FROM 2D TO CONSOLES: A THREE-SEMESTER COMPUTER GAME DEVELOPMENT CURRICULUM

William Birmingham, David Adams
Computer Science Department
Grove City College

Introduction

Computer-game development is immensely popular with undergraduate computer-science and computer-engineering students. More importantly, the design and development of computer-games is an excellent pedagogical opportunity: developing games integrates a great number of the subjects students learn throughout their undergraduate experience[1]. This integration of topics, coupled with student driven motivation to learn, is an important step for students allowing them to utilize tools from programming and graphics to calculus and physics; from data structures and algorithms to computer hardware to name just a few subjects[2]. From a teaching perspective, computer-game development is great fun to teach as the students are highly motivated and the subject matter, while very challenging, is fun!

At Grove City College (GCC), we have developed a comprehensive three-semester sequence in computer-game development. The sequence is designed to take students from interactive fiction and 2D arcade-style games to sophisticated console game development. The first two courses in our three course sequence stress computer gaming fundamentals in 2D (the first term) and then 3D (the second term). In these courses, we cover a wide range of topics from software architectures for game design to fundamentals of game development including algorithms, data structures, graphics (including Microsoft XNA) and techniques for good game play.

In the third term, our students gain experience with console gaming using the Microsoft Xbox 360. Console game development is challenging, as the students not only must use sophisticated game development techniques learned in the previous two courses, but they must apply their knowledge of networking.

In the paper, we describe our curriculum, as well as the pedagogical techniques we use. In addition, we discuss many of the issues in delivering the curriculum, particularly at a small college.

Related programs

Overcoming a reputation synonymous with wasting time, game programming is being incorporated into academic programs creating new classes and opportunities for students to work on very sophisticated and technically relevant applications during their undergraduate education. Programs, like that of North Texas, incorporate game design with a focus on getting students into the gaming industry and have had reasonable success[3].

In contrast, many programs are aimed at simply increasing student motivation to explore current hot technologies and programming techniques on a large project and to work in multi-disciplinary teams. For example, the College of New Jersey offers a design course where students from a variety of disciplines, including the arts, work on a game. Other programs range from minor incorporation of gaming into programming assignments and capstone projects to the kind of full scale degree programs offered by Full Sail, Digipen, and The Guildhall at SMU. Still others have programs that focus on the use of gaming as an aid or driver for learning concepts using 3D environments in novel ways like at the University of North Carolina Charlotte4 or the M.U.P.P.E.T.S project at RIT.

A number of universities have research programs in gaming and related technologies. Examples of these schools include: The University of Michigan, Michigan State University, University of Southern California and Carnegie Mellon. The last two schools have academic units for their

research programs. These schools also offer a variety of gaming classes.

Pedagogy and course sequence

The catalog descriptions for the games courses are given in Figure 1. Each course is three credit hours and runs for an entire semester. The first course, Comp 441, is to be taken in the first semester, junior year; the second course, Comp 446, is taken second semester, junior year; and the final course, Comp 447, is to be taken first term, senior year. While we do not have a formal course, some of our students participate in a research project in the second term, senior year, concerning multiplayer gaming or mobile gaming.

The courses are all in a lecture format, with some design work done in class. GCC has a comprehensive Tablet PC program, where all students are given Tablet PCs as freshmen. These machines are powerful enough for much of their work in all three classes, aside from some specialized assignments. Because all students have machines, we do not have labs, aside from one to house the console gaming equipment (as we describe later).

By the time they are juniors, our students have taken three semesters of programming, and have in-depth experience with c++, programming environments, and the c++ Standard Template Library. Moreover, the students have also taken data structures and algorithms, and so are comfortable with designing, analyzing, and coding high-performance algorithms and complex data structures. In addition, they have taken computer organization and operating systems. Thus, they know about threading, memory organization, and processes. All of these things—not just programming—are needed for writing games.

