The International JOURNAL of LEARNING

Volume 15, Number 2

Performamatics: Experiences with Connecting a Computer Science Course to a Design Arts Course

Jesse M. Heines, Jim Jeffers and Sarah Kuhn

COMMON GROUND

www.learning-journal.com

THE INTERNATIONAL JOURNAL OF LEARNING
http://www.Learning-Journal.com

First published in 2008 in Melbourne, Australia by Common Ground Publishing Pty Ltd
www.CommonGroundPublishing.com.

ISSN: 1447-9494
Publisher Site: http://www.Learning-Journal.com

THE INTERNATIONAL JOURNAL OF LEARNING is a peer refereed journal. Full papers submitted for publication are refereed by Associate Editors through anonymous referee processes.

Typeset in Common Ground Markup Language using CGCreator multichannel typesetting system
http://www.CommonGroundSoftware.com.

Performamatics: Experiences with Connecting a Computer Science Course to a Design Arts Course

Jesse M. Heines, University of Massachusetts Lowell, MA, UNITED STATES
Jim Jeffers, University of Massachusetts Lowell, MA, UNITED STATES
Sarah Kuhn, University of Massachusetts Lowell, MA, UNITED STATES

Abstract: Our work is based on a partnership between the a Computer Science (CS) and Art, Music, and English departments in the area of exhibition and performance technologies. We define these areas broadly to encompass all CS applications in the creative and performing arts. These areas not only resonate with today's media-rich culture, but reinforce the fact that virtually all computer applications now require the integration of creative elements. CS majors must learn to work with specialists in areas where the perspective is often quite different from their own. We believe that computer scientists have much to learn from those trained in the arts and vice versa. The common thread in performamatics projects is that many tasks, performed by multiple people, must come together on a tight schedule by a specific date to achieve a desired result. Performamatics also implies that each team member must "perform" his or her task(s) in a way that can be integrated into a final product, regardless of whether that team member participates visibly in the culminating event. Our paper reports on initial attempts to couple CS courses and integrate CS elements with courses in Art, Music, and Theater. We describe the techniques we used that were designed to increase the scope and level of creativity in student projects and the impact these techniques and the presence of interdisciplinary teams had on those projects. We discuss changes we will make to improve the experience for both groups of students in the future and suggest new techniques we may try to better achieve our goals. This work is supported by NSF Award No. CNS-0722161. Principal Investigator: Jesse Heines. Co-Principal Investigators: Fred Martin, Gena Greher, Jim Jeffers, and Karen Roehr. Senior Personnel: Sarah Kuhn and Nancy Selleck. Further information is available at: www.performatics.org.

Keywords: Interdisciplinary Programs, Computer Science and the Arts, Performamatics

A Common Problem

RECENT DECLINES IN computer science (CS) enrollments are well documented in both the U.S. [2, 9, 12] and abroad [3, 15]. Many reasons are cited, including a "negative view of the CS profession by pre-college students, especially females" [14], "media portrayals of computing as stodgy and nerdy compared to other fields" [6], and "the image that computer science is a field of programmers" [5]. Few CS programs enjoy the enrollments they had during the dot-com boom, a fact that concerns business and industry leaders as well as academics [13].

How We're Trying to Address It

One way to address this problem is to work harder to retain students who have already chosen to major in CS, especially women [4, 8]. Toward this end, we are introducing innovations into our curriculum to give students more flexibility in course selection, especially in the freshman and sophomore years. This approach follows the lead of Downey's and Stein's "small footprint curriculum" [7, 17] and Georgia Tech's "Threads" program [10, 11]. The trick is to introduce curriculum changes without putting our accreditation at risk. Fortunately, new standards recently approved by the ABET Computing Accreditation Commission [1] make it feasible to do so. We want to encourage CS students to explore application areas earlier in their academic careers and to move through the program along various "tracks," such as graphics, robotics, and user interfaces.

The Performamatics Concept

One of the new tracks we're developing is based on a partnership between the CS and Art, Music, and English departments in the area of exhibition and performance technologies. We define these areas broadly to encompass all CS applications in the creative and performing arts. These areas not only resonate with today's media-rich culture, but reinforce the fact that virtually all computer applications now require the integration of creative elements. CS majors must learn to work with specialists in areas where the perspective is often quite different from their own, and we believe that computer scientists have much to learn from those trained in the arts and vice versa.

