Journal of Information Technology Education: Innovations in Practice **Volume 9, 2010**

Algorithm Visualization System for Teaching Spatial Data Algorithms

Jussi Nikander, Juha Helminen, and Ari Korhonen
Aalto University, Finland

jtn@cs.hut.fi; jhhelmi2@cs.hut.fi; archie@cs.hut.fi

Executive Summary

TRAKLA2 is a web-based learning environment for data structures and algorithms. The system delivers automatically assessed algorithm simulation exercises that are solved using a graphical user interface.

In this work, we introduce a novel learning environment for spatial data algorithms, SDA-TRAKLA2, which has been implemented on top of the TRAKLA2 system. Spatial data items are identified by a set of coordinates, such as x and y for two-dimensional spatial data. The spatial environment contains new visualizations for representing spatial data, and a number of new exercises that cover a variety of spatial data algorithms.

The new exercises have been used in the spatial data algorithms course at the Helsinki University of Technology since spring 2007. Here, we summarize previous research and report on an analysis of the quantitative data on the students' learning outcomes for years 2007 and 2008. We have also analyzed the students' learning results using qualitative methods in order to discover how the new system affects the students' learning outcomes.

Keywords: Algorithm visualization, automatic assessment, learning environments, spatial data algorithms, TRAKLA2

Introduction

Spatial data is data that is located in a multidimensional space (Laurini & Thompson, 1992). In other words, each spatial data item is identified by a set of coordinates, which define its location in relation to other data items. Thus, each spatial data item contains *spatial information* (coordinates) that describes the location of the item and associated *attribute data* that describes what it represents. Spatial data is used in numerous disciplines, such as geographic information systems (GIS), computer graphics, robotics, virtual reality, computer–aided design, biology, VLSI design, and many others. In the context of GIS and related disciplines, the data is assumed to model geographic locations on the Earth's surface and their associated properties. In this paper, we discuss spatial data in this context. In GIS, there are at least two coordinate axes (x and y, which represent geographical longitude and latitude), and two additional ones (height and time) can also be used. *Spatial data algorithms* (SDA) are algorithms designed to process and manipulate such data and *spatial data structures* are entities used to store the data.

Spatial data structures are based on regular non–spatial data structures such

Editor: Kam Vat

as arrays and trees, as well as algorithms that manipulate these basic structures. However, the multidimensional nature of spatial data makes them more complex than the basic data structures. This also makes the design and implementation of efficient spatial data algorithms more difficult. For example, dictionary structures for spatial data must be able to locate data items according to their coordinates instead of using one-dimensional key values.

Furthermore, the spatial nature of the data makes teaching spatial data algorithms more challenging than basic non–spatial data structures and algorithms. For example, in order to show both how data items are related to each other and how they are arranged in a data structure, typically two illustrations are required: one to show the data items and the space they occupy, and another to show the data structures. Consequently, the learner must be able to connect the illustrations in order to comprehend how the data is organized. In basic data structures, on the other hand, the relationships between the data items are typically distinguishable with just a single picture.

SDA and associated data structures are an integral part of *geoinformatics*, a branch of science where computer science is applied to cartography and geosciences. The data geoinformatics studies is location data on the Earth's surface, and therefore SDA are required for efficient storage and processing. Geoinformatics is also closely related to cartography, and therefore many different kinds of illustrations are used. Maps are the most fundamental way to represent spatial information. Visualization of maps is in itself a large, varied and important field (Slocum, McMaster, Kessler, & Howard, 2004). However, since many spatial data sets have several properties for each location, other visualization methods are also required for understanding the data. For example, multivariate visualization techniques such as parallel coordinate plots or star plots can be used in conjunction with map views. A map view shows how the data is distributed geographically, while other views show what information the data contains.

Software Visualization (SV) is a branch of software engineering that aims to use graphics and animation to illustrate the different aspects of software (Stasko, Domingue, Brown, & Price, 1998). Price, Baecker, and Small (1993) divide SV into two subcategories: program visualization (PV) and algorithm visualization (AV). PV is the use of various visual techniques to enhance the human understanding of computer programs. It is typically used to illustrate actual, implemented programs. AV, on the other hand, illustrates abstractions of algorithmic concepts and is independent of any actual algorithm implementation. In this paper, we will concentrate on AV.

One of the primary uses for SV is education. Numerous SV systems have been developed for teaching purposes. It has been noted by Hundhausen, Douglas, and Stasko (2002) that merely introducing SV to teaching does not seem to improve learning results. In order to benefit from the use of SV, the learner must become an active participant in the learning process by interacting with the visualizations. They must, for example, construct an animation or simulate the workflow of a data structure.

One way to activate the learner is to use exercises where interaction with the algorithm visualizations is required for solving the problem. For example, the learner could manipulate data structure visualizations in order to simulate the modifications a real algorithm would do to the data structures in question. We call these exercises *visual algorithm simulation exercises* (Korhonen, 2003; Malmi et al., 2004). Such exercises have been implemented in the TRAKLA2 system. Previous studies show that TRAKLA2 is an effective teaching tool (Laakso et al., 2005; Malmi, Korhonen, & Saikkonen, 2002) for basic data structures and algorithms. The exercises incorporate *automatic assessment* that allows the learners to practice without the need for instructor participation. Automatic assessment is a computerized procedure that takes a learner-made solution to an exercise as an input and produces a mark for the solution without human intervention. This novel approach seems to be a promising alternative also for explaining how the hard-to-grasp spatial data algorithms and data structures work.

In this paper, we describe and evaluate how the idea of visual algorithm simulation exercises can be used in the learning of spatial data structures and algorithms (SDA). We have created a learning environment for spatial data algorithms, based on the TRAKLA2 system (Nikander & Helminen, 2007). In this work, we call this spatial extension "SDA-TRAKLA2" to differentiate it from the base system. We have already done preliminary analysis on the use of the system in teaching SDA (Nikander, Helminen, & Korhonen, 2008). The system has been in use in the SDA course at the Helsinki University of Technology since spring 2007. Here, we report on the findings from the course in years 2007 and 2008. The results are based on several data sources and analysis methods including linear regression comparison of TRAKLA2 exercise performance with exam results, content analysis of the exam answers, student interviews, attitude studies (questionnaire), and course feedback. The overall attitude of the students has been positive, and they generally wish to have more TRAKLA2 exercises, as the system does not yet cover all the course topics. Furthermore, there is a strong correlation between successfully solving TRAKLA2 exercises and performing well in the final examination. Finally, the students' examination answers show influence of TRAKLA2 exercises.

The rest of this paper is organized as follows. In the background and related work section, we survey related work on software visualization and automatic assessment. In the system description section, we give an overview of the SDA-TRAKLA2 system and a detailed description of one spatial algorithm exercise. In the research setting and data collection section, we introduce the course in detail, and enumerate the research methods we have used to evaluate the use of the system on the course. In the prior studies section we summarize some of our previous publications, and do a new analysis on some of the old results. The qualitative results section describes the qualitative analysis of the students' learning outcomes. The discussion section contains discussion about the results and the use of the system, and last section contains our conclusions.

Background and Related Work

Geoinformatics contains elements from both cartography and earth sciences as well as elements from computer science. However, the focus in teaching geoinformatics is often on how to use available tools and techniques instead of how to develop them. Therefore, geoinformatics courses often deal with how to run different types of analyses, such as numerical modeling (Karssenberg, Burrough, Sluiter, & de Jong, 2001), simulations (Crouch, Shen, Austin, & Dinniman, 2008), or exploratory data analysis (Andrienko, Andrienko, Fischer, Mues, & Schuck, 2006) and how to use standard tools such Matlab (Degrande & Geraedts, 2008). Internet and various eLearning environments have also been utilized (Degrande & Geraedts, 2008; Purves, Medyckyj-Scott, & Mackaness, 2005).

Information visualization is often used in teaching geoinformatics as well, but the use of software visualization is rare. Several visualization systems exist that can be used to illustrate some spatial data structures, algorithms, and concepts such as geometric algorithms (Hausner & Dobkin, 1998; Hipke & Schuierer, 1998; Shneerson & Tal, 2000) or Voronoi diagrams (Fisher, 2004). There are also numerous applets and applet collections in the web that handle the topic, but their quality varies greatly, and the maintenance or course integration such applets have is unknown. For applet examples, see http://www.cosc.canterbury.ac.nz/mukundan/JavaP.html
http://www.diku.dk/hjemmesider/studerende/duff/Fortune/
http://www.cs.cornell.edu/home/chew/Delaunay.html .

It seems, however, that most existing SDA visualization systems are designed for and used in the computer science domain as opposed to geoinformatics. Furthermore, none of the systems come even close to covering the basic set of spatial data structures typically used in geoinformatics. In addition, many of the existing systems seem to use only one view, a two-dimensional area, to show how the algorithm operates on the data. Such limited visualization completely omits how the data structures used in the algorithm are modified as the algorithm executes and thus gives the viewer only a limited view of how the algorithm operates. It also seems that the visualization of spatial structures and algorithms is not currently an active area of research. A recent paper on the MAVIS algorithm visualization tool has one spatial algorithm example (Koifman, Shimshoni, & Tal, 2008), but we are not aware of other recent progress in the field.

