2012

Creative Coding and Visual Portfolios for CS1

Ira Greenberg

Deepak Kumar
Bryn Mawr College, dkumar@brynmawr.edu

Dianna Xu
Bryn Mawr College, dxu@brynmawr.edu

Custom Citation
Greenberg, I. et al. 2012.Creative coding and visual portfolios for CS1, Proceedings of the 43rd ACM technical symposium on Computer Science Education, February 29-March 03, 2012, Raleigh, North Carolina: 247-252.

Creative Coding and Visual Portfolios for CS1

Ira Greenberg
Center of Creative Computation
Dept. of Comp. Sci. & Engineering
Southern Methodist University
Dallas, TX (USA)
igreenberg@smu.edu

Deepak Kumar
Department of Computer Science
Bryn Mawr College
Bryn Mawr, PA (USA)
(1) 610-526-7485
dkumar@brynmawr.edu

Dianna Xu
Department of Computer Science
Bryn Mawr College
Bryn Mawr, PA (USA)
(1) 610-526-6502
dxu@brynmawr.edu

ABSTRACT

In this paper, we present the design and development of a new approach to teaching the college-level introductory computing course (CS1) using the context of art and creative coding. Over the course of a semester, students create a portfolio of aesthetic visual designs that employ basic computing structures typically taught in traditional CS1 courses using the *Processing* programming language. The goal of this approach is to bring the excitement, creativity, and innovation fostered by the context of creative coding. We also present results from a comparative study involving two offerings of the new course at two different institutions. Additionally, we compare our results with another successful approach that uses personal robots to teach CS1.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education-*computer science education*.

General Terms

Design, Experimentation.

Keywords

CS1, computer science, education, pedagogy, creative coding, art, visual portfolio, Processing.

1. INTRODUCTION

Many theories have been put forward to explain the waning CS enrollments. While there is little agreement on a single cause or solution, there is some consensus that Computer Science has an image problem [Yardi & Bruckman 2007, Guzdial 2009]. The most wired and computationally involved student population ever perceives Computer Science as tedious, antisocial and irrelevant. A lagging response by CS departments to modify less relevant pedagogical approaches, perhaps too fettered to the mathematical and engineering legacy of the discipline [Hillberg & Meiselwitz 2008] does little to improve the image.

The contextualized approach to introduce students to Computer Science has gained momentum and recognition in recent years. Notable efforts include media computation [Guzdial 2004], robots [Summet et al 2009], games/animation [Moskal et al 2004, Bayless & Stout 2006], and music [Beck et al 2011]. Arguments in favor of teaching CS1 in context are persuasive, and may provide a much needed trigger for sustained student interest well beyond the introductory courses [Kay 2011, Guzdial 2010].

We present the design and development of a new context to teaching CS1, using generative art and creative coding. Over the course of a semester, students create a portfolio of aesthetic visual designs that employ basic computing structures typically taught in traditional CS1 courses using the *Processing* programming language [Processing Group]. The goal of this approach is to present computing as a medium of creativity and nurture an accessible, engaging environment that attracts a modern, diverse student body that appreciates the excitement, creativity, and innovation that computing brings.

In this paper we present results from a comparative study involving two parallel offerings of the new course at two different institutions, Bryn Mawr College and Southern Methodist University (SMU). Additionally, we compare our results with another successful approach that uses personal robots to teach CS1.