Finally, by their junior year, all the students have taken physics (statics, dynamics and kinematics), three courses in calculus, and linear algebra. Since students need to create all the physics in their games (e.g., how a ball will bounce off the end of the screen), they need to be familiar with physics. Calculus and linear algebra are used extensively in collision detection, graphics and "implementing" physics.

The idea behind teaching a three course sequence in game design is to expose the students to a broad picture of development including art, sound, level design, story, balance, and game engine creation. Our philosophy is to teach the students to create games with great gameplay, not simply interesting technology: the courses focus not only on the programming side of each of these things, but encourage the students to create games that are actually fun to play. We teach algorithms and data structures applicable for use in games. The first two courses involve the creation of graphics and game engines, programming them based on XNA and code from the books we use in

Comp 441. COMPUTER GAME DESIGN AND DEVELOPMENT. This course covers concepts and methods for the design and development of computer games. Topics include: graphics and animation, sprites, software design, game design, user interfaces, game development environments.

Comp 446. ADVANCED COMPUTER GAME DESIGN AND DEVELOPMENT. This course is a continuation of Computer Science 441 and is focused on the development of 3D games and other advanced game programming techniques.

Comp 447. CONSOLE GAME DESIGN AND DEVELOPMENT. This course is a continuation of Computer Science 441 and is focused on the development of console games, with emphasis on both hardware and software design issues. The course will explore sophisticated programming techniques and advanced algorithms. Prerequisites: Computer science 441, 446, and permission of instructor.

Figure 1: Catalog descriptions of the game sequence.

class. The students make significant modifications and extensions to these engines.

Each course builds on what the students have learned in the previous course allowing for us, in a three course sequence, to bring the students from interactive fiction to complex, graphical games. None of the games projects have unyielding specifications, thus students are open to explore and experiment with different ways to improve the gameplay of their projects.

Comp 441

The main objective for Comp 441 is to teach students the basics of game development. The course is organized around three projects that unroll over the term. The first project is a simple arcade-style 2D game, like Pong. With XNA, simple 2D graphics are easy to program, so we do not need to spend a lot of time on technical issues and can cover game design. Thus, they get immediate experience with game play.

The second project is a more advanced arcade-style game (e.g., see Figure 2) that has added sound effects, background music, and other graphical elements, like blending. The final project requires students to develop complete game, with multiple levels, splash screens, cut scenes, high scores, and well balanced game play . For all three projects, students choose the game to implement and are required to write specifications.

In Comp 441, we also set the stage for later classes by having students develop a graphics engine and a game engine. They also develop physics and artificial intelligence (AI) subsystems. Some students take our AI course while taking games; these students then get to use sophisticated AI methods in their games.

In Comp 441, the final project is the culmination of the term, and takes the place of a final exam. In fact, during the final exam period, the students play and evaluate each other's games. The projects are graded according to the following criteria[6] (note: these criteria are applied to the final projects in later classes):
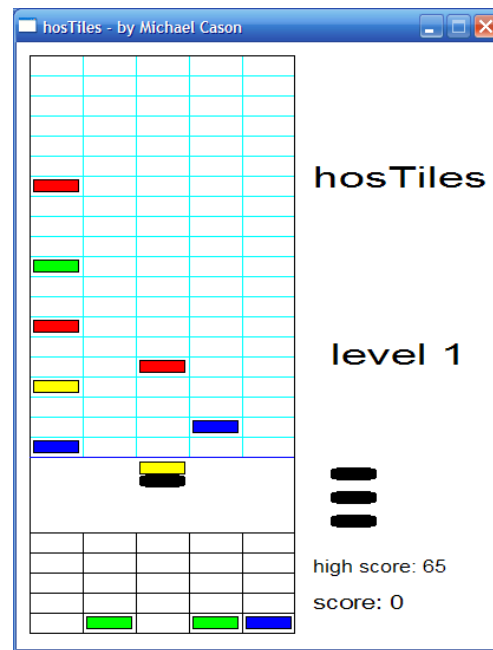

Figure 2: Example arcade-style game (written by Michael Cason).

