Breadth-Last Technical Electives: Integrating the CS Core via Computer Games and Mobile Robotics

William W. White
Southern Illinois University Edwardsville
Department of Computer Science
Edwardsville, IL 62026-1656
(618)650-3483
wwhite@siue.edu

Jerry B. Weinberg
Southern Illinois University Edwardsville
Department of Computer Science
Edwardsville, IL 62026-1656
(618)650-2368
jweinbe@siue.edu

ABSTRACT
In this paper, we introduce the concept of *breadth-last* technical elective courses, which are designed to assist undergraduate CS students in integrating their entire core curriculum into a coherent whole at the end of their degree programs. Specific breadth-last courses in intelligent mobile robotics and computer game development have been implemented and are presented here to demonstrate the pedagogical concepts being discussed.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms
Algorithms, Design, Experimentation, Human Factors.

Keywords
Breadth-last, breadth-first, breadth-second, game design, integration, mobile robotics.

1. INTRODUCTION
Technical electives in undergraduate CS curricula frequently focus upon application areas and algorithmic disciplines that integrate only portions of the underlying CS core areas. For instance, courses in computer networks and data communications commonly integrate computer architecture, operating systems, and graph-based algorithms, but rarely touch upon the core areas of software engineering, programming languages, and human-computer interaction. Similar limitations exist in other common technical elective courses, such as database management systems, artificial intelligence, automata theory, computer graphics, compiler design, and numerical analysis.

The integration of the entire CS core into particular technical electives that a student takes late in the degree program serves two significant purposes. First, the student is afforded an opportunity to see how the entire core fits together within an integrated system. The core is usually presented in a segregated fashion, not merely separated along the applications/systems fault line, but split even further into isolated cliques of specializations. Operating systems courses and computer architecture courses, for example, rarely encroach upon each other's domains, in spite of the strong interdependence of the two areas in real development environments. Technical electives that strive to integrate the entire core have the potential to lower these somewhat artificial boundaries between CS disciplines.

In addition, technical electives that integrate the entire CS core have the potential to facilitate the ability of students to envision the entirety of software systems and to consider the more far-reaching ramifications of the solutions that are being proposed to particular development problems. Superior alternative solutions may be formulated based upon a student's thorough understanding of, say, a certain algorithm's data retrieval process when accessing a particular database system on a specific hardware platform. Moreover, students may envision solutions that span hardware, systems, and application levels, rather than merely devising workarounds for the platforms on which they happen to be working.

2. BACKGROUND
The merits of exposing CS undergraduates to an integrated core have been acknowledged for some time. The most common approach has been to require students to take a *breadth-first* CS0 course that introduces the primary core areas in a broad, pre-programming environment [1]. Such courses are widespread in CS curricula and have proven to be quite helpful in motivating new CS students and sketching the discipline for them in broad strokes [11].

A more recent, and less common, technique for illustrating how the CS core may be integrated is the *breadth-second* approach [2], which affords a survey of the core after CS majors complete their introductory programming courses, but before they take more advanced core courses in algorithms, computer architecture, operating systems, software engineering, database management, etc. Like breadth-first, this technique has been quite motivational to CS students, with the additional benefit that the students have already had some early hands-on experience within the discipline, which assists in their understanding of the concepts being presented [4].

The primary mechanism for integrating the CS core later in the curriculum has been the use of capstone projects [3]. Undertaken by students in their senior year, these projects are often designed to provide students with software development experience analogous to what they might see in industry. While such projects frequently integrate various portions of the core, like most technical electives, they rarely endeavor to provide students with an opportunity to integrate the entire CS core.

In an attempt to address the need for core integration at the end of the CS curriculum, we have developed two technical elective courses that utilize the core topics that are presented throughout the major curriculum and apply them in an integrated fashion to specific application areas. Designed to complement our capstone course sequence, these courses focus upon areas that are particularly conducive to the notion of reviewing and integrating the entire CS core: game design and development and intelligent mobile robotics. While previous efforts have focused upon enhancing various aspects of the CS core by means of both game design [6,9] and robotics [12], our work has demonstrated that the entire core may be integrated via such courses.

While it has been suggested that breadth-first courses for non-majors might also be considered *breadth-last* [7], we apply the term breadth-last to CS technical electives which expose majors to the integration of the entire CS core.

