Int. J. Engng Ed. Vol. 25, No. 1, pp. 93–101, 2009
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2009 TEMPUS Publications.

Educational Support for Computational Geometry Course—The Delaunay Triangulation Tester*

DENIS ŠPELIC
Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia

FRANC NOVAK
Jožef Stefan Institute, Ljubljana. Slovenia

BORUT ŽALIK
Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia.
E-mail: zalik@uni-mb.si

The paper presents a tool to verify the correctness of the generated 2D Delaunay triangulation that has been developed as an educational support for a computational geometry course. This tool allows students to discover possible flaws in implemented triangulations such as unused points, missing edges, non-Delaunay triangles and degenerated triangles. The associated benchmark data sets provide common checkpoints for the implemented solutions. The tool and the benchmark data set also assist teachers to evaluate students' work fairly.

Keywords: computational geometry course; Delaunay triangulation; benchmarks

INTRODUCTION

COMPUTATIONAL GEOMETRY has become one of the fastest growing areas of algorithmic research. It involves the design and analysis of algorithms and data structures for application solutions, such as geographical information systems (GIS), medical imaging, computer vision, computer graphics, and computer-aided design and manufacturing. It is well-suited to undergraduate and postgraduate computer science education, both within the classroom curriculum and for independent research.

Within the framework of individual research projects, students study and implement different algorithms for a variety of computational geometry problems. Once the work has been carried-out, the solutions are discussed and compared with regard to their efficiency and robustness. The implemented algorithms also need to be checked for correctness, which may, in some cases, require considerable effort. To date, numerous demos have been developed and offered free on the web in order to help students gain a deeper understanding and experimental skills before starting their own research projects. However, evaluation of the developed solutions is often done in an ad hoc fashion.

Relying on benchmarks and supporting tools for checking the correctness of an implemented algorithm is a step towards a more uniform approach. The goal of a benchmark is to check the validity and produce a score for an algorithm's performance, so that different algorithms can be compared in terms of speed, effectiveness, and quality results. While no single test (i.e., benchmark) can fully characterize a targeted system's performance, the results from a set of representative benchmarks can provide valuable insight into the actually expected performance. Typically, a benchmark set contains a collection of items described within a common format, representing a range of problem tasks within any given problem domain.

The computational geometry society, as yet has failed to set-up such benchmarks despite the fact that the Strategic Directions in Computational Geometry Working Group Report [1] (1996) clearly identifies the need for a collection of benchmark data drawn from a wide range of applications, analogous to those SPEC benchmarks used in computer architecture. The report concludes: "no such collections of benchmark data are currently available, and creating such collections would be a valuable service to the community." The report also addresses the issue of teaching computational geometry: except for a few cases, computational geometry does not provided the simple, flexible tools that would enable the kind of widespread use that other disciplines have available.

Today, more than a decade later, the situation has not drastically improved. In the absence of

* Accepted 30 October 2008.

commonly accepted examples or benchmarks for evaluating the results of students' individual research projects, we have taken a step towards setting-up a set of benchmarks for a specific problem domain, supported by a tool for checking the correctness of the developed solutions [2]. This approach has been successfully introduced into the learning process and has resulted in positive initial results, encouraging us to offer the practice to a wider community.

DELAUNAY TRIANGULATION—ONE OF THE CLASSIC ISSUES OF THE COMPUTATIONAL GEOMETRY COURSE

The computational geometry course, which is part of the Computer Science BSc Curriculum (fourth semester) at the Faculty of Electrical Engineering and Computer Science, starts with wide-ranging introductory lessons that expose students to the main themes of modern computational geometry. This course covers those fundamental algorithms for solving geometric problems and includes theoretical background, analyses and discussions regarding the applications of geometric algorithms, and practical exercises. This paper describes our teaching experience when introducing Delaunay triangulation as one of the basic issues, and as a background for different students' individual research projects.

Delaunay triangulation is one of the most widely studied problems in computational geometry. Efficient solutions attract the interests of both academia and commercial tool designers. The triangulation of a set of points P in a plane is a process of forming triangles by connecting neighboring points, with the restriction that each triangle's side is entirely shared by two adjacent triangles. A Delaunay triangulation is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P) [3]. (A circumcircle is defined as the unique circle that passes through each of a triangle's vertices. The reader can find interactive construction at [4].) The Delaunay criterion optimizes the minimal inner angles of triangles. An example of Delaunay and non-Delaunay triangulation is shown in Fig. 1. The vertex C in Fig. 1(b) lies within the circumcircle of triangle ABD, which violates the empty circle property.

