

On Decreased Importance of Algorithmic Strategies in Current ACM Computing Curricula for Graphics & Visual Computing

Andrej Ferko
Graz University of Technology
Comenius University
ferko@fmph.uniba.sk

Abstract

The vividly discussed *ACM Computing Curriculum* initiatives identify the *CS Body of Knowledge*. There are many online educational resources. As observed in [Cunningham02]: „However, the quality of many if not most of this material is not clearly or immediately discernible, thus diminishing its value by requiring substantial evaluation and modification work to make it usable by educators.“ The refereed online resources should hierarchize the **quality** and establish the set of superior resources. In our opinion, for the quality of courseware we have to discuss the quality of curriculum again. The algorithmic strategies are the fundamental and universal tool for creating graphics and vision solutions. Their central role in the currently proposed *CS Body of Knowledge* seems to be underestimated. The paper introduces the context and discusses the importance giving the most beautiful examples from the practical classes.

Keywords

ACM Computing Curriculum, algorithmic strategies, graphics and vision education, computational geometry

1. INTRODUCTION AND MOTIVATION

Algorithmic strategies (or paradigms) provide useful tools for discovering the algorithmic solution of many geometric and computer science problems. Taking into account the deep observation by David Dobkin and Seth Teller [Dobkin97] that *computer graphics is a radiometrically weighted counterpart of computational geometry* we should consider consequences and implications. It might be worth to emphasize these algorithmic paradigms in graphics and vision education as early as possible. Moreover, the conscious application of problem solving methodology is not restricted to computational geometry although computational geometry provides excellent illustrations, especially of the power of algorithmic paradigms. First, we are going to embed the algorithmic paradigms into the broader context of problem solving methodology.

All the easy-to-understand strategies may help students and/or researchers speed up having outline & insight with those aspects of algorithmisation which could be observed (e. g. similarities in the work of various algorithms) but not always expected! Algorithmic strategies provide top quality instruments for consciously application for creating solution of a new problem or improving known algorithm in speed or space requirements.

The deterministic strategies (brute force, iteration, sorting, sweeping, divide & conquer, locus approach, duality,

combinatorial analysis, prune & search, dynamic programming...) and stochastic strategies (simulated annealing, genetic and memetic algorithms...) proved extreme power and elegant results in many fields.

Note, that the context of algorithmic paradigms can be seen in two directions. First, many algorithmic paradigms apply outside of science, e. g. the divide & conquer method was successfully used for maintaining the power of the pope over middle age Italy and Germany which were divided into the absurd number of very small (and powerless) states. Second, there is a scientific methodology framework, namely the methodology of mathematical modeling:

1. *Analysis of the problem.* The output is the mathematic model of the problem.
2. *Analysis of the model.* The output is an algorithm for solution of the problem under assumptions (and error) of the model.
3. *Analysis of the algorithm.* The result is the program.
4. *Computing of experiments* delivers results (alphanumerical, audio, video).
5. *Analysis of the results.* The solution returns into the application area where the problem came from.

Algorithmic paradigms are employed in step 2 of the mathematic modeling methodology. For example, in computer graphics we have for global illumination problem the rendering equation model given in step 1.

Step 2 fails to give the exact algorithm and therefore we repeat step 1 to simplify the model – obtaining either ray tracing or radiosity formulation. Step 2 gives us slow solution in both cases and therefore we employ prune&search strategy for speeding up.

Similarly motivated methodologies apply to other areas (or lower levels) of mathematic modeling. Their cleverness and knowledgeability support creative intuition by reasoning ways which proved competent. For instance, the displaying of results using the (passive) computer graphics methods is summarised into clearly focused four steps strategy. This happens in the step 4 of the above methodology. The same number of steps has the design of user interface for interactive computer graphics [Foley90]. The step 3 of mathematic modeling is where live the programming methodologies (structured programming methodology, object oriented paradigm). Another methodologic framework is used for rendering pipeline (computer graphics reference model), scientific visualisation, software design, even the software quality assurance is standardised by ISO. Methodology and algorithmic strategy appears quite sparse in the discussed body of knowledge. Our educational experience shows that making clear the methodology improves technology perception. [Preparata85] states that there are three equivalent ways of teaching computational geometry – problem oriented, datastructure oriented, and oriented to algorithmic paradigms. If *computer graphics is a radiometrically weighted counterpart of computational geometry* then we should think about implications.

The rest of the paper we structure as follows. First, we try to have an experimental student, who searches for the study method based on computing curricula today. The survey of algorithmic strategies (deterministic strategies: iteration, sweeping, sorting, divide and conquer, prune and search, locus approach, combinatorial analysis... and the stochastic ones: simulated annealing, genetic and memetic algorithms...) will be given. The CS Body of Knowledge from this point of view will be evaluated. The discussion and conclusions will finalize the paper.

