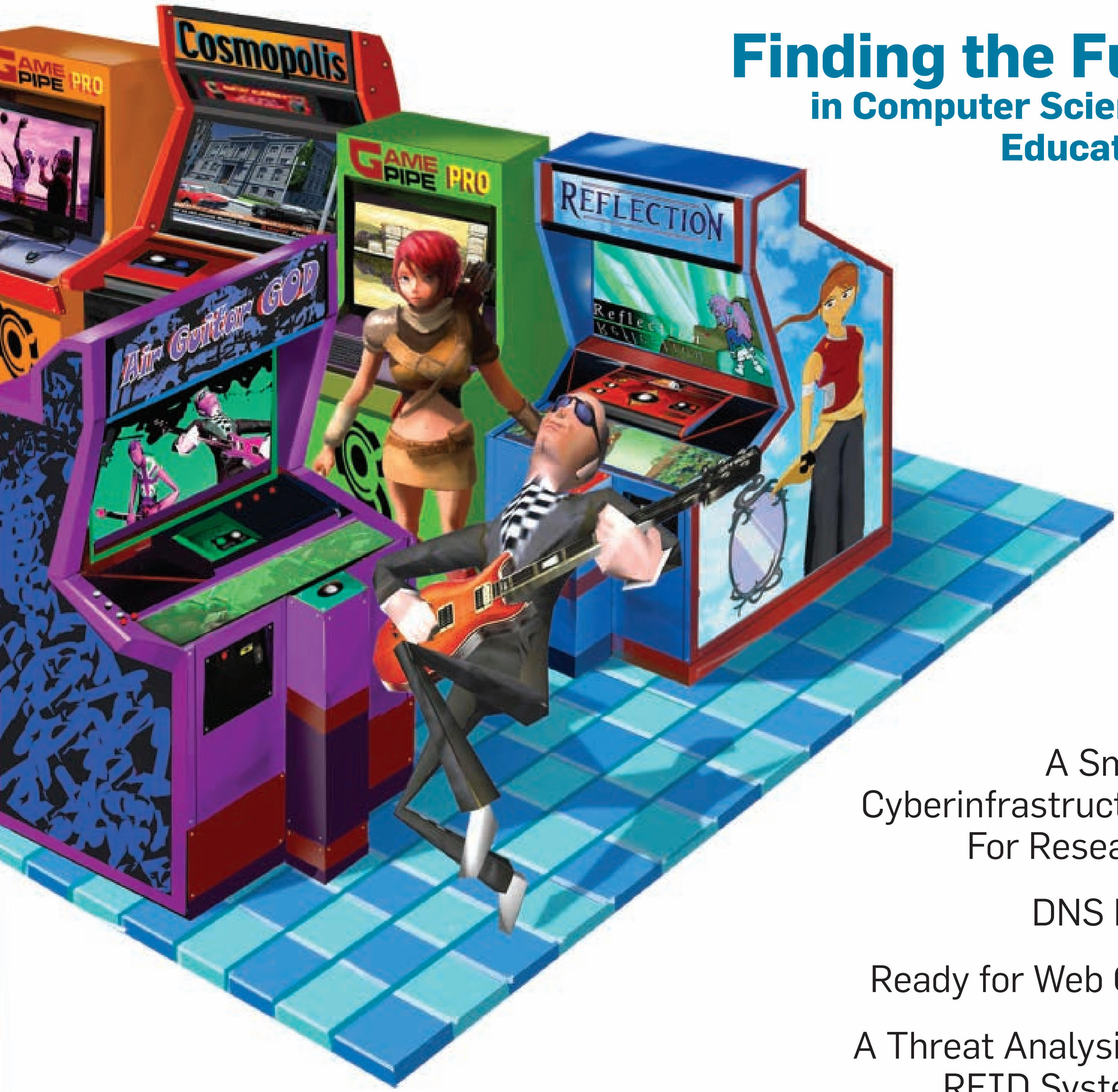
# COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

12/2009 VOL.52 NO.12



## Finding the Fun in Computer Science Education

A Smart Cyberinfrastructure For Research

DNS Lies

Ready for Web OS?

