Student Teamwork: A Capstone Course in Game Programming

Robert Kerbs
California State Polytechnic University, Pomona, Computer Science Department,
Pomona, CA 91768 rwkerbs@csupomona.edu

Abstract – **Many computer science (CS) undergraduate programs have experienced dramatic reductions in the number of incoming freshman and transfer students in recent years. Simultaneously, industry feedback informs us that graduating students should have experience working with others and on long-term software projects. This requirement is due to the fact that most industry projects are team-based, often take months or years to complete, and are often global in nature. Institutions have addressed these issues in a number of ways. Cal Poly Pomona's CS department created a one-term ten-week project course in game programming offered during spring 2006. Teams were assembled comprising of four or five students. The final deliverable for each team was a CD which included a working 3D game and all design documentation (i.e. UML diagrams, source code, etc). This paper discusses the design, implementation, and results of this course. It also describes the outstanding efforts put forth by the students.**

Index Terms - Game programming, Interactive multimedia, Software engineering, Retention.

INTRODUCTION

While the Bureau of Labor Statistics expects *computer scientists* to be among the fastest growing occupations through 2014 [1], the number of incoming freshman majoring in Computer Science (CS) has decreased more than 60% between Fall 2000 and 2004 [2]. Compounding this decrease is the major's high student attrition rate [3]. Simultaneously, our industry advisory board informs us that graduating CS students should gain experience working with others and on long-term software projects. Most computer science courses assign homework to students individually which can usually be finished in a short period of time. Most software projects in industry, however, require computer scientists to work in teams and on projects that may take months or years to complete. Consequently, many institutions are taking a variety of approaches to deal with declining enrollment and industry need. These approaches include the following: training high-school teachers to better understand, inform, and motivate students about the CS major [4]; introducing team-related orientation activities to incoming freshman [5]; redesigning the first year programming curriculum to include innovations such as the Alice interactive animation system [6].

STUDENT INTEREST AND MOTIVATION

A student's first experience using a computer is often with a gaming system of some type [7]. In fact, upon entering college, students typically have had extensive experience playing games, but do not possess the background in how they are made. This domain knowledge can help motivate students [8] to enter an undergraduate computer science program [9] and successfully learn and apply core computer science topics [10] – yielding higher retention levels in the major. Additionally, because students perceive themselves as solving real-world gaming-related problems [11], the CS major is an attractive one as it potentially prepares students for post-graduate career opportunities. This student profile matches well with the incoming freshman and transfer students characteristic of Cal Poly Pomona (one of two polytechnic universities in the 23-school California state university system).

To address decreasing enrollment, industry concerns, and student preferences, a ten-week course was created and titled Introduction to Game Development. This course focused on undergraduate computer science majors who were seniors and was intended to be a capstone course. This paper outlines the design, implementation, and results of this first course offering and may help others at different universities and institutions.

THE COURSE

The *Curriculum Framework* of the International Game Developer's Association (IGDA) served as the basis for this course [12]. The framework was developed in 2001 to aid institutions in creating game-related courses, programs, and degrees. Cal Poly Pomona is a partner with IDGA and is participating in the update of the document. The course we developed was similar to others [7] as it focused primarily on the *Game Design* and *Game Programming* elements of the *Curriculum Framework.* We had a difficult time finding a book that fit our needs and finally settled on *Introduction to Game Development* [13]. Not only did it provide depth-of-coverage, but it was also based upon IGDA's *Curriculum Framework.*

By design, the course targeted graduating seniors. The idea was for students to work in teams for ten-weeks to design and implement a 3D game that would run from a CD. Not only would the CD contain their working game, but it would also include all the materials utilized in making the game such as storyboards, source code, documentation, critical analysis, and

1-4244-1084-3/07/$25.00 ©2007 IEEE

UML diagrams. The CD would serve as memento of the team's work and could be copied and distributed to potential employers and serve as a portfolio for our CS undergraduates.

