

Web with art and computer science

Anna Ursyn
School of Art & Design
University of Northern Colorado
ursyn@unco.edu

Terry Scott
School of Natural and Health Sciences
University of Northern Colorado
tscott@fisher.unco.edu

Abstract

This paper describes the integrative, cooperative instruction in web design that applies art in technical instruction (programming). Students work in an interdisciplinary style drawing upon concepts from programming, web design, computer graphics, art, and design, both in the form of interdisciplinary studies and in a web construction and design class. Examples of student work support presentation.

Categories and Subject Descriptors

K.3.1. [Computer Uses in Education] Computer-assisted Instruction

J.5. [Fine Arts]

General Terms

Algorithms, Design, Languages

Keywords

Collaborative practice, interdisciplinary instruction, Science–art connection, programming, web design, computer graphics, art, and design.

1. Introduction

In this paper, we describe a proactive environment aimed to integrate computer graphics, web design, and visual learning skills. We describe instruction where students enhance their creative problem solving using various programming languages and combine both the precise and the expressive way of thinking. First, we provide a review of research on the benefits of applying visual science in technical instruction, then a review of relevant literature on an integrative, interdisciplinary way of teaching, followed by a narrative about the interdisciplinary course in web design, that includes methods, analysis, discussion and conclusions, the challenges, successes, and outcomes coming from the experience gained through our long-term collaboration.

Our teamwork has been possible two ways: through an interdisciplinary Computer Science/Art program that allows students to create their own course of studies and the Honors Program that offers opportunities for the faculty to merge their areas of expertise. In the spring semester of 2006, as an interdisciplinary team of professionals in computer science and art, we offered an integrative course of studies entitled “Web Construction and Design” aimed to improve students’ understanding and skills in the areas of both computer science and art.

Both faculty members have been interested for some time in the overlap between thinking in terms of art and computer science [Ursyn et al. 1997]. This project was a continuation of previous studies on the effects of art instruction on student achievement in science [Ursyn 1997, 1991].

2. Rationale

There is a growing need for people who possess both programming skills and art and design skills. Usually, students tend to be better in either Art/Design or in Computer Science, but the best opportunities are for those who excel in both areas. The authors have been working on providing students with ways to merge the two disciplines. Integrative instruction was intended to develop educational and training methods that involve students in learning programming through art (one can view the source code).

The environment for web creation is a natural place to start with programming. Web design has become a significant part of media and its social, cultural, political, and critical forces. Since the number of people designing websites is growing rapidly and there is growing competition, people need to understand concepts behind art and design. The same applies to the industries that involve animation. Thus, it is expected that more and more students will find a way to combine their art and programming skills.

3. Conceptual Framework

3.1. Overlap between thinking in terms of art and computer graphics

Thanks to classical works of the precursors of cognitive science such as Benjamin Bloom [e. g., 1971], J. P. Guilford [1967], and then R. J. Sternberg [1991, 1992] it has been generally accepted that intellectual behavior in learning occurs in the cognitive, psychomotor, and affective domains, with knowledge, comprehension, application, analysis, synthesis, evaluation, induction and deduction processes. According to the Guilford’s structure-of-intellect model, the levels of cognitive activity include: operation (evaluation, convergent production, divergent production, memory, cognition), product (units, classes, relations, systems, transformations, implications), and content (figural, symbolic, semantic, behavioral). Cognitive thinking is often referred to problem solving, hypothesis testing, and concept acquisition. Sternberg’s model of human intelligence included components or executive planning, monitoring, and evaluation processes, performance processes, and knowledge acquisition processes, all of them allowing learning, comprehending, and

remembering information. He defined orders of relations, with first-order relations between primitive terms, second-order relations between relations, and so on.

Critical thinking skills entail logical thinking and reasoning that include comparison, classification, sequencing, cause/effect, patterning, webbing, analogies, deductive and inductive reasoning, forecasting, planning, hypothesizing, and critiquing [Dunbar 1997]. Critical thinking that traditionally has been ascribed to left-brain reasoning is typified as analytic, convergent, verbal, linear, objective, judgmental, focused on a subject, and probability of its change. Scientific thinking, used in investigating processes, events, acquiring new and integrating previous scientific knowledge, involve cognitive processes for gathering observable, empirical, measurable evidence, generation of a theory, designing an experiment, hypothesis testing, and data interpretation. Intelligence is creative and self-creative within our inner realities, in process of self-actualization, as in T.S.Eliot's verse, Little Gidding, "Either you had no purpose or the purpose is beyond the end you figured and is altered in fulfillment."