A Threat Analysis of RFID Systems

Association for Computing Machinery

DOI:10.1145/1610252.1610272

**University CS departments are incorporating game design and development to prepare their students for the game industry's expectations.**

BY MICHAEL ZYDA

# Computer Science in the Conceptual Age

IN TECHNOLOGY, THE conceptual age is defined by cognitive or creative assets, including design, storytelling, artistry, empathy, play, and emotion. Good engineering or good computer science is no longer enough; design must be just as good.[7] The transition from information age to conceptual age has been overlooked by most academics in computer science, yet many of the consequences of the transition have been apparent for the past decade at least, including a drop in undergraduate interest, the outsourcing of U.S. programming jobs, and the decline in research laboratories focused on advanced computer science research. Today, another aspect of the transition is emerging: integration of game development into computer science curricula. Here, I discuss what it looks like, how it affects computer science departments, and how it helps drive the overall transition.

The last quarter of the 20th century saw the U.S. economy transition from the industrial age to the information age, a move characterized by major changes in profession, as many people chose to be knowledge workers, rather than manual laborers. Computer science was the driving intellectual, social, and cultural force behind the information age. Computer researchers and developers spearheaded widespread adoption of new technology and were paid well. Academic computer science departments grew in terms of numbers of students, faculty, and facilities; almost all major universities received donations, including new "computer science buildings," and enrollment was so large that standardized tests had to be created for the field. These were the fattest of times, when computer science research drove the growth in the national, as well as the global, economy.

The dot-com crash closing out the 20th century signaled the end of the era. Computer science was now viewed as an unstable career choice, with corresponding drop in interest by young people.[8] Additionally, many computer science departments heard complaints from industry that their recent graduates were unprepared to be part of the modern work force. So, following their economic self-interest, students migrated to other fields, and the leading U.S. universities saw major growth in undergraduate business programs.

We can hypothesize that the drop in interest in computer science was part of the natural ebb and flow and fashion of career choice. But doing so is to stick one's head in the sand and hope for the return of yesteryear. A more progressive view is to focus on the concurrent transition from information age to conceptual age. The old

**Scene from the *Artemis Chronicle* PC game built with Microsoft's XNA toolkit and the USC GamePipe Laboratory NitroX game engine.**

ILLUSTRATION BY ANDREA TSENG / CHIMERA GAME STUDIOS

focus on "tractable abstractions separate from real problems"[2] is no longer acceptable. In the conceptual age, we need to change the direction of the field or continue an unwelcome slide toward irrelevance.

## Cognitive and Creative Assets

The conceptual age of technology is defined by cognitive and creative assets, with the design side being just as good. We see harbingers in recent business success. For example, Apple has done well with design-driven products enabled by great engineering. On the other hand, Microsoft is a great engineering company that deemphasizes design at its own peril; the Vista operating system, despite its technical success, was not built with user experience as its ultimate goal and had great difficulty with respect to adoption. Much of the rest of the traditional computing industry is shrinking, but the game industry is a segment that continues to grow due to its focus on design backed up by great supporting engineering. The demand for computer scientists capable of building games is high, with large companies like Electronic Arts reporting that 65% of their hiring demand is for programmers skilled in building games. Unfortunately, the kind of computer scientist required by the game industry is not exactly what traditional computer science departments produce.

The game industry wants graduates who are strong programmers and system developers, skilled in game design, and capable of and experienced in game development in large, cross-disciplinary collaborative teams. Some computer science departments produce graduates who are strong programmers and system developers, but most do not produce graduates skilled in game design. Moreover, most do not produce graduates facile in large, cross-disciplinary collaborative teams. The typical computer science graduate has little experience with team software development beyond one or two projects with three to five other computer science students.

