Design and Implementation of Computer Games: A Capstone Course for Undergraduate Computer Science Education

Randolph M. Jones
Computer Science Department
Colby College
5847 Mayflower Hill Drive
Waterville, ME 04901-8858
rjones@colby.edu

Abstract

This paper presents a course in the design and implementation of computer games, offered as an upper-division computer science course at Colby College during the winter semester, 1999. The paper describes the material, topics, and projects included in the course. More generally, I argue that this course provides an ideal environment for students to integrate a wide base of computer knowledge and skills. The paper supports this argument by presenting the variety of computer science concepts covered in the course, as well as pointing out potential areas of variation in future courses, depending on the tastes and priorities of the instructor.

1 Background

Computer games are a hugely popular and successful application of computer technology. Many computer games developed these days have production budgets that rival Hollywood movie budgets. On top of this, computer games make some of the most extreme demands on computer technology, requiring more speed, memory, and state-of-the-art peripherals than most standard application software. Because computer games push so many envelopes at the same time, they create a ripe area for a variety of research in computer science. More importantly for this presentation, serious game-oriented projects expose students to a wide variety of technology issues and development and design skills.

In spite of this, before teaching the course presented here, I was personally aware of only two computer game courses at the college and university level. Professor John Laird offers one course at the University of Michigan [8] and Professor Ken Forbus offers the other at Northwestern University [3]. I have since learned of two other computer games courses taught by Jessica Hodgins and Amy Bruckman at the Georgia Institute of Technology [4] and Ian Parberry at the University of North Texas [13].

2 Motivation

I was attracted to the idea of a course on computer games for a variety of reasons. On the surface, such a course appears to offer at least the following advantages:

1 It provides an extremely project-oriented, upper-division course to exercise and enhance the programming and problem-solving skills of advanced students.

2 There is presumably little effort required to motivate the students, to the extent that students can reasonably be expected to expend a fairly large amount of effort on the course.

3 A final project in such a course requires the students to go through a fairly complete software cycle from initial design through implementation to testing and demonstration before their peers.

4 The integration of concepts and techniques required to design and build computer games covers many of the topics offered in an undergraduate computer science curriculum, allowing the students concrete application of much of the theory, concepts, and skills they have been exposed to.

5 Computer games provide excellent examples of object-oriented environments, for which object-oriented design and programming is ideal. This fits in especially well at Colby College, where the lower-division curriculum centers on object-oriented design and programming using the Java programming language.

6 The history and design of computer games include a variety of interesting technological and sociological stories, the study of which fit in well with the liberal arts philosophy at Colby College.

The remainder of the paper presents the details of the course I offered. In the final analysis, although there were

- Computer game genres
- Game design and implementation issues/evaluating computer games
  - What makes a computer game "good"/"popular"?
- History of video and computer games
- General architecture of computer games
  - Event loops
  - State-based programming
- Introduction to Animation
  - Frame rates; physiological and technical issues
  - Simulation frame rate vs. animation frame rate
- Introduction to "sprites"
- Object-oriented design and computer games
- More on sprites and agents
  - Viewing sprites as objects in an object-oriented world
- Building game applets
  - Game manager object
- Threads and multi-threaded programming
  - Threads in Java
- More on animation; drawing and animation in Java
  - Video cards; technical details of drawing
  - Double buffering, blitting, clipping
  - Making animation efficient
- A library of sprite-related classes in Java
- Game input; input devices
  - Mouse and keyboard events in Java
- Graphics and image processing in Java
- Sound processing in Java
- Using application software to develop sounds, images, 3-D graphics
- Extending Java sprite class to have bitmapped sprites
- Technical details of computer graphics and graphics formats
  - Pixels, raster vs. vector graphics, anti-aliasing, gamma correction, alpha channel, resolution, color look-up table, dithering, compression
- 2-D graphics; rotation, translation, scaling, reflection
  - Using matrices to transform 2-D graphics
  - Using a 2-D graphics library
- Animated bitmaps; bitmap loops/sequences
  - Extending the sprite class for animated bitmaps
- Special effects using the alpha channel in Java
- A framework for analyzing/categorizing game interfaces
  - Presentation, world structure, world quantization, projection type, projection angle, point of view
- The art of computer game design
- The craft of computer game design
- Modeling physics; position, time, velocity, acceleration, collisions, mass, force, momentum, conservation of momentum, gravity, friction, resistance
- 3-D graphics, objects, and worlds
  - Understanding and using a 3-D graphics library in Java
  - The 3-D pipeline
  - Coordinate systems, projections, and transformations (in 3D); model coordinate system, world coordinate system, view coordinate system, screen coordinate system
- Graphical rendering of 3-D worlds
  - Painters algorithm
  - Z-buffering
  - Texture mapping
  - Shading, shadowing, and other lighting effects
  - Ray tracing
  - Ray casting
  - Binary space partition trees
- Case studies: Castle Wolfenstein, Doom, Quake
- Networked games
  - Client-server/peer-to-peer
  - Organizing a client-server game as a set of Java classes
  - Interface and game-play issues for networked games
- Selling a computer game
- Review important points in game design and game implementation