Delaunay triangulation solutions represent a key issue within the different areas of computational geometry [5, 6]. Several algorithms for constructing Delaunay triangulation have been proposed over recent years. These algorithms differ in their time-scales regarding response, memory consumption, implementation demands, numerical stability, and sensibility to different distributions of input points. Delaunay triangulation finds application in different fields such as computer graphics, communication, GIS, signal processing, multimedia, micromechanics, etc. A list of references for the proposed Delaunay triangulation algorithms and associated applications [7–19] is given purely for illustration, and is by no means exhaustive. Certain algorithms may perform better or worse, depending on the types of data sets, which is of crucial importance for the targeted applications. Individual attempts at testing the efficiency of triangulation algorithms on various sets of points have been reported (e.g., Su and Drysdale [20]), yet they have not resulted in a general verification platform.

Knowledge and understanding of fundamental algorithms for constructing Delaunay triangulation, together with experience regarding their implementation, is a prerequisite for advanced solutions in practice and one of the initial goals of the computational geometry course. The implementation of a stable and fast Delaunay triangulation has always been a challenge, especially for students.

The following describes the tool and associated materials (test data sets) that we have developed for educational support.

VERIFICATION OF TRIANGULATIONS

As an educational support for a computational geometry course, we have prepared a set of benchmark data (i.e., collections of artificial and engineering data sets) that provide a means of

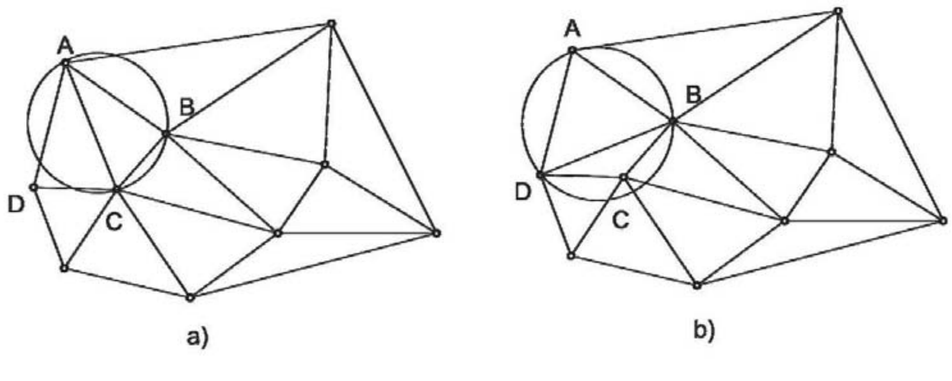


Fig. 1. Delaunay triangulation (a) and non-Delaunay triangulation (b).

comparing different implementations of algorithms for constructing 2D Delaunay triangulation. Further details are provided in the Appendix.

In addition, the Delaunay Triangulation Tester (DTT)—a tool for verifying the generated triangulation correctness of a given benchmark data set, has been developed [2]. This tool has proved to be an important aid in the development of implemented algorithms. The included diagnostic features considerably shorten the debugging time in the case of detected anomalies.

Verification procedure

The verification procedure for Delaunay triangulation is as follows.

1. A benchmark file is selected from the benchmark datasets.
2. Delaunay triangulation is performed by the algorithm to be tested.
3. The resulting triangulation is checked by the Delaunay Triangulation Tester tool. The inputs to the DTT tool are the original benchmark and the resulting triangulation file. The DTT tool checks the correctness of the resulting triangulation and reports any possible errors. Error messages are described in the Appendix of [2].

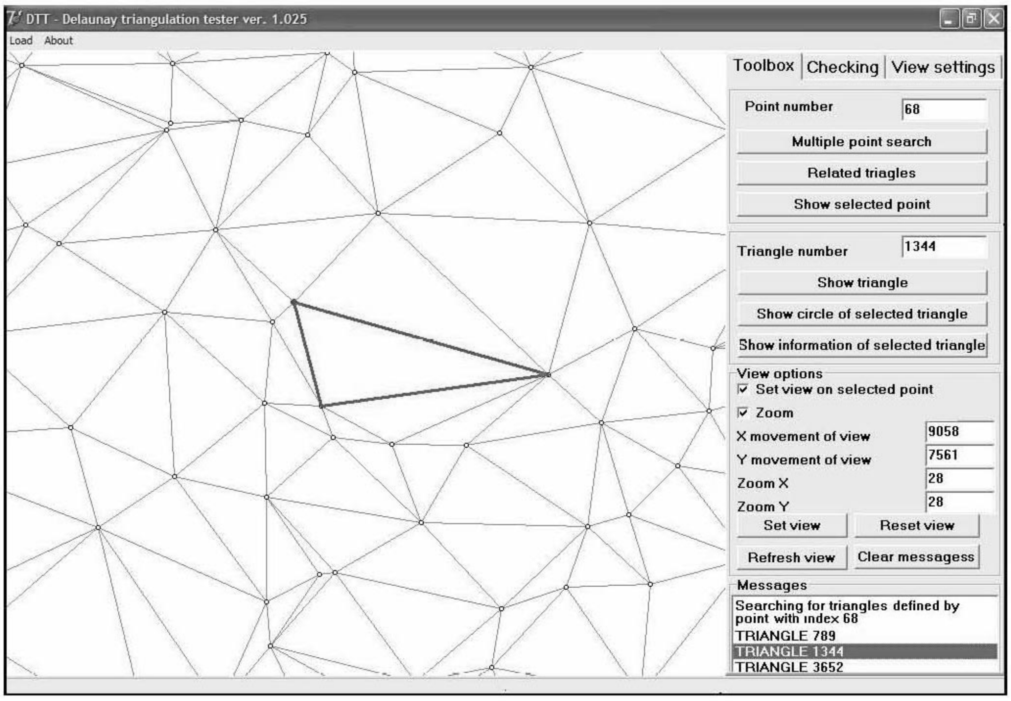
DTT is available free by downloading at http://gemma.uni-mb.si/dtt. The compressed package includes the DTT tool, and the user manual.

Delaunay triangulation tester

The graphical user interface of the DTT tool consists of a menu at the top, working window (main screen space), and the toolbar on the right. The menu contains the option of loading input data (points and the triangulation). The input file formats are simple and are described on our web page http://gemma.uni-mb.si/dtt/specifications.html.

When the data are loaded, they are visualized within the working window. In this window, the user can select points and apply zooming and panning functions (Fig. 2 shows the screen of the DTT after zooming). A point is selected by clicking on it using the middle mouse button. Zooming is performed by drawing a rectangle. To draw a rectangle, the user should click the first point using the left mouse button, drag the mouse, and then release the button to end drawing the rectangle. The DTT will zoom onto the area inside the drawn rectangle. The zoom-in and zoom-out option are also achieved by rotating the mouse wheel. If the wheel is rotated downwards, the zoom is increased by a factor of 2 and when rotated upwards, it is decreased by the same factor. Pan function is carried-out using the right mouse click and dragging the mouse in the desired direction until the button is released.

The toolbar consists of three tabs: toolbox, checking, and view settings. The toolbox tab offers utilities for managing the points and triangles, setting viewing parameters, and displaying information about points, triangles, and errors. This section integrates those functions needed for visually debugging a triangulation. When the user selects a point, the point index is displayed in the text box at the top. The point index can also be inputted manually via the keyboard. When the point index is set, you can verify whether there is



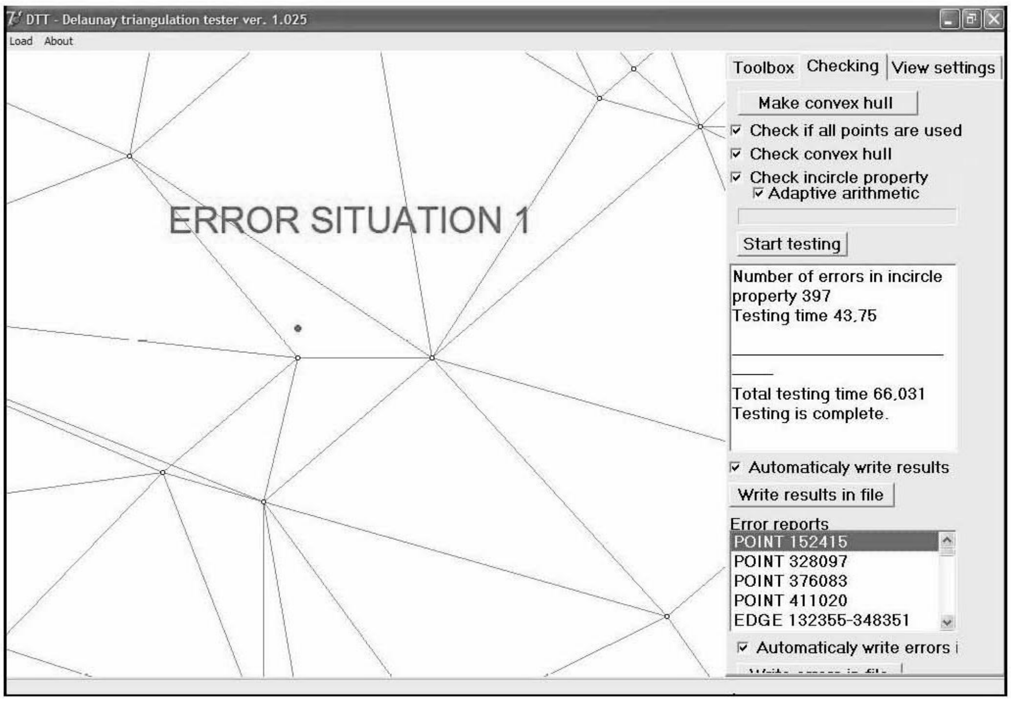
Point number 68

Triangle number 1344

View options
[✓] Set view on selected point
[✓] Zoom
X movement of view 9058
Y movement of view 7561
Zoom X 28
Zoom Y 28

Messages
Searching for triangles defined by point with index 68
TRIANGLE 789
TRIANGLE 1344
TRIANGLE 3652

Fig. 2. An example of zoomed triangulation.



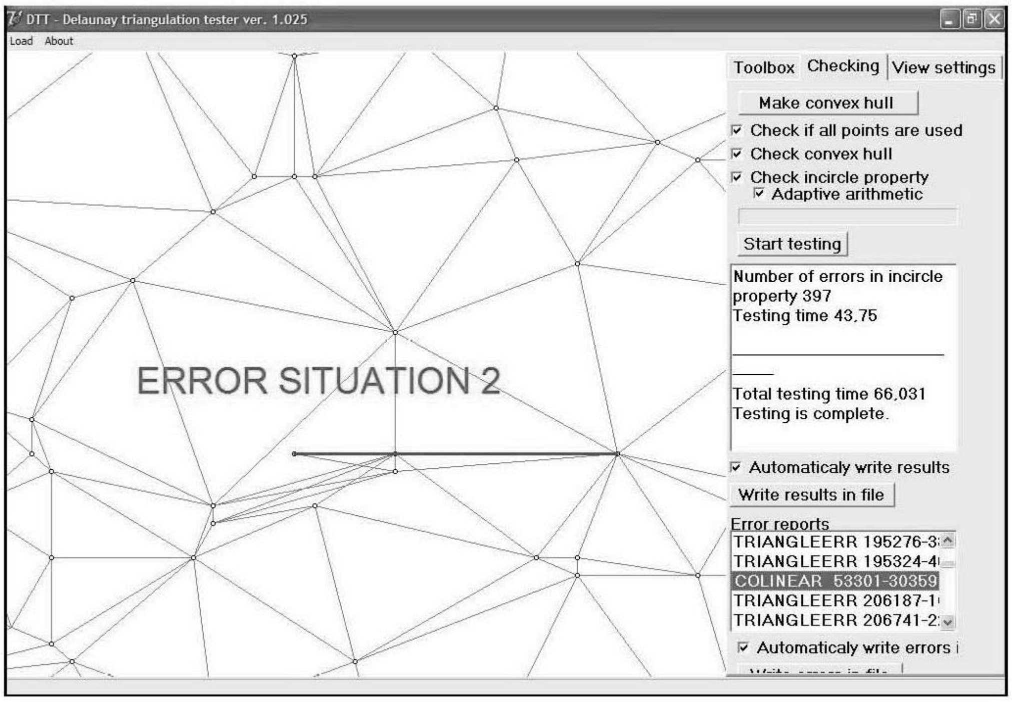
```
Number of errors in incircle
property 397
Testing time 43.75

Total testing time 66.031
Testing is complete.
```

```
POINT 152415
POINT 328097
POINT 376083
POINT 411020
EDGE 132355-348351
```

Fig. 3. Error situation—unused point.

any other point with the same coordinates in the dataset by clicking the button *Multiple point search*. By clicking the button *Related triangles*, all triangle indices in the list box, which share a selected point, are listed. By clicking the triangle index, the selected triangle is highlighted on the screen in a different color. The button *Show selected point* selects and displays the point, the index of which is given in the text box. Similarly, the user can write a triangle index in the triangle text box. By clicking the button *Show triangle* the triangle with the given index is displayed. The circumcircle of the selected triangle is displayed by clicking the button *Show circle*. In this way the user can visually check whether the empty circle property is valid.

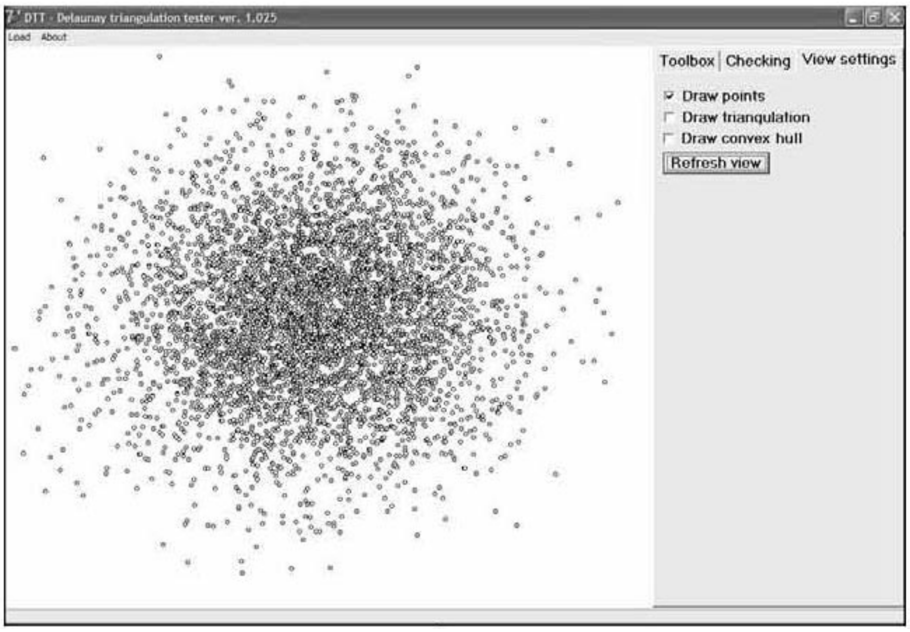


```
Number of errors in incircle
property 397
Testing time 43.75

Total testing time 66.031
Testing is complete.
```

```
TRIANGLEERR 195276-3
TRIANGLEERR 195324-4
COLINEAR 53301-30359
TRIANGLEERR 206187-1
TRIANGLEERR 206741-2
```

Fig. 4. Error situation—incorrect triangles, the one with all three vertices on the same line is marked in green.

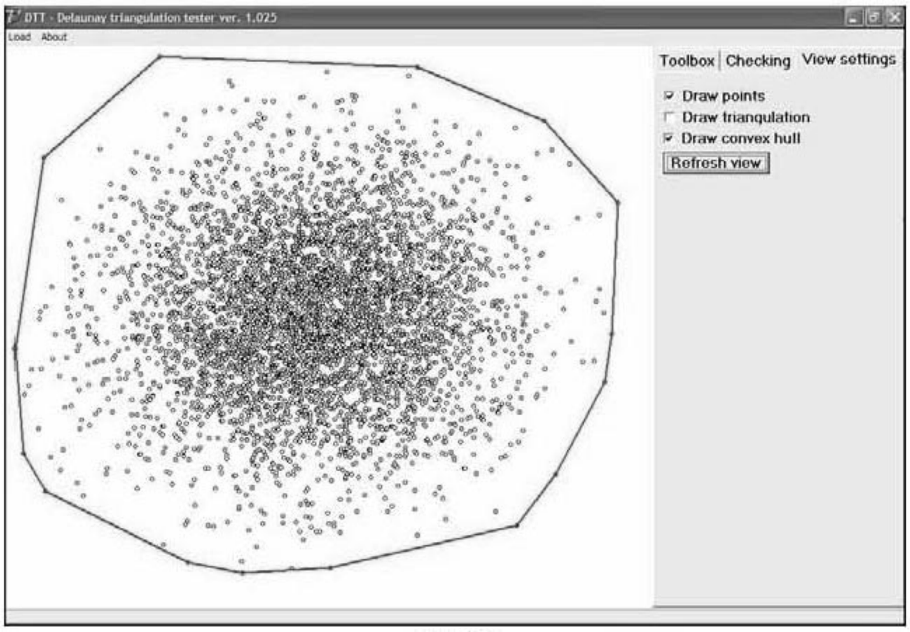


☑ Draw points
☐ Draw triangulation
☐ Draw convex hull

Fig. 5. View only points.

Checking the tab helps the user to easily identify any possible errors in the constructed triangulations. Typical error situations are:

- **Unused point** In this case, at least one non-doubled input point is unused in the final triangulation. The DTT tool outputs the error message in which the unused points are reported.
- **Missing edge** This error occurs if the convex hull property of the Delaunay triangulation is violated. (Informally, a convex hull is a rubber band wrapped around the "outer" points of the data set.) If, for example, edge *ij* is missing, the corresponding error message is reported.
- **Non-Delaunay triangle** The empty circle property is violated if a non-Delaunay triangle *t* exists in the triangulation. The error message *Error: non-Delaunay triangle t* is generated in this case.
- **Degenerated triangles** Two possible cases of degenerated triangles are detected:
  - duplicate vertices: vertices *i and j* coincide (i.e. are located in the same point). The error message *Error: triangle t is degenerated because of duplicated vertices i, j.* is reported.
  - collinear vertices: if triangle *t* is determined by three collinear vertices *i, j, k*, the error message *Error: triangle t is degenerated because of collinear vertices i, j, k* is reported.



☑ Draw points
☐ Draw triangulation
☑ Draw convex hull

Fig. 6. View points and convex hull.

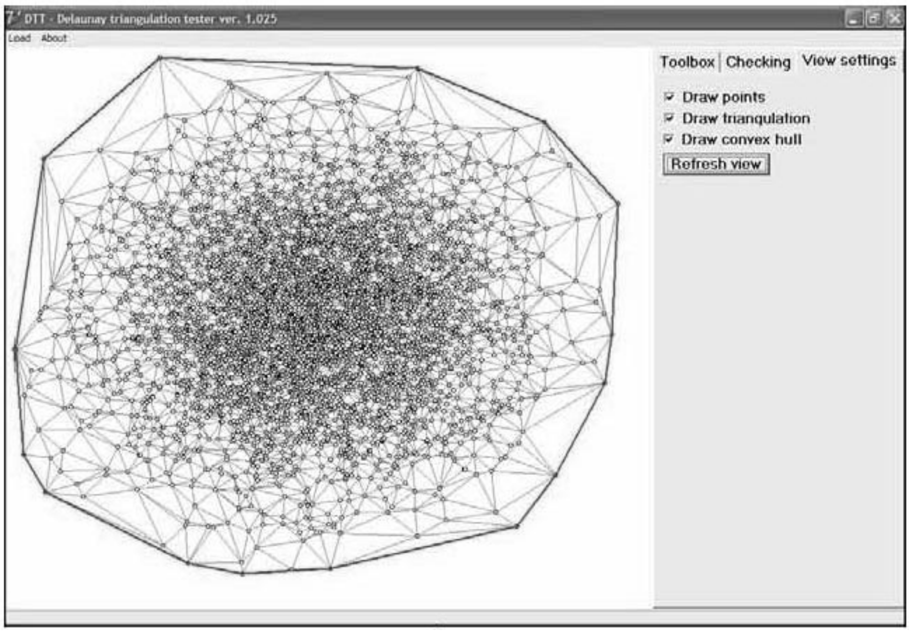


Fig. 7. View points, convex hull and triangulation.

An example of an error situation is shown in Fig. 3. In this example it can be seen that the point with index 152415 is unused in triangulation. Triangulation is incomplete.

Figure 4 depicts another example of an error situation. An incorrect triangle, having all three vertices on the same line, is shown in green. Within its neighborhood there are also other incorrect triangles. Each of them is marked in a different color from its surroundings when selected from the list of error reports.

The View settings tab allows the user to select from the information displayed on the screen. The user can switch between displaying the input points, triangulation, and convex hull.

EXPERIENCES FROM TEACHING COURSES AND MENTORING INDIVIDUAL STUDENT'S RESEARCH PROJECTS

As stated previously, the DTT tool is used during the Computational Geometry course, which is a part of the Computer Science BSc Curriculum (fourth semester) at the Faculty of Electrical Engineering and Computer Science. The number of students enrolled is from 20 to 40, depending on the year. The students have strong skills in programming (C/C++) but their experience in programming geometric problems is inadequate. They are unaware of those pitfalls relating to finite arithmetic (rounding errors, propagation of rounding errors), and are inexperienced regarding the debugging of programs dealing with large geometric data sets.

Various triangulation algorithms are discussed after the theoretical background of the Delaunay triangulation has been explained. They are analyzed from the theoretical (time and space complexity) and practical points of view (actual run-time, clarity of implementation, depending on the distribution of input points, stability, robustness, and the correctness of the obtained results). The DTT tool is used to show the impact of finite arithmetic on the correctness and robustness of various implementations of Delaunay triangulation. Debugging features using the DTT tool are also demonstrated.

During the lab exercise work, students are required to select one of the algorithms described in the introductory lessons, and program it. They can use the DTT tool for debugging. Once the implementation is completed with most of the bugs removed, the program is tested on the benchmarking data sets. The remaining bugs are corrected and then the performance is analyzed, and implementation issues are discussed. Possible improvements are suggested. After some iterations, the

Table 1. Evaluation results

Question	**5**	**4**	**3**	**2**	**1**
Has the course met your expectations?	17	3	4	–	–
How would you assess the knowledge gained?	12	10	2	–	–
How would you rate the practical experience you received in the course?	15	6	3	–	–
How helpful is the DTT tool for individual research projects?	15	8	–	1	–
How appropriate is the available benchmark data?	18	3	2	–	–

final implementation is documented within a technical report and orally presented at a seminar.

An evaluation aimed at providing an insight into the educational value of the course and the received practical skills and experience was performed for a group of 24 students. The results are summarized in Table 1. Evaluation marks were given from 1 to 5 where a higher value means better quality.

Observations and remarks

We summarize our experiences of the Computational Geometry course as follows.

We found that algorithms for solving geometrical problems are quite attractive for students. Computational geometry is a relatively new research field motivated by those needs arising within application domains such as robotics, VLSI, computer graphic, GIS, etc. Many students foresee their careers in advanced manufacturing where computational geometry solutions play an important role in the design stage, prototyping, and quality control.

Delaunay triangulation is a good initial problem to work on in order to introduce programming and debugging strategies to the students. The complexity level of the problem is appropriate for the students and offers several possibilities for exercising different approaches towards efficient programming solutions.

Although the students were highly motivated and with good programming practice, their approach to collating and interpreting massive geometric data needs additional guidance. They do not have the practical experience of dealing with rounding errors and how to minimize their effects. Furthermore, they do not have experience of debugging the application in the case of larger amounts of input data.

We noticed that students usually test their programs on a small, easily manageable amount of data (up to 100 points). On real data sets with several millions of points the programs may become inefficient or may even perform incorrectly.

Algorithms typically use a number of parameters, the values of which affect the performance, robustness and, in some cases, even the correctness of the code. Analyses of performances for different values of parameters and different input data sets is a valuable experience in student practice.

The introduction of the DTT tool and a set of benchmark data was positively reflected in student feedback regarding the increasing quality of the students' individual research projects, as indicated by the following observations.

- The students can concentrate on the given task instead of programming support codes such as viewing, zooming, or identifying geometric objects,
- The interactive nature of the DTT tool motivates students. The availability of immediate feedback facilitates programming and allows the students to correct mistakes within their programming code,
- The prepared benchmark data sets play an important role in the success of this subject. They provide the possibility for a fair evaluation of the implemented algorithms and are a positive stimulus for the students,
- The DTT and benchmarks increase competition amongst the students,
- When using the benchmark data, the students are made aware of the effect the input data has on the algorithm's efficiency,
- The DTT and associated benchmark data sets allow the teacher to evaluate the students' work fairly.

The DTT tool is, of course, not the only educational support tool for computational geometry, including Delaunay triangulation. Different tools for educators have been reported, such as, for example, the set of teaching blocks proposed by S. Bischoff and L. Kobbelt [21]. While most of them have the advantage of being in the public domain, they are still specific to targeted curricula, which makes them less applicable in practice.

Alternatively, the computer algebraic system, Mathematica (Wolfram Research) has a computational geometry package that includes a command, Delaunay Triangulation, which performs Delaunay triangulation. This package also has some error checking capabilities [22]. However, our aim is that students program Delaunay triangulation by themselves by developing those programming and debugging skills, motivated by the development of the DTT tool.

CONCLUSIONS

Usage of the described DTT tool and benchmark data sets significantly improve the programming skills of students, thus allowing them to gain experience using realistic case studies within the area of computational geometry. Students are highly motivated to achieve established course objectives. They appreciate the opportunity to have practical demonstrations of the strengths and weaknesses of their own implementations. Faculty members also find the process of students providing feedback educationally useful.

On the basis of the student feedback collected during the first year, certain modifications and additions have been made, mainly concerning the user interface. However, since the tool and test data sets are available free to external users, we are soliciting comments from the public. Any proposals for additional data sets that would contribute to the efficiency of the verification process are also welcome. The contact address: denis.spelic@uni-mb.si.

REFERENCES

1. R. Tamassia, Strategic directions in computational geometry, *Working group report, ACM Computing Surveys*, **28**(4), 1996, pp. 591–606.
2. D. Špelic, F. Novak and B. Žalik, Delaunay triangulation benchmarks, *Journal of Electrical Engineering*, **59**(1), 2008, pp. 49–52.
3. http://iris.elf.stuba.sk/JEEEC/data/pdf/1_108-9.pdf (accessed 29 September 2008).
4. Delaunay triangulation at http://www.wikipedia.org. Complete URL http://en.wikipedia.org/wiki/Delaunay_triangulation (accessed 29 September 2008).
5. Definition of circumcircle at http://www.mathopenref.com. Complete URL http://www.mathopenref.com/constcircumcircle.html (accessed 29. September 2008).
6. M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry*, Springer-Verlag, Berlin, (2000).
7. J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, Cambridge, (2001).
8. I. Kolingerova, Modified DAG location for Delaunay triangulation, *Computational Science—ICCS 2002*, Part III, Amsterdam—The Netherlands, LNCS 2331, Springer-Verlag, (2002) pp. 125–134.
9. T. P. Fang and L. Piegl, Delaunay triangulation using a uniform grid, *IEEE Computer & Graphics Applications,* **13**(3), 1993, pp. 36–47.
10. P. Cignoni, C. Montani and R. Scopigno, DeWall: A fast divide and conquer Delaunay triangulation algorithm, *Computer-Aided Design*, **30**(5), 1998, pp. 333– 341.
11. H. Edelsbrunner and R. Seidel, Voronoi diagrams in linear expected time, *Discrete Computational Geometry,* **1**(1), 1986, pp. 25–44.
12. S. J. Fortune, A sweep-line algorithm for Voronoi diagrams, *Algorithmica*, **2**(1), 1987, pp. 153–174.
13. B. Žalik, An efficient sweep-line Delaunay triangulation algorithm, *Computer-Aided Design*, **37**(10), 2005, pp. 1027–1038.
14. J. Kohout, I. Kolingerová and J. Žára, Parallel Delaunay triangulation in E2 and E3 for computers with shared memory, *Parallel Computing*, **31**(5), 2005, pp. 491–522.
15. S. Meguerdichian, F. Koushanfar, G. Qu and M. Potkonjak, Exposure in wireless ad-hoc sensor networks. *ACM SIGMOBILE,* **7**(1), 2001, pp. 139–150.
16. J. Liebeherr, M. Nahas, and W. Si, Application-layer multicasting with Delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications*, **20**(8), 2002, pp. 1472–1488.
17. D. G. Park, H. G. Cho and Y. S. Kim, A TIN compression method using Delaunay triangulation *Int. Journal of Geographical Information Science*, **15**(3), 2001, pp. 255–269.
18. J. Plantier, L. Boutté and S. Lelandais, Defect detection on inclined textured planes using the shape from texture method and the Delaunay triangulation, *EURASIP Journal on Applied Signal Processing*, **7**, 2002, pp. 659–666.
19. Y. Yaoping and W. Chengke, A novel video coding scheme using Delaunay triangulation, *Journal of Visual Communication and Image Representation*, **9**(1), 1998, pp. 80–86.
20. Y. Zhao, F. E. H. Tay, F. S. Chau and G. Zhou, A nonlinearity compensation approach based on Delaunay triangulation to linearize the scanning field of dual-axis micromirror, *Journal of Micromechanics and Microengineering*, **15**(10), 2005, pp. 1972–1978.
21. P. Su and R. L. S. Drysdale, A comparison of sequential Delaunay triangulation algorithms. *Computational Geometry: Theory and Applications,* 7(5–6), 1997, pp. 361–385.
22. S. Bischoff, and L. Kobbelt, Teaching meshes, subdivision and multiresolution techniques. *Computer-Aided Design (CAD Education)*, **20**(14), 2004, pp. 1483–1500.
23. Mathematical software MATHEMATICA, Complete URL http://www.wolfram.com/products/mathematica/index.html (accessed 29. September 2008).