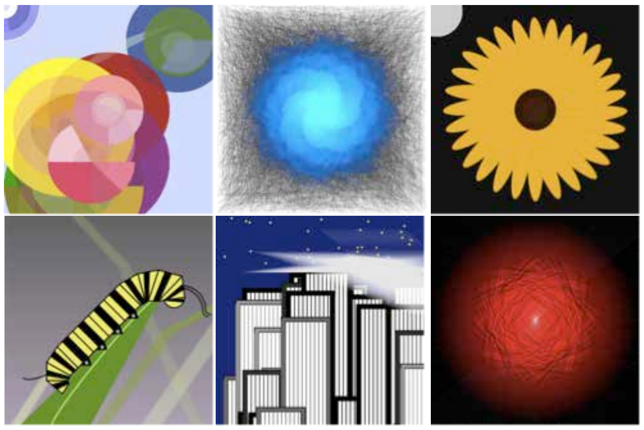


Figure 1: Student artwork in Bryn Mawr and SMU classes

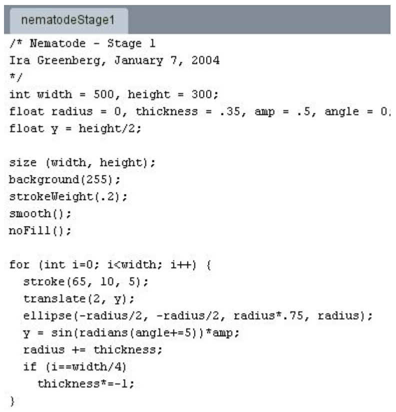
1.1 Related Work

There have been various attempts at incorporating graphics and creativity in introductory computing courses, one of the earliest being Niguidula & van Dam [N&vD 1987]. Perhaps the most successful efforts to date are those of Guzdial using media computation as a context [Guzdial 2004, 2009, Guzdial & Ericsson 2006] and using Alice to introduce non-majors to computing [Cooper et al 2003, Moskal et al 2004]. These approaches have since been expanded into CS1 courses with documented success [Balter & Bailey 2010]. Other notable efforts in this arena include a course taught by Prof. Ursula Wolz (at The College of New jersey) on *Introduction to Interactive Multimedia*, the Artbotics project which also uses robotics to engage students in creating creative artifacts [Yanco et al 2007] and the *Computational Thinking* course at Colby College taught by Professor Bruce Maxwell, which uses Python and Turtle Graphics for 2D graphics as a medium of creativity, expression, communication and experimentation. Despite its appeal, the concept of generative art and creative computing are relatively underutilized in creating introductory computing curricula.

Besides Processing there are several other projects that are focused on creating design and art using computational techniques [VVVV, PD, Zimmer 2009, Scratch, Resnick 2007a, 2007b, Maloney et al 2008, Monroy-Hemandez & Resnick 2008, Panda3D, Arduino, Wiring]. Most of these projects either build on the context of robotics and creativity, or are still in early development stages (still in alpha-releases). Their uses in formal computing education are mostly localized to the development groups and their institutions or their immediate communities (not in Computer Science). However, they represent an exciting direction for bringing computing to a much larger community of students and practitioners.

2. CREATIVE CODING & PROCESSING

The concept of creative coding offers a different conceptual lens to the task of programming and computing. Our approach is firmly grounded in the innovative and rigorous explorations of the *Aesthetics + Computation Group*, headed by Prof. John Maeda, at the MIT Media Lab [Maeda 1999, 2004]. Maeda's group explored computation, including pedagogy, from the context of the arts classroom. We believe that the lessons learned and ultimately tools that have been created have a very direct bearing on the discussion of CS pedagogy and ultimately enrollment and retention. John Maeda is both a formally trained artist and computer scientist, who pioneered an approach to "Creative coding" that radically *recontextualized* computer code–from an applied math notation to a creative medium, on par with charcoal, paint, clay, etc. Lessons learned from these explorations led to the design of the *Processing* language [Reas & Fry 2006, 2007].



```
/* Nematode - Stage 1
Ira Greenberg, January 7, 2004
*/
int width = 500, height = 300;
float radius = 0, thickness = .35, amp = .5, angle = 0;
float y = height/2;

size (width, height);
background(255);
strokeWeight(.2);
smooth();
noFill();

for (int i=0; i<width; i++) {
  stroke(65, 10, 5);
  translate(2, y);
  ellipse(-radius/2, -radius/2, radius*.75, radius);
  y = sin(radians(angle+=5))*amp;
  radius += thickness;
  if (i==width/4)
    thickness*=-1;
}
```

Figure 2: Nematode program

Processing is a robust and full-featured language that has use both in the classroom and beyond; it is used widely in industry and growing quickly in popularity. Factors contributing to its growing popularity include: it is integrated into Java; it is stable; it has a very flat learning curve; it has a simple, intuitive, and easy-to-use IDE; it is fun to work with; it is open source and easily extendable; and runs on all popular computing platforms.