Table 1. Detailed outline of course topics.

the usual rough spots one might expect from offering a course for the first time, the course succeeded in many respects beyond my expectations. Each of the potential advantages listed above proved true, and I became convinced that courses developed along these lines create excellent opportunities for upper-division students to "bring it all together", integrating many of the lessons of a typical computer science curriculum into a coherent and educational package.

3 Course Materials

As there are so far not many college courses on computer games, neither has anyone so far developed a college textbook on the subject. The books that do exist on the subject mostly present themselves as "self-teaching" books. I did identify one book, *Black art of Java game programming*, by Fan, Ries, and Tenitchi [2], that served the purpose of a textbook fairly well. I was also able to find a few books that I used primarily for my own reference in developing projects and lectures [5, 10, 12, 17]. Finally, I supplemented the textbook with a variety of readings available on the world-wide web [1, 11, 14].

One of the primary difficulties in finding a good textbook was a problem that is endemic to much of computer science education; these books quickly become outdated. One of the books that best described computer game concepts [12] was written in 1994, and its examples were written for DOS using a mixture of C and assembly language. In just

those few years, much of that book's content has simply become obsolete. Even the textbook I selected used version 1.1 of Java, while I encouraged the students to use the most current version (Java version 1.2). However, I feel the Fan, Ries, and Tenitchi book provided the best combination of conceptual discussion, emphasis on object-oriented design, and illustration of examples with understandable code. This particular book pretends not even to assume any prior knowledge of Java (although it certainly requires previous programming experience), but I did not make that assumption about my own students.

4 Language Issues

I may justly be criticized simply for selecting Java as the programming language for a course on computer games, because many of the most modern computer games demand much more processing efficiency, and much more sophisticated libraries, than Java currently comes anywhere close to providing. However, in addition to a variety of pedagogical reasons, I felt that Java at least makes for a very clean prototyping language, and it is easy enough to use that the students could concentrate on the higher-level concepts of game design and implementation. The sacrifice was that we ignored some important issues of efficient design, because there were some types of games we simply could not build in Java. Happily, in the time since I taught this course, Sun Microsystems has made significant strides in developing more sophisticated Java libraries. New APIs exist for two- and three-dimensional graphics processing, as well as processing of sound and multi-media data [15, 16], so a future version of the course might find Java even better suited for game development.

5 Course content

Table 1 provides a somewhat extensive outline of topics covered in the course. Because of the level of detail in the table, I will not describe each topic in further detail in this text. Suffice it to say that the course touched on a number of issues from a variety of subdisciplines in computer science, including object-oriented design, computer graphics, operating systems, threaded programming, simulation, system design, interface design, networking, peripheral management, sound processing, software engineering, and 3-D processing. A key element of this course is that the final project (described in the next section) required students to understand and integrate each of these elements into a coherent, robust, and entertaining piece of software. Because of this, the final project provides an appropriate "punctuation mark" on an undergraduate study of computer science. With an added focus on history, social and design issues, and the mathematics of physical simulation, this point becomes even more prominent for the study of computer science at a liberal arts institution.

6 Course Projects, Assignments, and Tests

Much of the course involved small projects and assignments to exercise the concepts covered in the reading and lectures. The initial project went together with discussions on game design, genres of computer games, and interface issues. It also provided a nice "ice breaker" for the course. The students were required to play and rigorously review a commercial computer game. The intent of this project was to get students used to thinking about what makes different types of computer games good or popular, and what types of design issues they should keep in mind for their own games. The discussion focused particularly on identifying unifying themes of design and implementation across the wide variety of types of games.