- Completeness: Does the game feel finished? Does it have a title screen, credits, and instructions? Is there support for keyboard, mouse, or joystick input? Does the game have sound support? Does the game run without crashing?

- Game Play: How polished is the game play? Is the game fun? If there is a story, is the story immersive and connected?

- Creativity and Design: Did the student explore beyond what was explicitly covered in class lectures to include additional graphical effects like particles (for later classes, we will evaluate things like shadows, directional light or environmental effects like water distortion, heat waves, and lense flare)? Is the design of the game novel in some way to include elements that are unique to their design or clever combinations of multiple game design techniques?

These criteria are then weighted heavily toward game play and completeness, while the creativity and exploratory design aspects differentiate the impressive games from the average and good

COMPUTERS IN EDUCATION JOURNAL

games. While this may seem vague (e.g., average versus good), it is very apparent to both the students and faculty how the games should be categorized.

Comp 446

In Comp 446, the students will apply their game development knowledge to creating 3D games.. The course plan is to create a specification at the start of the class describing a simple 3D-shooter type game that will be built from the ground up. The 3D-shooter game is a perfect example to begin 3D game development as it involves cameras, multiple styles of player movement, physics simulations, and more advanced collision detection. The course lectures build on example projects that demonstrate concepts in code that can be adopted and transformed by the students in their own projects.

Some of the topics covered include extending the game engine developed in Comp 441 to three dimensions, as well as extending the graphics engine. This is a significant technological leap, as the simple mathematics used in 2D games must be significantly extended. For example, detecting collisions between objects is much more complex in 3D than it is in 2D. In addition, we need to cover more sophisticated AI methods than we used in 2D games.

Student grades in the class are based on their final project, but regular project iterations are also used. These demonstrations allows the class to see the progress of their peers and learn from the experiences other students are encountering on their own projects. Using this iterative methodology for game development, the students end up building three games over the term.

Comp 447

Writing console games is demanding, as consoles do not have the same types of runtime environments enjoyed by a PC programmer and more of the hardware interfaces are visible to the programmer. However, with XNA, Microsoft has provided a good transition to console programming, as many of the things that work on the PC will work on the console. Students enjoy

programming consoles, as they have a certain *panache* that makes them very attractive.

The console games course is organized in a similar way to Comp 441, in that there are a series of projects of increasing difficulty. The graphics and game engine knowledge developed in the first two classes is applied to this course, with changes made appropriate to new hardware.

One of the important additions to this class is the use of the Xbox's networking capabilities. The main focus of this course is developing multiplayer, networked games.

Since students have experience with 2D and 3D game development, we do not need to cover the technical aspects of these things. Rather, we concentrate on things important to consoles. A summary list of topics we cover is as follows:

- Since consoles use gamepads, input is different. This has significant effect on game design. For example, first-person shooters play differently with a gamepad as opposed to a keyboard and mouse.

- Cameras for multiplayer games are different than single player ones. We discuss how to use screen real estate effectively, and what types of camera work well for different games. For example, if two avatars are controlled by two different players and viewed with a single camera, how is avatar movement handled so neither can leave the camera's view?

- Networking provides a variety of opportunities and problems. Students need to learn the technical details of network programming using the XNA model. While not as complex as some 3D topics, networking is a challenging topic. Part of this challenge comes from dealing with latency and packet loss. In a high-performance game, such as a first-person shooter, these things can make a game unplayable. Thus, students need to learn how to adjust quality of service guarantees to balance reliable packet delivery with bandwidth and latency concerns.

Furthermore, they learn to use simulation in their physics engines to account for latency in packet delivery. For example, using latency estimates, they "speed up" their local physics engines to predict where a remote player is in a shared world.Networking also affects gameplay. Students must learn what opportunities there are with screens that are not shared among player, and with player coordination issues (e.g., starting a game in a fair way so that no player has an advantage.)

The projects are as follows:

- The first game is a 2D game to give the students experience with networking and console input devices.

- The second game is a 3D game, which can be a reimplementation of a Comp 446 game. This game emphasizes multiplayer gameplay with extensions to the game engine and emphasis on good camera work.

- The third and final project is a full-fledged networked, multiplayer game. The students are expected to connect with Microsoft's LIVE server, which requires going over the Internet. This means that latency increases (compared with local networking) and packet loss increases. They need to address these issues to keep their games playing well. This is typically done by adjusting quality of service and using simulation to account for latency.

We note that network programming in Comp 447 is the most significant networking experience our students get. Since XNA provides an abstraction over the typical network layer (e.g., sockets), students can concentrate on issues important to their games, such as handling latency and packet loss while meeting real-time deadlines (i.e., frame rates).

Programming Environment

All students at GCC get the latest HP Tablet PCs when they are freshmen. For this year's freshmen, the CPU and GPU are fairly powerful and can easily handle anything for the courses; for Comp 447, the Tablet PC is used to compile and then deploy a solution to the Xbox. Thus, computer hardware is not a major concern,

Developing games requires XNA, and that requires Visual Studio (VS). Microsoft makes VS and the XNA SDK (software development kit) freely available, as well as a re-distributable runtime environment.

Creating textures and other artwork is done with various programs. In class, we use the Autodesk 3D Studio Max, and Microsoft Paint, among other programs. For 3D work with articulated models or complex meshes, a full-feature system, like 3D Studio Max, is needed.

Students use a variety of systems for audio. The recorder that comes in Windows is surprisingly good for creating sound effects, particularly when processed with Audacity. Students create MIDI with a variety of sequencers, such as Cakewalk. Of course, as with graphics, there are a variety of good quality shareware and freeware systems available.

The software and hardware needed for PC gaming is surprisingly easy to get, and most campuses can supply their students with the necessary software for no cost if they have Microsoft site licenses. Thus, it is straightforward to offer gaming courses.

Experience

We have offered the games classes many times over the past several years The courses have been very well accepted by the students. Our survey data supports this with overall high ratings for the course (in fact, these classes are among the most highly rated of our senior elective classes), including students' satisfaction with their projects.

In total, over 100 students have taken all the games classes. As a point of comparison, our senior class size is about twenty students. Game programming is very popular, and, interestingly, we have enrolled students from related disciplines, such as EE, in the course sequence.

There are a number of interesting things reported by the students, such as the following:

- The amount of effort required to develop a game is much greater than the students expected. What is interesting to note is that they are not necessarily commenting on the effort for coding, rather it includes the design, coding, creating graphics and music, and balancing (e.g., developing a good scoring system to make the game challenging, but still playable, adjusting the strength of characters, writing the backstory, and so forth). We try to manage their expectations at the start of class, but to no avail.

- The students spent a lot of time on their games, but enjoyed the process. We have received relatively few complaints about the amount of effort and time spent. While hard to measure exactly, we estimate that the students spend about ten to fifteen hours per week, on average, on their projects; this is for the successful projects.

- The students gained an appreciation for the non-programming aspects of game development, particularly developing bitmaps, sounds, and the importance of user testing.

The first time we offered Comp 441 the student performance was mixed. Some students produced excellent final games; but about half developed games that were short of their specifications and were hardly games at all (the user moved a figure around the screen without any gameplay). Exit interviews indicated this was because they waited too long to start their games and could not finish before the due date. These students did not fully appreciate the complexity of the games they were building.

The second time we offered Comp 441 and in all classes subsequently, the final projects were uniformly excellent. All the students produced fully functional games. One change we instituted was to have periodic demos in class through the assignment period. Thus, the students were motivated to keep up with development. Note, interestingly enough, that we did not assign a grade for these demos. Rather, it seems that the students did not want to be shown up in class.

Our experience with Comp 446 mirror that of Comp 441. Students are generally pleased with the class, particularly as they are able to produce excellent, albeit short, games. The mathematical rigors and some of the programming issues, such as shaders, does give students pause. Further, the amount of time needed to create models and other meshes (e.g., terrain) is significant.

Since the 3D are more complex than 2D games, students must work in groups and learn how to divide up work. Students will tend to specialize in areas that interest them, such as physics engines or model creation. This poses a problem to the instructor, as we want them to understand all aspects of game development. Without having a good solution to this problem, we try to ameliorate it with comprehensive exams and periodic code reviews.

By the time students get to Comp 447, they are experienced game developers and programmers. While we do need to teach specific technical elements for the console, namely networking, the number of technical lectures is significantly less than for the previous game classes. Thus, the students are able to spend more time on game design and balancing. This leads to some very sophisticated networked, multiplayer games.

One of the most significant issues facing faculty members wishing to build a gaming curriculum is the dearth of textbooks. While there are many books available on gaming, and some of them are well written and very informative, they are not textbooks. Much of the underlying theory on algorithms and programming methods is missing or treated in a cursory way; or, one must use several texts to get the right combination of topics and depth.

Summary

Our intention in creating the three-course sequence in gaming is to provide our students extensive experience in game development. We want to educate game developers who, first and foremost, create games with excellent play, and

who have the skills and expertise to manipulate the underlying technology as they need for creative purposes. In addition, the games courses utilize much of what students have learned in their first three years of undergraduate schooling and provide excellent ways of building on and incorporating material from courses students take concurrently with the games courses (e.g., AI).

Since GCC is a small comprehensive college, we hope to demonstrate that a gaming curriculum is accessible to a broad range of colleges and universities, both big and small. PC-based gaming classes require very little beyond what is needed to teach standard programming classes. Console gaming requires more equipment and licensing, but is a very popular and interesting topic.

References

1. Maxim, B. "Game development is more than programming," In Proceedings of the 2006 American Society for Engineering Education Annual Conference and Exposition.

2. Jones, R. "Design and implementation of computer games: a capstone course for undergraduate computer science education," In Proceedings of the 31st SIGCSE Technical Symposium, ACM Press, New York, NY, 2000.

3. I. Parberry, M.B. Kazemzadeh, and T. Roden 2006. The Art and Science of Game Programming. In *Proceedings of the 2006 ACM Technical Symposium on Computer Science Education* (Houston, TX, Mar. 1-5, 2005). pp. 510-514.

4. Wolz, U., Barnes, T., Parberry, I., and Wick, M. 2006. Digital gaming as a vehicle for learning. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, Texas, USA, March 03 - 05, 2006). SIGCSE '06. ACM Press, New York, NY, 394-395.

5. LaMothe, A., Tricks of the windows game programming gurus 2nd Edition. Sams, 2002.

6. I. Parberry, T. Roden, and M.B. Kazemzadeh 2005. Experience with an Industry-Driven Capstone Course on Game Programming. In *Proceedings of the 2005 ACM Technical Symposium on Computer Science Education* (St. Louis, MO, Feb. 23-27, 2005). pp. 91-95.

7. LaMothe, A., Tricks of the windows game programming gurus-Advanced 3D graphics and Rasterization. Sams, 2002.

Biographical Information

Dr. Birmingham is the chair of the Computer Science Department at Grove City College. Before coming to Grove City College, he was a tenured Associate Professor in the EECS Department at the University of Michigan, Ann Arbor. His research interests are in AI, computer gaming, mobile computing and communications, and computer-science pedagogy. He received his Ph.D, M.S., and B.S. all from Carnegie Mellon University.

David B. Adams received a B.S. in Computer Information Systems and a B.S. in Mathematics at The University of Virginia's College at Wise in December of 1997. He moved on to graduate work in Computer Science at Virginia Tech obtaining a Masters (2003) and Ph.D. in Computer Science in August, 2005. He is Associate Professor of Computer Science at Grove City College in Grove City, PA. His research interests include parallel and scientific computing, multiplayer client-server models for games, programming languages and computer science education.