2. STUDENT THE EXPERIMENTAL OR

A MENTAL EXPERIMENT: MECHANIC STUDENT

The metaphor here borrows from the Denis Diderot's mechanic actor (the ideally programmable one). Assume we have a mechanically working graphics and vision student who (she-who) will literally follow all the instructions given by the teacher. The set of instructions is simple:

1. Identify global curricula
2. Search for the recent textbooks and courseware - and study them
3. If (Your knowledge not converged) refine identification, search and study

Obviously, this is an algorithm. It has an input (set of rules), the finite number of seemingly deterministic steps, and an output (converged knowledge which is both output data record and halting condition). As a whole, the algorithm has the most erotic structure among fundamental program structures - repeat-until. There are some tricks, in addition. The first step expands one rule to the set of rules. So, this is a meta-rule. (Recall the Aladdin and the ghost or the golden fish fairy tale. The first wish of a logically educated Aladdin is: fulfill one thousand of wishes!)

Instead, the first step should be 1. Identify (one) global curriculum. This might be very helpful for our perfect student. If she will be a non-perfectionist, she will stop with the first global curriculum identified. This is one of the curricula by ACM (3 versions) or ACM SIGGRAPH or ACM SIGCHI or ECET... (six different curricula at all). If the perfect student is perfectionist she will have about six global curricula and, eventually, in addition, the ACM Classification and the ISO/IEC JTC1/SC24 set of double standards, eventually some of them completed with ITU liaisons... (RFC standards excluded.)

This leads to very serious questions. Who is enabled to use fundamental notions like curriculum or definition? What is the status of ISO definition against the other definitions? In math, the situation is simple having the equivalency and proofs of it - like for about 9 different definitions of convex hull.

In Computing we utilize the uniqueness, except for definitions and curricula. It would be useful to adopt the practice from ISO that the current document release finishes the validity of the previous one. More precise, we utilize the uniqueness in computing but not in Computing.

Assume that the latest ACM Computing Curriculum is the right one.

There is no detailed content and there are no books or paper references given. The student has to study another documents to understand the structure of the chapters. It would be useful to adopt the practice from ISO that the current document contains the relevant content of the previous one. The search for on-line books and slide shows will probably contain the recent versions of course notes by Dave Mount [Mount01], Steve Cunningham [Cunningham02], Franz W. Leberl [Leberl01], and Forsyth & Ponce [Forsyth02]. None of them fits with the curricula. In other words, they are excellent new sources of knowledge but they were created asynchronously with curricula. So, our student will search for sets of – not yet refereed - on-line resources, standards, web pages, even e-learning tools, see [Czanner02] for the more info.

Some findings will be again somewhat confusing. The ACM SIGGRAPH Curriculum part for Visualization has matte similarity - eight themes versus five topics - with the ACM SIGGRAPH Curriculum for Visualization.

- Theme 1: Introduction to Visualization
- Theme 2: Data
- Theme 3: User and Tasks
- Theme 4: Mapping
- Theme 5: Representations
- Theme 6: Interaction Issues
- Theme 7: Concepts of the Visualization Process
- Theme 8: Systems and Tools

GV9. Visualization [elective]: Topics:

- Basic viewing and interrogation functions for visualization
- Visualization of vector fields, tensors, and flow data
- Visualization of scalar field or height field: isosurface by the marching cube method
- Direct volume data rendering: ray-casting, transfer functions, segmentation, hardware
- Information visualization: projection and parallel-coordinates methods

The vision oriented student may ask the question about the graphics/vision proportionality.

GV11. Computer vision [elective]: Topics:

- Image acquisition
- The digital image and its properties
- Image preprocessing
- Segmentation (thresholding, edge- and region-based segmentation)
- Shape representation and object recognition
- Motion analysis
- Case studies (object recognition, object tracking)

The system oriented student can visit a taxonomy [Morie01] (ACM SIGGRAPH page) and The Computing Classification System (1998). Another picture appears:

- I.3 COMPUTER GRAPHICS
 - I.3.0 General
 - I.3.1 Hardware Architecture (B.4.2)
 - I.3.2 Graphics Systems (C.2.1, C.2.4, C.3)
 - I.3.3 Picture/Image Generation
 - I.3.4 Graphics Utilities
 - I.3.5 Computational Geometry and Object Modeling
 - I.3.6 Methodology and Techniques
 - I.3.7 Three-Dimensional Graphics and Realism
 - I.3.8 Applications
 - I.3.m Miscellaneous

The ACM Computing Classification System (1998) says:

- I.4 IMAGE PROCESSING AND COMPUTER VISION
 - I.4.0 General
 - I.4.1 Digitization and Image Capture
 - I.4.2 Compression (Coding) (E.4)
 - I.4.3 Enhancement
 - I.4.4 Restoration
 - I.4.5 Reconstruction
 - I.4.6 Segmentation
 - I.4.7 Feature Measurement
 - I.4.8 Scene Analysis
 - I.4.9 Applications
 - I.4.10 Image Representation
 - I.4.m Miscellaneous

Unlike in ACM Computing Curriculum., computer graphics and vision belongs in Classification to Computing methodologies, where they are separated.