Planning for an event such as an exhibition or performance gives very sharp focus to one's work. We interpret these types of activities in a very broad sense and group them under the heading of "performamatics," building on the "artbotics" concept pioneered by Yanco *et al*. [18].

The common thread in performamatics projects is that many tasks, performed by multiple people, must come together on a tight schedule by a specific date to achieve a desired result. Performamatics also implies that each team member must "perform" his or her task(s) in a way that can be integrated into a final product, regardless of whether that team member participates visibly in the culminating event.

We believe that focusing CS assignments around a theme and making those assignments part of a larger project, doing the assignments in conjunction with arts and humanities majors, and having those assignments result in a broadly interpreted exhibition or performance can invigorate traditional CS programs such as ours and both attract and help retain CS majors.

Adding Art Students' Designs to GUI Programming Students' Projects

As a first step toward developing the performamatics concept, we attempted to design a project for the Fall 2007 semester that would capture the interests of both CS and Art/Design students while fitting within the structures of the existing GUI Programming and Web Art & Design courses we were already scheduled to teach. This resulted in the following plan for an Extended Art Media Performamatics (eAMP) project to be developed during the semester.

- CS students would cover the principles of object-oriented programming (OOP) required to create a GUI application for assembling visual data such as JPEG files, ordering those files in some way, and disseminating them via the Internet to e-mail addresses. Ideally, the images would be formatted for small devices such as cell phones or PDAs, and the application would exhibit a user-friendly interface using drag-and-drop modality.
- Art/Design students with advanced imaging skills and understanding of web-based design would generate the raw content and overall visual concept for the project. They would use their knowledge of applications such as Photoshop, Dreamweaver, and Flash to make visual models, aiding the team to find not only a practical CS solution, but also a stylish and visually compelling art piece.

The professors then worked independently to incorporate this plan into their respective courses. They brought their students together formally a number of times, but each course remained autonomous.

Effect on GUI Programming Course Structure

The CS course participating in this effort was GUI (Graphical User Interface) Programming I. (The course website is publicly available at teaching.cs.uml.edu/~heines/91.461.) This is the first course in a two-semester senior capstone project sequence that focuses on building GUIs and understanding the human factors of user interfaces. The course is taught using Java, and the majority of our CS courses are taught using C and C++. Therefore, in the past, this first course mostly taught the underlying skills needed to do a project in the second semester: getting students up to speed on the Java language and Swing toolkit, reviewing and applying OOP concepts, learning to read an API, and implementing progressively complex GUIs specified by the instructor. The assignments had only a loose relation to one another, but students usually started each program from scratch, only occasionally including code from one assignment in another.

With the focus of the performamatics project, however, the course was structured specifically to give students the skills needed to implement all required facets of the target application. Each assignment built on the preceding one, using almost all the previously developed code and extending it to provide new functionality and features and enhanced human factors considerations. The titles of the semester's assignments are listed below. (The full text of these assignments and supporting "startup" code are publicly available on the course website referenced above.)

1. Using Text Fields, Buttons, and Labels
2. Populating List Boxes
3. Creating an E-Mail Repository
4. Using a Custom Cell Renderer
5. Implementing Drag-and-Drop
6. Usability Report
7. Final Project Submission

This structure gave the course an entirely different feel from that of previous semesters. Since students knew where the course was going, class interactions had much less of an instructor-centered nature. Not only did student questions drive classroom discussions, but the professor's choice of lecture topics was strongly influenced by the problems that needed to be solved to implement the application. This was completely opposite from the approach used in the past, which centered around specific GUI components and examples of their use.

Effect on Art/Design Course Structure

The Art course in this pairing was Web Art & Design II. (See its public website at classroom.uml.edu /art /webart0201/.) This course builds on knowledge (gained in Web Art & Design I) of intermediate HTML, JavaScript, and CSS. The course uses Adobe Dreamweaver as a design interface, with more complex animation and interactivity provided by Adobe Flash. The students in this course have completed a six-course art/design foundations program and have taken several other computer art and/or design courses.