Documentation of student work in the form of portfolios is a common practice of many disciplines; namely, those which are visually oriented. The portfolios are commonly used during the interview process to ascertain a candidate's suitability for a given position. While the advantages of constructing student portfolios has led to their use in scientific disciplines [14], portfolios in computer science are primarily used as an assessment tool [15] because institutions [16] have found them to be useful for measuring programmatic outcomes in their curriculum - a necessity for accredited programs. We believe the CD-ROM deliverable described in this paper is a unique portfolio that demonstrates how students progressed from concept to completion of a moderately-sized software project.

The only pre-requisite for this course was the CS department's one-term C++ programming course (our core CS undergraduate sequence uses Java). While it was understood that some students would have completed our computer graphics course, and others had not, it was believed that by working in teams there would be sufficient challenge for everyone due to the wide spectrum of tasks required such as AI, physics, collision detection and handling, and scripting. This approach has been successful elsewhere and introduces students to the variety of challenges associated with large software development projects [17].

We originally offered one section of this course but it filled up in one day, so we opened up a second section. In total, 58 students completed the course.

The course was held in the Department of Computer Science's Software Engineering Laboratory. In 2005 it was redesigned to include hemi-circle workstation tables which allowed each group to work in close proximity. The computers were all 2Ghz P4s with 1GB of RAM, however, the laboratory provides wireless Internet access so students could also use their own laptops if desired.

I. Course Structure

A milestone chart directed the ten-week term (see table I). During the eleventh week (finals week) each team presented their games and a critical analysis of their work to the rest of the class.

In order to gain insight into student expectations and perceptions, a survey was conducted on the first day of class. Results of the survey showed that 39% of the students definitely wanted to become game programmers, 36% of the students wanted to learn how games are made and may seek employment in the games industry, and 25% of the students were taking the course to find out how games are made and would not seek employment in the gaming industry.

TABLE I
MILESTONE CHART

Game Development (CS 499) - Spring 2006 - Milestone Chart - 5/10/06 update
Instructor - Dr. Robert Kerbs

Week	1	2	3	4	5	6	7	8	9	10	Finals
Course Background, Organization, Activities											
Survey	x										
Course Overview	x										
Lua Style Sheet		x	x	x	x	x	x	x	x	x	⊕
C++ Style Sheet			x	x	x	x	x	x	x	x	⊕
Groups Chosen		x									
Critical Stage Analysis										x	⊕
Game Platform (Windows) and Technologies											
Lua	x	x	x	x	x	x	x	x	x	x	⊕
OpenGL			x	x	x	x	x	x	x	x	⊕
OpenAL						x	x	x	x	x	⊕
AutoRun								x	x	x	⊕
CD Music Streaming									x	x	⊕
Game Design											
Concept / Story	x	x	⊕	•							⊕
Storyboard			x	⊕	x	⊕	x	⊕			
Level 1 - design					x	⊕	•	•			⊕
Level 2 - design							x	⊕	•	•	⊕
Level 3 - design									x	⊕	⊕
Menus / Screens			x	x	x	x	⊕	•	•		⊕
Artwork				x	x	x	x	x	x	x	⊕
Sound Effects								x	⊕	•	⊕
Music									x	⊕	⊕
Player-vs-Machine				x	x	⊕	•	•	•	•	⊕
Player-vs-Player						x	x	⊕	•	•	⊕
Kiosk Mode						x	x	x	x	⊕	⊕
Game Systems											
Game Engine		x	x	⊕	•	•					⊕
Modeling			x	x	x	x	⊕	•	•	•	⊕
Memory Manager			x	x	⊕	•					⊕
Collisions, Physics, AI (Levels 1 & 2, 3)					x	x	x	⊕	x	⊕	⊕
Week	1	2	3	4	5	6	7	8	9	10	
Group Presentations			⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕

x = Development
⊕ = Major Milestone + Design Review + Deliverable Due Date
• = Iterative Refinement (as your time permits)

Regardless of the motivating factor, it was anticipated that students who did not play games would do as well as those who do play games [18]. This turned out to be true; however, only 10% of students did not play games.