I. Computing Methodologies

- I.0 GENERAL
- I.1 SYMBOLIC & ALGEBRAIC MANIPULATION
- I.2 ARTIFICIAL INTELLIGENCE
- I.3 COMPUTER GRAPHICS
- I.4 IMAGE PROCESSING & COMPUTER VISION
- I.5 PATTERN RECOGNITION
- I.6 SIMULATION AND MODELING (G.3)
- I.7 DOCUMENT & TEXT PROCESSING (H.4, H.5)
- I.m MISCELLANEOUS

If the student discovers the ECET page only then her conclusion might be that no curriculum in computing has been completed. The following text is condensed a bit:

The objectives of large network European Computing Education And Training (ECET), aimed at establishing a model of Virtual European department of computing (VEDoC) include Creation of Virtual Recommended Curricula and Syllabi in Computing, Creation of a WEB based Courses in Computing, and Creation of a Virtual Library in Computing. Their motivation is Use and development of the European Credit Transfer System (ECTS) and the System for Quality Control (SQC).

How should a poor teacher answer the possible questions? Well, Leonardo da Vinci was probably the first scientist having relatively unlimited access to all contemporary knowledge. He identified and solved many problems (e. g. medical and flow visualization). As far as we know he had no problems with contradictions in his wide scientific production. Currently, we all are in the best-informed Leonardo's position - sharing web and approaching the semantic web. Too many "co-authors" independently formulate and solve problems, including the sequential unifying the language (definitions, taxonomies, standards, curricula...) and the future "classic" view on our field.

And if the student prefers ISO standards, methodology an invaluable algorithmic strategies, so important in computational geometry and complexity, she will be very surprised only. Is this the right context and content for computer graphics?

3. ALGORITHMIC STRATEGIES SURVEY

An algorithmic problem in computational geometry presents a set of geometric tasks (instances) of given type which is possible to solve by given deterministic algorithm in finite time and space. It is necessary for given input to compute the unique output (solution). If the algorithm is not deterministic then the computational model uses another set of given operations, e. g. random selection of the next step. Just for the sake of the completeness: there are algorithmic unsolvable problems but we assume here the algorithmic solvable problems.

In computer graphics, however, it is possible to solve unsolvable problems. A lovely example: "Since there is not a known reference on the subject, we had to imagine what dragon fire should look like," [Sande01]. Even, we can display our imagination of the spheric lightning. The science has up to now no guess how the model of this natural phenomenon can be derived. Similarly with Shrek's dragon fireball the bombshell has never been measured. The mathematical modeling methodology fails twice but moving images work well...

The solvable geometric problem may be either off-line (the complete input is given at the beginning of computation) or on-line (input is given in some parts during the computation; if the input is provided by the real process we have the real-time problem).

The first known computational geometry problems were probably the tasks to construct some planar object with the ruler and compass (the ancient predecessors of information technology hardware).

A typical example of an algorithmic problem is the construction of a convex hull of N planar points. The notion of convex hull $CH(P)$ of a finite set P of points in the plane is natural and easy to understand at the high school level. According to the definition it is the smallest convex set over P . Unfortunately this simple definition has not the constructive nature.

The solution of the problem means to search for the solution in the set of possible candidates (in state space or in searching space). The trivial algorithm is the exhaustive searching - well-known as the **brute force** method. Using brute force is almost always unfeasible. Therefore we use it in practice for small state spaces only. The brute force method has usually the only advantage - it is often simple for programming. If we search the space of all candidates randomly we have the example of non-deterministic procedure called the **blind search**. While brute force guarantees the exact solution the blind search may neither provide the exact solution nor guarantee the stop of the procedure. This is another important difference between deterministic and stochastic

algorithms - deterministic algorithm always stops after the finite number of steps.

The set of all candidates for the convex hull in the plane are for instance all convex polygons which contain the input point set. The solution is the smallest of them. Searching for the minimal convex polygon by brute force or by blind search is not clever as the state space has enormous size.

This motivates for creating methods for constructing efficient algorithms which are frequently named algorithmic paradigms or techniques/methods/approaches for creating feasible algorithms. The efficient algorithm must be more economic, faster, much more feasible than the brute force method. It is usually more difficult for programming because it may use sophisticated data structures and/or methods with better knowledge of the nature of the problem. If we know for the given problem the lower bound of the problem complexity and we have the algorithm of the same complexity then the algorithm is said to be optimal. Up to now there are not known many optimal algorithms.