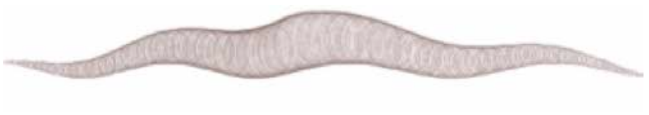


Figure 3: Nematode Sketch (see Figure 2 for program listing)

Processing was designed for the construction of 2D and 3D visual forms. Its IDE is light-weight, but well-suited for the kind of rapid proto-typing needed for dynamic visual work. Novice programmers respond well to programming environments where small snippets of code can be quickly tested and minimal effort is needed to run code. Despite the ease with which beginners take to Processing, it is a full featured programming language capable of rendering stunning graphics and animations, ones that rival the capabilities of OpenGL and other standard Graphics libraries, with little learning curve and much less actual code. For example, Figure 2 shows a complete Processing program [Greenberg 2007] which generated the sketch in Figure 3. It comes as no surprise that Processing is being embraced by academia–from departments of Art and Design and Architecture; to the Sciences, for visualization; to computer science departments, to teach CS1. In addition, Processing is built on top of Java, but uses a simplified syntax and graphics programming model. It is fully integrated in that straight Java code can be embedded freely in any Processing program/sketch, and every Processing program/sketch can be exported to a Java applet as well as a Java application for Linux, Mac and Windows.

3. COURSE DESIGN

Our pilot offering consisted of the following topics, in the order of presentation during the semester:

Course Introduction: What is computing? Algorithms, programming. What is creative computing? Creative computing examples.

Drawing Primitives: point, line, shapes, color, curves, text, images, 3D objects.

Interactivity and Simulation: I/O, mouse and keyboard events, animation and gaming, simple physics.

Control Structures: Syntax, variables and data types, expressions, conditionals, loops.

Functions: Procedural abstraction, modularity, parameters, return values.

Mathematical Concepts: Coordinate systems, polar-coordinates, basic trigonometry and geometry.

Arrays & Objects: Introduction to reference types; arrays, ArrayLists and indexing; introduction to OOP, including encapsulation, inheritance and polymorphism.

Creative Coding Concepts: Transformations, including translate, rotate and scale; iteration and randomization; algorithmic drawing; introduction to image processing.

Text, Data & Visualization: Strings, displaying text, fonts; file I/O; introduction to data mining; acquiring, parsing, filtering and cleaning data; visualization design.

Each topic was structured around 1-2 weeks of lectures and laboratory exercises followed by a creative coding assignment. Figure 1 shows some example student work. Over the course of a semester students built a visual portfolio of their work, exhibited it in a public website (www.openprocessing.org) where they could also watch and interact with the work of others, and learned the core concepts in computing. A larger collaborative project (virtual fish tank) between the students of the two partner institutions Bryn Mawr College and SMU was given mid-semester. Instructors provided an interface (the fish tank) where students shared code (fish class, physics class, etc) and learned team programming. The course culminated in a final design project where students chose, designed, and built a data visualization artifact of their own interest. These design projects ranged from visualizations of box office movie earnings, weather data, restaurant reviews (Zagat), aural data (music), demographics, etc. Some students also built games and animations.

As we continue to develop the structure of such a course we are examining ways in which we can reorganize the course content, both from design and computing perspectives. We are currently engaged in creating a set of structured instructional materials that reflect our course design. These materials will be tested in the coming year and also made available for wider adoption.