In computing education, software visualization is utilized more often. Especially for the topic of basic data structures and algorithms, numerous visualization systems have been implemented at many different institutions. For examples see Akingbade, Finley, Jackson, Patel, and Rodger (2003), Hundhausen and Brown (2005), Karavirta, Korhonen, Malmi, and Stålnacke (2004), and Rößling (2002). Early systems focused on creating clear and attractive visualizations. This is reflected in Price et al. (1993), which grouped systems based on how well the different aspects of the graphical representation have been implemented and how versatile the graphical representations are. A more recent work by Hundhausen et al. (2002) recognized that the pedagogical effectiveness of a visualization depends more on the level of engagement the learners have with the visualizations than on the attractiveness of the graphical representation. Therefore, many modern systems are more concerned about the type and amount of interaction they offer than how visually pleasing the graphical representations are. Types of interaction can be categorized using, for example, the engagement taxonomy (Naps et al., 2003).

Software visualization can be used to create interactive exercises that the students can solve by manipulating the visual elements (Janhunen, Jussia, Järvisalo, & Oikarinen, 2004; Malmi et al., 2004; Tscherter, Lamprecht, & Nievergelt, 2002). When such exercises are combined with automatic assessment, the learners can get feedback on their solutions without human intervention. Automatic assessment is also commonly used in computing education separately from software visualization. Solutions range from multiple-choice question systems (Denny, Hamer, Luxton-Reilly, & Purchase, 2008) to those assessing programming assignments (Benford, Burke, Foxley, Gutteridge, & Zin, 1993; Saikkonen, Malmi, & Korhonen, 2001) and style (Ala-Mutka, Uimonen, & Järvinen, 2004). For more in-depth investigation on the use of automatic assessment, see Carter et al. (2003).

System Description

Visual Algorithm Simulation Exercises

Visual algorithm simulation (Korhonen, 2003; Malmi et al., 2004) is a novel technique for allowing a user to directly manipulate data structures on the screen. Not only does it enable user interaction with data structure visualizations, but it also makes it possible to modify real data structures by using simple GUI operations. The user can *simulate* real algorithms by manipulating elements on the screen. This concept can easily be extended to automatically assessed *visual algorithm simulation exercises* where the grading is based on comparing the student-made simulation sequence to a sequence produced by an actual algorithm implementation.

In *tracing exercises* the task is to simulate a specific algorithm with the given input. Thus, the focus of the exercise is on the understanding of the given algorithm and how that particular algorithm manipulates data. The primary method of interaction is clicking and drag-and-dropping data item visualizations, although other basic GUI components, such as buttons and combo boxes, are also used in some exercises. A typical interface operation in an exercise is to drag an item from its current location to another. This simulates the action of deleting a data item from one structure and inserting it into another. The new location could be another position in the same structure or in another data structure altogether.

In tracing exercises, the input of the algorithm fully determines the control flow of the algorithm's execution, and the learner's task is to reproduce the execution by appropriately manipulating visualizations. At each step, the learner traces the pseudo code and is forced to think of how the algorithm works in order to carry out the next operation. For example, the learner might drag an item from a stack visualization onto a visualization of a search tree and drop it there, thus inserting the item into the tree. In essence, they are constructing an animation of the algorithm's execution. The assessment of the solution is then carried out by comparing the learner's animation sequence to a sequence generated by running an actual implementation of the algorithm. Feedback is given in terms of the number of correct steps from the beginning of the sequence and by showing the correct solution as an algorithm animation.

Open tracing exercises, on the other hand, are more exploratory in nature and are used to allow students to examine a specific concept through visual exploration. In these exercises, the correctness of the solution is evaluated based on the end state. An example of such an exercise is the coloring of a red black tree: the student is not required to follow a *specific* algorithm, but is asked to give any correct coloring. Thus, the focus in this exercise is on the *conceptual knowledge* and understanding of the rules that govern the creation of red black trees (such as "the root node is black" or "red nodes cannot have red children"). These conditions are easy to check, and the feedback can explicitly state which constraints are satisfied and which are not. For more discussion about different types of exercises, see Korhonen and Malmi (2004).

TRAKLA2

TRAKLA2 is a web-based learning environment aimed at teaching data structures and algorithms (Malmi et al., 2004). The system contains a number of visual algorithm simulation exercises and instructional material, help files, and links to additional material. The basic exercise set includes assignments for basic data structures, sorting, hashing, graph algorithms, and algorithm analysis. For going beyond the basic data structures and algorithms of core computer science, the system was extended to support the visualization and manipulation of spatial primitives and data structures.

TRAKLA2 visualizations are built using a general-purpose application framework written in Java

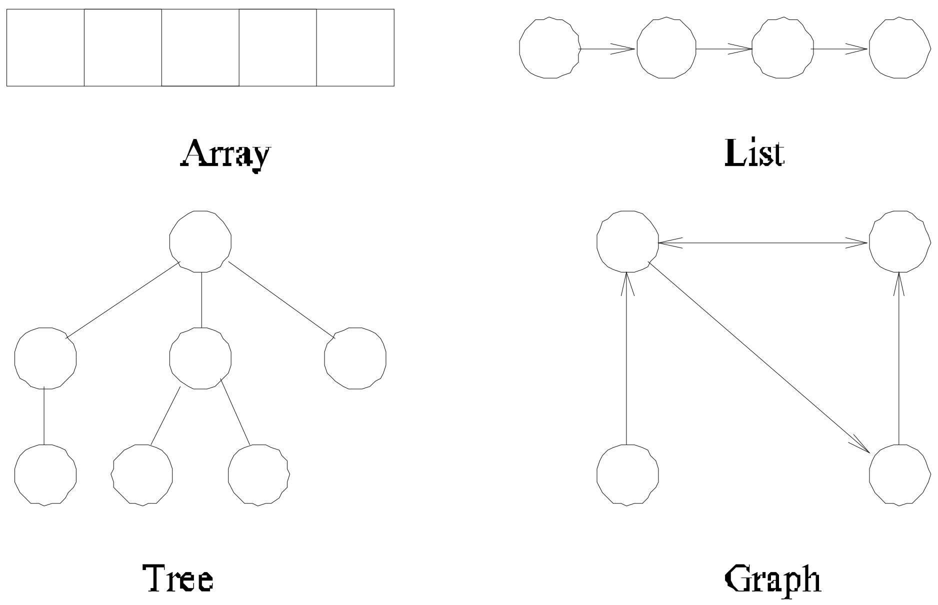


Figure 1: The canonical views of basic data structures.

for creating algorithm animations and simulations. In the system, all data structures are visualized by combining instances and variations of a small number of well-known visualizations, which we call *canonical views* (Nikander, Korhonen, Valanto, & Virrantaus, 2007). We use four canonical views that represent the data structures array, list, tree and graph. Examples of canonical views can be seen in Figure 1.

The framework also allows for many simultaneous synchronized views of the same underlying data. In other words, we can, for example, have both an array and a tree visualization of a heap data structure, where invoking operations via either of them modifies the same underlying data structure and triggers an update of both visualizations. Furthermore, visualizations can be combined hierarchically. For example, a B-tree may be visualized as a tree in which each node contains a data item that is visualized as an array.

The SDA-TRAKLA2 Environment

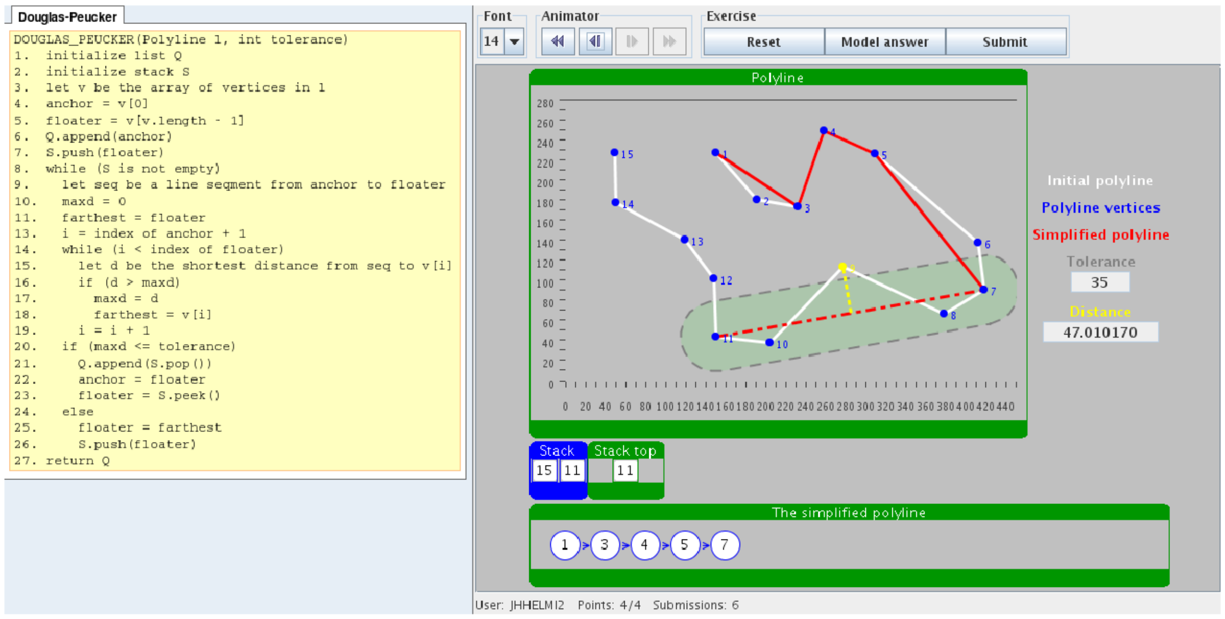
The canonical views in TRAKLA2 can accurately depict the internal hierarchy of a data structure. For example, a tree is visualized as a layered hierarchy of nodes that are connected by edges that represent links between the data items. However, because of spatial properties such as proximity, length, and direction, these types of visualizations cannot adequately represent the relationships between spatial data items. Although we might use a list of coordinate values to represent spatial data, this type of rudimentary visualization is insufficient for conveying spatial information. For example, a polygon could be represented by a tuple of the coordinates of its vertices. However, using such visualization it would be hard to make any observations on the area of the polygon or the overlap of several polygons. To tackle this, we included a new visualization to the TRAKLA2 system called the *area view*.