The schedule of Web Art & Design II was minimally impacted by the addition of performamatics content, as CS students were able to come to the regularly scheduled art classes. (These classes meet twice a week for almost three hours, making it nearly impossible to reschedule without overlapping other classes.) Enrollment was five students. This actually created a good "design team" environment by putting the designers in high demand by the CS students. However, the class dynamic would have been better in a larger section. This would have allowed more peer-to-peer learning among the art students and spread out the work that needed to be done by each art/design student to keep the CS students on schedule. The joint eAMP project was added as another assignment to an already full semester of projects. In retrospect, it perhaps should have been emphasized in terms of being a project for a professional design portfolio from the start of the course. Art/design students with the clearest and most frequent communication with their CS counterparts seemed to have the best experience.

What We've Learned

Despite some logistical rough spots, we judge this first iteration to have been a very real success, and we will be continuing our collaboration in future semesters. Students and faculty from both CS and Art all gave feedback that was on balance very positive, with the most enthusiastic endorsement coming from the CS professor, and the most measured responses from the art students, who felt the collaboration had value but was problematic.

Rough Spots

Several issues arose because we received our funding too late to create a new course, or to reschedule existing courses, during the first semester of the project. As a result, the meeting times of the two courses had no overlap, and expectations for student work and collaboration were not always well aligned between the two courses. The art course had unusually low enrollment this particular semester, which also created some collaboration problems. However, we expect that these problems will not continue with future iterations of the course.

Other difficulties are less easily solved. Our university is divided into two main campuses ("North" and "South") separated by a river. Though only 1.2 miles apart, getting from one to the other can be time-consuming. To further complicate things, the computer science course met on North on a Monday-Wednesday-Friday schedule, while the art course met on South on a Tuesday-Thursday schedule. Class meeting days can be changed in the future, but changes will interfere with the customary scheduling practices of either CS or Art.

Difficulties That Are Also Opportunities

One issue involved software and hardware. Art coursework was done in Flash, while CS programs were written in Java. In addition, the Art students worked under MacOS, while CS students worked under Windows. These seemingly deal-breaking incompatibilities turned out to be a huge benefit. Overcoming them not only significantly broadened both sets of students' understanding about software, but also forced them to view design and implementation from a higher perspective and fostered fascinating and highly creative approaches to implementing sophisticated user interfaces (see "Impact on Programming Projects").

Another issue was one of perception and style, and this one involved the professors as well as the students. CS teaching tends to be highly structured, while studio art teaching is far less so.

To the CS professor, the studio art classroom environment was chaotic, although he quickly became aware of the creative energy that this environment brought out of the CS students as they discussed designs with their art student partners (see Figure 1) and struggled to figure out how they could possibly implement some of the design elements dreamed up by their counterparts (see Figures 2).

Figure 1: Art/CS Design Collaboration

Figure 2: Design Dilemma for a CS Student

Impact on Programming Projects

The CS students' GUI programming projects were far more creative than in past. As shown in Figures 3 and 4, the art students' influences are clearly evident in the CS projects. Integration of the graphic elements challenged CS students to fully comprehend advanced GUI concepts such as custom cell renderers, layered panes, and non-rectangular buttons, as well as the full capabilities of lists, trees, and tables. In previous years, many of these concepts weren't reached until the second semester, if at all. Thus, the course was significantly accelerated. The more interesting designs and faster pace forced students to explore more topics than can be covered in class and to dig deeper into the Java API on their own, resulting in more lively class discussions and open-ended questions.

Figure 3: CS Implementation of an Art Student's Design with a Scrolling Image Interface that Allows Images to be Dragged-and-Dropped for Sending via E-Mail