The lower bound for the convex hull construction was proved by reduction to the well known lower bound of sorting problem - $O(N \cdot \log N)$. Searching for the minimal convex polygon by divide and conquer algorithm has the worst-case complexity $O(N \cdot \log N)$ and therefore this is the optimal algorithm.

In the following we briefly survey the most popular algorithmic strategies.

3.1 Iteration

Iteration means the gradual adding of elements from which is the result created/composed. Iteration is evaluated as the most simple and the most intuitive method of construction of efficient algorithms. We illustrate the iterative approach by an example which shows that iteration can give the optimal algorithm.

Let us create the convex hull of N planar points. We start with the first three input (noncollinear) points which form the initial triangle (the first iteration of the final convex hull). Now we add the next point. There may occur two cases. Either the added point lies inside the initial convex hull or it is outside. In the first case we ignore the new point. In the other case we search for the two new edges and we remove the interior points and edges from the previous convex hull. It has been proved that this can be realised optimally in $O(N \cdot \log N)$ time.

3.2 Sweeping

Sweeping represents a specialised approach for solution of geometric problems and can be extended to higher dimensions. The fundamentals of this technique we explain in the plane. We use the sweepline which sweeps the plane from left to right (or from top to bottom). The sweepline stops only in the special points named the "event points". The intersection of sweepline with the data structures of the solved problem contains all the

relevant information for the continuation of the sweep. The sweepline never returns. From this point of view we have a greedy method in contrast to the back-track methods. Sweeping requires two basic structures:

A. The event point schedule which is a sequence of x-coordinates ordered from left to right which define the halting positions of the sweepline.

B. Sweepline status - adequately describes the intersection of sweepline with the geometric structure being swept. The status we actualise in each important point.

Sweeping technique enables solution of many problems. E. g. the contour edge filling in computer graphics is an excellent use of sweeping. In the construction of convex hull we must return in some cases and therefore sweeping is not an appropriate paradigm for convex hull computation. There exist some modifications of sweeping technique, e. g. lazy sweeping, [Boissonnat98].

3.3 Sorting

Sorting is frequently used in efficient algorithms construction. The beautiful examples in computer graphics include polygon filling, painter's visibility algorithm, and sorting patches in radiosity.

3.4 Divide & Conquer

This paradigm is probably the most successful method for creating efficient algorithms. Informally it can be described as follows. Let $V(N)$ denote a computational task for processing an input in the size of N . Then for some appropriately large N the divide & conquer method works in the following 3 steps:

1. Divide the task $V(N)$ into K tasks $V(1), \dots, V(K)$. This step divides the original problem into subproblems of smaller size.
2. Compute solutions of tasks $V(1), \dots, V(K)$.
3. Merge the subsolutions of $V(1), \dots, V(K)$ so that the final solution is the solution of the original task $V(N)$.

The complexity of the proposed algorithm is the sum of complexities of the tasks $V(M1), \dots, V(MK)$ + divide step complexity + merge step complexity.

Because of computing solutions of smaller tasks we can express the complete complexity by a recurrent relation. Another advantage of divide & conquer method is the relatively simple proving of correctness.

The example in graphics is Warnock's algorithm.

3.5 Locus Approach

To illustrate the locus approach we employ a very interesting geometric construct - Voronoi diagram [Aurenhammer91], a subdivision of the plane into the convex regions which contain all points which are closer to the input point than to any another input point.

Voronoi diagram for a point is the whole plane, for two points it consists of two halfplanes bordered by a bisector line which separates the two points. The

separator itself is a locus of all points which have equal distance to the two input points. Voronoi diagram for certain (infinite countable) sets of regularly placed points can be the regular tessellation of the plane consisting of identical triangular, rectangular or hexagonal tiles.

The lower bound for Voronoi diagram problem is $O(N \cdot \log N)$ and there are known optimal algorithms. Voronoi diagram gives a preprocessing instrument for optimal solution of seemingly unrelated family of proximity problems: closest pair, all nearest neighbours, triangulation, convex hull, etc.

What is their solution? For the closest pair problem we search for two points, all nearest neighbours define a relation over P , i. e. the oriented graph with at least N oriented edges, and for triangulation we need at least the list of edges for all triangles. The power of locus approach is demonstrated by construction of optimal algorithms for all (and many other) problems which work in two phases. First, the Voronoi diagram is computed. Then the final problem is solved using the Voronoi diagram data structure and the locus properties. For instance, for the triangulation we have to identify edges of all triangles simple by joining by the edge all pairs of Voronoi neighbouring points instead of testing all quadratic number of cases.

3.6 Duality

Many solutions of geometric problems use the transforming of the problem. Among these transformation the duality transform seems to provide maybe the most elegant answers for many questions. The duality transforms points of a d -dimensional space to the hyperplanes and vice versa.

The example of duality transformation is the solution of the following problem. Given a set of points P . Are there 3 collinear points?

Dualising the problem reformulates the question using lines. Given set of lines in dual space. Are there 3 lines which intersect in one point? The dual problem we can understand better and the solution is straightforward using the standard line segments intersection procedure. Another nice duality trick has been used by Ivana Kolingerová for reducing the number of ray-polygon intersections in [Kolinger97].

Note that there is another duality in graph theory. In this sense any planar triangulation is dual to a binary tree.

3.7 Combinatorial Analysis

The example of this approach can be the following reasoning. The triangulation of N points is a planar graph with N vertices. Therefore it has maximum $3N-6$ edges. The solution must consist of the list of the edges.

There is a quadratic number of all possible edges. If we search for any triangulation we can write immediately the unfeasible algorithm. Generate all possible edges. Sort them increasingly. Accept the shortest edge. Accept the next edge from the list if it is not crossed by any accepted

(shorter or equal length) edge. If we have accepted $3N-6$ edges, stop. If the list of all edges was made empty before, stop.

The algorithm is known as a greedy triangulation. Combinatorial analysis gives us at least the better orientation in the state space.

3.8 Prune & Search

The idea with prune and search is to focus not on the solution but on quick identification on the parts of the state space where the solution cannot be.

This paradigm is suitable for solving problems with small size of output – ray tracing of a single ray. It seems the best example for educational purposes is the search for a zero point of a given function in a given interval by the popular interval bisection method. Another perfect example is given by sieve of Erasthones. In fact, prune and search is named sieve in [Rein77]. Prominent uses of prune & search strategy represent speeding-up techniques for ray-tracing.

3.9 Dynamic Programming

Dynamic programming is based on the philosophy of recursively decomposing a problem into a number of smaller problems. Efficiency is gained by avoiding recomputation of the solution of common subproblems. Even though the solution to a subproblem may not be unique, standard dynamic programming involves situations where an arbitrarily chosen single solution to the subproblem suffices for the solution of the parent problem.

3.10 Simulated Annealing

This powerful stochastic approach inspired by thermodynamic optimisation has been discovered independently by Cerný [Cerný82], and Kirkpatrick et al. [Kirkpatrick82]. They assumed the following:

1. The problem solution must be represented in a state space.
2. Each state of the space has a cost.
3. The change of the state must be easy.
4. Some cooling schedule should be applied, e.g. $T_{NEW} = 0.9 * T_{OLD}$, where T_{NEW} and T_{OLD} are thresholds for the "temperature" (weight, cost) of the system state.

The procedure randomly changes the states and with some probability it accepts worse choice, too. This locally worse choice precludes being locked in local optima. The finding of suboptimal solution is guaranteed. The original method utilizes Metropolis algorithm. If we understand to ray-tracing as selecting representative individuals in the space of rays, we can employ simulated annealing.

3.11 Genetic Algorithms

Instead of thermodynamical inspiration the genetic algorithms work with the groups of individuals each of which has the "chromosome". The chromosomes from

two "parents" can be combined by a crossover (or sex) procedure. The chromosome of an individual can be changed by "mutations". Again the chromosome (like the state in simulated annealing) represents one choice in the space of all possible solutions. An example of trivial genetic algorithm is the blind search procedure. Genetic algorithms model the biological evolution. Methodologically related approaches are Neural Networks and DNA Computations.

3.12 Memetic Algorithms

Instead of biological inspiration, the memetic algorithms use the model of cultural evolution or social processes. The extension if compared with genetic algorithms is that the individuals can mutually cooperate and compete. Memetic algorithm instead of one searching individual (simulated annealing) uses a group of independently searching individuals. This makes memetic algorithms especially suitable for parallel implementation.

3.13 Heuristic

If we cannot compute the exact solution of a given problem the we could try to satisfy ourselves by a heuristic, from the famous Archimed's Heureka! - I have discovered! Heuristic procedure can be understood in two senses. First, it provides some close solution, i. e. we have algorithmic *solution of a modified problem*. Second, it provides *exact solution which is not complete* solution of the original problem. Note that the solution of a heuristic is frequently named an approximation. It is worth to know the quality of the approximation.

3.14 More Strategies

Dynamic programming was the last deterministic strategy we have shown here. One can append fractional cascading, edge insertion, look-ahead search, and others. The paradigms for stochastic algorithmisation have excellent results in searching (sub)optimal solutions of complex combinatorial optimisation problems (VLSI design, Euclidean Travelling Salesperson Problem, etc.). There exist another stochastic strategies, i. e. Markov Random Fields. The collection of algorithmic strategies is under the construction.

If we go higher with our focusing on methodology we are in general in the field of creatology. One of the first creatology books is Koestler's The Act of Creation [Koestler64]. He considers the creativity as the transfer of facts from one to another context. In contrast to association he defines the bisociation: evidence of one fact in two contexts. His example of bisociation is the transfer of tides interaction with the small pieces of paper to the gravity theory which applies for the movement of the Moon. Our example for this could be the duality. We have transformed the problem using duality transformation to another context. One of recent creatologic books is [Kim90] oriented to organisation and self-organisation of a research work(er) where is a quite recent bibliography in the field. Let us return back and down to deterministic strategies.

And what about *coherence*? Jed Lengyel in [Lengyel98] says: Another longstanding trend in computer graphics is to pursue coherence whenever it can be found... There is a complete book on coherence [Gröller94].

4. SWEEPING THE ORPHAN OR ALGORITHMIC STRATEGIES IN ACM CC

This very short paragraph is in defense of sweeping. It can be understood as a close relative to sorting. In our opinion, sweeping is so powerful, so easy, and so fundamental that it deserves to be identified and taught in the course - and online refereed educational resources - based on *ACM Computing Curriculum, CS Body of Knowledge*.

This defense applies for all geometric algorithmic strategies. Sweeping stands here for the whole group of brilliant and easy-to-understand ideas with deep impact, not only for graphics. The question goes further. Synchronisation of global online materials, the role of methodology, ISO standards and definitions, algorithmic strategies, taxonomy, classification - and new curricula are the issues. We cannot explain the discrepancies and incomensurabilities to our students very long. The quality of online refereed educational resources cannot be absolute. On the other hand, it cannot be relativized by the lack of synchronisation, at least.

5. CONCLUSION

There is not known the exhaustive number of algorithmic paradigms. Their systematic taxonomy like and could be one of the goals of future work. Today, algorithmic strategies are neglected in the curriculum considerations. A few of them can be found. Irrelevant ones for graphics. Methodology and especially the algorithmic paradigms are the golden treasure of creative thinking in computational geometry (see all above uncited really clever books in the References: Mehlhorn 1984, Edelsbrunner 1987, etc.) and consequently in computer graphics and vision.

Maybe the computer graphics and vision educational community can smuggle them to the online refereed set of resources.

6. ACKNOWLEDGEMENTS

We would like to express our sincere thanks to anonymous referees who asked for better focusing the paper.

7. REFERENCES

[ACMCCS98] The ACM Computing Classification System.. 1998.

<<http://www.acm.org/class/1998/I.3.html>>

[Aurenhammer91] AURENHAMMER, F. 1991. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. pp. 345 - 405. ACM Computing Surveys. Vol. 23. No. 3. September 1991.

[Bern92] BERN, M. - EPPSTEIN, D. 1992. Mesh Generation and Optimal Triangulation. In: DU, D.-Z. - HWANG, F. K. eds. *Computing in Euclidean Geometry*. pp. 23-90. Singapore: World Scientific Publishing 1992.

[Berners-Lee01] Berners-Lee, T. et al. 2001. The semantic Web. Pp. 28-37 in *Scientific American*. May 2001.

[Blahová97] BLAHOVÁ, V., ed. 1997. Nové trendy v informatike (New Trends in Computer Science). Bratislava: ŠPÚ 1997.

[Boissonnat98] BOISSONNAT, J.-D. - YVINEC, M. 1998. *Algorithmic Geometry*. 519 p. Cambridge: Cambridge University Press 1998. ISBN 0-521-56529-4.

[Chalmovianský01] CHALMOVIANSKÝ, P. et al. 2001. *Zložitosť geometrických algoritmov*. (Complexity of Geometric Algorithms). 188 pages. ISBN 80-223-1598-2. Bratislava: Comenius University 2001.

[Chazelle96] CHAZELLE, B. et al. 1996. Application Challenges to Computational Geometry. TR-521-96. Princeton University: April 1996.

<<http://www.cs.princeton.edu/~Chazelle/taskforce/CGreport.ps>>

[Cunningham02] CUNNINGHAM, S. 2002. SIGCSE 2002 BOF on Computer Graphics. Defining the Field of Computer Graphics.

<<http://www.cs.csustan.edu/~rsc/BOF.html>>

[Cunningham02b] CUNNINGHAM, S. 2002. Computer Graphics: Programming, Problem Solving, and Visual Communication.