The second project required the students to take an existing game that I developed (a very simple pong game, where a paddle plays "hand ball" against walls), and extend the existing object-oriented structure into a simple version of a "breakout" game (where a paddle bounces a ball into bricks, which then disappear from the screen). Aside from implementing initial game concepts (such as sprites and animation), this required students to build a new system within a pre-established framework involving a moderate number of classes and objects. Students also learned how well "game worlds" can be represented by objects and their interactions. Thus, this project also served as an exercise in the usefulness of object-oriented design where its application is very natural, and as a small-sized exercise in software engineering.

The third project required the students to turn their "prototype" breakout game into a professionally styled game including sound effects, bitmaps and animation, game-play extensions, and an overall atmosphere or theme for the game. Again, as well as focusing on game-specific concepts (such as sound, bitmapped animation, collision detection, and sophisticated sprite management), the students had to make significant extensions to their own design and implementation. This once again provided a glimpse at a more realistic software cycle than students experience in many computer science courses.

The final two significant homework projects focused more specifically on isolated issues discussed in class. The first was an exercise in using matrices to transform coordinate systems for computer graphics. Although most graphics libraries insulate today's programmer from such low-level issues, this homework reflected my opinion that programmers should understand to some extent the tools they are using. This also provided an opportunity for students to experience some "hard-core" elements of computer graphics, at a college that otherwise does not regularly offer a course on computer graphics. This experience was supplemented by the availability of sophisticated tools for 2-D and 3-D graphics, which the students used to build graphics for their games. The final assignment involved various methods for modeling different types of physics within a game world. This assignment required students not only to learn some simple types of mechanics, but how to simulate those mechanics by programming a simple virtual world.

A primary focus for the course was a large final project (there was no final exam in the course). Students worked individually, or in groups of 2 or 3, on a game of their choosing. They could choose almost any genre of game they desired, and the amount of work required depended on the type of game and number of group members. Everyone was at least required to produce a working prototype by the end of the course (the games were then put on display for the campus community). In addition, the project requirements specified that each student demonstrate competency in a set of core issues, including good quality graphics and animation, sound effects and music, efficient sprite management, modeling the world with physics-based and/or state-based programming, good user interface design and implementation, and building efficient and modular code. The students were also encouraged to incorporate other elements of computer game technology, such as three-dimensional graphics and animation, high-fidelity physics modeling, multi-player or networked game playing, and sophisticated sprites or intelligent agents. Following is a summary of the final projects generated by the members of this class:

1 Three-dimensional automobile racing

2 Graphical adventure through the sewers of Manhattan

3 Networked, multiplayer blackjack

4 Strategic exploration and combat

5 Role playing and puzzle solving in a fantasy world

6 Graphical interactive fiction for children

7 "Super breakout" with puzzles inspired by academic life at Colby College

The final project began with initial design proposals just before the halfway point in the semester. In retrospect, and in response to student comments, I recommend that future courses start the initial design very much toward the beginning of the course, and involve the final project throughout the course, in order to give the students more time to implement various design ideas. Naturally, this can sometimes be a tricky tradeoff against the smaller projects that are also part of the course.

Finally, there was a single midterm examination about two thirds of the way through the course. This exam enabled me to have an additional point of evaluation for the students, particularly with regard to some of the course concepts that might not get adequately exercised in the course projects. At the time of the midterm, the course had covered most of the basic concepts of game design and implementation, and we were yet to discuss some advanced concepts such as three-dimensional graphics and networking.

7 Ulterior Motives

Aside from my own interest in (and enjoyment of) computer games, I did have some ulterior motives when developing this course. My own research involves the development of intelligent, synthetic characters, for real-time, simulated environments [7, 9]. Aside from the commercial possibilities, state-of-the-art computer games provide a wonderful environment for research into interactive intelligent agents. Although I was not able to include a unit on "game AI" in this version of the course, I certainly hope to draw some students from the course into this interest. It is no coincidence that instructors for *all four* of the other courses I have mentioned are active researchers in the field of artificial intelligence. As it turns out, all of the other courses have included more of an emphasis on "game AI" in their courses than I did in mine.

