AC 2007-1786: FROM 2D TO CONSOLES: A THREE-SEMESTER COMPUTER GAME DEVELOPMENT CURRICULUM

William Birmingham, Grove City College

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. Birmingham's research interests are in AI, computer gaming, mobile computing and communications, and computer-science pedagogy. He received is Ph.D., M.S., and B.S. all from Carnegie Mellon University.

David Adams, Grove City College

David B. Adams received a BS in Computer Information Systems and a BS 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 PhD in Computer Science in August, 2005. Adams is Assistant 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.

From 2D to Consoles: A Three-Semester Computer Game Development Curriculum

1. 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 OpenGL and DirectX) and techniques for good game play. We have offered the first class twice and we will offer the second course during the coming academic year.

In the third term, our students gain experience with console gaming. