Developing a Game Programming Course For Computer Science Majors In A Liberal Arts College

Connie Yu

Abstract—This position paper examines pedagogical issues related to game programming courses in small computer science departments in liberal arts colleges in the United States of America.

Index Terms—Video game, 3D game programming, computer science education.

I. INTRODUCTION

Electronic game programming courses are rapidly gaining acceptance by institutions of higher education in the United States. Game design programs vary widely and are mostly interdisciplinary [1], [2]. While some game making courses are more suitable for design school students than computer programmers, other courses draw programmers of the highest caliber. There are non-credit vocational training courses as well.

I teach students who belong to another type of student body. The computer science department of New Jersey City University has a program typical of a program in a liberal arts college. The students graduate from the program either continuing their graduate education or pursuing careers in IT. The students are expected to have a broad-based, not vocationally focused, liberal arts education. Most of our students have no engineering background.

Does game programming have a place for the large number of students enrolled in computer science programs in liberal arts colleges just like mine? If so, what should the course components be, why are they important, what are the academic justifications for them, and, to be practical, how can such curricular additions be implemented in terms of faculty training and laboratory facilities?

II. 3D GAMES AND HIGHER EDUCATION, THE LARGE PICTURE

There are many enthusiastic faculty members of computer science departments in institutions like mine in the United States who want to add game programming courses to their programs. Their number one task is to convince colleagues and administration that game production is a serious line of inquiry.

At my institution, students, in groups or as individuals, asked when, if ever, we would offer a game programming course. These students are fully aware of the popularity of 3D video games, and ask for computer science courses to get them familiar with the production technologies.

There are people who have serious doubts about the addition of game courses: When UC Irvine rejected a Game Design degree proposal, according to Wired [3], a university official explained that the university did not want to run “the strong risk of attracting people on the basis of prurient interest.” Luckily, there has been a decrease in resistance to the addition of game development as a respected academic subject.

By electronic games, I refer to the latest 3D video or computer games played on a PC or Macintosh, consoles such as PlayStation 2, Xbox and GameCube, or a handheld games console or other mobile devices like cell phones. Game programming hence can include the technology of programming of 3D games for PCs, game consoles, and network games.

In the last couple years, the acceptance of electronic game development as a legitimate academic program has been slowly gaining ground in the United States. The highlight of the educators’ program in the SIGGRAPH 2002 Conference held in San Antonio, Texas during July 21-26, was IGDA (the International Game Development Association)’s presentation of its game design curriculum guidelines and framework proposal. IGDA has a High School Outreach program, a Student Scholarship program, and a very active Education Committee. However, in the United States, the availability of game programming education is still lagging behind Europe and other areas.

Computer gaming and the associated culture has been with us for a long time [4]. Many seasoned programmers will tell you that their original motivation for getting into computer science 20 years ago was to make computer games. Many computer science students today are the same. This generation of computer science students have grown with the maturity of the video game industry and been aware of the existence of potential careers in this industry. The games are part of them; they want to have a deeper understanding of the technology involved in making these games. Wisely, most students in our department are not limiting themselves to today’s hot technologies, but are interested in completing a traditional broad-based computer science education. They don’t
necessarily plan to specialize in game development, yet many
consider 3D game programming in today’s technology a
worthy elective subject. Game programming has been one of
the most challenging fields in software development. The high
demand for real-time graphic rendering and intensive
calculation continuously drives hardware technological
advancement.

Without a doubt, vocational schools will develop and
implement more programs before higher education institutions
in the United States add game programming to their
curriculum. These alternatives can be a very good alternative
to a four-year college for some students. However, game
programming courses in computer science programs in liberal
arts colleges have broader educational goals. It to serve these
liberal arts educational goals that motivated me to develop
game programming courses for computer science majors.

III. GAME PROGRAMMING IN COMPUTER SCIENCE EDUCATION

My department, typical of this kind of program in the
United States, has a well established Computer Graphics
course that follows ACM topics closely. As a typical CG
course, it covers hardware and software for computer graphics,
2D and 3D graphics basics including geometry, and
algorithms. It is typically implemented in C, and lately, using
the graphic library openGL. Students have high expectations
and demands for the course, yet some were disappointed with
the outcome: As a computer graphics course, game specific
materials are rarely included. Many computer science students
who successfully completed the computer graphics course
wanted a project-based follow-up course, specifically in 3D
game programming. Additionally, there are students who do not
want the full treatment of computer graphics. Instead, they
want a course centered on techniques related to 3D games with
emphasis on real time rendering.