The area view is fundamentally different from the canonical views. The canonical views depict the internal hierarchy of a data structure. An area view, on the other hand, depicts a rectangular region of a 2-dimensional plane and can, therefore, be used to show data item positions in a 2-dimensional coordinate space. For example, in the area visualization we can show how spatial primitives, such as points, lines, and polygons, are laid out in the associated space, as shown in Figure 3. The elements can be differentiated using colors, transparency, line thickness, and dash types. However, while this does allow us to illustrate spatial relationships well, it is equally difficult to depict the structural relationships of data in this view as it is to display spatial attributes with canonical visualizations. Thus, in order to get the complete picture, we generally need two simultaneous views of the data: a data structure visualization based on canonical views to show how the data is stored and an area view to show how the data items relate to each other in terms of their coordinates. An example of using several different views simultaneously can be seen in Figure 4. In addition, it is possible to include extra visual cues in the area view that represent conceptual elements. An example of this is drawing a sweep line in sweep line algorithms, or the Delaunay triangulation circumcircle depicted in Figure 3.

By using an area view and associated two-dimensional data items we created SDA-TRAKLA2, a novel learning environment for spatial data algorithms. Using the area view in SDA-TRAKLA2 the learner can see how the spatial data items are arranged in the space they occupy. By including also canonical views we can create even more engaging exercises where the learner can also see how the data is arranged in data structures and how the contents of these structures change as the algorithm executes. Furthermore, by designing the exercises so that both the area view and the canonical views are required in order to solve the exercise, we can force the learner to interact with all relevant visualizations. Since interaction between the user and the visualizations is shown to be crucial in order to facilitate learning, having the learner to interact with the visualization increases the value of the exercises.

The Exercises

The set of exercises in TRAKLA2 is divided into rounds with defined publish dates, deadlines, and minimum requirements. Figure 2 shows a screenshot of an exercise. An exercise consists of a description and instructions for solving the exercise (not shown in the picture), pseudo code for the algorithm (in tracing exercises), and an interactive Java applet for carrying out the assignment.



```
DOUGLAS_PEUCKER(Polyline l, int tolerance)
initialize list Q
initialize stack S
let v be the array of vertices in l
anchor = v[0]
floater = v[v.length - 1]
Q.append(anchor)
S.push(floater)
while (S is not empty)
  let seq be a line segment from anchor to floater
  maxd = 0
  farthest = floater
  i = index of anchor + 1
  while (i < index of floater)
    let d be the shortest distance from seq to v[i]
    if (d > maxd)
      maxd = d
      farthest = v[i]
    i = i + 1
  if (maxd <= tolerance)
    Q.append(S.pop())
    anchor = floater
    floater = S.peek()
  else
    floater = farthest
    S.push(floater)
return Q
```

Tolerance 35

Distance 47.010170

User: JHHELMI2 Points: 4/4 Submissions: 6

Figure 2: SDA-TRAKLA2 exercise for the Douglas-Peucker line simplification. Pseudo code for the algorithm is on the left, and the Java applet for solving the exercise is on the right.

For SDA-TRAKLA2 we have implemented two types of exercises (as defined above in the sub-section on Visual Algorithm Simulation Exercises): tracing and open tracing exercises. For example, in an exercise exploring the construction of a Voronoi diagram by means of the Delaunay triangulation (see Figure 3), the learner is provided with interface operations for modifying the edges of the triangulation and testing for Delaunay conditions. Success is evaluated based on whether the points are correctly connected to create the new triangulation. One example exercise is described in more detail in the next sub-section, and a list of all implemented exercises is given in the Appendix.

Finally, one of the key ideas in TRAKLA2 is to allow multiple tries for each assignment. This is possible because, for each attempt, a new set of random input is generated. The randomized input also prevents the students from using trial-and-error problem solving methods and prevents plagiarism. Care must be taken, however, to ensure that exercise instances are neither impossible to solve nor trivial. This is typically achieved by setting a number of constraints that the input must conform to. For example, if the task is to insert items to a balanced binary tree, then the input should be such that the tree needs to be balanced during the simulation sequence.

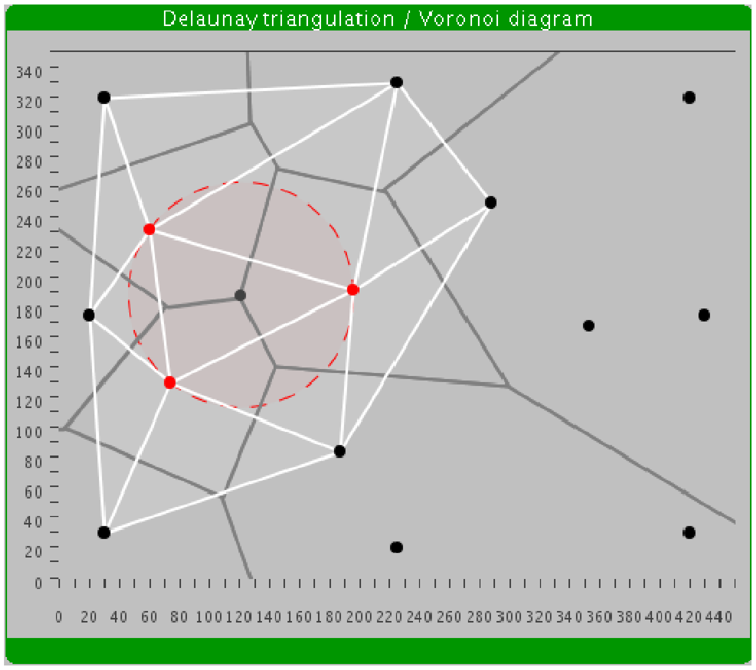


Figure 3: TRAKLA2 exercise for Delaunay triangulation construction. The Figure shows a partially constructed Delaunay triangulation using white lines, the corresponding Voronoi diagram using dark lines, and one Delaunay circumcircle using a dashed red line.

An Example Exercise: Line Segment Intersections

The line segment intersections problem is to find all intersections in a given set of line segments. Such a problem is trivial for a human to answer, once the line segments have been drawn. A computer, however, cannot see the intersections, and therefore the problem is algorithmically non-trivial.

One efficient solution to the problem is to use a *line sweep* (Bentley & Ottmann, 1979). In the line sweep approach a conceptual sweep line is moved across the plane, stopping at each line segment endpoint or intersection. The algorithm finds intersections by using an adjacency structure, which contains the line segments that intersect the sweep line at any given point. The adjacency structure can be used to search for line segment intersections effectively. The line sweep algorithm is capable of finding all line segment intersections in $O(nlogn+klogn)$ time, where k is the number intersections in the data set.

Figure 4 shows a screenshot of the SDA-TRAKLA2 exercise applet for the problem. The applet contains four data structure visualizations. In the top row there is a tree view of a binary heap, which is used as a priority queue, and an area view showing the line segments, their intersections and the current position of the sweep line. In the bottom row there is the adjacency structure used by the algorithm, visualized as an array, and a linked list for storing the output. In the figure, the student has started solving the exercise. As can be seen from the area view, three line segment intersections have been discovered in the simulation and are explicitly shown. Intersections are drawn to the area view as points when the corresponding line segments are next to each other in the adjacency structure, simulating how the real algorithm finds the intersections.

The visualizations used in the exercise give two very different views to the problem. The priority queue and adjacency structure views show internal details of the data structures used in the algorithm. The area view, on the other hand, shows how the data is arranged spatially. The output visualization is used to store the intersections.

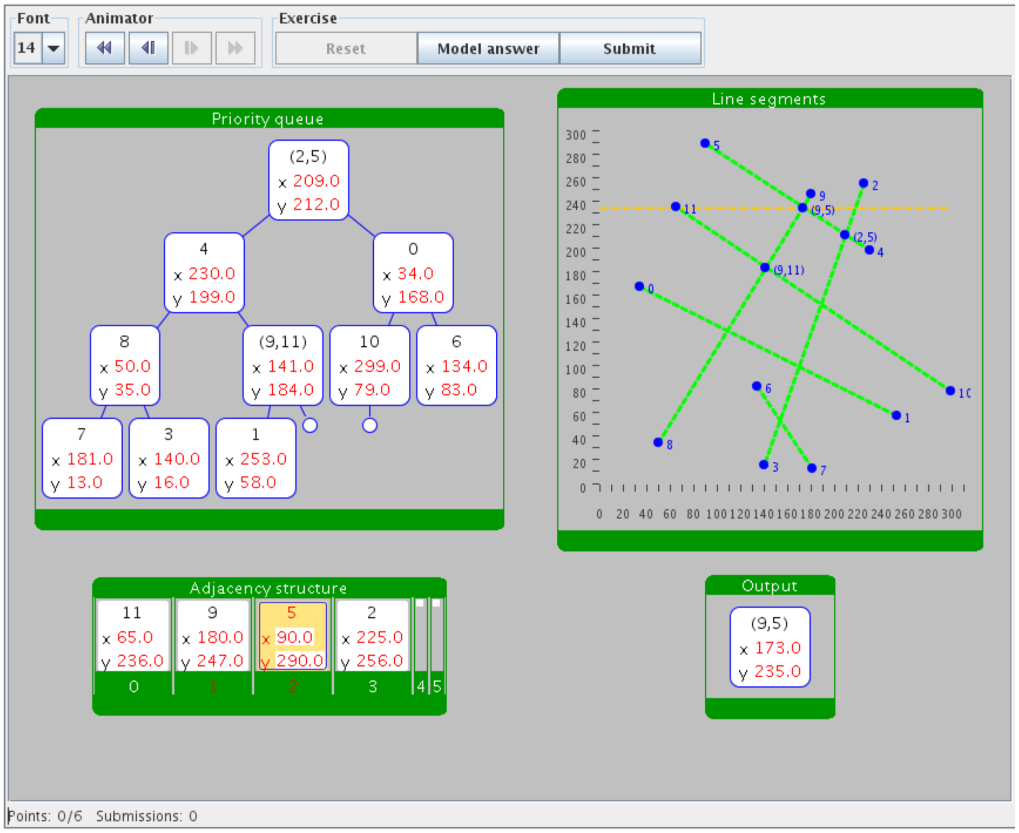


Figure 4: Line segment intersections with line sweep.