After the survey was conducted, students were presented the milestone chart and lectures were given describing the game development process. Students were introduced to the Lua scripting language to handle parameterized input and would later facilitate changes to each game without requiring recompilation. Most students did not have experience with scripting languages, so this early introduction was imperative. Each student was required to devise their own story/concept for the game they wished to undertake. This allowed students to explore their creativity in a way they probably would not have experienced previously or otherwise [18].

During the second week, students selected their own teams based upon one another's computer science background and the type of game each student wanted to develop. We ended up with twelve teams with four members and two teams with five members. Virtually each of the fourteen teams had an idea of the type of game they were interested in producing. Since others have experienced success relegating this decision to the students [19], we decided to follow this practice. The profile of selected game genres were as follows:

- Four adventure games
- Five puzzle games (example shown in figure 1)
- Two shooters (example shown in figure 2)
- One racing game
- One rhythm game
- One RPG (Role-Playing Game)

FIGURE 1
STUDENT'S VIRTUAL PUZZLE WORLD MIMICKING CAL POLY'S QUAD

FIGURE 2
STUDENT'S COMPUTER VIRUS WORLD – THE ENEMY IS THE STEALTH VIRUS

For the remaining weeks, short theoretical lectures and straight-forward examples were presented. Students were given time to plan out strategies for completing upcoming milestones and iron-out implementation details. Experience with this format served as a helpful method for guiding students. When large programming APIs are utilized by students, they may be overwhelmed with the possibilities and consequently experience difficulty selecting the subset necessary to best address the task at hand [19].

II. Programming Environment

Microsoft Visual Studio was used as the Integrated Development Environment. As with some other programs [20], we utilized cross-platform, non-proprietary APIs. Open Graphics Library (OpenGL) 3D graphics API was used for rendering and Open Audio Library (OpenAL) audio API was used for audio. The scripting language Lua was used for setting state variables. C and C++ were used for the programming languages.

III. Software Engineering

Software engineering courses are a natural fit to introduce game-related problems because game design entails foundational software engineering principles [8]. Additionally, by requiring students to complete their projects in one-term, they are exposed to a fairly complete software cycle [17]. Since this was not strictly a software engineering course, a subset of these principles was selected.

The primary software engineering tool used was Unified Modeling Language (UML) class diagrams. These diagrams articulate the structure of a software system by showing its classes, class attributes, and the relationship among classes (see figure 3).

COURSE RESULTS

Of the fourteen teams, twelve successfully finished their games. Interestingly, the two teams that were unsuccessful were those that had five people on their team (all the other teams had four members). While we followed [19]'s lead by offering a single-project course, we handled grading differently. At each milestone (see table I), each team was graded on how well they met the requirements. Halfway through the course, team members confidentially graded one another. Upon course completion, team members once again graded one another. It was interesting to see how the grading changed from the halfway point – in many cases, a leader emerged. Finally, each team presented their games to the rest of the class and was graded by the other teams.

Part of the final presentation included a variation of a critical stage analysis (CSA) [21]. A CSA is a way to query all team members at different points in the development process. The purpose is to determine what is going right, what is going wrong, and what should be changed at future stages in the project. Since the class was only ten weeks long, we had teams perform the analysis once, after the game was completed, so it served as more of an overall critical analysis rather than a formal CSA. Feedback obtained from these analyses will be invaluable for future offerings of the course.

Students were encouraged to use simple OpenGL primitives to represent their virtual worlds. The aim was for students to concentrate on the programmatic elements of building a game and not spend excessive time learning a specific modeling program (artists would serve this function in