Arguably, the essential elements of game programming are
covered by a broad-based computer curriculum already.
Mathematics and physics courses teaching the mathematics
and physics concepts needed for game programming are
available for liberal arts students. In addition to computer
graphics, we offer an introductory course in algorithms and
one in artificial intelligence. Including a video game related
project assignment in either course, hypothetically, is possible.
Certain software-level game programming techniques, for
example, particle systems, can be included as a case study in
any programming course introducing the concept of objects.
Game development process management can be included in a
software engineering course. On the hardware side,
programmable GPUs and instruction sets like MMX can be
covered when assembly language is taught, and console
architectures can be included in computer architecture courses.
And so on.

This suggestion I made in the previous paragraph for
introduction of game programming without a specific course
makes sense only for a large department with a research
faculty and wide range of electives. On the other hand, these
departments can easily offer a game programming course if
they decide to do so. Sadly, small computer science
departments in liberal arts colleges have neither option readily
available to them. Realistically, we can not expect game
programming related projects to be included in any of the
traditionally offered courses.

Many computer science degree programs have a capstone
project requirement. In departments in institutions with a low
student/faculty ratio, a student can elect to make his/her
capstone project a game design project. Unfortunately, in my
department, there is a high student/faculty ratio, and directing
individual student game design projects are out of the
question.

If we want to expose students to game technology and
designs, our best strategy is to offer an elective course that
introduces them to all aspects of electronic game production
including programming for real time, console architecture, and
game development process management.

The main supporting argument for a game programming
course is the importance of teaching collaboration and process
management. The game production process has become so
complex that talents have to be drawn from many different
fields, including people trained in computer science and
people who are not.

Interestingly, it is only after video game making left the
realm of computer science and entered everyday life
entertainment that it rapidly grew into a multi-billion industry.
Admittedly, the development of new programs in game
development and game study will mostly likely be outside
computer science or at least interdisciplinary. Programming
skills are downplayed in the process of game design, and game
development jobs will open up more for artists than computer
science graduates.

On the other hand, the game programmers now need a more
theoretic understanding of mathematics and physics, a grasp of
the architecture of the game equipment, be it a PC or a
console, algorithm analysis and implementation, optimization,
artificial intelligence, computer graphics, object-orientation,
and software engineering. All these are at the core of an
undergraduate computer science education. Putting all of the
necessary pieces into an introductory, group-project based
course is where its values lay; the whole is more than the sum
of its parts.

IV. EQUALITY ISSUE

There are a large number of studies on the imbalance of
genders in electronic gaming [5]. While this is an obvious
reason to worry, it is more worrisome that the imbalance exists
in electronic game production. It is not clear to me how the
lack of video gaming experience will harm a girl or woman,
which may not be at all, but it is clear that the lack of women’s
participation in game production is not healthy for the game
industry and the society. The same is true concerning the lack
of minorities in game production.

Computer science majors in liberal arts colleges generally
enroll more females. Institutions like New Jersey City University have a large percentage of minority students in the computer science program. Students enrolled in game design vocational programs tend to be self-selected with clearly defined career goals. In contrast, many students in a liberal arts environment are “undecided” on their career goals, and are encouraged to try without fully committing themselves. My hope is that the game programming courses under development in New Jersey City University will attract a wide range of students, especially females and under-represented minorities. Ideally, more game design courses can be developed in collaboration with other departments later.

V. PEDAGOGY OF THE GAME PROGRAMMING COURSE

A game programming course for computer science students is not easy to design, aside from convincing the curriculum committee that the subjects are important to a computer science education. The typical concerns of designing a course in a liberal arts education are:

- student learning outcomes and course goals
- instructional procedures
- pre-requisites
- contents
- assessment and evaluation
- programmatic outcomes
- resources

Here I only highlight a few issues related to the course design.

A. Learning outcomes, goals and assessment

Due to the complexity of the process of game production in the real world, collaboration is required. It is only natural to make game programming courses group-project based. It is clear that groups are to produce completed “products,” typically a 3D game engine as in today’s technology, or other implementations of algorithms. Students should be evaluated on their grasp of the elements of game programming and mastery of certain techniques. While students may view a video game CD that they may one day show a potential employer as the outcome of the course, we view the course outcome as something broader.

The learning outcomes and goals must be thought of carefully because of the rapid advancement and short life span of any technology related to game programming. Whatever specific techniques students learn now may be passe techniques the day they are out of college. While computer science in general is rapidly evolving, this applies to electronic game programming even more so. Learning game programming means dealing with emerging technologies. What is done in software now may be picked up in hardware in the next generation of video cards. Certain techniques hard to master today may be obsolete when the bandwidth bottleneck of today is resolved. While it is fascinating to learn the architecture of PlayStation 2 console’s unique design, PlayStation 3s may be unrecognizable and completely different. With this in mind, one goal of the course must be to help students become better learners of new computing technology. A lot of hardware and software used for game programming will be new to everyone, students as well as instructors. This provides a perfect opportunity for students to experience the IT professionals’ life of continually upgrading skills.

Therefore, the assessment and evaluation of student performance is also based on a combination of tangible measurements of their research of available technology and their applications of concepts. Measurement of students as team members is also a vital part of the assessment process.

B. Mathematical rigor and applications

Broad-based computer curricular stress theoretical foundations and require a certain level of mathematical sophistication and an understanding of science [6]. While game design is now considered closer to art than to computer science, paradoxically, the mathematical preparation of next generation game programmers increased significantly. From my own experience in game programming, the majority of current computer science students in liberal arts colleges do not have the game mathematics or physics background. For example, mathematics courses like differential equations are not required for computer science majors because our program is not an engineering program. No matter what the pre-requisites of the game programming course are, the hours will have to be spent on the mathematical foundation. However, the emphases have to be on applications rather than theory. The course actually offers an opportunity to motivate students who usually try to avoid mathematics, if the applications of mathematics in game development are introduced in a non-trivial, hands-on and interesting manner, for example, the demonstration or even implementation of quaternion operations in animation.

C. Content and instructional procedures

The contents of the course need to be flexible. This is essential, because it is hard to tell for sure what will still be worth teaching in the near future, like, next semester. I am an “opportunist” when it comes to changing course content from semester to semester. I view this as an opportunity to revisit some ignored core computer science knowledge. For example, most contemporary students learn programming on fairly powerful PCs. I consider it lucky that we still teach low level programming, and only introduce students to IDEs later to a limited extend. The programming skills of graduates of computer science programs decline when they see no reasons for efficiency. A central issue of real time game programming, however, is optimization. Hopefully, students will learn lessons from solving the mystery of how PlayStation2 gets those stunning visuals with only a 32MB system memory and 4MB graphic memory [7].

Since the majority of my students are beginning programmers, they must be given a safety net of some sort. Course materials like ready-made models and additional course specific library need be developed to make even
beginning game programmers feel rewarded from learning. On the other hand, we need to concentrate on game programming specific skills, and leave other topics like windows programming and 3D modeling to a minimum.

D. Resources

The most difficult part of developing this course is the issue of resources. Clearly, we cannot afford professional level hardware and software. Luckily, the rapid expansion of the electronic game industry has made free or cheap tools readily available. Overspending on department laboratory may not be really necessary. Game programming puts students much closer to hardware than they would for other programming projects. Students generally prefer to set up the development environments on their own computers rather than department laboratory computers for project. However, it can be very hard for any instructor when many different machines and video cards may be used.

Training faculty in game technology is another issue. Ideally, students will be benefit most if the instructor is actively engaged in professional practice. We are lucky at this aspect because New Jersey City University is located in Central New Jersey. It is not difficult to link game industry professionals with instructors and students, or to find guest lecturers. However, small colleges not located near a major electronic game technology hub will have a very hard time getting their students in contact with professionals.

VI. CONCLUSION

The greatest educational benefit of any technology is reserved for those who produce rather than just consume. As complex and cutting-edge as real time electronic games are, more people, including females and minorities need to have access to an education in the process of making them. Small computer science departments in liberal arts colleges need to follow the lead of the programs in research universities that make game programming available as an elective, with significant modifications to meet the needs of liberal arts students.

The strategy for small departments with limited faculty expertise and resources is to offer a group-project based introductory game programming course stressing collaboration and the need to adapt to rapid advancement in hardware and software.

REFERENCES