<<http://www.cs.csustan.edu/~rsc/NSF/Notes.pdf>>

[Cerný85] CERNÝ, V. 1985. A Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm. pp. 41-52. J. Opt. Th. and Appl. Vol. 45 No. 1. January 1985

[deBerg96] DE BERG, M. et al. 1997. *Computational Geometry, Algorithms and Applications*, 365 p. Berlin: Springer-Verlag 1997 ISBN 3-540-61270-X

[Dobkin97] DOBKIN, D. - TELLER, S. 1999. Computer Graphics. Pp. 779-797 in [Goodman97]

[Domik99] DOMIK, G. 1999. *HyperVis - Teaching Scientific Visualization*.

<<http://www.siggraph.org/education/materials/HyperVis/domik/foalien.html>>

[ECET02] EUROPEAN COMPUTING EDUCATION and TRAINING (ECET) Home Page.

<<http://ecet.ecs.ru.acad.bg/index.php?command=displayPage&pid=2>>

[Edelsbrunner87] EDELSBRUNNER, H. 1987. *Algorithms in Combinatorial Geometry*. 423 p. Berlin: Springer-Verlag 1987 ISBN 3-540-13722-X.

[Ferko97] FERKO, A. 1997. Metódy tvorby efektívnych algoritmov ako možný aspekt vo výuke informatiky (On Methods of Efficient Algorithms Construction as a Possi-

ble Aspect in Teaching Computer Science). In: [Blahová97], pp. 57-62.

[Czanner02] CZANNER et al. 2002. Developing ACM Computing Curriculum Courseware on Graphics & Visual Computing. CGE02, Eurographics/SIGGRAPH Workshop On Computer Graphics Education 2002. Bristol, June 6-7, 2002.

[Fisher00] R. FISHER, R. - PERKINS, S. - WALKER, A. - WOLFART, E. 2000. Hypermedia Image Processing Reference (HIPR). Edinburgh: University of Edinburgh 2000.
<<http://www.dai.ed.ac.uk/HIPR2/>>

[Foley90] FOLEY, J. et al. 1990. *Computer Graphics: Principles and Practice*. 1174 p. ISBN 0-201-84840-6. Reading, MA: Addison Wesley 1990.

[Forsyth02] FORSYTH, D. - PONCE, J. 2002. *Computer Vision -- A modern approach*.
<<http://www.cs.berkeley.edu/~daf/book.html>>

[Goodman97] GOODMAN, J. E. - O'ROURKE, J., Eds. 1997. *Handbook of Discrete and Computational Geometry*. Boca Raton - New York: CRC Press 1997.

[Gonzales93] GONZALES, R. C. - WOODS, R. E. 1993. *Digital Image Processing*. 716 p. ISBN 0-201-50803-6. Reading MA: Addison-Wesley Publishing Company.

[Gröller94] GRÖLLER, E. 1994. *Coherence in Computer Graphics*. PhD Thesis. TU Vienna 1994.

[Gross94] GROSS, M. 1994. *Visual Computing*. ISBN 3-540-57222-8. 334 p. Berlin: Springer-Verlag 1994.

[Holland75] HOLLAND, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press 1975. Reprint edition (June 1992) Bradford Books. ISBN 0262581116.

[Holzschuch97] HOLZSCHUCH, N. 1997. *N. Holzschuch Lectures*.
<<http://www.loria.fr/~holzschu/cours/>>

[Ironman01a] IRONMAN Report a. 2001.
<<http://www.computer.org/education/cc2001/index.htm>>

[Ironman01b] IRONMAN Report b. 2001.
<<http://www.computer.org/education/cc2001/ironman/cc2001/v2-courses.html>>

[Kim93] KIM, S. H. 1990. *Essence of Creativity*. ISBN 0195060172. Oxford University Press 1990.
<<http://www.oup-usa.org/isbn/0195060172.html>>

[Kirk82] KIRKPATRICK, S. - GELATT jr., C. D. - VECCHI, M. P. 1982. Optimization by Simulated Annealing. IBM Tech. Report, 1982, also: Science 220 (1983) pp. 671-680, 1983

[Kolingerová97] KOLINGEROVÁ I. 1997. Convex Polyhedron - Line Intersection Detection Using Dual Representation. Pp.42-49 in *The Visual Computer*. Vol.13. No.1. 1997.

[Koes64] KOESTLER, A. 1964. *The Act of Creation*. New York: MacMillan 1964

[Leberl01] LEBERL, F. 2001. *Bildanalyse und Computergrafik*. Web Lecture Notes.
<<http://www.icg.tu-graz.ac.at/~Education/Vorlesung/>>

[Lengyel98] LENGYEL, J. 1998. The Convergence of Graphics and Vision. Pp. 46-53. IEEE. Computer. July 1998.