APPENDIX

2D Delaunay triangulation test data sets

Provisional benchmarks are collections of:

- artificial data sets,
- engineering data sets.

Artificial data sets are generated using different rules and distributions, as shown in Fig. 1. These data sets differ in the number of points (we use 1000, 10 000, 100 000 and 1 000 000 points), types of coordinates, and the range of the bounding box surrounding the points (Table 1A). For denoting the data sets, we use the following convention:

(type of distribution)_(number of points)_(type of coordinates).pnt.

For example, the data sets for the uniform distribution with integer coordinates 10 000, 100 000 and 1 000 000 points are denoted as U_10K_I.pnt, U_100K_I.pnt and U_1M_I.pnt.

The benchmark data sets are stored in ASCII files and students can download them from the web site http://gemma.uni-mb.si/dtt.

Table 1A. Artificial benchmark data sets Engineering benchmark data (Table 2A) are obtained from different engineering disciplines (i.e., GIS, mechanics, electromagnetics).

Name	**Type of distribution**	**Type of coordinates**
U_1K_I.pnt	Uniform	Integer
U_1K_F.pnt	Uniform	Floating-point
G_1K_I.pnt	Gaussian	Integer
G_1K_F.pnt	Gaussian	Floating-point
Gr_1K_I.pnt	Grid	Integer
Gr_1K_F.pnt	Grid	Floating-point
C_1K_I.pnt	Clusters	Integer
C_1K_F.pnt	Clusters	Floating-point
L_1K_I.pnt	Line	Integer
L_1K_F.pnt	Line	Floating-point
Ci_1K_I.pnt	Circle	Integer
Ci_1K_F.pnt	Circle	Floating-point

Table 2A. Engineering benchmark data sets.

Name	**Description**	**Type of coordinates**
	Electromagnetic data	
EM_46625_F.pnt	Electrical field around high-power lines	Floating-point
	Mechanical data	
MD_15559_F.pnt	Measurement of forces in a sheet-metal plate	Floating-point
	GIS data	
GIS_4897_F.pnt	Points from irregular triangular network representing terrain	Floating-point
GIS_193360_F.pnt	Points representing boundary stones of a city land cadastre	Floating-point

Denis Špelic is a Ph.D. candidate at the Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests are in computational geometry, computer graphics, and the compression of geometric data.

Franc Novak is Head of the Computer Systems Department at the Jozef Stefan Institute, Ljubljana, and associate professor at the Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests are in the areas of electronic testing and diagnosis, fault-tolerant computing, and design for testability. Novak has an MSc (1977) and a Ph.D. (1988) in electrical engineering, both from the University of Ljubljana.

Borut Žalik is a professor at the Department of Computer Science, Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. He is Head of the Laboratory for geometric modeling and multimedia algorithms. His research interests include computational geometry, geometric modeling, GIS, and multimedia. Žalik has an MSc (1989) and a Ph.D. (1993) in computer science, both from the University of Maribor.