To solve the exercise, the student needs to simulate how the algorithm manipulates the priority queue, the adjacency structure, and the output. Snapshots of the solution process can be seen in Figure 5. At the beginning of the simulation, the priority queue contains all line segment end-points, and the sweep line is at the top of the area visualization, as shown in the top left snapshot. To solve the exercise, the student then dequeues items from the priority queue, maintains the adjacency list, and adds intersections to the priority queue and the output. All these actions can be done by drag-and-dropping items to the appropriate positions in the visualizations. The second snapshot, at the top right, shows the visualizations after four items from the priority queue have been handled. The sweep line has advanced and currently is at y-coordinate of the last dequeued item. In this case, it is the line segment endpoint labeled 0. At this point, two intersections have been found and added to the priority queue. The next item to be processed is the intersection labeled (0,6), after the top endpoints of the corresponding line segments. At the bottom left snapshot, the algorithm simulation is at a point where the last item handled was the intersection (4,10). The bottom right snapshot shows the situation at the end of the simulation process. Priority queue and adjacency structure are empty, and all intersections have been added to the output.

The line segment intersection exercise has been used on the SDA course since the year 2007. It was completely re-designed after the first year, and Figures 4 and 5 illustrate the current version. In the original design the sweep line swept from left to right and was not visualized in the area view, and the line segment endpoints and intersections used much larger visualizations. The original version of the exercise was much harder for the students to solve than the current version, and most gained approximately 50% of the points. Using the new version students typically gain 80-100% of the points from the exercise.

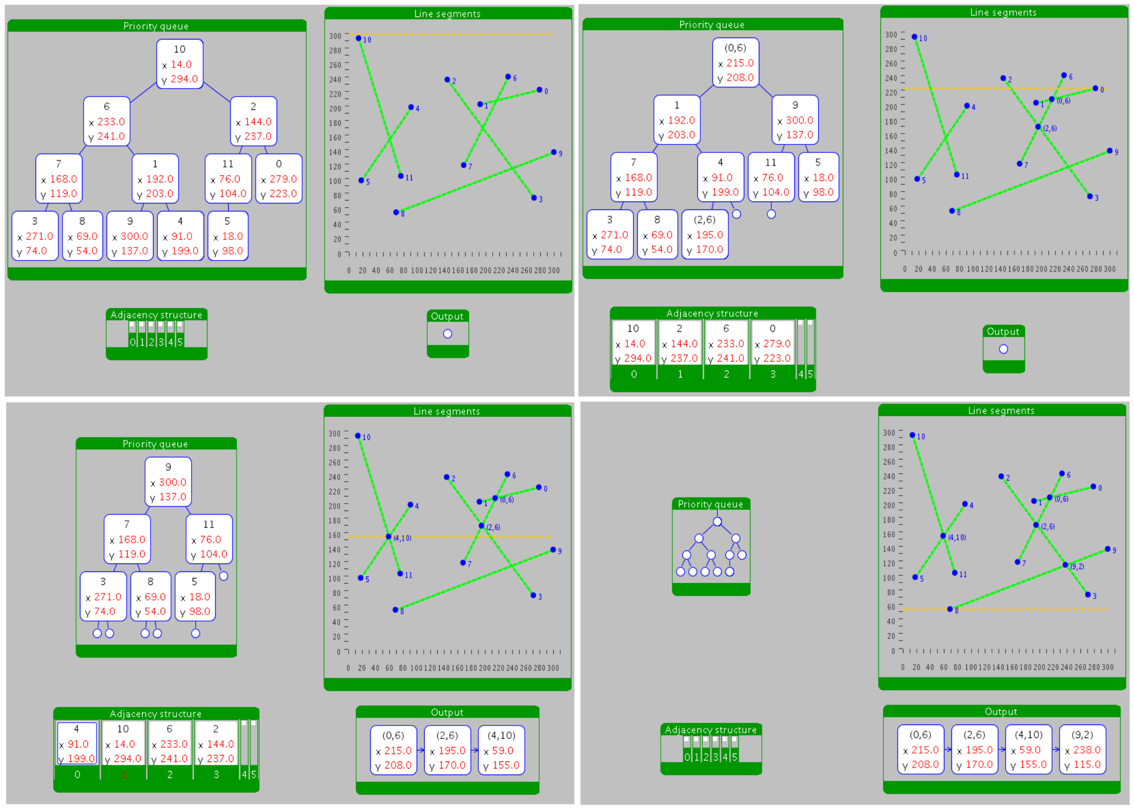


Figure 5: Four snapshots of solving the line segment exercise.
Snapshots start at the top left, and end at the bottom right picture.

Research Setting and Data Collection

The SDA-TRAKLA2 was implemented for the Spatial Data Algorithms course (SDA course) at the Helsinki University of Technology. The course is given by the Department of Surveying each spring. The prerequisites are basic knowledge on geoinformatics, programming, data structures, and algorithms. Thus, the students should have passed CS1 and CS2, as well as the geoinformatics equivalents of those before enrolling for the course. The SDA course typically has 15 to 25 students per year. Most participants are third year students of the Department of Surveying.

Details of the Spatial Data Algorithm Course

The SDA course is worth 6 ECTS credits and is an obligatory part of the bachelor's studies for students who specialize in geoinformatics. After having completed the course, the students should be able to define and compare spatial data algorithms, be able to select efficient algorithms for spatial problems, and be able to describe how spatial algorithms and data structures work. Furthermore, the students should be able to implement simple spatial data structures and algorithms. It should be noted, however, that the SDA course is intended to be just an introduction to spatial algorithms and covers only a small fraction of the whole field.

The data structures and algorithms on the course are mostly presented on a conceptual level, as opposed to a specific algorithm implementation. Thus, the minute details of the algorithm implementation are not discussed, and the focus is more on how different problems can be solved and how the algorithms work on a high level of abstraction. Thus, it is more important to know how the sweep line problem solving method works and what problems can be solved using it, than to know how the intersection of two line segments can be calculated, or how the pointers in an adjacency structure need to be modified in order to maintain it.

Before the introduction of SDA-TRAKLA2, the course consisted of combined lecture and studio sessions, a programming project, and a final examination (exam). The lecture and studio sessions were the first part of the course. There were 6 or 7 four-hour sessions on the course, held once a week. Each session consisted of a small lecture – typically one hour – which introduced the topic to the students. At the end of the lecture the students were divided into groups of three or four. Each group was given one spatial data structure or algorithm, which they then had two hours to familiarize themselves with. During the last hour of the session each group presented their results to the rest of the class. The goal of the studio sessions was for the students to familiarize themselves with one data structure or algorithm on a conceptual level. They could use lecture slides, handouts, book excerpts or research articles as study material.

After the studio sessions were over, the students had their first chance to take the exam. The exam consisted of four or five questions, which typically tested the students' conceptual understanding of the topics. A typical exam question was to describe an algorithm for solving a given problem. In case they failed their first attempt, the students typically had at least two additional opportunities to take the exam.

The students also started the programming project after the sessions had ended. The students worked in the project either alone or in pairs. They were required to study and implement one spatial algorithm or data structure and also present their results to the rest of the class. They had approximately seven weeks to complete the project.

It was deemed that there was room on the course for additional learning activities. Thus, the SDA-TRAKLA2 system was included in spring 2007 without touching the other parts of the course. The motivation for including SDA-TRAKLA2 was to increase the amount of practical exercises the students had and to strengthen their understanding on how algorithms work. Furthermore, the system allowed the students to learn how certain data structures and algorithms work in more detail by giving them hands-on experience with the algorithms on a conceptual level. The course staff was also interested in the system, since by using SDA-TRAKLA2 the students could solve the problems independent of time and place, and because of the automatic assessment functionality the use of the system would not require additional human resources.

The SDA-TRAKLA2 exercises on the course were divided into a couple of rounds in order to even up the students' workload and to prevent them from trying to do too much work at once. Deadlines had one or two weeks between them. In order to pass the course the students were required to gain at least 50% of the points awarded from the exercises. There was no further advantage they could gain from solving more exercises.

Research Methods and Goals of the Evaluation

In order to ascertain whether adding SDA-TRAKLA2 to the course affected student learning, we evaluated the use of the system using several methods. The goal was also to ascertain whether it was possible to successfully extend TRAKLA2 beyond basic data structures and algorithms. Thus, during the years 2007 and 2008 we gathered data on both student learning outcomes and their attitudes towards the system.