Visual thinking and learning involves cognition and refers to the idea that communication occurs through visual symbols, as opposed to verbal symbols, or words. It takes place through visual, often nonlinear processing happening beyond the definitions of language and often produce personal referents and insights to meaning that cannot be translated into linear manner using language. Creative thinking that enables students to create something new or original involves originality, elaboration, brainstorming, imagery, metaphorical and associative thinking. Guilford [1968] described different types of creative abilities: sensitivity to problems, fluency factor, novel ideas, flexibility of mind, synthesizing and analyzing ability, reorganization or redefinition of organized wholes, complexity, evaluation, motivational factors and temperament. Creative thinking that traditionally has been ascribed to right-brain activity may be characterized as generative, divergent, visual, associative, subjective, and open to possibilities and novelty.

It is possible to discern several areas for overlapping between thinking in terms of art and computer graphics. Thanks to the ingenious work of Rudolf Arnheim [1969, 1974, 1990] and then the master work of Edward Tufte [1983, 1990, 1997], the mode of applying the visual along with the verbal way of communication has been accepted, developed and taught in the precise and thoughtful way, with all advantages coming from the chances for experiencing the insight and using mental shortcuts provided by this way of thinking. The perception of a shape requires the grasping of its essential structural features. Research on scientific thinking [Dunbar 1997] revealed that much of the scientists' reasoning and over 50% of the findings resulted from interpreting unexpected findings that were very different from the hypotheses based on literature. It was also found that scientists use analogies from similar domains in proposing new hypotheses. The visual way of thinking is related to the methods of simulation and visualization because it brings forth an ability to perceive complex systems.

3.2. Learning and Thinking Visually

Computer graphics has a multidisciplinary nature, involving components of cognitive psychology, geometry, imaging science, technology, art and design, and computer science [Bertoline and

Laxer 2002]. Computer Graphics has become a discipline of study involving the two tracks, artistic and technical. Knowledge-based curricula contain the fusion of artistic and technical theories, necessary for developing skills that are attractive to both computer science and art majors [Alley et al. 2006]. An understanding of the structured type of thought that goes into computer science could improve everyday interactions and work environments for artists and scientists. It may be valuable to those who are pursuing a career in computer graphics, animation, and web. According to Eppler and Burkhard [2006], visualization outperforms text alone and increases our ability to think and communicate. People think in pictures, so knowledge must be recreated in the mind of the receiver. Visual metaphors combine the creative leap of sketches with the analytic rationality of conceptual diagrams. They also organize and structure information in a meaningful way, and convey a through the key characteristics or associations of the metaphor. Communication calls for language, imaginary or articulated; pictures need a caption. Because language is metaphorical, there is not non-metaphorical thought. Thus, visualization is pictorial and linguistic at the same time, both kinds being complementary parts of communication. Thus, according to Bertschi and Bubenhofer [2005], the metaphor is a tool of conceptual economy, but also a tool of discovery of structures within novel or unfamiliar situations.

A growing number of authors have adopted an idea of teaching problem solving and decision-making skills with the use of mental imagery. Hartman and Bertoline [2005] asserted that the computer graphics learning environment takes advantage of a learner's ability to quickly process and remember visual information, as about 80% of sensory input comes from our visual system. The authors postulate that a body of knowledge called Visual Science should be studied, practiced, and scientifically verified as a discipline. They define Visual Science as the study of the processes that produce real images or images in the mind.