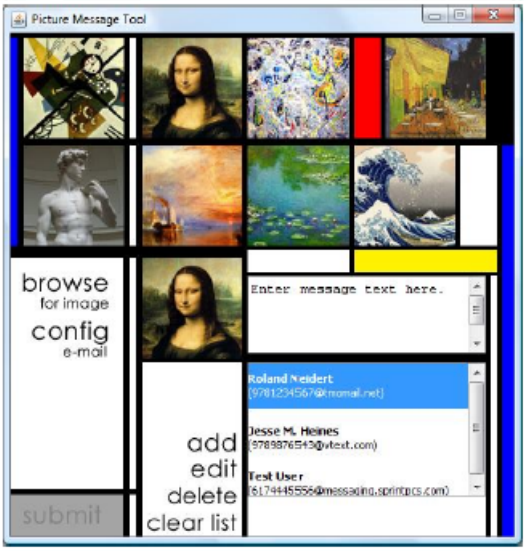

Figure 4: CS Implementation of an Art Student's Design Inspired by Piet Mondrian (1872-1944)

CS students were exposed to the difficulty of coordinating project tasks with students in a different department with different priorities. Differences in style and expectation were amplified by the fact that the art students had a different professor and their work is evaluated on different criteria. This situation is common in today's industrial software development environments and one that is well worth experiencing in college. It fostered interesting class discussions as we wrestled with how to address issues introduced by interdisciplinary work.

Impact on CS Students

In a focus group meeting with the project evaluator, a social science faculty member at the university, the CS students acknowledged the difficulties of collaboration across disciplinary and geographic boundaries, but expressed strong enthusiasm for their overall course experience. The greatest positive impact came from organizing the course as a whole around a single, open-ended project. The students felt that they gained valuable "real world" experience by doing a substantial project that they could show to others, that used all (or most) of the skills taught in the course, and that required them to collaborate with others who did not share their disciplinary perspective and had somewhat different values and habits.

The students contrasted this with their experiences in some other classes, where they worked individually on the same problem sets, and where there were right and wrong answers rather than open-ended challenges. They found the freedom and variety of the performamatics project extremely motivating,

and many consequences followed from their enhanced motivation and engagement.

Students said that they became more self-directed, finding new tools and code on the web. They liked the fact that they were able to use the code of others with attribution, of course and that each student was doing something different. They learned from looking at the other students' projects, and from showing their own work to other students.

They experienced the course as coherent, rather than as a sequence of unrelated assignments. "Everything had a reason [for being in the course]" said one student. Some students wanted to be able to do additional things not taught in the course, and in some cases spent extra time finding outside resources and incorporating them into their projects, teaching them to be self-reliant and entrepreneurial.

Because the student projects resulted in "something we could show, not just code," students were pleased that they could share their work with outsiders. One of the few improvements they suggested for future iterations was to have a have a page on the course website where projects could reside and be publicly available.

The CS faculty member teaching the course judged the student work to be "light years ahead of what has been done [by students in this class] in the past, absolutely light years, as far as visual appeal, and also in terms of coding sophistication."

Impact on Web Design Projects

The Web Art & Design class had only five students, making it difficult to judge what impact performamatics may have had on their projects. However, three of the five were fully engaged with the course material and made advances in their understanding of web design, and some of the performamatics projects looked really great from a design perspective. Some of these projects will surely be included in the art/design students' professional design portfolios as a result of the collaboration.

In a focus group with the evaluator, the art students made the same point as the CS students: they felt this was a "real world" experience, and therefore extremely valuable preparation for the world of work. Only one student had collaborated with students outside his major; for the rest, this was a new and useful exposure.

Impact on Course Faculty

The most dramatic effect of our pilot project—and an unexpected result—was the impact of this new collaboration on the CS faculty member. To integrate his class with the art class and create a substantial project, in effect he turned his existing course inside-out, so that course timing and content were driven by the needs of the project. He described himself as switching to "just-in-time" teaching, pulling in topics when the students needed them for their work. This created the effect that the students found so motivating—they were learning things because they needed to use them immediately, not just because they had come to the week in the semester that had been set aside for that topic.

Rethinking his course, having students work on their own varied and creative projects, and seeing highly engaged students also engaged the CS professor. He also seemed to enjoy his collaboration with his counterpart, the Art professor. They communicated frequently, both before and during the semester, and the CS professor made frequent visits to the Art professor's studio classroom. "It was tough to get him out," said the Art professor. "I would come back from eating lunch and he would still be there."