I believe such ulterior motives are entirely appropriate, and, for a computer games course, should by no means be restricted to the area of artificial intelligence. I can easily imagine computer games courses with special emphases on other areas of computer science, such as computer graphics, networking, user interface design, social implications of computing, etc. As I hope I have conveyed, each of these areas ought to be addressed during the course of computer game design and implementation, and there is ample room for an instructor to emphasize favorite issues.

8 Closing

A course in computer game design and implementation provides a wonderful integration of many concepts in computer science for a heavily project-oriented course. Judging from my final evaluations of the students, together with their evaluations of me, the students were enthusiastic (even if slightly over-worked) and effectively learned quite a bit of material and a number of skills, even though they had been exposed to many of these concepts in relative isolation in previous courses. I personally was pleased to be able to cover such a wide variety of technical concepts in front of an engaged audience.

Because this was an experimental course at Colby College, it does not currently appear as a regular entry in the catalog or curriculum. Whether it will be taught again at Colby depends in large part on resource availability and the plans for future academic directions within the department. In any event, I endorse the notion that an upper-division course on computer game design and development would fit quite well into most undergraduate computer science curricula. The courses at the University of North Texas and the University of Michigan are regular offerings. I am also aware of one university in the United Kingdom that now offers *two* separate courses on developing computer games. I believe that additional colleges and universities should strongly consider offering such a course.

Materials from my course, including lecture notes, project descriptions, and some software, are currently available on the world-wide web [6]. This web site also includes some of the student projects, which can be run (at least on some browsers) as Java applets. The final projects (as of this writing) are not yet available on the web, but are available for educational use by getting in touch with me.

References

[1] Crawford, C. The art of computer game design, 1997. Online. Internet. [November 30, 1999]. Available WWW: http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html.

[2] Fan, J., Ries, E., and Tenitchi, C. *Black art of Java game programming*. Waite Group, 1996.

[3] Forbus, K., and Bachmann, A. Computer game design, 1998. Online. Internet. [November 30, 1999]. Available WWW: http://www.cs.nwu.edu/academics/courses/c95-gd/.

[4] Hodgins, J., Bruckman, A., and Metoyer, R. Electronic game programming, 1998. Online. Internet. [November 30, 1999]. Available WWW: http://www.cc.gatech.edu/classes/cs4803_98_winter/.

[5] Holder, W., and Bell, D. *Java game programming for dummies*. IDG Books Worldwide, 1998.

[6] Jones, R. M. Design and implementation of computer games, 1999. Online. Internet. [November 30, 1999]. Available WWW: http://www.cs.colby.edu/~rjones/courses/cs398/.

[7] Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P. G., and Koss, F. Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*(1) (1999), 27-41.

[8] Laird, J. E., and Assanie, M. Computer game design and implementation, 1998. Online. Internet. [November 30, 1999]. Available WWW: http://ai.eecs.umich.edu/soar/Classes/494/.

[9] Laird, J. E., and Jones, R. M. Building advanced autonomous AI systems for large scale real time simulations. *Computer Games Development Conference* (1998). Long Beach, CA.

[10] Lamothe, A. *Windows game programming for dummies*. IDG Books Worldwide, 1998.

[11] Nelson, G. The craft of the adventure, 1995. Online. Internet. [November 30, 1999]. Available WWW: ftp://ftp.gmd.de/if-archive/info/Craft.Of.Adventure.letter.ps.

[12] Lamothe, A., Ratcliff, J., Seminatore, M., and Tyler, D. *Tricks of the game programming gurus*. Sams Publishing, 1996.

[13] Parberry, I. Computer game programming, 1999. Online. Internet. [November 30, 1999]. Available WWW: http://mycroft.csci.unt.edu/csci4050/.

[14] Sawyer, B. The getting started guide to game development, 1995. Online. Internet. [November 30, 1999]. Available WWW: http://www.cs.colby.edu/~rjones/courses/cs398/lectures/gamfaq.txt.

[15] Sun Microsystems. Java 2D API, 1999. Online. Internet. [November 30, 1999]. Available WWW: http://www.javasoft.com/products/java-media/2D/index.html.

[16] Sun Microsystems. Java platform optional packages, 1999. Online. Internet. [November 30, 1999]. Available WWW: http://www.javasoft.com/products/OV_stdExt.html.

[17] Tieskotter, K. *Black art of Macintosh game programming*. Waite Group, 1996.