The data on student learning outcomes was gathered from SDA-TRAKLA2 and the students' exam answers. The system stores all student submissions, enabling us to see how many times each student has submitted each exercise and how each submission was assessed. The course staff gave us the students' exam answers. Two methods were used for collecting data on student attitudes. First, at the end of the course, all students filled out an anonymous course feedback form.

This consisted of questions both about the students' attitudes towards the course as a whole and towards the SDA-TRAKLA2 system. Second, after the spring 2008 course, four students were interviewed (Nikander et al., 2008). We used both quantitative and qualitative methods to analyze the data gathered. The decision to use both methods rose partially out of necessity, as the small size of the course prevented us from using a formal research setting where the students were divided into test and control groups. Partially the decision to include qualitative methods came from the available data. We had access to the students' exam answers and also had used interviews as one data gathering method.

Quantitative learning data we decided to analyze using standard statistical methods, namely linear regression analysis on the learning outcomes. The output of the analysis shows us – if you accept its validity despite the small sample sizes – whether there is statistical correlation between the student learning activities and their learning outcomes. The students' exam answers we decided to analyze also using content analysis (Krippendorff, 2004). We used content analysis to search how the effect of SDA-TRAKLA2 exercises could be found in the students' exam answers and inferred from that whether the system played a part in the students' learning.

The student attitudes we analyzed using the feedback questionnaires and interviews. On the feedback questionnaires we used simple summaries of student attitudes and compared the students' answers to various different questions to one other. Thus, we could achieve a ranking for the students' attitudes towards different learning activities on the course. Similarly, from the verbal feedback part of the questionnaires we could infer what parts of the course the students felt strongly about and what kinds of attitudes they had. In the interviews we used the interview guide approach (Patton, 2002), where the interviewer has an outline of topics to be covered, but may vary the wording and order of the questions to some extent.

We have already published some results on using the SDA-TRAKLA2 in teaching. The results discussed in the next section have, for the most part, already been published in Nikander et al. (2008). The quantitative analysis of students' learning outcomes presented in the first part of the section is, however, revised from the previous publication. First, the material used in Nikander et al. (2008) later turned out to be incomplete. Here we have used the whole dataset. Second, the two years' exams shared one question which was, unfortunately, graded using very different criteria on the different years. We therefore decided to regrade the students' answers in order to make the results comparable. Third, in the next section we give a more detailed analysis of the course details and overall results than in Nikander et al. (2008).

Prior Studies

The details of the 2007 and 2008 courses are shown in Table 1. The table shows how many students started the course each year (# stu), how many of them participated in the final exam (in exam), how many SDA-TRAKLA2 exercises there were in the course (and how many of those were spatial data algorithm exercises) (exer. (SDA)), the average number of submissions per student (subs/stu) and the average number of submissions per student per exercise(subs/ex), as well as the students' average SDA-TRAKLA2 scores (avg. score). The number of students in the course includes only those who actually participated in some learning activities on the course. In both years, there were students who enrolled to the course but did not do anything. In year 2007, there were five of them, but in 2008 only one.

Table 1: Basic course data for the Spatial Data Algorithm course.

year	# stu	in exam	exer. (SDA)	subs/stu	subs/ex	avg. score
2007	16	10	15 (9 SDA)	45,1	3,0	67%
2008	20	16	16 (10 SDA)	51,8	3,2	83%

In spring 2007, the course was given by a professor who had been lecturing it for several years. Our plan was to have her lecture the course also in spring 2008. Unfortunately, she was able to give only half of the lectures in 2008 due to other obligations. Therefore, half of the lectures in spring 2008 were given by other people. One of the authors gave two of the six lectures, and a visiting lecturer gave one. Course contents were the same in both years.

In spring 2007, the spatial exercises were divided into five rounds, each round consisting of 1–3 exercises. In spring 2008, there were four rounds. In order to pass the course, the students needed to gain at least 50% of the SDA-TRAKLA2 points available. A student could submit each exercise as many times as he or she wanted, and they were not penalized for submitting the exercises late.

As can be seen in Table 1 students did fewer submissions in the year 2007 than in the year 2008 (45.2 vs. 51.8 submissions) and gained a lower average score (67% vs. 83% of maximum). However, in 2007 there was one exercise less than in 2008, and thus the number of submissions per student per exercise was similar on both courses (3.0 vs. 3.2 submissions).

The most likely reason for the increase in scores is the improvements to SDA-TRAKLA2 between the two courses. The improvements are: one exercise completely redesigned, one exercise removed, and two new exercises added to the system. In 2007, the students gained the lowest average scores on the exercises that were redesigned (56%) or removed (57%) before the 2008 course. In 2008, students gained much better scores on the redesigned exercise (average score 80%), and extremely good scores on the new exercises (average score 98% in both). The affected exercises were worth a total of 16 points on each course. However, the maximum points in 2007 were higher (88 compared with 72 in 2008). Therefore, the affected exercises were proportionally worth a bit more in 2008. In the exercises which were not modified between the two years, there were no significant changes in the average scores.

Learning Outcomes

Linear regression analysis was used to ascertain whether the students' SDA-TRAKLA2 performance was a good indicator of their exam results. The analysis was made both between overall SDA-TRAKLA2 and exam results as well as between exercise and exam results covering the R-tree data structure, which is a multi-dimensional generalization of the B^+-tree (Guttman, 1984). The R-tree was selected for special study, since it is a complex data structure that is hard to comprehend, and therefore the students' SDA-TRAKLA2 and exam answers were assumed to follow roughly normal distribution. In contrast, with easy exercises most student answers tend to be close to maximum points. In both years, there were two SDA-TRAKLA2 exercises about R-trees. In the first exercise, the students were required to insert data items into an R-tree, and in the second, they needed to search for specific data items in an R-tree. Both years' exams contained one R-tree question, where the students were required to describe the structure of an R-tree and what problems can be solved using it.

As the criteria used in the original grading of the R-tree exercises were very different on the two years, we regraded the students' answers. Two of us regraded the exercises independently, and we compared the two regradings using linearly weighted Cohen's Kappa (Cohen, 1968), which measures the inter-rater reliability. Linearly weighted version was used since standard Cohen's Kappa measures just how many ratings both raters have put in the same category. In the linearly weighted model the distance between the two ratings is also taken into account. In our opinion, this better reflects the fact that if one rater gives a grade 4 and the other a grade 5, the two ratings are still quite close to each other. The agreement between the two gradings was substantial (Kappa 0.64), and the average of the two was used in the linear regression analysis.

Table 2 contains the results of the regression analysis. The table is divided into three categories. First category contains the course information (the year and the number of participants in the exam). The second category contains the results of the linear regression analysis for the whole course, while the third category contains the results for the R-tree exercises. For the linear regression analysis ρ (correlation), adjusted R^2 (strength of relationship) and p (statistical significance) are reported.

Table 2: Learning results for the spatial data algorithms course.

Course info		**Whole exam**			**R-trees**		
Year	N	ρ	adj. R^2	p	ρ	adj. R^2	p
2007	10	0.74	0.50	0.01	0.90	0.78	<0.01
2008	16	0.48	0.18	0.058	0.55	0.25	0.03

As can be seen in Table 2, there was a strong correlation between students' performance in SDA-TRAKLA2 and the course exams. The correlation is significant and, with the exception of the overall 2008 exam results, the relationship is statistically significant ($p<0.05$).

The number of students in this study is low, thus making it hard to draw any strong conclusions. However, similar results have been observed on large basic data structures and algorithms courses (Korhonen, Malmi, Myllyselkä, & Scheinin, 2002). In that study, TRAKLA2 exercises were compared with similar exercises done with pen and paper. The study showed that there was no difference in learning outcomes if the exercises were the same.

Due to the small class sizes in the SDA course, a similar research setup was not possible here. However, if we accept that the results we gained here indicate similar learning outcomes also in this context, our results show that TRAKLA2 can be used in the context of spatial data algorithms the same way that it can be used with basic data structures and algorithms. Of course, given the small class size on the SDA course and lack of a control group, these results can be considered merely indicative.

Student Attitudes

The students' attitudes to the SDA-TRAKLA2 system were primarily measured in two ways: using a course feedback questionnaire and through interviews. Some hints about student attitudes could also be found in the students' SDA-TRAKLA2 data.

The **course feedback questionnaire** was given almost identically in both years. The only difference between the two years was in one series of questions, where the scale used was changed. The questions discussed the different teaching methods used in the course. In 2007, the scale was from 1 to 4, but in 2008 it was changed to conform to the 0 to 5 scale used to grade exams and courses. The change was done in order to make it easier for the students to answer to the question since they had a direct analogy in the exam grades. In the questionnaire, there were several questions regarding SDA-TRAKLA2. One was to give it a grade, while other questions discussed the system's usefulness (on a scale from 1–4 where 1 was "not useful" and 4 "very useful"), and the various aspects of the system from pseudo code to the applet's user interface.

In 2007, SDA-TRAKLA2 was among the lower ranking teaching methods on the course. Most students regarded the system as either a somewhat useful or a quite useful learning method and were of the opinion that it helped their learning. All students regarded SDA-TRAKLA2 exercises as suitably challenging. However, many students expressed the opinion that the system was their least favorite part of the course.

In 2008, on the other hand, SDA-TRAKLA2 was the highest ranking teaching method on the course. Most students regarded the exercises as either quite useful or very useful for learning and were of the opinion that the system helped their learning. All students also thought that the exercises were suitably challenging. This year, no student regarded SDA-TRAKLA2 as their least favorite part of the course, and a few actually named it the part of the course they enjoyed the most.