3. GAME DESIGN AND DEVELOPMENT

While computer game development remains a niche area within the grand scheme of computer science, it provides opportunities to illustrate the confluence of many of the primary fields within the discipline. Table 1 illustrates the wide array of such areas that we present in our Game Design and Development course, which is offered to senior CS students as a technical elective.

Based loosely on the curriculum framework developed by the International Game Developers Association [8], this course devotes approximately one week to each of the following topics:

1. History & Social Impact of Video Games
2. Game Design Principles
3. Game Software Engineering
4. Game Programming
5. Game Hardware & Operating System Considerations
6. Game Mathematics
7. Game Physics
8. Game Graphics & Animation
9. Game Intelligence
10. Game Audio
11. Multiplayer Games
12. Game Interface Design
13. Game Textures & Lighting
14. Game Cinematography & Storytelling
15. Game Industry & Legal Perspectives

The parenthetical numbers next to each topic in Table 1 indicate the week in which that topic is covered.

Assignments in recent offerings of this course have included:

Programs:

- Interactive target practice, throwing virtual darts at virtual balloons that generate a particle system of confetti when popped (Figure 1). This assignment emphasizes data structures, HCI, and graphics.

Table 1. CS Areas and Corresponding Topics within the Game Design and Development Course

CS Area	Corresponding Topic in Course
Computer Architecture	• Graphics Processors (5) • Peripheral Devices (5) • Display Technologies (5)
Operating Systems	• Multimedia Coordination (5) • Cross-Platform Compatibility (5) • Multithreaded Race Conditions (5)
Data Structures	• Isometric Grids (6) • Music Sequencers (10) • Subdivision Surfaces (8) • Binary Space Partitioning (8)
Algorithms	• Collision Detection (6) • Rigid-Body Dynamics (7) • Fractal Terrain Generation (6)
Programming Languages	• Scripting Languages (4) • APIs (4)
Software Engineering	• System Design (2) • Agile Programming (3) • Rapid Development (3)
Networks/Data Communications	• Multiplayer Platforms (11) • Wireless Connectivity (11) • Game Latency (5) • Mobile Gaming (11)
Database Management	• Game Asset Management (3)
Human-Computer Interaction	• Intuitive Game Interfaces (12) • Heads-Up Displays (12)
Artificial Intelligence	• Non-Player Characters (9) • Emergent Behavior (9) • Finite State Machines (9) • Pathfinding & Dead Reckoning (6)
Computer Graphics/ Image Processing	• Real-Time Rendering (5) • Character Animation (8) • Texture Mapping (13) • Lighting & Radiosity (13)
Social Implications	• Game Content/Censorship (1) • Social Isolation (1) • Software Piracy (15)
Interdisciplinary Interaction	• Art & Design (14) • Storytelling & Culture (14)

Figure 1. Data Target Practice, Before and After Hit.

- Intelligent flocking behavior of non-player characters (merely screen vertices), with interactive controls to modify the extent to which particular behaviors are applied (Figure 2). This assignment stresses algorithms, AI, and graphics.

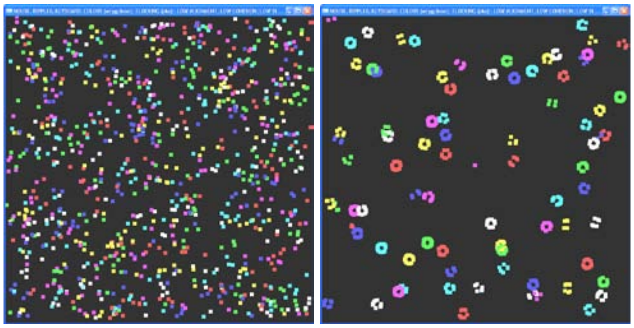

Figure 2. Character Configuration, Initial and After Several Seconds of Flocking Behavior.

- Third-person shooter in which a player-controlled character battles a game-controlled character, with both characters expressing emotional reactions to game events (Figure 3). This assignment emphasizes data structures, algorithms, HCI, AI, and graphics.

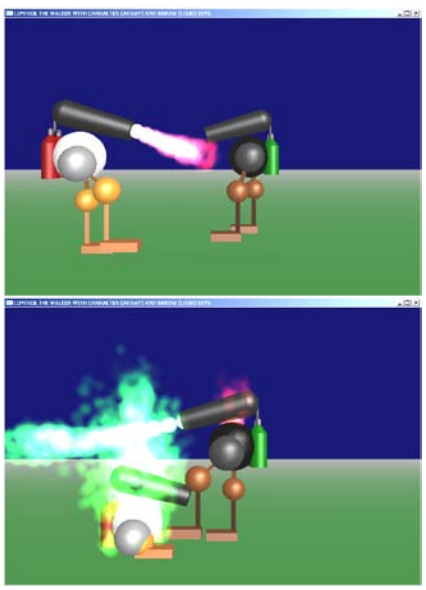

Figure 3. Player-Controlled Character Attacks Game-Controlled Character and Is Destroyed in Return.

Paper:

- Evaluation of recent game “postmortem” articles published in *Game Developer* magazine, emphasizing good and bad development practices. This assignment places particular emphasis upon software engineering and interdisciplinary interaction.

Presentation:

- Explanation of how the course enhanced each student’s understanding of an assigned CS core area, as well as how that core area helped the student better understand the areas of game design and development. This assignment serves as a review of the entire game design and development course and how it relates to the CS core.

Student enthusiasm for game design and development is quite high in CS, and the ready availability of fast, inexpensive graphics processing units and advanced graphical APIs, such as OpenGL and DirectX, facilitates the ability to offer courses in this popular area.

Our course primarily emphasizes the aspects of computer graphics and artificial intelligence that pertain to the development of modern games, but it also gives serious consideration to the hardware, systems software, and algorithmic efficiency concerns that must be addressed in order to yield such results. In addition, significant time is spent in the course discussing practical concerns within the game industry, including rapid software development paradigms, competing and often incompatible game platforms, emerging game opportunities in Web-based and mobile communications, and issues regarding the social impact of games and their sometimes questionable content.

In collected survey data, students have indicated that the course greatly enhanced their understanding of the practical significance of software engineering, human-computer interaction, data structures, and algorithms. Core areas that were not as clearly enhanced included database management and programming languages. Efforts to fortify these areas will be made in future versions of the course. In addition, we are developing mechanisms to gauge the success of the course in illustrating the integration of the CS core, including pre-tests and post-tests to directly measure the students’ understanding of that integration.

4. INTELLIGENT MOBILE ROBOTICS

The advent of affordable, versatile, and robust robotics components in recent years has sparked intense activity in the utilization of mobile robotics throughout CS curricula. In addition to a multidisciplinary robotics course involving teams of students majoring in computer science, electrical engineering, and mechanical engineering [14], we have developed a CS technical elective that effectively uses intelligent mobile robotics to demonstrate the integration of the bulk of the CS core, as illustrated in Table 2. Approximately one week is devoted to each of the following course topics:

1. Overview of Robotics & Control Paradigms
2. Reactive Robot Control
3. Proportional Integral Derivative (PID) Control
4. Robot Sensing
5. Analog-to-Digital Conversion & Signal Noise
6. Computer Vision
7. Knowledge Representation & Robot Deliberation
8. Path Planning
9. Navigation
10. Mapping
11. Localization
12. Recent Mobile Robotics Research
13. Digital Signal Processing
14. Robotics Applications & Social Issues
15. Team Project Workshop

The parenthetical numbers next to each topic in Table 2 indicate the week in which that topic is covered.

Table 2. CS Areas and Corresponding Topics within the Intelligent Mobile Robotics Course