3.3. Visualizing Problems, Creating Simulations

There are a growing number of industries that request the ability to create graphic images directly from data and algorithmically manipulate this data, whether linear, 3D, or multidimensional [Graham 2005]. Another demand is for the use of interactive visual representations of abstract data to amplify cognition [Bederson and Shneiderman 2003]. Interactive visualization is an essential issue in organizational communication or knowledge media design [Klein 2005]. Burkhard et al. [2005] list types of visualization as sketches, diagrams, images, maps, objects, interactive visualizations, and stories. Frameworks for knowledge maps involve questions about their function type (what?), recipient type (whom?), and map type (how?) that may include a heuristic, diagrammatic, metaphoric, geographic, 3D, interactive, or mental map. Klein [2005] postulates that simulation models lead to actions in the real world, because real world figures and interrelations go into the simulation.

3.4. Integrative Curricula

Computer graphics has become an attractive field of study to both computer science and art majors, as it often provides creative and self-motivating job opportunities. Tasks are made easier when students have expertise in both these areas and are likely to work together. Specialists working in game design need skills provided

by art courses. Art students need technical skills for multimedia productions, and advanced Web page design. Magazine and newspaper publishers need people with programming skills when they launch online video advertising. The same opportunities are in the TV networks, newly established firms, and web portals.

Digital art programs have additional components that focus on software instruction and technological literacy [Wands 2006]. Computer graphics specialists design programs to provide research-oriented interdisciplinary collaborative experiences that engage students from varying disciplines [Palazzi et al. 2006]. Several universities have already developed collaborative interdisciplinary curricula designed for art and computer science undergraduates in a liberal arts environment. For example, students are provided with experiences in computer graphics at the Advanced Computing Center for the Arts and Design at the Ohio State University [Palazzi et al. 2006], the interdisciplinary program in computer graphics at the Departments of Art and Computer Science at Shippensburg University [Mooney 2006], Clemson University [Matzko and Davis 2006], Minneapolis College of Art and Design [Border 2006], and many others. Several authors stress the relationship to spatial visualization abilities and computer graphics education [Hartman and Bertoline 2006]. In several cases, using computers as design tools and applying computer animation and graphics served for teaching elementary school level mathematics [Eisenberg et al. 2005].

4. Pedagogical Strategies: Interdisciplinary Approach

The main objective of our integrative instruction was to improve students' technical expertise in computer science, their art skills, and their understanding of the connections between computer science, art, and design. Students worked in an interdisciplinary style, drawing upon concepts from programming, web design, computer graphics, art, and design. We have adopted a constructivist approach: students made drawings rather than read about theoretical drawings. Graphics they created made a tool to learn with. At the same time, we have adopted a cognitive framework for solving technical challenges by introducing the integrative environment in the classroom.

Students came from many different areas of study, although there were more technical students than those from the humanities. Some of them had no prior experience with programming. Students enrolled in Interdisciplinary Program have chosen C++, Open GL, and Python for their Computer Science part of the training, to name just a few languages. Students also used commercial programs, such as Adobe Studio, Macromedia Studio, 3dsmax and more. It allowed for changing and improving image files.

During the Spring 2006 Web Design course, the Computer Science professor offered instruction in technical areas, with consultation from the art professor. It involved HTML and CSS (cascading style sheets used extensively on the Web to make web pages have the same formatting and layout) [Hart-Davis 2005; Niederst 2001] and JavaScript used extensively to make web pages. It was included because it uses client-side processing [Pollock 2004]. PHP was chosen because it is used on the server side. It allows the code to be placed inside tags that are nested inside HTML tags, it is open source, and interfaces easily with MySQL [Harris 2004]. MySQL was chosen because it is open

source and because database is required to maintain information on many websites [Harris 2004]. Blogs use this technology, as well as Wiki.

The chosen areas were open source and may be recognized by the initials LAMP, which stands for Linux, Apache (the web server), MySQL, and PHP. Students and instructors alike learned much about the coding during the class. The computer server for the class ran Slackware Linux. A separate directory was created for the class and each student was given a directory of her/his own that was owned by them and had a symbolic link from their home directories to the individual web directories.

A unique feature of both the class and the program were that assignments were open ended. This means, students were receiving the sources for inspiration within a framework provided, rather than a description of an expected outcome. They could go any direction they'd choose. Some of them were looking for answers on the Internet, while some tried to create something totally new. The art professor gave assignments that required the class members to use the new technical concept following a common design idea. Class members created their own individual designs around this common idea. The students were very good at creating interesting and varied projects around the given concept.