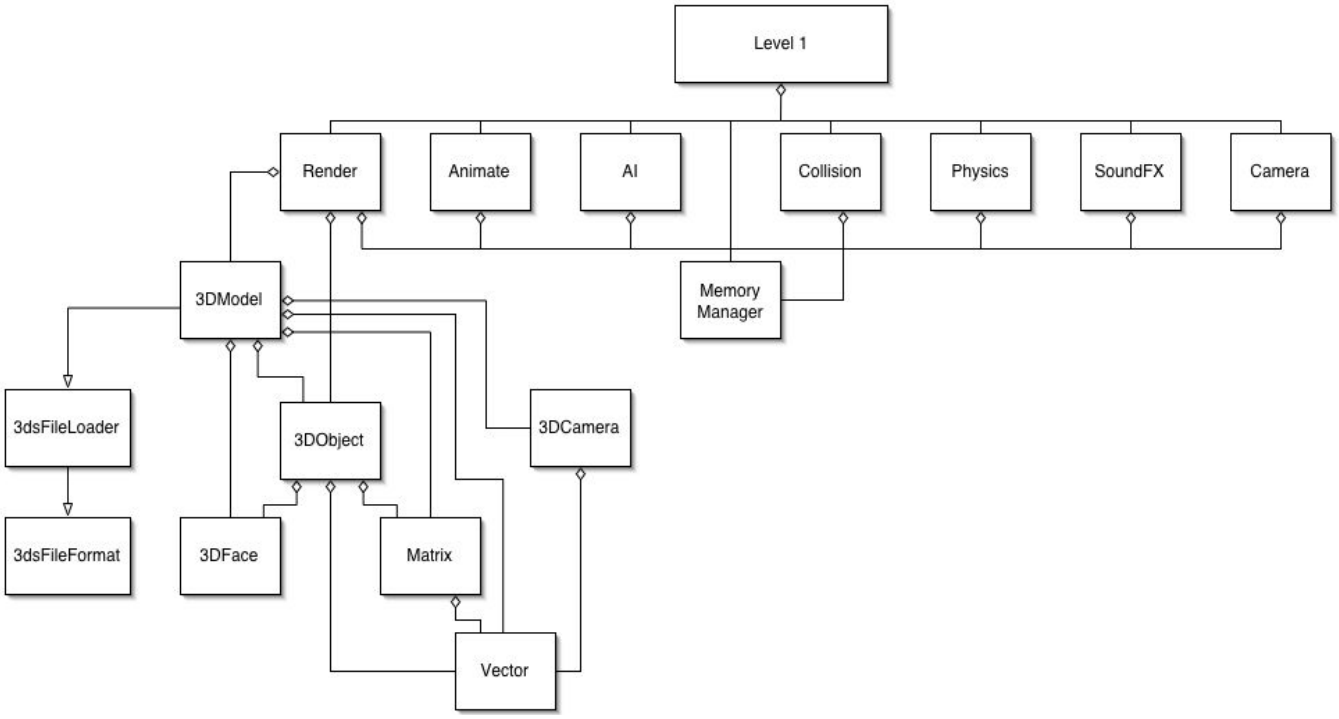

FIGURE **3**
STUDENT UML DIAGRAM

the real world anyway). Students were discouraged to utilize modeling programs such as Maya, 3D Studio Max, and MilkShape 3D. The most interesting and complete games were those that did not us the aforementioned programs – perhaps demonstrating the demanding nature of the programmatic elements of the course. Overall, the games proved to be of better quality than anticipated. Additionally, because each game was different, and students had to work together to complete each milestone, the completed projects were all unique. In addition, students felt empowered [22] (see figure 4).

DISCUSSION AND FUTURE WORK

The team format resulted in students enhancing their programming and problem-solving skills. The two five-person teams did not complete their projects on time – the leadership element seemed to be missing. In the future, we plan to limit teams to no more than four people in size for a 10-week (four hours per week) course.

Students were extremely motivated. They spent more time than anticipated ([18] experienced the same result). In fact, two teams invested so much time that they created over 10,000 lines of code (LOC) in ten weeks (i.e original code, not repackaged code from the APIs). The median LOC output was about 6,000.

There were some areas that could be improved with future offerings of the course. A stable source-code control would allow students to better utilize their time. While students were able to individually get their code to work, when coalesced with the remaining team member's code, an exorbitant amount of time was spent getting it to work together. In the future, we plan to implement a source-code control system where students would be able to check files in and out of a repository - similar to industry practices. Additionally, it would useful to administer the student survey at the end of the term to see how the course experience might have affected student expectations, perceptions, and future ambitions.