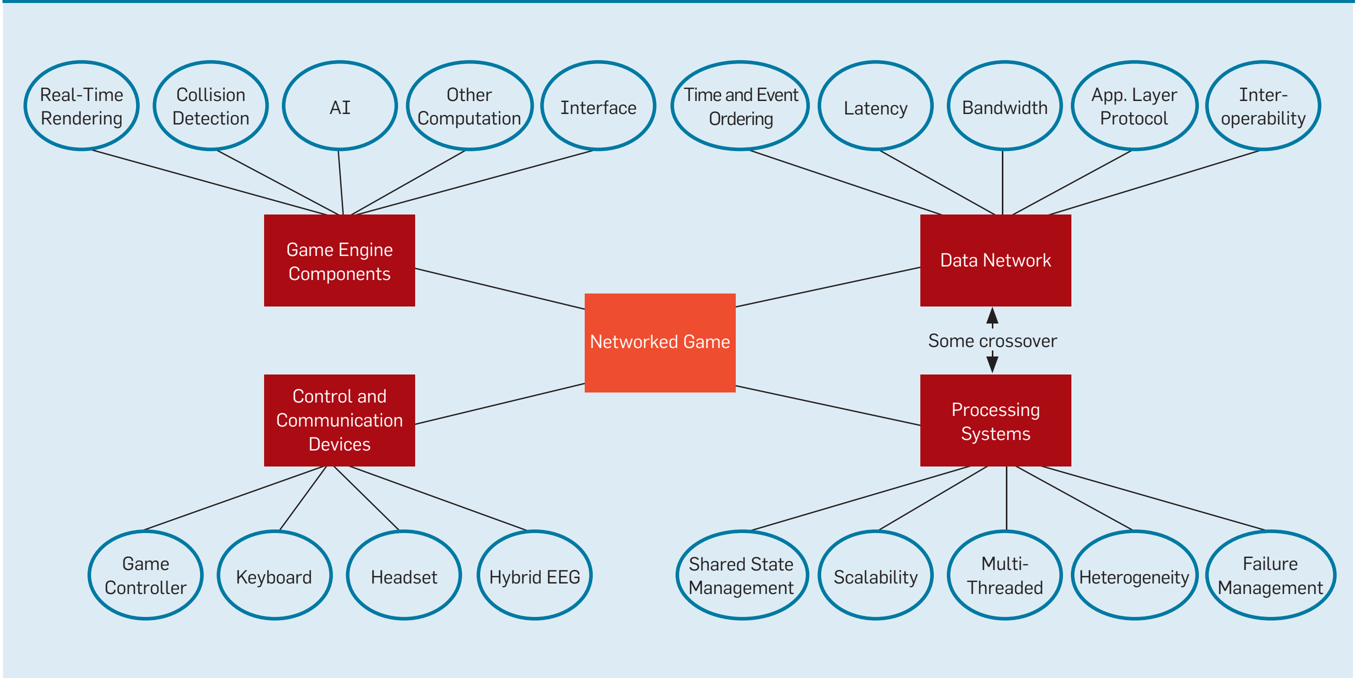
Computer science departments can retool themselves to meet these challenges, but, for game development, doing so requires a strong, experienced champion and proper resources. Here, I discuss a particular approach we take at the University of Southern California, outcomes from that program, and questions with respect to transitioning a mature field toward the conceptual age.

## Game Development and CS Education

If computer science departments are indeed to make the transition, what will they transition into? The game industry also wants graduates with a strong background in computer science. It does not want graduates with watered-down computer science degrees, but rather an enhanced set of skills. This is good news for the departments, meaning they can transition some courses to new material or new foci, create new courses, and still not abandon decades of hard-won knowledge. The best way to think about the transition to the conceptual age is to make the focus the "big idea or big concept," with a follow-on focus on how to build the concept and with what technologies.