Figure 1 shows one of the class projects. A student, Royce Wood, designed a robot. The user had 10 points to distribute between robot's intelligence, weaponry, and speed. Depending on the visitors' choices, different robots would show up on the screen.

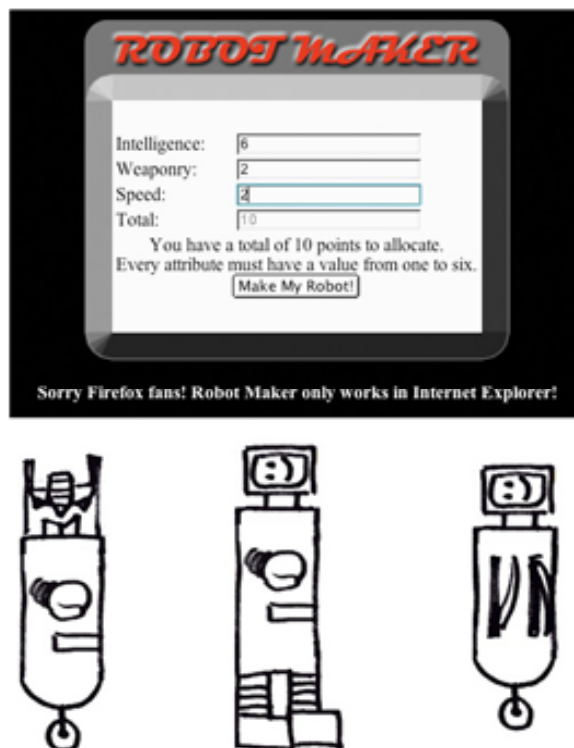


Fig. 1. Example of class assignments – design a robot.

The two instructors had different teaching styles. The computer science instructor presented information in a lecture style. On the other hand, apart from an introductory talk on a particular subject, the art professor gave the assignments so that they were not prescriptive and allowed for much creativity. The art instructor delivered some lectures on design, art, information visualization, visual data organization, and presented some examples of computer generated art. Students answered some questions prepared by the instructor for a proof of their understanding.

The technical aspects were relatively easy to evaluate since either the software worked or it did not. The art and design assessment was handled by having the whole class participate in critiques of each class member's work. The class critiques were aimed to look at the progress made

5. The Assignments

Students were required to post their web pages on the server so that all could observe what they had accomplished. The interdisciplinary students were asked to develop projects and also to present the outcomes in front of their Interdisciplinary Committee.

The students were encouraged to organize explicitly:

1. The assignment or test requirements
2. The objectives of the assignment
3. The programmer's thoughts, and what they learned
4. Pertinent links for the assignment
5. The created webpage(s).

In one of the class assignments, the students were asked to create a game environment that promoted them as programmers/designers. This assignment was meant to serve as their own portfolio/resume, outlining their skills, knowledge, and achievements. It was recommended that the portfolio be interactive, informational, and entertaining so the audience would remember them. The user might be forced to do something or there might be an element of being challenged, if they'd prefer. Description of this assignment should include words: playful, interactive, challenge, and surprise. It was recommended that this web page (one or more pages) be a "show and tell" and be active rather than passive. Students were encouraged to try to showcase their skills in web design, specifically in making the web pages very interactive for the user.


Another assignment, entitled a "C++ Cow program" shows a visual learner how the basic structure of C++ works by connecting its user-defined elements with visual symbols. Graphics created by a student Ben Hobgood describe some simple aspects of the C++ class as an abstract data type. The class is used in a program that asks for the pounds of food that one wishes to feed a cow named Betsy, and tells how many steaks can be made from her. This graphical representation of C++ has source code and an executable to back it up (compiled for DOS). Note: some words are in bold typeface and indicate the definitions. The names of the files are in *italics*.