As the CS professor observed in a lengthy, open-ended interview with the evaluator, "We're talking about revitalizing computer science, but one thing that's important is to revitalize the computer science *professors*. … I've taught this course many different ways, and this was by far the most exciting and most fun way to teach it. There is absolutely no doubt in my mind about that, and that the students also got the most out of it. … The more enthusiastic the professor, the better the course is going to be, without a doubt. There are plenty of smart professors, but if they're not enthusiastic about what they're doing, the students get bored, too. … To do something like this is *exciting* … *I* enjoyed this semester. *I* enjoyed going to South Campus."

The Art professor was also affected by the collaboration. He reconsidered how he taught students coding, an essential part of their advanced web design work. "[In the future,] I'll teach them to code the way I teach them art. I used to teach it like calisthenics." That is, in the past he assigned programming exercises in the form of small, insignificant, and typically unrelated projects to try to teach students the coding basics by focusing on syntax. In the future, he plans to assign exercises driven by what students need to make their projects function, giving them more reason to explore topics above and beyond those covered in class.

The Art professor also commented on how the opportunity to be part of a National Science Foundation grant was highly unusual for Art faculty and a boon to a field that expects very little in terms of outside funding. He felt that his participation in the grant gave him both credibility and visibility at the university. The long-term effects on the value system of this technical university could be significant, raising the profile of the arts and humanities and establishing their importance to the work of science and technology education and research.

What We're Now Doing Differently

The first change we made for the new semester was to schedule the companion CS course on Tuesdays and Thursdays so that its timing better aligns with the courses taught on South Campus. This may seem like a simple thing to do, but it is counter to the prevailing culture for science and engineering students and causes conflicts with courses in some other departments. Nonetheless, the rescheduling allows us to get students in the different disciplines to meet more often and to work together more closely.

We have also added more joint classes at the beginning of the semester to introduce students in both classes to the entire project simultaneously, rather than having each professor introduce the project to his or her own class. This is intended to "get everyone on the same page" more quickly and help build the relationships necessary for a successful project.

We are also coordinating assignments more closely so that the entire project builds functionality in each class throughout the semester and so that more concrete deadlines are set for code and graphic deliverables. This may involve a joint grade on some part of the project, but doing that requires more thinking about its impact before we are ready to implement it.

Overall, we feel that this first pilot iteration was remarkably successful, and justifies the extra work required to coordinate and overcome scheduling, geographic, and cultural barriers.

Additional Performamatics Initiatives

The effort discussed in this paper is the first of four in UMass Lowell's Performamatics program that build on the success of the earlier Artbotics program [18]. A similar collaboration is taking place between another pair of CS and Art professors to create a course in Tangible Interaction Design.

The Spring 2008 semester pairs a Music course with the second semester of the GUI Programming course, giving us an opportunity to address some of the issues mentioned in the previous section. Also, two professors from English and CS will team teach a Theater History course that includes performamatics elements. This is a "general education" (university core) course on the role of theater in society that is open to all students, and it incorporates web-based research, techniques for archiving that research in web pages, sharing findings and ideas via wikis, creating dynamic bibliographies, and using applications like Word and PowerPoint in ways that take advantage of their real power.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0722161, "CPATH CB: Performamatics: Connecting Computer Science to the Performing, Fine, and Design Arts." *Principal Investigator:* Jesse M. Heines. *Co-Principal Investigators:* Fred G. Martin, Gena Greher, Jim Jeffers, and Karen Roehr. *Senior Personnel:* Sarah Kuhn and Nancy Selleck. Further information is available at: www.performatics.org. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] ABET Computing Accreditation Comm. (2007). *Criteria for Accrediting Computing Programs*. Baltimore, MD.

[2] Campus Technology (2007). *Microsoft: Time to 'Panic' in Search for CompSci Workers*. campustechnology.com/news_article.asp?id=19886&typeid=150 *accessed* January 5, 2007.

[3] Carter, J. (2001). *Arresting the Decline: conversations with female CS undergraduates*. Computing Laboratory, Univ. of Kent at Canterbury. www.cs.kent.ac.uk/pubs/2001/1218 *accessed* January 21, 2007.

[4] Cohoon, J.M. (2001). *Toward improving female retention in the computer science major.* Comm. of the ACM 44(5):108-114.

[5] Denning, P.J. (2004). *The field of programmers myth.* Comm. of the ACM 47(7):15-20.