Strong programming skills is the first item on the list, meaning an undergraduate program's first four computer science courses—CS-101 Fundamentals of Computer Programming, CS-102 Data Structures, CS-105 Object Oriented Programming, and CS-201 Principles of Software Development—must be taught in a rigorous manner by excellent practicing programmers. While computer programming languages abound, and the historic computer science attitude "You learn one language, and it's easy to pick up the next one" is not shared by industry. The game industry will also tell you that it wants the first four programming classes in C++, not Java, according to M.M. McGill[4] and

Components of a networked game.



my own private communications with directors of human resources in major game-development companies. Many universities switched to Java with the Internet boom, as it is easy to resource a programming laboratory for Java, and Java support for the user interface is extensive. But the game industry programs mainly in C++. Many game companies say they will not interview or hire someone whose first programming language is Java. Computer scientists might argue about the virtues of programming languages, but most console and PC games are built in C++. From a C++ perspective, our students can teach themselves other useful languages, learning Objective-C for iPhone game development or C# for games with XNA in a single semester. The industry is basically saying it wants strong programmers with multiple courses in C++, its primary development language. The USC experience is that 100% of its students interviewed for programming positions are given three-to-four-hour-long programming tests, with almost all companies administering the tests in C++. We had one company (NCSoft) test in C and another (Microsoft) indicate the test could be done in C++, C, or C#. Programming interns/job seekers from our program Spring 2009 (35 interviewed in the game industry) found no companies administering programming tests in Java.

Strong skills in system development is another must. The figure here outlines the components that must be juggled by a programmer developing a typical networked game, touching on much of computer science and then some. Moreover, the game software must run multithreaded in a well-balanced manner and provide an immersive and entertaining experience to the game player. Game development is viewed by some as systems design.

Programmers comfortable in cross-disciplinary groups is third on the list. Industry wants programmers who are able to generate software based on the vision of designers, work with artists to generate the right display and feel, and know how to participate in large-team development efforts. The traditional computer science student is far from this.

**Programming interns/job seekers from our program Spring 2009 (35 interviewed in the game industry) found no companies administering programming tests in Java.**

Computer science departments must provide the necessary experience so game programmers are industry-ready on graduation day. This means that graduates must have built a significant game by collaborating with other students, not all of whom have backgrounds in computer science. Grads should also be familiar with the pipeline-development process and asset-management systems, both aspects of game-industry development potentially useful throughout the program. Such experience is very different from traditional computer science-degree programs in which software-development teams are small, and there is no strong requirement for asset management or source-code base sharing or versioning.

## Big Game Program or Baby Steps?

Computer science departments changing their focus toward game development is an obvious approach toward preparing students for the conceptual age. Many universities are building game-development programs within or aligned with their computer science departments.[1,3,5,6,9,10] The annual Foundations of Digital Games conference (http://www.foundationsofdigitalgames.org/) focuses on this important transition.

How does all this directly affect the departments? Consider two separate efforts: the USC GamePipe Laboratory and the University of Washington, Bothell, the latter covered in the article "Computer Games and Computer Science" by Kelvin Sung (on page 74).

## USC GamePipe Laboratory

The USC Department of Computer Science offers both a bachelor's degree in computer science (games) and a master's degree in computer science (game development).[12] Students interested in Ph.D.-level topics are encouraged (for the moment) to apply to the traditional computer science Ph.D. program. The bachelor's in computer science (games) program includes 37 units of traditional computer science and 42 units of game-development courses. The computer science courses are the same as those taken by regular students in USC's bachelor's computer-science program, except for the following modifications:

*Programming courses.* We rewrote

three of the first four programming courses to be in C++ and created game-oriented examples and exercises. The game focus helps motivate students, getting them excited about programming. For Fall 2008, results showed a 28% increase in the number of students with letter grade A in the games-oriented CS-101 Introduction to the Fundamentals of Programming and for Spring 2009 a 49% increase (http://gamepipe.usc.edu/~zyda/GamePipe/Ghyam-Final-MS-Study-2009.pdf). Further analysis of the results is underway to determine whether they reflect superior skills, superior understanding of programming, superior motivation, or some other cause;

*Replaced EE with CS.* We replaced four electrical-engineering courses on circuit design with computer-science-focused EE-352 Computer Organization & Architecture and parallel-programming-focused EE-452 Game Hardware Architecture. The removed courses represent an older style of computer-science-degree program. We felt it more important that students learn how computer architectures affect programming rather than how to make such architectures. We also felt that parallel programming was highly relevant to both the multithreaded nature of modern game development and multicore processors;