This is a declaration of the class called cow or more formally known as a class declaration. Defined as an ordinary every day cow. Nothing fancy, just some of the basic stuff about a cow.


cow.h (the declaration of the cow class)


```

#include <iostream.h>


class  {


  private:


     ;


     ;


  public:

     () {

    void  ( int );

    void  ();

    int  ();

    int  ();

  };

```

Fig.2. Example of a C++ header file or class declaration
 This is a definition of the class called cow that was declared above in the cow.h file. More formally known as a class definition this file describes exactly how the abstract data type, cow, works.

cow.cpp (The definition of the cow class)

```

#include "cow.h"

Cow::Cow() {
    weight = 0;
    height = 0;
}

void Cow::Cow(int hay) {
    weight = (hay / 7);
}

void Cow::Cow(Cow c) {
    weight = (c.weight) * 16 / 12;
}

int Cow::Cow(Cow c) {
    return c.weight;
}

int Cow::Cow(Cow c) {
    return c.height;
}

```

Fig. 3. Example of a C++ class definition
 The abstract data type cow that we declared in cow.h, and defined in cow.cpp is used as a data type to collect information about an instance of the cow class called Betsy (black and white cow with a little more character)

betsy.cpp (Implementing the cow class in a program)

```

#include "cow.h"

void main() {
    Cow Betsy;
    int hay;
    int weight;
    int height;

    cout << "How many pounds of hay "
         << "do you want to feed Betsy? ";

    cin >> hay;

    Cow Betsy(hay);
    Cow Betsy(Betsy);
    weight = Betsy.weight;
    height = Betsy.height;

    cout << "Betsy weighs " << Betsy.weight
         << " pounds and can yeild"
         << Betsy.height << " yummy 12 ounce steaks!";
}

```

Fig. 4. Example of a C++ file that uses the cow data type

This program was small and simple to start things out but the topic could easily be used to describe many aspects of object oriented programming. Topics such as data encapsulation could easily be described while visually contrasting this class with classes describing other animals, farmers or perhaps milking cows. Inheritance could be illustrated by creating a basic cow class in which cows bread for eating, milking, and bull fighting could each derive from. Likewise the subject of polymorphism

naturally fits in with the topic since a classical approach is to describe it using animals. Operators such as the input and output operators used in this class may need a little bit of explaining for a discussion about this class but could easily be illustrated using a cow and its products.

The nice thing about illustrating programming concepts is that we can decide which parts of the program to distinguish from others instead of letting a glob of text overwhelm us. This approach may not be great for everyone but I believe that it could help alleviate some of the initial fear of programming and overwhelming information overload that scares off many potential programmers as it nearly did I.

The following are the files that would be compiled by a C++ compiler and run on an intel machine. This would follow the graphical explanation to reassure the student that the concept works. This element is defiantly necessary until we write a visual compiler!

cow.h (the declaration of the cow class)

```
#include <iostream.h> // cout, cin functions
```

```
class Cow{
private:
    int TwelveOunceSteaks;
    int CowPounds;
public:
    Cow();
    void FeedCow(int);
    void MakeSteaks();
    int GetCowSize();
    int GetSteaks();
};
```

cow.cpp (The definition of the cow class)

```
#include "cow.h"

Cow::Cow(){
    TwelveOunceSteaks = 0;
    CowPounds = 0;
}

void
Cow::FeedCow(int foodPounds){
    // grain to beaf conversion is 1 to 7
    CowPounds = int(foodPounds/7);
}

void
Cow::MakeSteaks(){
    // Assuming that the entire cow can be made into steaks...
    // Number of 12 ounce steaks equals the weight in pounds times
    16 ounces
    // in a pound divided by 12 ounces per steak
    TwelveOunceSteaks = ((CowPounds * 16) / 12);
}
int
Cow::GetSteaks(){
    return TwelveOunceSteaks;
}
```

```
int
Cow::GetCowSize(){
    return CowPounds;
}
betsy.cpp (Implementing the cow class in a program)

#include "cow.h"

void
main(){
    Cow Betsy;
    int poundsOfFood;
    int numberOfSteaks;
    int BetsySize;
    cout << "How many pounds of hay do you want to feed Betsy?:";
    cin >> poundsOfFood;
    Betsy.FeedCow(poundsOfFood);
    Betsy.MakeSteaks();
    numberOfSteaks = Betsy.GetSteaks();
    BetsySize = Betsy.GetCowSize();
    cout << "Betsy weighs " << BetsySize << " pounds and can
    yeild "
        << numberOfSteaks << " yummy 12 ounce
    steaks!";
}
```

6. Discussion and Conclusions

In continuation of efforts to improve student understanding [Scott 2006], the educational environment described in this paper was aimed to engage the students in combining both the precise and expressive way of thinking. The authors discussed what should be done to improve the class. First would be to find a really good JavaScript analyzer. There was a free one on the Web but it was not sufficient [Crockford 2002]. The problem with the Java Script was that if there was an error in the JavaScript, it failed to do anything, and it was not possible to easily determine where the error was or what caused it. Because of the JavaScript issues, it would have been better to teach it after PHP and MySQL.