CS Area	Corresponding Topic in Course
Computer Architecture	• Exteroceptive Sensors (4) • Proprioceptive Sensors (4) • Microcontrollers (2) • Motor Controllers (2) • Digital Signal Processing (5, 13)
Operating Systems	• Multitasking (1, 2) • Interprocess Communication (1, 2) • Resource Scheduling (12)
Data Structures	• Grid- and Graph-Based Maps (10) • Quadtree-Based Maps (10) • Schemas (10) • Voxels (4)
Algorithms	• Position Sensing (11) • Navigation & Spatial Mapping (9) • Collision Avoidance (9)
Programming Languages	• Concurrent Programming (1) • Functional Languages (1)
Software Engineering	• Behavior Diagrams (15) • Sequence Diagramming (15)
Networks/Data Communications	• Wireless Connectivity (5) • Multi-Robot Coordination (12,15) • Web Interfacing (11)
Database Management	• Resource Management (12)
Human-Computer Interaction	• Tele-Operation (5) • Tele-Presence (5) • Heuristic Design (2) • Cognitive Modeling (2)
Artificial Intelligence	• Intelligent Agents (7) • Speech Recognition (7) • Reinforcement Learning (7) • Neural Networks (12) • Swarm Intelligence (12)
Computer Graphics/ Image Processing	• Computer Vision (6) • Color Tracking (6) • Pattern Recognition (6) • Image-Map Correspondence (10)
Social Implications	• Industrial Automation (1) • Assistive Robotics (14) • Anthropomorphism (1)
Interdisciplinary Interaction	• Electrical Engineering (5) • Mechanical Engineering (3)

Assignments in the most recent offering of the course included:

Programs:

- Development of a predator-prey robot exhibiting navigation, obstacle avoidance, and chasing and fleeing behaviors. This assignment integrates algorithms, AI, and image processing.
- Development of a PID controller for moving straight and in circles. This program stresses computer architecture and operating systems.
- Development of a map-based module for a robot to use for pathfinding. This assignment emphasizes data structures, algorithms, and AI.
- Development of a dynamic mission-planning module for a robot to use for determining optimal paths to multiple destinations. This program integrates operating systems, data communications, and algorithms.

Paper:

- Evaluation of recent robotic-related technical paper dealing with research in assistive robotics, robotic agents, sensing, navigation, etc. This assignment serves as a review of the entire intelligent mobile robotics course and how it relates to the CS core.

Robots are commonly used in artificial intelligence courses as a tool to illustrate knowledge-based behavior [10], and more recently they have been introduced in other CS courses, primarily as a motivational tool [5, 13]. However, mobile robotics have proven to be a worthwhile forum for demonstrating how the entire CS core may be integrated into a specific application area. The course is taught using an open architecture microcontroller (Figure 4) that provides digital and analog sensor input, control for servo motors and DC motors, and an integrated color camera. The use of the microcontroller provides an opportunity to cover various topics in architecture and operating systems, such as H-bridge circuits for motor control and FPGA programming.

Figure 4. The XBC Microcontroller (www.botball.org).

While the primary focus of the course is the examination of the control structures and computational mechanisms needed for effective robotics navigation and planning, course participants must also keep in mind the hardware limitations of these systems and the requisite memory management needed to efficiently utilize resources. Using their knowledge of programming languages, data structures, and algorithms, students must learn to develop efficient code that runs within the 32K of programming space provided by the firmware. This can be quite challenging in the later assignments, which require the robot to store and use a map of its environment.

The integrated color camera allows us to assign projects that incorporate image processing concepts into the course (Figure 5). Reactive and behavior-based robot control architectures provide a natural area to introduce students to multitasking and multi-process programming. The current version of this course uses a radio communication module for networked inter-robot communication, which provides an excellent medium for presenting material on networking and wireless communication, while affording an opportunity to assign projects that address multi-robot coordination, which may include multi-robot mapping and task completion.

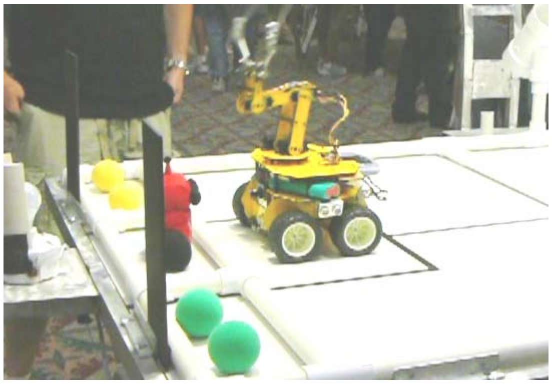

Figure 5. XBC Mounted to Robot Base Used to Detect Items of Different Color and Control Robot Movement and Arm.