Thus, there was a significant shift in overall student attitudes between the two years. In 2007 SDA-TRAKLA2 was ranked rather low when compared to other teaching methods and several students complained about having to use the system. In 2008, on the other hand, SDA-TRAKLA2 got the top rank, and some students mentioned that the system was their favorite part of the course. Furthermore, in 2008 students felt that the system was more helpful for their learning than in 2007.

The **interviews** revealed some pros and cons of the system. The interviewees were one Finnish female, one Finnish male, one foreigner female and one foreigner male, aged between 22 and 28 years. The interviews were conducted by the two authors not involved in teaching the course.

The problems were mostly related to usability issues while the positive comments were related to the better learning experience. A more thorough report on the interviews is published elsewhere (Nikander et al., 2008). In the following we sum up the findings.

All of the interviewees criticized the GUI and the feedback received from the exercises. Even though the GUI is quite simple, some students felt that it is too difficult to learn quickly. They especially complained about how the GUI changed between exercises. In addition, the feedback is merely a number of correct steps in the simulation sequence, and the students hoped for more detailed feedback on the mistakes they made. Moreover, a mistake early on results in lost points, thus a more humane grading scheme would have been appreciated.

Most of the comments were positive, however. Especially the model answer was appreciated as well as the animations and visualizations in general. The students' subjective opinion of the system was that it helped them to learn and memorize the topic more easily compared to other teaching methods and learning materials. With the system, the students can actually practice the algorithm, which makes it easier to remember it later. In addition, there is a certain analogy between this and *learning by doing*.

The **students' SDA-TRAKLA2 data** shows that many students continued to solve the exercises even after they had gained points required for passing the course. The students needed to gain at least 50% of the TRAKLA2 score in order to pass the course and did not gain any further benefit from getting more points. However, as can be seen from Table 1, the average student score is much higher than 50%. Especially in the 2008 course, students seemed to solve many more exercises than necessary to pass the course. Similar behavior has also been observed on basic data structures and algorithms courses (Malmi, Karavirta, Korhonen, & Nikander, 2005). It seems that TRAKLA2 inspires students to do extra work also in the SDA course.

Qualitative Results

The small number of students and the lack of a control group caused by this prevents us from drawing definitive conclusions from the quantitative analysis. Furthermore, all the relationships in the analysis were not statistically significant. Therefore other analysis methods need to be used in order to further ascertain whether there actually is any effect of using SDA-TRAKLA2. We decided to use content analysis (Krippendorff, 2004) on the students' exam papers in order to investigate how students' experiences with the exercises are reflected in their exam answers. This helps us ascertain whether there is any effect of using the system on the course or, in other words, do students learn anything by using it.

In the analysis, the answer to each question in a student's exam paper was considered separately, and the analysis was mainly done per question and not per student. All exam questions were considered in the analysis, but most effort was spent on the R-tree questions. R-tree was the only data structure or algorithm that was a part of both years' exam. Furthermore, the students' answers to the R-tree question had already been analyzed quantitatively.

Effect of SDA-TRAKLA2 Exercises

Depending on the exam question, we were able to see none, slight, or even rather extensive evidence of the effects of SDA-TRAKLA2. The effect was primarily observed in either the diagrams a student drew as part of their answer or on which parts of a given data structure or algorithm their answers concentrated.

These two effects we were able to distinguish, since SDA-TRAKLA2 visualizations are quite distinct from the pictures included in other learning material, and some of the exercises concentrated on rather specific parts of a data structure or algorithm. Thus, if the diagrams reflected SDA-TRAKLA2 visualizations or the focus of the answer was the same as in the corresponding SDA-TRAKLA2 exercise, we could be reasonably certain that such effects were due to the influence of the system.

Rather surprisingly, in the R-tree exercises, there is not much evidence that SDA-TRAKLA2 had any direct influence on students' exam answers, especially in the 2008 exam. In the 2007 exam, many students did not remember R-tree at all, or remembered it completely incorrectly, and thus received no points from the exam question. Those who had some recollection of the structure typically drew R-trees where each non-leaf node had either two or three children, similar to the SDA-TRAKLA2 exercise. There was no other possible direct influence from the system observed. Average student score, using the R-tree regrading described earlier in the sub-section on learning outcomes, was 1.4 points out of the maximum of 6. In that year, the students' learning material on R-trees included only the lecture notes and the SDA-TRAKLA2 exercises. In the lecture notes there were no pictures of R-trees, and therefore the students only saw R-tree visualizations in the system.

In 2008, the lecture material on R-trees was redesigned and relevant parts from Worboys and Duckham (2004) and Samet (1989) were included as additional teaching material. The redesigned lecture material also contained R-tree pictures created using SDA-TRAKLA2. In 2008, the students' average score was substantially better than in the previous year: 3.75 out of the maximum of 6. Furthermore, the students' answers showed practically no direct influence from the SDA-TRAKLA2 exercises. The students' diagrams typically were very close to the ones included in Worboys and Duckham (2004), where each non-leaf node had two child nodes and each leaf node could contain two data items.

In other exam questions where there was a corresponding SDA-TRAKLA2 exercise, influence of the system could clearly be seen. In the 2007 exam, one question was to describe algorithms for creating a Delaunay triangulation. The course material contained several algorithms for calculating a Delaunay triangulation, including the random incremental algorithm (Guibas, Knuth, & Sharir, 1992), Fortune's algorithm (Fortune, 1986), and the expanding wave-algorithm (McCullagh & Ross, 1980). Of all the algorithms included in the course, only the expanding wave had a corresponding SDA-TRAKLA2 exercise. Other learning material for this algorithm included the lecture slides and the original research paper (McCullagh & Ross, 1980). Seven of the ten students described the expanding wave in their answers, and five of them used illustrations. Four out of five used illustrations that were very similar to the visualizations used in the system, while one used an illustration similar to those found in McCullagh and Ross (1980). One student actually illustrated the whole algorithm step-by-step using visualizations functionally identical to the ones used in the SDA-TRAKLA2 exercise.

In the 2008 exam, one question was about different algorithms for finding the shortest path in various situations. One of the algorithms discussed in the course was about using a visibility graph to find the shortest path in free space with polygon obstacles. The learning material for this consisted of lecture slides, relevant parts of de Berg, Kreveld, Overmars, and Schwarzkopf (2000) and a SDA-TRAKLA2 exercise concentrating on a single rotation of the rotational sweep algorithm. In the exam, students who described the visibility graph almost exclusively concentrated on describing how the algorithm finds the points visible from a given point. Other parts of the algorithm that construct the visibility graph were glossed over by most students. Those who drew pictures, however, typically included the whole visibility graph even if they did not describe it in the text. Only three students included the rotational sweep algorithm in the pictures they drew.

Thus, it is clear that the students do remember the SDA-TRAKLA2 exercises in the exam and, thus, that the system plays a part in the students' learning. The amount of influence the exercises have can, however, vary a lot. In some cases, such as the expanding wave, it seems that most students remembered the topic mainly through SDA-TRAKLA2. However, as seen in the case of R-tree, other teaching material can also have a large effect, and sometimes completely hide possible effects of the exercises.

Discussion

Spatial data structures are an important branch in the study of data structures and algorithms and are needed in numerous disciplines. Many advanced courses on data structures in computer science, geoinformatics, and several other domains cover topics such as line sweep algorithms, finding the closest pair of points, and dictionaries for multi-dimensional data. Teaching and learning these topics is hard and typically requires heavy use of visualizations and other similar learning materials in order to be effective.

We have demonstrated a novel tool for providing exercises for students to practice the topics and build feasible mental models of the functionality of spatial data structures and algorithms. The list of exercises in the SDA-TRAKLA2 system can be found in the Appendix. Although, we have implemented exercises for the topics most often covered in courses targeted to students in geoinformatics, many of the topics are covered also in computer science courses, such as computational geometry, computer graphics, and advanced algorithms.

Our previous studies have shown that the exercises based on visual algorithm simulation are successful in basic CS courses. The exercises can be personally tailored, which prevents trial–and–error problem solving and plagiarism. The feedback for the student is immediate and, thus, enhances learning as the student can continue to work with the exercise until he or she has mastered the algorithm. Model solutions provide important and timely feedback in order to self-study the topic. By logging the points received from the exercises, the system can also be utilized to assess student performance. In our institution, this motivates students to do more exercises compared with the case where they are not part of the assessment.

All the above benefits can be achieved also in case of spatial data structures. However, spatial data structures are intrinsically more complex in their nature than basic data structures. In this paper, we have studied the feasibility of the spatial data algorithm version of the TRAKLA2 system, the SDA-TRAKLA2. First, we have implemented new visualizations for the system, which include a two-dimensional area visualization regularly used in spatial data algorithm exercises. The visualization can contain different graphical primitives, spatial elements, and conceptual elements such as sweep lines or circumcircles. Second, we have used several methods to evaluate the system in teaching spatial data algorithms including comparison of SDA-TRAKLA2 exercise points and examination points, qualitative analysis of exam answers, interviews, and attitude surveys. We have also compared the results with previous similar studies where the TRAKLA2 system was utilized to teach basic data structures and algorithms. Our results indicate that the system can be used to teach SDA the same way it can be used to teach basic algorithms, and that we gain the same benefits from using the system, which also conforms to the results of our previous studies.

Although there are similar systems, we have compared SDA-TRAKLA2 with other systems and found several advantages. First, many of the other tools use only one view, the area view, in the visualization. SDA-TRAKLA2 uses area and data structure views together. Therefore, changes in one view are reflected in all other relevant views, and the student can gain a more complete understanding on how the algorithms work.