3.1 Assessment

Assessment of the students included traditional instruments such as quizzes, exams and presentations and also critiques. The critique process comes out of the arts classroom, where students and the instructor openly discuss individual projects. In the arts classroom, student work is assessed primarily through a formal analysis of design and aesthetic principles. In the Processing CS1 classroom, discussions included both technical aspects of the projects, such as a review of the source code, and also basic aesthetic issues. The aesthetic issues, as listed in any introductory design book [Rowena & Hannah 2002], included factors such as symmetry, repetition, contrast, balance, focal point, rhythm, emphasis, movement, pattern, variety, unity, etc.

One of the main benefits of utilizing the critique process was increased student engagement. Students were consistently motivated to improve their programs following the critique. This invariably led to a rethinking of both technical and aesthetic factors. In addition, students were motivated to auto-didactically learn beyond the course curriculum to extend their classroom projects.

3.2 Statistics and Early Results

In the fall semester of 2010, two CS1 classes were offered, one at each partner institution involving a total of 39 students (23 at Bryn Mawr and 16 at SMU). Both classes were capped and pre-registration numbers far exceeded available seats (closed at 45 when the cap was 23) at Bryn Mawr and lotteries were conducted. In the spring semester of 2011, Bryn Mawr offered two sections (each capped at 23) of the Processing-based CS1 due to demand, again both sections required lotteries. Bryn Mawr was not able to offer more sections due to staffing constraints.

The success of the Processing-based CS1 has led SMU to start offering an entire curriculum, including a three-course introductory sequence starting with a Processing-based CS1 and a new minor and major in creative computing. A new graduate program is also being built.

4. EVALUATION

Survey results from our pilot classes show that our approach is successful and appears particularly appealing to women. Note the enrollment trends (quoted in section 3.2) and survey results for Bryn Mawr, an all-women's institution.

4.1 Cross-institutional Comparisons

Identical surveys were collected from pilot classes at both Bryn Mawr College and SMU, which are very different institutions. Bryn Mawr is a small all-women's liberal arts college with 1,300 undergraduate students while SMU offers a more conventional CS and Engineering program with a much larger student body (11,000 total with 6,000 undergraduates). SMU also offers an Introduction to CS for Non-majors (CS0) while Bryn Mawr does not, making SMU's CS1 less likely to attract non-majors, particularly non-STEM majors. Bryn Mawr's CS1 tends to be 90% non-majors or undecided.

In addition, near identical surveys were also given to traditional Java-based CS1 sections running concurrently at SMU as a control, with two art and/or Processing specific questions taken out. A total of 21 Bryn Mawr Processing CS1 students, 11 SMU Processing CS1 students and 39 SMU non-Processing Java-based CS1 students returned the surveys for the Fall 2010 semester. We had additional data collected from the Spring 2011 sections, but we did not have sufficient time to perform the data analysis in time for this paper. We also did not calculate Chi-square because these results are early and we have a relatively small sample size.

From the Fall 2010 data we observe the following:

- Students in the Processing sections appear more positively inclined to take additional CS courses. 42.86% at Bryn Mawr and 45.45% at SMU said yes to another CS class versus 28.21% non-Processing students at SMU (see Figure 6). We would like to point out the significance of the 42.86% at Bryn Mawr, where the majority of students in the CS1 are non-majors taking the class to fulfill the quantitative requirement of the college and thus the 'yes' is unlikely to be influenced by required course sequences in the major or minor. This is borne out by the answers to whether they expect to ever have to write another program in any language after this class. 61.9% of the Bryn Mawr students said no (see Figure 8). Also interesting were the answers to why they took this class. 76.77% of Bryn Mawr students said they were just checking to find out what CS is all about, versus 27.27% at SMU, which again clearly indicates SMU required course sequences.
- Students in the Processing sections are more likely to spend extra time on a homework assignment for "fun". 85.71% of Bryn Mawr students and 72.72% of SMU students reported spending extra time on at least one homework because it was cool, versus 30.77% non-Processing students at SMU (see Figure 4).
- Students in the Processing sections indicate strong tendency to talk to friends not in the class about the class. 80.96% at Bryn Mawr and 65.64% at SMU agreed or strongly agreed.
- Students in the Processing sections at both institutions disagree that CS and programming are the same thing. 52.38% at Bryn Mawr and 54.55% at SMU disagreed or strongly disagreed.
- SMU students exhibit stronger confidence in knowledge and technical knowhow (compared to peer in class) than Bryn Mawr students. Although Bryn Mawr's distribution approximates the normal curve, possibly influenced by the all-

female student body and is likely also a more accurate depiction of relative knowledge of peers.