*Added EE.* We added EE-450 Introduction to Computer Networks as a degree requirement. Networking is offered as an elective in the regular computer science curriculum. Because games need networking, all our students take it;

*Cut compiler courses.* We eliminated the two compiler courses taken by regular computer science students. ACM eliminated compilers as a requirement from the CS core in 1979. The USC Computer Science Department uses the two courses as large programming-project capstone courses, so we felt we would rather have our students build games. The replacement for the games curriculum is the two-semester CS-491A/B Final Game Projects course. An interesting result is that the Computer Science Department is weighing whether to allow general computer science students to take the Final Game Projects course instead of the compiler sequence; and

**We felt that parallel programming was highly relevant to the multithreaded nature of modern game development and multicore processors.**

*No foreign languages.* General education requirements for the bachelor's degree in computer science (games) are approximately the same as for USC's regular computer science degree. The degree lacks a four-course foreign-language requirement as in all other USC College degrees, an accident of planning rather than a recommendation.

When we began planning the degree, the dean of engineering said, "I don't want a weak degree." So we made these changes to the computer science component of the program and are confident we offer an academically strong and industry-viable undergraduate degree.

For the games-side of the bachelor's degree, we replaced 42 units of electives from the general computer science degree with a full course in game development. Hence, the degree looks more like a double major than a specialization. We also have a set of courses on game engineering, game design, and game cross-disciplinary.

Game engineering covers video-game programming, parallel programming on consoles and graphics processing units, and programming game engines, all of which are straightforward software-development courses and all important for game development.

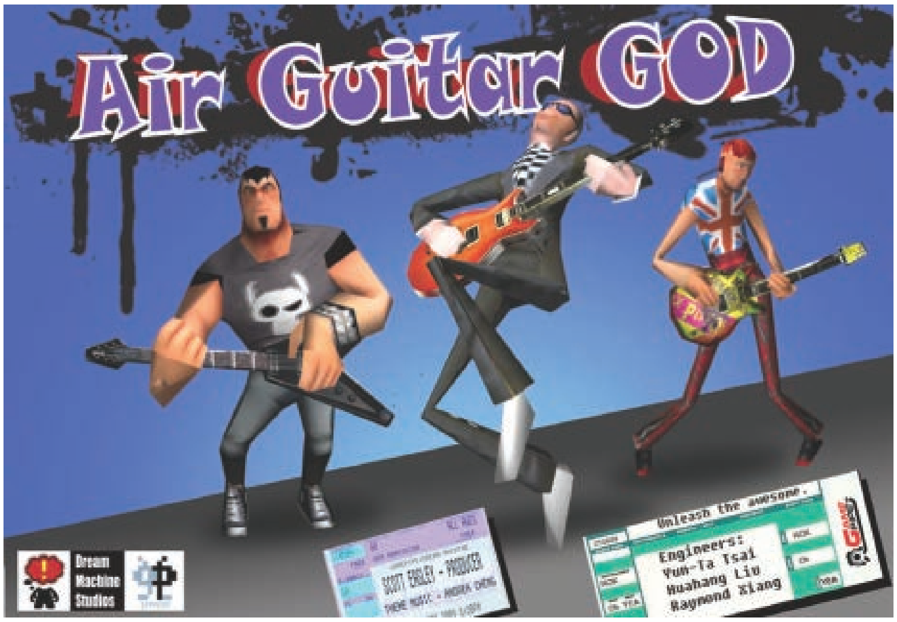
For game design, we send our students to a three-course Game Design Workshop sequence in the USC School of Cinematic Arts Interactive Media Program. Engineers are immersed in gameplay design as taught by master game designers. The first course in the sequence—CTIN-488 Game Design Workshop—teaches students how to prototype gameplay using cardboard and hand-drawn art; they basically build board games. We get an interesting response from the engineers we send there. Some rave about it, saying it's the best thing they've ever taken. Some hate it, feeling frustrated they cannot immediately code-up a game. Some hate the first few weeks but in the end come back and say it was a great course. Ultimately, the students who understand the importance of the course and express satisfaction end up with great internships/jobs in the game industry. It's where we see the future game industry technical directors of the conceptual age.

