat kolmiot. Niiden oikealla olevat vihreäksi väritetyt pisteet ovat sen hetkisiä naapurikandidaatteja. Käyttäjä on ottanut jonosta tukipisteen sekä löytänyt sen tunnetun naapurin. Nyt hän etsii uusia naapuripisteitä tukipisteelle käyttämällä “get candidates” -toimintoa. Ympyrää ei tehtävässä ole piirretty johtuen käytetyn sovelluskehyksen rajoituksista.

Tehtävä ratkaistaan aloittamalla pistejoukon käsittely vasemmassa alakulmassa olevasta pisteestä. Käyttäjä etsii sen vieruspisteet ja lisää ne jonoon. Tämän jälkeen hän ottaa jonosta ensimmäisen pisteen ja etsii sen uudet naapurit lisäten ne jonoon. Tehtävän tekeminen loppuu, kun jono on tyhjä, jolloin alueelle on muodostunut Delaunay-kolmiointi. Tehtävän tekemisessä käyttäjällä on apunaan mahdollisuus etsiä sen hetkisen tuki- ja naapuripisteen avulla uudet naapurikandidaatit, sekä mitata kolmen pisteen muodostaman kulman suuruus.

6 Yhteenveto ja jatkokehitys

Spatial TRAKLA2 on sijaintitietorakenteiden ja algoritmien laajennus laajassa käytössä olevaan TRAKLA2-järjestelmään. Sitä on tarkoitus käyttää TKK:ssa osana kartografian ja geoinformatiikan opetusta. Laajennuksen käyttäjäkunta, aihealueen hyvin graafinen luonne, sekä korkean käsitetason visualisointien luomat ongelmat tekevät toteutuksesta haastavan.

Laajennuksessa on päätetty toteuttaa joukko tietorakenteita ja algoritmeja jotka käsittelevät sijaintitietoa lähinnä käyttämällä vektorimallia. Tällaiset tehtävät soveltuvat hyvin TRAKLA2-järjestelmän tehtäviksi. Tehtäviä varten on kuitenkin täytynyt kehittää järjestelmään uusia havainnollistamisnäkymiä. Lisäksi useat algoritmit ovat monimutkaisia, ja niiden simulointi järjestelmän käyttöliittymän läpi voi olla monimutkaista, kuten esimerkistä osiossa 5.1 nähtiin.

Vektorimallin lisäksi sijaintitiedon käsittelyssä käytetään myös rasterimallia, jota käsittelevät algoritmit eivät kuitenkaan sovellu nykyisiin TRAKLA2-tehtäviin. Osana järjestelmän jatkokehitystä onkin tarkoitus tarkastella, millaisia tehtäviä rasterimallin algoritmeista olisi mahdollista tehdä. Voi olla, että rasteritehtäviä varten on käytettävä jotain muuta sovelluskehystä, tai tuotava TRAKLA2-järjestelmään aivan uudenlaisia vuorovaikutusmalleja sekä havainnollistamismenetelmiä.

Viitteet

[1] F. Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[2] J. Carter, J. English, K. Ala-Mutka, M. Dick, W. Fone, U. Fuller, and J. Sheard. ITICSE working group report: How shall we assess this? *SIGCSE Bulletin*, 35(4):107–123, 2003.

[3] M. de Berg, M. V. Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[4] B. Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR*, 7:793–800, 1934.

[5] G. E. Forsythe and N. Wirth. Automatic grading programs. *Communications of the ACM*, 8(5):275–278, 1965.

[6] C. Higgins, P. Symeonidis, and A. Tsintsifas. The marking system for CourseMaster. In *Proceedings of the 7th annual conference on Innovation and Technology in Computer Science Education*, pages 46–50. ACM Press, 2002.

[7] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, June 2002.

[8] J. Hyvönen and L. Malmi. TRAKLA – a system for teaching algorithms using email and a graphical editor. In *Proceedings of HYPERMEDIA in Vaasa*, pages 141–147, 1993.

[9] P. Ihantola, V. Karavirta, A. Korhonen, and J. Nikander. Taxonomy of effortless creation of algorithm visualizations. In *ICER'05:*

Proceedings of the 2005 international workshop on Computing education research, pages 123–133, New York, NY, USA, 2005. ACM Press.

[10] V. Karavirta, A. Korhonen, L. Malmi, and K. Stålnacke. MatrixPro — A tool for on-the-fly demonstration of data structures and algorithms. In A. Korhonen, editor, *Proceedings of the Third Program Visualization Workshop*, Research Report CS-RR-407, pages 26–33, The University of Warwick, UK, July 2004. Department of Computer Science, University of Warwick, UK.

[11] M.-J. Laakso, T. Salakoski, L. Grandell, X. Qiu, A. Korhonen, and L. Malmi. Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system TRAKLA2. *Informatics in Education*, 4(1):49–68, 2005.

[12] P. LaFollette, J. Korsh, and R. Sangwan. A visual interface for effortless animation of C/C++ programs. *Journal of Visual Languages and Computing*, 11(1):27–48, 2000.

[13] L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, O. Seppälä, and P. Silvasti. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education*, 3(2):267–288, 2004.

[14] M. J. McCullagh and C. G. Ross. Delaunay triangulation of a random data set for isarithmic mapping. *The Cartographic Journal*, 17(2), December 1980.

[15] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari. Visualizing programs with jeliot 3. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 373–376, New York, NY, USA, 2004. ACM Press.

[16] F. Naharro-Berrocal, C. Pareja-Flores, J. Urquiza-Fuentes, J. Ángel Velázquez-Iturbide, and F. Gortázar-Bellas. Redesigning the animation capabilities of a functional programming environment under an educational framework. In *Second Program Visualization Workshop*, pages 59–68, HornstrupCentret, Denmark, 2002.

[17] G. Rößling, M. Schüler, and B. Freisleben. The ANIMAL algorithm animation tool. In *Proceedings of the 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'00*, pages 37–40, Helsinki, Finland, 2000. ACM Press, New York.

[18] V. Tscherter, R. Lamprecht, and J. Nievergelt. Exorciser: Automatic generation and interactive grading of exercises in the theory of computation. In *Fourth International Conference on New Educational Environments*, pages 47–50, 2002.

[19] S. Wise. *GIS Basics*. Taylor & Francis, 2002.

[20] M. Worboys and M. Duckham. *GIS: A Computing Perspective*. Taylor & Francis, 2004.