- SMU students also exhibit stronger confidence in math and science abilities, which is partly explained by the higher number of intended CS majors in the class. It may also have a strong correlation to gender rather than actual ability.
- Students at both institutions are quite positive about technology and trying new tools/products, with the SMU students indicating stronger tendencies.

4.2 Comparing with Another Contextualized Approach (Robots)

The Computer Science Department at Bryn Mawr College was part of the IPRE institution [Blank 2006, Kumar 2008, Summet et al 2009] that developed the personal robots approach to CS1. We have various data collected on other contextualized approaches from Bryn Mawr and Georgia Tech during the robot-based CS1 endeavors and this forms the basis for comparison. Different surveys were given out as the IPRE project evolved and they did not all contain questions comparable to our surveys. In this particular paper, we are comparing to survey data collected in the Spring 2007 (22 returned surveys) [IPRE 2007] and Spring 2008 (24 returned surveys) classes of CS1 with robots taught at Bryn Mawr, where equivalent or near equivalent questions were asked. Those were the first and second years of the IPRE project.

We would like to point out that this comparison is not intended to show which contextualized approach is better or worse. While the authors are reassured that the data supported that the new art/creative coding context is doing at least as well as another more established context, the important point is that bringing context, any context to CS1 makes a dramatic difference for student motivation and engagement on all levels, and we direct the readers' attention to the contrast of both contextualized approaches with the traditional approach.

The positive effects on student motivation demonstrated by other contextualized approaches continued consistently in our art/creative coding-based approach. Similar to results obtained from CS1 with robots, students showed strong tendency to spend extra time on assignments because they enjoyed it (see Figure 4).

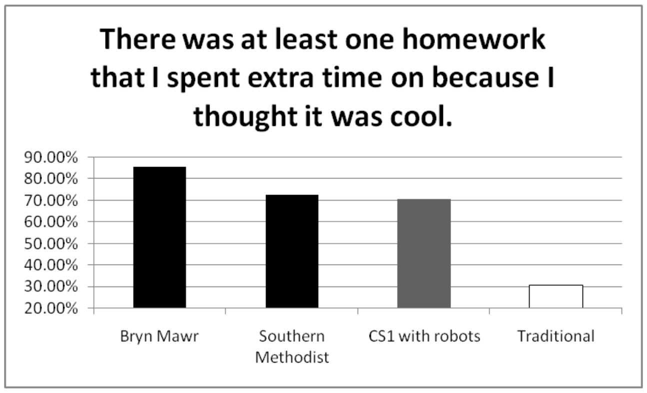


Figure 4: % of students responding Agree or Strongly Agree

As shown in Figure 5, students responded very favorably to the context of art and more found it appealing than robots. Note in Figure 4, the third data point was for the question "I enjoyed using the robot in class". We do not have a comparison data point for the traditional approach as this is a context-specific question and was taken out in the surveys given to the traditional sections.

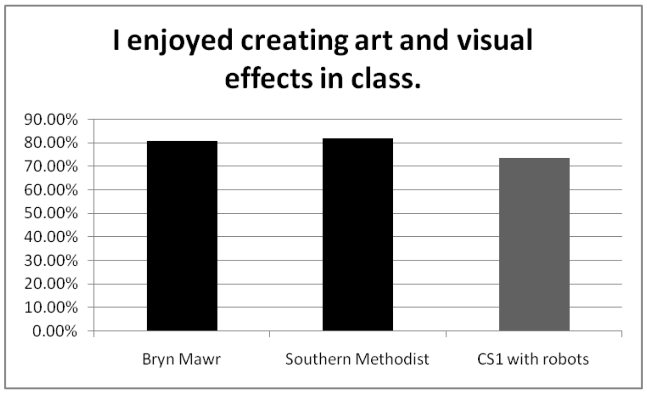


Figure 5: % of students responding Agree or Strongly Agree

Figure 6 shows that the art/creative coding context was very effective in initiating student interest and influencing students to consider taking a further course in Computer Science.