Our approach to creating a cross-disciplinary program was to design courses that could be taken by non-computer scientists, as well as by computer science majors. First-semester undergraduates survey game play, from the start of games to using (among other things) old consoles, old PC games, and emulators. Students come out of the course hooked on our degree program and wanting more. They then take a video-game production course to build individual games using GameMaker and hear from industry speakers on game development. Next is CS-281 Pipelines for Games & Interactives in which they learn how to create assets for games, including 3D models and animations. One achievement is a pipeline asset and source-code management system we designed that is shared by all game-development courses in the program. We teach our students how to use it early on, simplifying and enhancing their ability to develop games in subsequent courses. We also place all our students in a semester-long character-animation course.

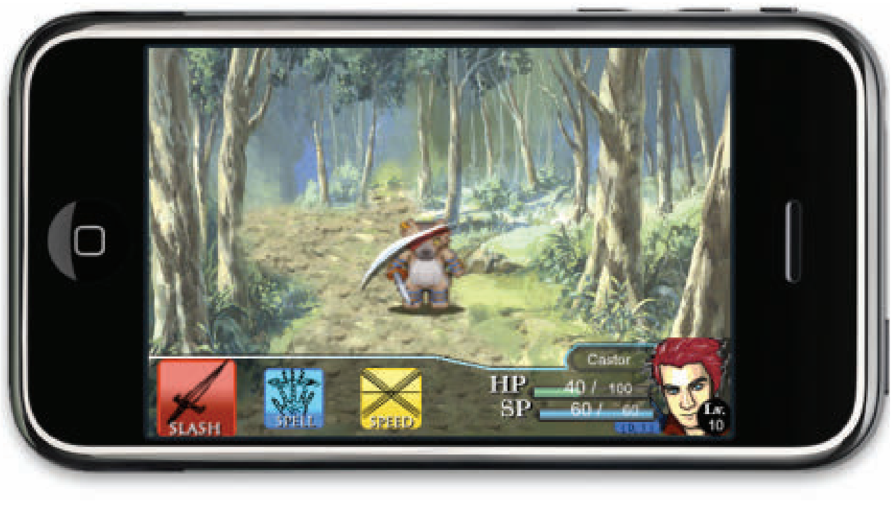
In their final one-and-a-half years before graduation, our students become ready for game development. All take a course developing a serious game in a large team for an interested sponsor. Their last year before graduation is in CS-491A/B Final Game Projects building games in large cross-disciplinary teams from August to May, with game designs selected through a design playoff the previous May. The Final Game Projects course is administered jointly by the School of Cinematic Arts and has students from computer science (bachelor's and master's students), interactive media (bachelor's of fine arts and master's of fine arts), fine arts (bachelor's of fine arts), animation (bachelor's of fine arts), music composition, and film scoring. Teams in this class include from 11 to 25 students building significant games over its two-semester run, aiming for contest submission by the end of the second semester.

Strong cross-disciplinary collaboration occurs, with results presented at the end of each semester on Demo Day when game-industry executives are invited to review the students' work; the accompanying screenshots are indicative of the visual quality. *Artemis Chronicle*, a beautiful action-adventure title (see page 67), demonstrates the powerful features of the NitroX Engine, a revolutionary, complete development framework for creating XNA games. Both the game and the NitroX engine were built in the CS-491AB Final Games course over two semesters. *Air Guitar God*, a beat-matching iPhone game (below), incorporates a student-designed algorithm for automatically computing beat detection from any song imported into the game. And *Slice*, an action role-playing game (below), puts battles at your fingertips on the iPhone. Videos of the most recent Demo Day are at http://gamepipe.usc.edu/USC_GamePipe_Laboratory/R%26D/R%26D.html.