[MacGregor92] MACGREGOR SMITH J. - WINTER, P. 1992. Computational Geometry and Topological Network Design. pp. 287-385. In: DU, D.-Z. - HWANG, F. K. eds. *Computing in Euclidean Geometry*. Singapore: World Scientific Publishing 1992

[Mariano95] MARIANO, A. - MOSCATO, P. - NORMAN, M. G. 1995. Using L-systems to Generate Arbitrarily Large Instances of the Euclidean Traveling Salesman Problem with Known Optimal Tours. Anales del XXVII Simposio Brasileiro de Pesquisa Operacional. Vitoria. Brazil. 6-8 Nov. 1995.

<ftp://ftp.ing.unlp.edu.ar/pub/papers/memetic/final.ps.Z>

[Mehlhorn84] MEHLHORN, K. 1984. *Multidimensional Searching and Computational Geometry 3: Data Structures and Algorithms*. 284 p. Heidelberg: Springer-Verlag 1984.

[Moritz90] MORITZ, E. 1990. Memetic Science: I - General Introduction. Journal of Ideas. Vol. 1 1990. 28 p.
<http://www.geocities.com/ResearchTriangle/3123/ms2.html>

[Moscato92] MOSCATO, P. - NORMAN, M. G. 1992. A "memetic" approach for the travelling salesman problem - implementation of a computational ecology for combinatorial optimisation on message-passing systems. In: Proceedings of the International Conference on Parallel Computing and Transputer Applications. Amsterdam: IOS Press 1992.

[Moscato97] MOSCATO, P. 1997. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts, Towards Memetic Algorithms. Caltech Concurrent Computation Program 158-79. Pasadena 1997?

[Mount00] MOUNT, D. 2000. *CMSC 427: Computer Graphics Lecture Notes*.
<<http://www.cs.umd.edu/~mount/427/>>

[Owen01] OWEN, S. 2001. *HyperGraph Homepage*. A project of the ACM SIGGRAPH Education Committee, the Hypermedia and Visualization Laboratory, Georgia State University, and the National Science Foundation, headed by G. Scott Owen.
<<http://www.siggraph.org/education/materials/HyperGraph/hypergraph.htm>>

[O'Rourke94] O'ROURKE, J. 1994. *Computational Geometry in C*. Cambridge: Cambridge University Press 1994. 346 p. ISBN 0-521-445922

[Preparata85] PREPARATA, F. P. - SHAMOS, M. I. 1985. *Computational Geometry: An Introduction*. 390 p. New

York: Springer-Verlag 1985. ISBN 0-387-96131-3 ISBN 0-540-96131-3.

[Reingold77] REINGOLD, E. M. - NIEVERGELT, J. - DEO, N. 1977. *Combinatorial Algorithms: Theory and Practice*. 433 p. Englewood Cliffs: Prentice-Hall 1977 ISBN 0-13-152447-X.

[Ružický95] RUŽICKÝ, E. - FERKO, A. 1995. *Počítačová grafika a spracovanie obrazu. [Computer Graphics and Image Processing, in Slovak]*. 316 p. ISBN 80-967180-2-9. Bratislava: SAPIENTIA 1995.

[Sande01] SANDE, L. R. 2001. Shrek: the Story behind the Screen. ACM SIGGRAPH 2001 Course Notes 33

[Steelman01] STEELMAN Report. 2001.
<http://www.computer.org/education/cc2001/steelman/cc2001/index.htm>

[Strawman00] STRAWMAN Report. 2000.
<<http://www.computer.org/education/cc2001/report/index.htm>>

[Strasser95] STRASSER, W. 1995. *Graphische Datenverarbeitung*. Course Notes in German. Hagen: Gesamthochschule 1995.

[Morie01] MORIE, J. F. 2001. *CGI Taxonomy Project*.

<<http://www.siggraph.org/education/curriculum/projects/Taxonomy2001.htm>>

[Toussaint85] TOUSSAINT, G. T., ed. 1985[Tous85] TOUSSAINT, G. T., ed. 1985. *Computational Geometry*. ISBN 0-444-87806-8. Elsevier 1985.

[VanDam01] VAN DAM, A. 2001. Exploratory: Applets.
<<http://www.cs.brown.edu/research/graphics/research/exploratory/research/applets/>>

[Watt92] WATT, A. - WATT, M. 1992. *Advanced Animation and Rendering Techniques*. 455 p. ISBN 0-201-54412-1. ACM Press. Workingham: Addison-Wesley 1992.

[Watt95] WATT, A. 1995. *3D Computer Graphics*. Second edition. 500 p. ISBN 0-201-63186-5. Workingham: Addison-Wesley 1995.

[Watt00] WATT, A. 2000. *Three-Dimensional Computer Graphics*. Third edition. ISBN 0-201-39855-9. New York: Addison-Wesley 2000.