Another technical issue was a problem with psftp for Windows. The connection to the server was lost if the mouse left the application window. The authors are still not certain what the problem was, but this should be fixed before the class is taught again. The important issue that was discussed but not enforced was to have students place files for each assignment and test in separate directories. This kind of organization helps keep the web pages organized and makes it easier to fix problems when they occur. Instructors insisted that students should provide their own images to prevent copyright violation; however, some of the students tried to obtain images from the web. Students rated the class above average. Former interdisciplinary students found their place in the industry and some of them presented at conferences. Some of the student projects were selected for presentations and at the conferences, other will be chosen for the Consortium for Computing Sciences in Colleges, Rocky Mountain Section Conference presentation and discussion.

References

- ALLEY, T. et al. 2006. Knowledge Base for the Emerging Discipline of Computer Graphics. The ACM SIGGRAPH Education Committee. Full Conference DVD, a publication of ACM SIGGRAPH.
- ARNHEIM, R. 1969. *Visual Thinking*. University of California Press. (Also: London: Faber and Faber, 1969). Also:
- ARNHEIM, R. 1974. *Art and Visual Perception*. University of California Press.
- ARNHEIM, R. 1990. Language and the Early Cinema. *Leonardo Digital Image – Digital Cinema Supplemental Issue*, 3-4.
- BEDERSON, B. and SHNEIDERMAN, B. 2003. *The Craft of Information Visualization: Readings and Reflections*. Morgan Kaufmann Publishers, San Francisco, CA.
- BERTOLINE, G. AND LAXER, C. 2002. Forum: A Knowledge Base for the Computer Graphics Discipline, *ACM SIGGRAPH 2002 Educators Program*. <http://www.siggraph.org/s2006/downloads/bertoline.pdf>.
- BERTSCHI S. and BUBENHOFER B. 2005. *Proceedings of Ninth International Conference on Information Visualisation (iV 05)* (IEEE Computer Society). Los Alamitos, CA, Washington, Brussels, Tokyo), 383-389.
- BLOOM, B. S., HASTINGS, J. T., and MADAUS, G. F. 1971. *Handbook on Formative and Summative Evaluation of Student Learning*. McGraw-Hill Book Company.
- BORDER, P. M. 2006. A Data Visualization Course at an Art School. *The ACM SIGGRAPH Education Committee. Full Conference DVD*, a publication of ACM SIGGRAPH.
- BURKHARD, R. et al. 2005. Beyond Excel and PowerPoint: Knowledge Maps for the Transfer and Creation of Knowledge in Organizations. In *Proceedings of Ninth International Conference on Information Visualisation (iV 05)* (IEEE Computer Society). Los Alamitos, CA, Washington, Brussels, Tokyo), 76-81.
- CROCKFORD, D. 2002. *The Java Script Verifier*. JSLint, <http://www.jshint.com/>.
- DUNBAR, K. 1997. How scientists think: Online creativity and conceptual change in science. In *T. B. Ward, S. M. Smith, & S. Vaid (Eds.) Conceptual structures and processes: Emergence, discovery and change*. APA Press.
- EISENBERG, M. et al. 2005. Mathematical Crafts for Children: Beyond Scissors and Glue. In *Art+Math=X International Conference Proceedings*, Carla Farsi, Ed., 61-64.
- EPPLER, M. J. and BURKHARD, R. M. 2004. *Knowledge Visualization*, <http://www.knowledgemedia.org/modules/pub/view.php/knowledgemedia-67>, [01/08/2007]
- GRAHAM, D. 2005. Information Visualization Theory and Practice. In *Proceedings of Ninth International Conference on Information Visualisation (iV 05)* (IEEE Computer Society). Los Alamitos, CA, Washington, Brussels, Tokyo), 2005, 599-603.
- GUILFORD, J. P. 1967. *The Nature of Human Intelligence*. McGraw-Hill.
- GUILFORD, J. P. 1968. *Intelligence, Creativity and their Educational Implications*. Robert Knapp.
- HARRIS, A. 2004. *PHP 5/MySQL programming*. Thomson Course Technology.
- HART-DAVIS, G. 2005. *HTML QuickSteps*, New York, McGraw Hill.
- HARTMAN, N. W. and BERTOLINE, G. R. 2005. Spatial Abilities and Virtual Technologies: Examining the Computer Graphics Learning Environment. *Proceedings of Ninth International Conference on Information Visualisation (iV 05)* (IEEE Computer Society). Los Alamitos, CA, Washington, Brussels, Tokyo), 992-999.
- HARTMAN, N. W. & G. R. BERTOLINE, 2006. Virtual Reality-based Spatial Skills Assessment and Its Role in Computer Graphics Education. In *The ACM SIGGRAPH Education Committee. Full Conference DVD*, a publication of ACM SIGGRAPH.
- KLEIN, S. 2005. Knowledge Visualization in Practice: Challenges for Future Corporate Communication. In *Proceedings of Ninth International Conference on Information Visualisation (iV 05)* (IEEE Computer Society). Los Alamitos, CA, Washington, Brussels, Tokyo), 70-75.
- MATZKO, S. and DAVIS, T. 2006. Using Graphics Research to Teach Freshman Computer Science. In *The ACM SIGGRAPH Educators Program Panel, Full Conference DVD*, a publication of ACM SIGGRAPH.
- MOONEY, D. 2006. <http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/10731/33854/01612097.pdf>
- NIEDERST, J. 2001. *Web Design In a Nutshell: A Desktop Quick Reference* 2nd Ed. Cambridge.
- PALAZZI, M. et al. 2006. Designing Collaborative Interdisciplinary CG Experiences in the Curriculum. In *The ACM SIGGRAPH Educators Program Panel, Full Conference DVD*, a publication of ACM SIGGRAPH.
- POLLOCK, J. 2004. *JavaScript: A Beginner's Guide*, 2nd Ed., McGraw Hill/Osborne.
- SCOTT, T. 2006. In-Class Projects to Enhance Student Understanding. *The Journal of Computing Sciences in Colleges*, 21, 3, 147-153.
- STERNBERG, R. J. 1991. Higher-order reasoning in postformal operational thought. In: *Merlin C. Wittrock and Eva L. Baker (Eds.), Testing and Cognition*. Prentice Hall, pp. 31-39 and 74-91. Also:
- STERNBERG, R. J. and DAVIDSON, J. (Eds.). 1992. *Mechanisms of Insight*. MIT Press.