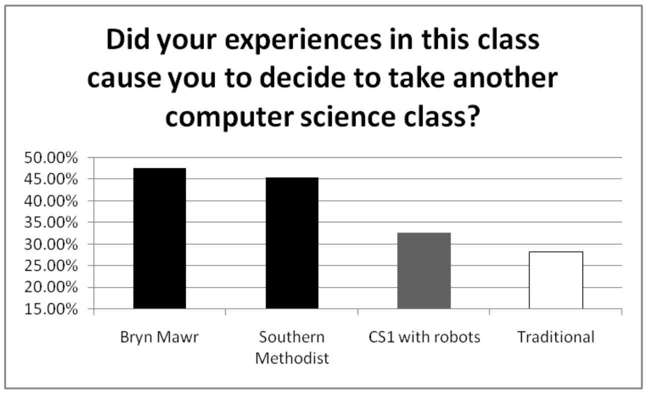


Figure 6: % of students responding Yes

Lastly, Figure 7 shows that 42.86% Bryn Mawr students indicated that they wrote additional Processing programs not assigned for this class, compared to 37.5% of students saying so for the robot approach. The SMU responses were comparable to the robot approach, at 36.36%. 28.21% of the traditional section responded Yes.

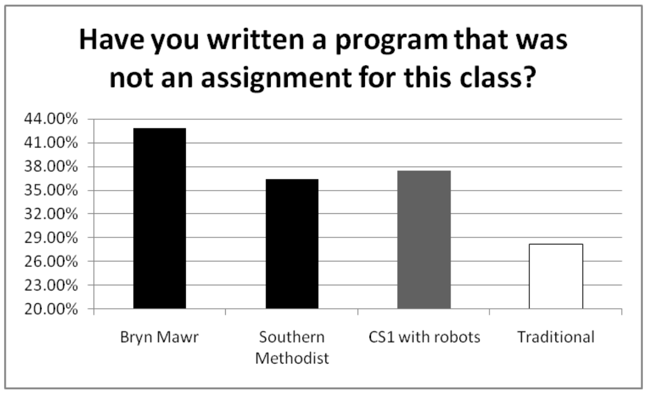


Figure 7: % of students responding Yes

Correlated with the responses to the question shown in Figure 8, we would like to point out that 61.9% of the Bryn Mawr students did not expect that they would ever have to write another program in any language after this class (likely non-STEM majors), yet the same Bryn Mawr students also showed the most interest in creating self-initiated programming projects outside of class. We believe this is a particularly encouraging sign that the art and creative coding context attracts and motivates women.

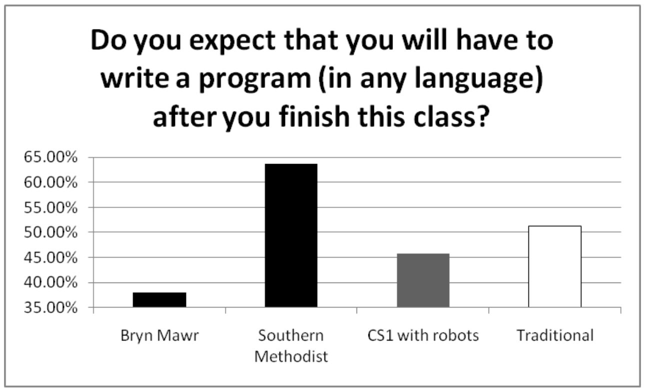


Figure 8: % of students responding Yes

5. FUTURE WORK

We continue to offer Processing-based CS1 in the 2011-2012 academic year at both institutions, using the opportunities to refine our course design and accumulate more course materials and student feedback. As we collect more survey data and build a more significant sample size, we will run more statistical analysis and look at cross-institutional differences as well as gender differences more closely. We will be conducting longitudinal studies to look at retention and enrollment numbers in CS2 and beyond. We are very excited about the development of the creative computing curriculum at SMU, which to our knowledge is the first of its kind nationwide.