At the end of each academic year, we now routinely place large numbers of students (typically around 35) in internships/jobs in the game industry where they are nearly instantly productive. In the Fall semesters in 2007 and 2008, a team on Demo Day was offered a commercial deal to turn their game into a company for further development or prepare to ship commercially. Spring 2009 included five student-built games under commercial consideration.



Opening screen from ***Air Guitar God***, **an iPhone beat-matching game developed by students in the USC GamePipe Laboratory's mobile games course.**



**Scene from** ***Slice***, **an iPhone gesture-based fighting game developed by students in the USC GamePipe Laboratory.**

**Industry wants programmers comfortable in cross-disciplinary teams.**

## Effect on the USC CS Department

How has the game-degree program affected USC's Computer Science Department, nudging it along toward the conceptual age? First, I must say we have not dismantled the traditional bachelor's program in computer science, and students continue to enroll in it. Computer science and games students share almost the same core computer science curriculum.

Enrollment concerns were part of our original motivation with respect to the games program, though they have eased, as they have for many departments, according to the Computing Research Association' annual Taulbee Survey (http://www.cra.org/statistics/). For the Fall 2009 semester, 29% of the students in the USC bachelor's in computer science program are in the games program, representing an important influence on the department.

While we have not measured it, probably the greatest effect we see is an apparent "joy of computing" feeling by our students who come to class highly motivated, pour their best ideas into their projects, and produce spectacularly creative results, some of which the game industry wants to commercialize. The graduating students who move into positions in the game industry return to subsequent Demo Days, bring their bosses, and hire more of our graduates. The addition of a creative-design component and making it student-driven and student-dependent is key to this success. Students take ownership of their educational program and aspire to make everything they do shine.

Another component worth mentioning is the commitment of the program's faculty and instructors. The students are in small classes where they have much say in direction and result. The faculty is available to provide technical guidance, mentorship, motivation, and support. It is physically draining if done properly, but if the students recognize their passion for games, they cannot help but be passionate game developers as well. Faculty working directly with our program are the executive producers of some 12 to 14 games per semester. The results of these efforts are visible on the USC GamePipe Web site (http://gamepipe.usc.edu).

We have also been able to create a line of research funding based on and around games.[11] It is more than difficult to build an R&D program on games without a pipeline of students learning to build them. Today, we have both. The games program has additionally strengthened the reputation of USC's Computer Science Department in terms of increased R&D funding, improved hiring rates for graduates, attention to the games program in the press, invitations to games faculty to speak at conferences and publish in traditional venues, and other universities desiring to copy USC's success. In Fall 2008, the director of the USC GamePipe Laboratory was appointed an ACM Distinguished Speaker (http://www.dsp.acm.org/) in recognition of the program's achievement. Hiring managers and developers in the games industry now regard us as one of the "best games programs," though we are only in our fourth year of operation. Moreover, traditional USC computer science faculty not currently involved in the games program have begun to ask how they can participate, some have changed their course projects to be game-related, and many ask how we can draft proposals together. The computer science faculty realizes that something important is happening, with some beginning to also guide their own programs toward the conceptual age. C