FIGURE **4**
STUDENTS CELEBRATE A JOB WELL DONE

ACKNOWLEDGMENT

Thanks are due to Dr. Mandayam Srinivas, Associate Dean, and Dr. Don Straney, Dean, College of Science at Cal Poly Pomona, for their encouragement and support for this effort. In addition, I would like to thank the faculty in the computer science department for their comments and feedback.

REFERENCES

[1] Bureau of Labor Statistics, "Occupational Outlook Handbook (OOH), 2006-2007 Edition", *http://www.bls.gov/oco*, accessed March 1, 2007.

[2] Vegso, J., "Interest in CS as Major Drops among Incoming Freshman", *Computing Research News*, 17, No 3, May 2005.

[3] Beaubouef, T. and Mason, J., "Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations", *ACM SIGCSE Inroads Bulletin*, 37, No 2, 2002, pp 103-106.

[4] Blum, L. and Cortina, T., "CS4HS: an outreach program for high school CS teachers", *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, 2007, 19-23.

[5] Talton, J., Peterson, D., Kamin, S., Israel, D., and Al-Muhtadi, J., "Scavenger hunt: computer science retention through orientation", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 2006, 443-447.

[6] Cooper, S., Dann, W., and Pausch, R., **"**Alice: a 3-D tool for introductory programming concepts", *Proceedings of the 5th annual CCSC northeastern conference*, 2000, 107-116.

[7] Coleman, R., Krembs, M., Labouseur, A., and Weir, J., "Game Design & Programming Concentration within the Computer Science Curriculum", *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 2005, 545-550.

[8] Sweedyk, E. and Keller, R., "Fun and Games: A New Software Engineering Course", *Proceedings of the 10th Annual SIGCSE Conference on Innovation and technology in Computer Science education*, 2005, 138-142.

[9] Chamillard, A. T., "Introductory Game Creation: No Programming Required", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 2006, 515-519.

[10] Wolz, U., Barnes, T., Parberry, I., and Wick, M., "Digital Gaming as a Vehicle for Learning", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 2006, 394-395.

[11] Stevenson, D. E., and Wagner, P. J., "Developing Real-World Programming Assignments for CS1", *Proceedings of the 11th Annual SIGCSE Conference on Innovation and technology in Computer Science*, 2006, 158-162.

[12] International Game Developer's Association, "IGDA Curriculum Framework. Report Version 2.3 Beta", 2003.

[13] Rabin, S., "Introduction to Game Development", Charles River Media, Boston, MA, 2005.

[14] Patten, A. and McGill, M., "Student Portfolios and Software Quality, Metrics in Computer Science Education", *Journal of Computing Sciences in Colleges,* 21, No 4, 2006, 42-48.

[15] Sanders, K. and McCartney, R, "Program Assessment Tools in Computer Science: A Report from the Trenches", *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education,* 35, No 1, 2003, 31-35.

[16] Blandford, D. and Hwang, D., "Five Easy but Effective Assessment Methods", *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education,* 2003, 41-44.

[17] Jones, R., "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*, 2000, 260-264.

[18] Lewis, M. and Massingill, B., "Graphical Game Development in CS2: A Flexible Infrastructure for a Semester Long Project", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 2006, 505-509.

[19] Parbarry, I., Kazemzadeh, M., and Roden, T., "The Art and Science of Game Programming", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 2006, 510-514.

[20] deLaet, M., Kuffman, J., Slatterly, M., and Sweedyk, E., "Computer Games and CS Education: Why and How", *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 2005, 256-257.

[21] Hamann, W., "Goodbye Postmortems, Hello Critical Stage Analysis", *http://www.gamasutra.com/resource_guide/20030714/hamann_01.shtm*, July, 2003.

[22] Lewis, M. and Massingill, B., "Graphical Game Development in CS2: A Flexible Infrastructure for a Semester Long Project", *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 2006, 505-509.