5. CONCLUSIONS

Admittedly, the job market in such fields as intelligent mobile robotics and computer game design and development is somewhat limited. From a pedagogical viewpoint, however, both topics represent very effective means for illustrating to CS majors the interrelationships between the various core areas within the discipline, with the additional positive factors of being remarkably affordable to institute and very popular among students.

We are convinced that the breadth-last approach affords students a substantial benefit as they complete their undergraduate CS degrees. Participants in both courses have commented on how the courses have helped to clarify the manner in which the core areas are interdependent. This increased awareness of the fundamental integration of the CS curriculum is expected to improve the ability of the students to develop more thorough and considered solutions to the software problems that they will ultimately encounter.

In addition to mobile robotics and game development, suggested breadth-last topics that might be developed in the future include mobile communications (with its strong emphases in networking, architecture, and operating systems, as well as significant contributions from HCI and social implications) and scientific visualization (with particular consideration given to parallel processing, graphics, and numerical data structures and algorithms, and additional interests in database management and customer-driven software development).

Early indications from students suggest that breadth-last courses are quite successful in their attempts to broaden student understanding of how the CS core may be integrated within individual technical electives. Future efforts will concentrate on measuring the extent to which this integration is digested by students in the breadth-last courses, as well as expanding the list of breadth-last technical electives from which students may choose.

6. REFERENCES

[1] ACM/IEEE Joint Task Force for Computing Curricula (2001). *Computing Curricula 2001: Computer Science - Final Report.* IEEE Computer Society, Los Alamitos, CA, 2002 (http://www.sigcse.org/cc2001/).

[2] ACM/IEEE Joint Task Force for Computing Curricula (2005). *Computing Curricula 2005: The Overview Report.* http://www.acm.org/education/curricula.html#CC2005.

[3] Beasley, R. E. Conducting a Successful Senior Capstone Course in Computing. *Journal of Computing Sciences in Colleges* 19, 1 (October 2003), 122-131.

[4] Brazier, P., Grabowski, L., and Dietrich, G. Closing the CS I – CS II Gap: A Breadth-Second Approach. *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference (FIE 2003)* (Boulder, CO, November 5-8, 2003), Volume 1. T3C-18-22.

[5] Chilton, J., and Gini, M. Using the AIBOs in a CS1 Course. *Proceedings of the 2007 AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education* (Stanford, CA, March 26-28, 2007), 24-28.

[6] Coulton, P., and Edwards, R. Could Computer Game Design Become a Core Subject for Engineering? *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference (FIE 2003)* (Boulder, CO, November 5-8, 2003), Volume 1. T2F-T21.

[7] Dodds, Z., Alvarado, C., Kuenning, G., and Libeskind-Hadas, R. Breadth-First CS1 for Scientists. *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '07)* (Dundee, Scotland, June 25-27, 2007), 23-27.

[8] IGDA Education Committee (2003). *IGDA Curriculum Framework: The Study of Games and Game Development.* http://www.igda.org/academia/curriculum_framework.php.

[9] Jones, R.M. Design and Implementation of Computer Games: A Capstone Course for Undergraduate Computer Science Education. *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (SIGCSE '00)* (Austin, TX, March 7-12, 2000), 260-264.

[10] Kumar, A. N. Three Years of Using Robots in an Artificial Intelligence Course: Lessons Learned. *Journal on Educational Resources in Computing* 4, 3 (September 2004).

[11] Phillips, A. T., Stevenson, D. E., and Wick, M. R. Implementing CC2001: A Breadth-First Introductory Course for a Just-in-Time Curriculum Design. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)* (Reno, NV, February 19-23, 2003), 238-242.

[12] Saad, A. Mobile Robotics as the Platform for Undergraduate Capstone Electrical and Computer Engineering Design Projects. *Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference (FIE 2004)* (Savannah, GA, October 20-23, 2004), Volume 3. S2G-7-11.

[13] Sklar, E., Parsons, S., and Azhar, M.Q. Robotics Across the Curriculum. *Proceedings of the 2007 AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education* (Stanford, CA, March 26-28, 2007), 142-147.

[14] Weinberg, J. B., White, W. W., Karacal, C., Engel, G., and Hu, A.-P. Multidisciplinary Teamwork in a Robotics Course. *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)* (St. Louis, MO, February 23-27, 2005), 446-450.