**References**

1. Barnes, T., Powell, E., Chaffin, A., and Lipford, H. Game2Learn: improving the motivation of CS1 students. ACM Game Development in Computer Science Education (Miami, FL, Feb. 26–Mar. 3, 2008). 1–5.
2. Brooks, Jr., F.P. The computer scientist as toolsmith II. *Commun. ACM 39*, 3 (Mar. 1996), 61–68.
3. Horswill, I. and Novak, M. Evolving the artist-technologist. *IEEE Computer 39*, 6 (June 2006), 62–69.
4. McGill, M.M. Defining the expectation gap: A comparison of industry needs and existing game development curricula. ACM Foundations of Digital Games (Orlando, FL, 2009), 129–136.
5. Parberry, I., Roden, T., and Kazemzadeh, M.B. Experience with an industry-driven capstone course on game programming. ACM SIG on Computer Science Education (Houston, TX, 2005), 91–95.
6. Phelps, A., Egert, C., and Bierre, K. Games first pedagogy: Using games and virtual worlds to enhance programming education. *Journal of Game Development 1*, 4 (May 2006), 45–64.
7. Pink, D.H. *A Whole New Mind: Moving from the Information Age to the Conceptual Age.* Riverhead Books, New York, 2005.
8. Thibodeau, P. Computer science graduating class of 2007 smallest this decade. *Computerworld Online* (Mar. 5, 2008); http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9066659.
9. Whitehead, J. Introduction to game design in the large classroom. ACM Game Development in Computer Science Education (Miami, FL, 2008), 61–65.
10. Wolz, U. and Pulimood, S.M. An integrated approach to project management through classic CS III and video game development. ACM SIG on Computer Science Education (Covington, KY, 2007), 322–326.
11. Zyda, M., Spraragen, M., and Ranganathan, B. Testing behavioral models with an online game. *IEEE Computer 42*, 4 (Apr. 2009), 103–105.
12. Zyda, M., Lacour, V., and Swain, C. Operating a computer science game degree program. ACM Game Development in Computer Science Education (Miami, FL, 2008), 71–75.

**Michael Zyda** (zyda@usc.edu) is the director of the USC GamePipe Laboratory and a professor of engineering practice in the Department of Computer Science at the University of Southern California, Los Angeles, CA.

© 2009 ACM 0001-0782/09/1200 $10.00

PHOTOGRAPH BY MICHAEL ZYDA/USC GAMEPIPE LABORATORY

**Acknowledgements**

We wish to thank Gerard Medioni, the Computer Science Department Chair who strongly supported the creation of this new academic program. Gerard let us create sixteen new Computer Science and two EE courses over an 18 month period and let us focus on that rather than proposal writing. Dean Yannis Yortsos of the Viterbi School of Engineering provided us laboratory support in the newest building on campus and helped us create a great program for our students. Scott Easley contributed enormously to the creation of this degree program and has operated both the Serious game research course structure and Final Game Projects sequence. Chris Swain of the School of Cinematic Arts Interactive Media Program provided guidance on how to utilize the game design workshop sequence in our degree program. Scott Easley and Chris Swain have now bravely created and operated the first joint Final Game Projects course sequence between the School of Cinematic Arts and the Viterbi School of Engineering. Steve Schrader made the paperwork happen for the degree program and he is an irreplaceable part of the Computer Science Department. Massoud Ghyam was a tremendous help in revising the computer programming and software development sequence of the undergraduate degree program. Jose Villeta developed the entire set of course notes for the EE-452 Game Hardware Architectures sequence. Louise Yates, Associate Dean for Admissions and Students Affairs, has been outstanding in handling admissions so that we get excellent students and has been wonderful in providing statistical support to show how the BS in Computer Science (Games) degree program has helped to strengthen and grow the Computer Science Department. Dean Ruth Weisberg of the School of Fine Arts was wonderfully receptive to the idea of creating a Game Art and Design minor program to complement her schools' degrees. Ann Page of Fine Arts carried the ball in designing that Fine Arts minor program. We had a host of graduate and undergraduate students that made the USC GamePipe Laboratory happen even when there was no financial compensation or reward – Pamela Fox, Dhruv Thukral, Sumeet Jakatdar, Fred Zyda, Danny Parks, Donna Djordjevich, Erin Reynolds, Rob Cheng, Marc Spraragen, Balakrishnan Ranganathan, Ricardo Chavarria, Keyur Bhulani, Devin Rosen, and Lily Cheng. We consider them the founding students for this laboratory and academic program. We had wonderful contributions to our degree program from Jose Villeta of Heavy Iron Studios; Steve Seabolt, Colleen McCreary, and Bing Gordon of Electronic Arts; John Bojorquez and Pat Griffith of Activision; Mike Lee of Emsense, and Anthony Borquez of Konami (formerly USC ITP). We had donations from Motorola Research Laboratories, Microsoft Research (John Nordlinger), the Humana Innovation Center, Lockheed Martin, Sandia National Laboratories, THQ Interactive, Activision, EA Mobile, and Alon Carmel that keep our program operating.