[6] Denning, P.J. & McGettrick, A. (2005). *Recentering computer science.* Comm. of the ACM 48(11):15-19.

[7] Downey, A.B. & Stein, L.A. (2006). *Designing a small-footprint curriculum in computer science*. Proc. of the 36th ASEE/IEEE Frontiers in Education Conference, pp. M4H-21 to M4H-26. San Diego, CA.

[8] Fisher, A., Margolis, J., & Miller, F. (1997). *Undergraduate women in computer science: experience, motivation and culture.* ACM SIGSCE Bulletin 29(1):106-110.

[9] Foster, A.A. (2005). *Student Interest in Computer Science Plummets.* Chron. of Higher Ed. 51(38):A31.

[10] Georgia Tech College of Computing (2007). *What is Threads?* www.cc.gatech.edu/education/what-is-threads *accessed* November 4, 2007.

[11] Isbell, C. (2007). *Threads: Restructuring a Computing Curriculum*. MIT Seminar. Cambridge, MA.

[12] Klawe, M. & Shneiderman, B. (2005). *Crisis & opportunity in CS.* Comm. of the ACM 48(11):27-28.

[13] Montalbano, E. (2005). *Gates Worries About Decline in U.S. Computer Scientists*. PC World. July 18. www.pcworld.com/printable/article/id,121857/printable.html *accessed* December 7, 2006.

[14] Patterson, D.A. (2005). *Restoring the popularity of computer science.* Comm. of the ACM 48(9):25-28.

[15] Round, A. (2006). *The Decline in Student Applications to Computer Science and IT Degree Courses in UK Universities*. Computing Research Association Conference Proceedings. Snowbird, UT: ACM.
[16] Sandoval, W.A. & Bell, P. (2004). *Design-Based Research Methods for Studying Learning in Context: Introduction.* Educational Psychologist 39(4):199-201.
[17] Stein, L.A. (2006). *A small footprint curriculum for computing: (and why on earth anyone would want such a thing)*. Journal of Computing Sciences in Colleges 21(6):3.
[18] Yanco, H.A., Kim, H.J., Martin, F., & Silka, L. (2007). *Artbotics: Combining Art and Robotics to Broaden Participation in Computing*. Proc. of the AAAI Spring Symposium on Robots & Robot Venues. Stanford Univ, CA.

About the Authors

Dr. Jesse M. Heines
Jesse is an Associate Professor of Computer Science at the University of Massachusetts Lowell. He specializes in the implementation and evaluation of interactive, user-centered programs with rich graphical user interfaces (GUIs), particularly those employing Dynamic HTML, JavaScript, Java, and XML and XSL and their related technologies. Prior to joining the UMass Lowell faculty, Jesse spent ten years with Digital Equipment Corporation, where he founded the Computer-Based Course Development Group and developed a large variety of CBT courseware.

Prof. Jim Jeffers
Jim is an Assistant Professor of Art at the University of Massachusetts Lowell and has been teaching at the college level for over eight years at Rutgers Univ., Seton Hall Univ., Drew Univ. and New York Univ. He has a background in performance, computer art (including video, digital photography, and web art), and conventional static and environmental media. Jim is a practicing artist and exhibits nationally and internationally. He teaches Web Art and Design and has large variety of interests.

Dr. Sarah Kuhn
Sarah is an Associate Professor at the University of Massachusetts Lowell in Regional Economic and Social Development, a multidisciplinary department focused on the economic and social development of the Merrimack Valley region. Sarah's commitment to innovation in learning, particularly blending technical education with the social sciences and arts, recently led her to create the new Laboratory for Interdisciplinary Design at UMass Lowell. She is a member of the Social Science Advisory Board of the NSF-funded National Center for Women in Information Technology, and she was a member of the National Research Council Committee on Workforce Needs in Information Technology. Sarah was Co-PI of Project TechForce, an NSF-funded study of women and men working in the Massachusetts software and Internet industry. She has been a Faculty Associate of the Center for Women and Work since 2004.

The International JOURNAL of LEARNING

COMMON GROUND

Please visit the Journal website at http://www.Learning-Journal.com
for further information about the Journal or to subscribe.