STERNBERG, R. J. and DAVIDSON, J. (Eds.). 1991. *Conceptions of Giftedness*. Cambridge University Press.

TUFTE, E. R. 1983. *The visual display of quantitative information*. Graphics Press.

TUFTE, E. R. 1990. *Envisioning information*. Graphics Press.

TUFTE, E. R. 1997. *Visual explanations. Images and quantities, evidence and narrative*. Graphics Press.

URSYN, A., MILLS, L., HOBGOOD, B. and SCOTT, T. 1997, Combining Art Skills with Programming in Teaching Computer Art Graphics. In *Proceedings of ACM SIGGRAPH 97*, Computer Graphics, 25, 3 (July 97).

URSYN, A. 1997. Computer Art Graphics Integration of Art and Science. *Learning and Instruction, The Journal of the European Association for Research on Learning and Instruction (EARLI)*, 7, 1, 65-87.

URSYN, A. 1991. The Use of Computer Art Graphics in Art and Science. In Abshire, K. Discovery Through Experimentation, Art and Educational Computing in Secondary Schools. In *Computer Graphics, 253, Proceedings of ACM SIGGRAPH 91 Educator's Program*. 168-171.

WANDS, B. 2006. Right Brain/Left Brain: Balancing Digital Art Curricula. In *The ACM SIGGRAPH Educators Program Panel, Full Conference DVD*, a publication of ACM SIGGRAPH.