As we continue to develop more course materials, they will be made freely available via our website. We are also writing a textbook for CS1 that showcases our approach and we are planning to wrap it up in time for the Spring 2012 classes. We hope that our dissemination efforts will attract signification adoption in peer institutions.

ACKNOWLEDGMENTS

The authors would like to thank Douglas Blank, Steve Cooper, Jennifer Burg, Eric Eaton, Ben Fry, Mark Guzdial, Darrel Ince, James Kiper, John Lee, Bruce Maxwell, Suku Nair, Jeffrey Nyhoff, Mark Russo, Daniel Shiffman, Allison Tew, Eugene Wallingford and Ursula Wolz for their support, encouragement, and assistance in several phases of this project. This project is supported in part by the National Science Foundation award DUE-0942626, Bryn Mawr College and Southern Methodist University and National Science Foundation award CCF-0939370.

6. REFERENCES

[1] Arduino. Web site: www.arduino.cc/

[2] Balter & Bailey 2010. Olle Balter and Duane Bailey. Enjoying Python, processing, and Java in CS1. *ACM Inroads, V(1)No.4*. ACM Press, December 2010.

[3] Bayless & Stout 2006. Jessica D. Bayless and Sean Strout. Games as a “Flavor” of CS1. In *proceedings of SIGCSE 2006*. ACM Press, 2006.

[4] Beck et al 2011. Robert E. Beck, Jennifer Burg, Jesse M. Heines, and Bill Manaris. Computing and Music: A Spectrum of Sound. *Special Session, SIGCSE 2011*. Dallas, TX, March 2011.

[5] Blank & Kumar 2003. Douglas Blank and Deepak Kumar. Patterns of Curriculum Design. In Cassel & Reis (editors), *Informatics Curricula and Teaching Methods*. Kluwer Academic Publishers/IFIP, 2003.

[6] Blank 2006. Douglas Blank. Robots make computer science personal. *Communications of the ACM*, 49(12), December 2006.

[7] Cooper et al 2003. Stephen Cooper, Wanda Dann, Randy Pausch. Teaching Objects-first in Introductory Computer Science. In *Proceedings of SIGCSE 2003*. ACM Press 2003.

[8] Greenberg 2007. Ira Greenberg. *Processing: Creative Coding and Computational Art*. Friends of Ed, 2007.

[9] Guzdial 2004. Mark Guzdial. *Introduction to computing and programming with Python: A Multimedia Approach*. Prentice-Hall, 2004.

[10] Guzdial & Ericsson 2006. Mark Guzdial and Barbara Ericsson. *Introduction to computing and programming with Java: A Multimedia Approach*. Prentice-Hall, 2006.

[11] Guzdial 2009. Mark Guzdial. Teaching Computing to Everyone. *Communications of the ACM (CACM)* 52(5):31-33. ACM Press, May 2009.

[12] Guzdial 2010. Mark Guzdial. Does Contextualized Computing Education Help?. *ACM Inroads, V(1)No.4*. ACM Press, December 2010.

[13] Hillberg & Meiselwitz 2008. J. Scott Hilberg, Gabriele Meiselwitz, Undergraduate fluency with information and communication technology: perceptions and reality, In *Proceedings of the 9th ACM SIGITE conference on Information technology education*, October 16-18, 2008, Cincinnati, OH, USA

[14] IPRE 2007. Institute for Personal Robots in Education. *IPRE 2007 Annual Report*. Institute for Personal Robots in Education (IPRE), 2007.