Second, typically other tools cover only a limited number of spatial data structures and algorithms and, thus, are suitable only for a very small fraction of an SDA course. We have covered much more of an SDA course, although there is still a long way to cover the whole course. All exercises include instructions on how to solve the exercise and a description of the algorithm in pseudo code. Many exercises also include links to additional learning material, which seems to be a lacking feature in many other tools.

Third, the interaction in SDA-TRAKLA2 is more robust and engaging than in many other tools, which merely provide the user with the ability to insert desired input for the algorithm and view the algorithm animation as it executes. In SDA-TRAKLA2, the user needs to think how the algorithm works and simulate its execution through the GUI. At the end of the simulation sequence, the user gets feedback of his or her performance. In addition, students can further explore the current problem instance by watching the model solution, which is similar to the algorithm animations found in other systems.

The results reported in this paper also back up our positive viewpoint on SDA-TRAKLA2. As can be seen in Table 2, there is a correlation between a student's SDA-TRAKLA2 results and their final examination results. Students performing well in SDA-TRAKLA2 exercises typically do well also in the examination. This result is similar to the results obtained in a prior study in which the use of the system was compared to exercises done with pen and paper (Korhonen et al., 2002). In our opinion this shows that the system is used similarly in learning both basic and spatial algorithms. In addition, the qualitative analysis of the students' exam answers shows that the students do remember the SDA-TRAKLA2 visualizations when they answer exam question, and typically, if there is a TRAKLA2 exercise, the exam answers reflect this. This shows that the students also remember what they have learned while using the system. The influence of the system on the answers seems, however, to be somewhat topic-specific. And, as seen in the students' answers to the R-tree question, other learning material can also play a very important role. Furthermore, the students' attitude towards the system is very positive. Actually, in the interviews, the students expressed a desire for more SDA-TRAKLA2 exercises. All this has assured us to continue to develop and utilize the system in the future.

From the teacher's point of view, integrating the system into the course was straightforward and requires only little effort from the course staff. In addition, it provides many benefits not studied nor reported directly, but which are obvious. For example, plagiarism seems to be a problem in many institutions. SDA-TRAKLA2 provides exercises that are individually tailored for each student, thus there is no point in copying answers from other students. Actually, the system provides the model answers itself, thus it can be used for learning purposes and not for bypassing the learning opportunity. The system also gives a meaningful way to increase the number of exercises in courses where there are limited resources to check the exercises and give feedback for the students. Thus, some of the resources could be reallocated, for example, to activities that cannot be automatically assessed.

Our way of using the SDA-TRAKLA2, however, is not the only possible one. Nowadays, most of the students in our institution have a home computer and internet connection, which makes the use of the system independent of time and place. Thus, it is natural that we have utilized the system most often in open labs. The system could also be utilized in closed labs, which of course would require computer labs to be available. This would make sense especially if the system is not intended to be used for assessment, but merely for practicing the algorithms. In our university, however, the tradition has been to give homework.

The system has been used since 2007 and is still in use and maintained. The system is open source and, thus, available also for other institutions. Interested parties can familiarize themselves with the system at the TRAKLA2 homepage < http://www.cse.hut.fi/en/research/SVG/TRAKLA2/ >. As the new spatial extension has been incorporated into the existing TRAKLA2 learning environment, one can also use all other exercises implemented in the system. These include data structures that are used as auxiliary structures in several spatial data structures and algorithms. A full list of implemented exercises can be found at the TRAKLA2 homepage.

Issues of Validity and Reliability

Due to the small number of students in the spatial data algorithm course, we were unable to create a proper experimental research setting. With less than 20 students on the course, it makes no sense to divide them into experimental and control groups.

The content of the course was kept the same after the introduction of SDA-TRAKLA2. However, the system was further developed between 2007 and 2008, and thus the students did not receive the same set of exercises on both years. Furthermore, in spring 2008, the course was lectured by different people than the previous year. Several independent variables changed and, therefore, the two courses are not entirely comparable. Thus, the quantitative data gathered in the course is not in itself sufficient for making any sort of definitive analysis. However, the data gathered from this study confirms the earlier findings; thus we have a reason to expect the conclusions to hold also in case we could repeat the study with larger cohort of students in the future. At least, we have no reason to believe the way SDA-TRAKLA2 delivers the exercises would be any worse than other methods, e.g., delivery by paper and pencil. However, compared to other teaching methods SDA-TRAKLA2 has many additional benefits.

Conclusions

In this work, we have described SDA-TRAKLA2, a novel learning environment for spatial data algorithms. The environment has been implemented on top of the successful TRAKLA2 system and has been used in teaching a spatial data algorithms course at Helsinki University of Technology since spring 2007. The course has 15 to 25 students a year, and it is part of the obligatory studies for the students specializing in geoinformatics. The course also has studio sessions, an exam, and a programming project. The new SDA-TRAKLA2 exercises were included without modifying other parts of the course.

We have implemented a set of visual algorithm simulation exercises to cover a number of SDA topics. Typically the exercises use several different algorithm visualizations at the same time to show both how spatial data items are related to one another and how they are arranged in data structures. To accomplish this, we have implemented novel visualizations to the SDA-TRAKLA2 system.

The students' learning results gained with the system are similar to those witnessed previously when using TRAKLA2 in the basic data structures and algorithms course (Korhonen et al., 2002). Furthermore, by qualitative analysis of students' examination papers, we have ascertained that using SDA-TRAKLA2 has an effect on the students' exam answers, and thus they seem to benefit of using the system in their studies. Therefore, using the system appears to be a viable alternative to similar classroom exercises, and the idea of visual algorithm simulation exercises can be extended to cover spatial algorithms. However, the small class size of the SDA course makes it hard to draw definitive conclusions. Furthermore, the effect of other learning material can be considerable and sometimes appears to mask the possible effects of using SDA-TRAKLA2.

The students' overall attitude towards the system is very positive. The students feel that the system helps them in learning, and they have asked for more aspects of the course to be covered by SDA-TRAKLA2. In 2007, however, the students' attitudes towards the system were rather poor. This was probably due to the large number of software bugs and the complicated UI the system had in its initial release. After bug fixes and some modifications to the UI, the student attitudes changed dramatically.

Overall, our opinion is that SDA-TRAKLA2 is a viable learning tool in teaching spatial data algorithms. This also proves that it is possible to use visual algorithm simulation exercises in teaching advanced data structures and algorithms. Previously, the system had covered only topics handled in basics algorithms courses.

Future Work

Several ways to improve the system remain. First, new exercises would be welcome to cover more spatial data algorithms. Second, several of the currently implemented exercises could be improved, however, not necessarily as radically as the complete redesign that was done to the line sweep exercise. Third, there are also some algorithms for which we have not been able to design a good exercise. For example, one exercise removed from the 2008 course was about a divide-and-conquer algorithm for finding the closest pair of points (Preparata & Shamos, 1985). The implementation was so clumsy that learning to use the exercise took a long time for the students. Also, the user interface contained so much functionality for the student that we came to the conclusion that even if a student was able to solve the exercise, their focus had been in "winning" the exercise and not in understanding the algorithm. Furthermore, in order to keep the number of steps in the exercise reasonable, the input for the exercise needed to be very small. Typically, with such input, it would have been better to just use the brute-force algorithm for finding the closest pair.

Finally, in the field of spatial data algorithms, there is a whole group of algorithms, which we deemed to be unsuitable for SDA-TRAKLA2 – at least with its current implementation. Spatial data can be divided into two models: object data model where spatial items are stored as distinct objects, and field data model where spatial data is stored as continuous fields. In the object data model, the execution of an algorithm is typically ruled by the input. With different inputs, different lines of the algorithm code are executed in different order. Therefore, when doing tracing exercises, at each step of the algorithm simulation, the student needs to think what his next action would be with the given input. With many field algorithms, however, the algorithm execution is not governed by the input. The same lines of code are typically executed in the same order, no matter what the input is. Only the numerical values carried from one subroutine to the next are different. Therefore such algorithms are not, in our opinion, suitable for visual algorithm simulation exercises. Regardless of the input, the algorithm simulation would always be more or less the same, thus making the exercises both boring and repetitive.

Overall, our opinion is that the spatial exercises in SDA-TRAKLA2 are already a useful learning resource for teaching spatial data algorithms. The system is still being maintained, and we hope to expand it further in the future. The system is available for free at our website < http://www.cse.hut.fi/en/research/SVG/TRAKLA2/ >.

References

Akingbade, A., Finley, T., Jackson, D., Patel, P., & Rodger, S. H. (2003). JAWAA: Easy web-based animation from CS0 to advanced CS courses. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education, SIGCSE'03*, pp. 162–166, Reno, Nevada, USA. ACM Press.

Ala-Mutka, K., Uimonen, T., & Järvinen, H.-M. (2004). Supporting students in C++ programming courses with automatic program style assessment. *Journal of Information Technology Education*, *3*, 245–262. Retrieved from http://www.jite.org/documents/Vol3/v3p245-262-135.pdf

Andrienko, G., Andrienko, N., Fischer, R., Mues, V., & Schuck, A. (2006). Reactions to geovisualization: An experience from a European project. *International Journal of Geographical Information Science*, *20*(10), 1149 – 1171.

Benford, S., Burke, E., Foxley, E., Gutteridge, N., & Zin, A. M. (1993). Ceilidh: A course administration and marking system. In *Proceedings of the 1st International Conference of Computer Based Learning*, Vienna, Austria.

Bentley, J., & Ottmann, T. (1979). Algorithms for reporting and counting geometric intersections. *Computers, IEEE Transactions on*, *C-28*(9), 643–647.

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

Cohen, J. (1968). Weighted kappa: Nominal scale agreement with provision for scale and disagreement or partial credit. *Psychological Bulletin*, *70*, 213–220.

Crouch, J. R., Shen, Y., Austin, J. A., & Dinniman, M. S. (2008). An educational interactive numerical model of the Chesapeake Bay. *Computers & Geosciences*, *34*(3), 247–258.

de Berg, M., Kreveld, M. V., Overmars, M., & Schwarzkopf, O. (2000). *Computational geometry: Algorithms and applications*. Springer.

Degrande, G., & Geraedts, K. (2008). An electronic learning environment for the study of seismic wave propagation. *Computers & Geosciences, 24*(6), 569–591.

Denny, P., Hamer, J., Luxton-Reilly, A., & Purchase, H. (2008). Peerwise: Students sharing their multiple choice questions. In *ICER '08: Proceeding of the fourth international workshop on Computing education research*, pp. 51–58, New York, NY, USA. ACM.

Fisher, J. (2004). Visualizing the connection among convex hull, Voronoi diagram and Delaunay triangulation. In *37th Midwest Instruction and Computing Symposium*.

Fortune, S. (1986). A sweepline algorithm for Voronoi diagrams. In *SCG '86: Proceedings of the Second Annual Symposium on Computational Geometry*, pp. 313–322, New York, NY, USA. ACM.

Guibas, L. J., Knuth, D. E., & Sharir, M. (1992). Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, *7*, 381–413.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pp. 47–57, New York, NY, USA. ACM Press.

Hausner, A., & Dobkin, D. P. (1998). GAWAIN: Visualizing geometric algorithms with web-based animation. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pp. 411–412, New York, NY, USA. ACM.

Hipke, C. A., & Schuierer, S. (1998). *Vega – A user-centered approach to the distributed visualization of geometric algorithms*. Technical report, University of Freiburg.

Hundhausen, C., & Brown, J. L. (2005). What you see is what you code: A radically dynamic algorithm visualization development model for novice learners. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 163–170.

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

Janhunen, T., Jussia, T., Järvisalo, M., & Oikarinen, E. (2004). Teaching Smullyan's analytic tableaux in a scalable learning environment. In A. Korhonen & L. Malmi (Eds.), editors, *Kolin Kolistelut/Koli Calling. Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education*, pp. 85 – 94. Helsinki University of Technology.

Karavirta, V., Korhonen, A., Malmi, L., & Stålnacke, K. (2004). MatrixPro – A tool for on-the-fly demonstration of data structures and algorithms. In *Proceedings of the Third Program Visualization Workshop*, pp. 26–33, Department of Computer Science, University of Warwick, UK.

Karssenberg, D., Burrough, P. A., Sluiter, R., & de Jong, K. (2001). The pcraster software and course materials for teaching numerical modelling in the environmental sciences. *Transactions in GIS*, *5*(2), 99–110.

Koifman, I., Shimshoni, I., & Tal, A. (2008). Mavis: A multi-level algorithm visualization system within a collaborative distance learning environment. *Journal of Visual Languages & Computing*, *19*(2), 182–202.

Korhonen, A. (2003). *Visual algorithm simulation*. Doctoral dissertation (tech rep. no. tko-a40/03), Helsinki University of Technology.

Korhonen, A., & Malmi, L. (2004). Taxonomy of visual algorithm simulation exercises. In *Proceedings of the Third Program Visualization Workshop*, pp. 118–125, The University of Warwick, UK.

Korhonen, A., Malmi, L., Myllyselkä, P., & Scheinin, P. (2002). Does it make a difference if students exercise on the web or in the classroom? In *Proceedings of The 7th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'02*, pp. 121–124, Aarhus, Denmark. ACM Press, New York.

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology*. Thousand Oaks, CA.

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

Laurini, R., & Thompson, D. (1992). *Fundamentals of spatial information systems*. Academic Press.

Malmi, L., Karavirta, V., Korhonen, A., & Nikander, J. (2005). Experiences on automatically assessed algorithm simulation exercises with different resubmission policies. *Journal of Educational Resources in Computing*, *5*(3).

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

Malmi, L., Korhonen, A., & Saikkonen, R. (2002). Experiences in automatic assessment on mass courses and issues for designing virtual courses. In *Proceedings of The 7th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'02*, pp. 55–59, Aarhus, Denmark. ACM Press, New York.

McCullagh, M. J., & Ross, C. G. (1980). Delaunay triangulation of a random data set for isarithmic mapping. *The Cartographic Journal*, *17*(2), 93-99.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., & Ángel Velázquez-Iturbide, J. (2003). Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin*, *35*(2), 131–152.

Nikander, J., & Helminen, J. (2007). Algorithm visualization in teaching spatial data algorithms. In *11th International Conference Information Visualization IV2007*, pp. 505–510. IEEE Computer Society.

Nikander, J., Helminen, J., & Korhonen, A. (2008). Experiences on using trakla2 to teach spatial data algorithms. In G. Rössling, G. (Ed.), *Proceedings of the Fifth Program Visualization Workshop (PVW 2008).*

Nikander, J., Korhonen, A., Valanto, E., & Virrantaus, K. (2007). Visualization of spatial data structures on different levels of abstraction. In G. Rössling (Ed.), *Proceedings of the Fourth Program Visualization Workshop (PVW 2006)*, volume 178, pp. 60–66, Florence, Italy.

Patton, M. (2002). *Qualitative research and evaluation methods*. Sage Publications.

Preparata, F., & Shamos, M. I. (1985). *Computational geometry: An introduction*. Springer-Verlag.

Price, B. A., Baecker, R. M., & Small, I. S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages and Computing*, *4*(3), 211–266.

Purves, R., Medyckyj-Scott, D., & Mackaness, W. (2005). The e-mapscholar project – An example of interoperability in giscience education. *Computers & Geosciences*, *31*(2), 189–198.

Rößling, G. (2002). *ANIMAL-FARM: An extensible framework for algorithm visualization*. Phd thesis, University of Siegen, Germany. Available at http://www.ub.uni-siegen.de/epub/diss/roessling.htm

Saikkonen, R., Malmi, L., & Korhonen, A. (2001). Fully automatic assessment of programming exercises. In *Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'01*, pp. 133–136, Canterbury, UK. ACM Press, New York.

Samet, H. (1989). *The design and analysis of spatial data structures*. Addison-Wesley.

Shneerson, M., & Tal, A. (2000). Interactive collaborative visualization environment for geometric computing. *Journal of Visual Languages & Computing*, *11*(6), 615–637.

Slocum, T. A., McMaster, R. B., Kessler, F. C., & Howard, H. H. (2004). *Thematic cartography and geographic visualization*. Prentice-Hall.

Stasko, J. T., Domingue, J. B., Brown, M. H., & Price, B. A. (1998). *Software visualization: Programming as a multimedia experience*. Cambridge, MA: MIT Press.

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

Worboys, M., and Duckham, M. (2004). *GIS: A computing perspective*. Taylor & Francis.

Appendix

Table 3: Spatial exercises in the TRAKLA2 system

Name	**Description**	**Years used**
Point in polygon	The learner is to find whether a point is inside a polygon by counting the number of intersections a half-line drawn from the point has with the polygon.	2007, 2008
Closest pair of points	The learner is to find the closest pair of points in a set of points by using a divide-and-conquer approach based on the mergesort algorithm.	2007
Doulas-Peucker line simplification	The learner is to simplify a polyline using the Douglas-Peucker line simplification algorithm.	2007,2008
Line sweep	The learner is to find line segment intersections in a set of line segments by using the line sweep algorithm. The exercise was completely redesigned between the two courses.	2007, 2008
Voronoi construction	The learner is to construct a valid Voronoi diagram from a set of points. There is no need to follow a specific algorithm, only the end result is assessed.	2008
Adding a point to TIN	The learner is to add three new points to a Delaunay triangulation and to modify the triangulation so that it still stays valid. There is no need to follow a specific algorithm, only the end result is assessed.	2008
Expanding wave-method	The learner is to construct a Delaunay triangulation using the expanding wave algorithm.	2007, 2008
Visibility with rotational sweep	The learner is to find polygon end points visible from a given point by using the rotational sweep algorithm.	2007,2008
R-tree insert	The learner is to insert a number of polygons into an R-tree.	2007,2008
Point-region quadtree insert	The learner is to insert a number of points into a point-region quadtree.	2007,2008
Point in polygon with R-tree	Point in polygon where the edges of the polygon are in an R-tree and the learner must search the R-tree for edges that may cross the half-line.	2007,2008

Biographies

Jussi Nikander received the degree of Master of Science (Technology) in 2005 and the degree of Licentiate of Science (Technology) in 2009 from the Helsinki University of Technology (HUT), Finland. Between 2001 and 2009 he worked at the Department of Computer Science and Engineering at the university, and in 2010 he moved to the Department of Surveying. He is currently a researcher and doctoral student. His research has concentrated on the use of visualization in the research and teaching of spatial data algorithms.

Juha Helminen received the degree of Master of Science (Technology) in 2009 from the Helsinki University of Technology (HUT), Finland. Since 2006, he has been associated with the Department of Computer Science and Engineering at the Aalto University School of Science and Technology (formerly HUT), Finland, currently as a researcher and a doctoral student. His research has dealt with software visualization and environments for teaching and learning of programming.

Ari Korhonen received the degree of Doctor of Science (Engineering) in 2003 from the Helsinki University of Technology (HUT), Finland. Since 1996, he has been associated with the Department of Computer Science and Engineering at the Aalto University (formerly HUT), Finland, currently as an adjunct professor. His main area of research is software visualization, and he has focused on automatic assessment software in virtual learning environments.