[15] Kay 2011. Jennifer Kay. Contextualized Approaches to Introductory Computer Science: The Key to Making Computer Science Relevant or Simply Bait and Switch?. In *Proceedings of SIGCSE 2011*. ACM Press, 2011.

[16] Kumar 2008. Deepak Kumar (editor). *Learning Computing With Robots*. Institute for Personal Robots in Education (IPRE), 2008.

[17] Maeda 1999. John Maeda. *Design By Numbers*. The MIT Press, 1999.

[18] Maeda 2004. John Maeda. *Creative Code*. Thames & Hudson Press, 2004

[19] Maloney et al 2008. Maloney, J., Peppler, K., Kafai, Y., Resnick, M., and Rusk, N. (2008). Programming by Choice: Urban Youth Learning Programming with Scratch. In *Proceedings of SIGCSE 2008*. ACM Press 2008

[20] Monroy-Hemandez & Resnick 2008. Monroy-Hernández, A. and Resnick, M. (2008). *Empowering kids to create and share programmable media*. Interactions, March-April 2008.

[21] Moskal et al 2004. Barb Moskal, Deborah Lurie, Stephen Cooper. Evaluating the Effectiveness of a New Instructional Approach. In *Proceedings of SIGCSE 2004*. ACM Press, 2004.

[22] Niguidula & van Dam 1987: David A. Niguidula and Andries van Dam. *Pascal on the Macintosh: A Graphical Approach*. Addison Wesley, 1987.

[23] Panda3D. Web Site: www.panda3d.org.

[24] PD. Pure Data. Web site: www.puredata.info.

[25] Processing Group. Main portal for all things Processing. Web site: www.processing.org.

[26] Resnick, M. (2007a). All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten. In *Proceedings of the SIGCHI Conference on Creativity and Cognition*, Washington, D.C.

[27] Resnick, M. (2007b). Sowing the Seeds for a More Creative Society. In *Learning and Leading with Technology*, 2007.

[28] Reas & Fry 2006. Casey Reas and Ben Fry. Processing Code: Programming within the Context of Visual Art and Design. In *Aesthetic Computing*. Paul A. Fishwick (editor). The MIT Press, 2006.

[29] Reas & Fry 2007. Casey Reas and Ben Fry. *Processing: A Programming Handbook For Visual Designers and Artists*. The MIT Press, 2007.

[30] Rowena & Hannah 2002. Rowena Kostellow and Gail Hannah. *Elements of Design and the Structure of Visual Relationships*. Princeton Architectural Press, 2002.

[31] Scratch. Web Site: www.scratch.org.

[32] Shiffman 2008. Daniel Shiffman. *Learning Processing: A Beginner's Guide toProgramming Images, Animation, and Interaction*. Morgan Kauffman Publishers, 2008.

[33] Summet et al 2009. Jay Summet, Deepak Kumar, Keith O'Hara, Daniel Walker, Lijun Ni, Doug Blank, Tucker Balch. Personalizing CS1 with Robots. In *Proceedings of ACM SIGCSE 2009*. March 2009.

[34] VVVV. A Multimedia Toolkit. Web site: www.vvvv.org.

[35] Wiring. Web site: www.wiring.org.co/

[36] Yanco et al. Holly A. Yanco, Hyun Ju Kim, Fred G. Martin and Linda Silka . *Artbotics:* Combining Art and Robotics to Broaden Participation in Computing. *2007 Workshop on Research in Robots for Education at the Robotics Science and Systems*. 2007.

[37] Yardi & Bruckman 2007. Sarita Yardi and Amy Bruckman.What is computing? bridging the gap between teenagers' perceptions and graduate students' experiences. In *Proceedings of the Third international Workshop on Computing Education Research* (Atlanta, Georgia, USA, September 15 - 16, 2007). ICER '07.

[38] Zimmer 2009. Frank Zimmer. *loadbang - Programming Electronic Music in Pd*. Wolke Publishing House, 2009.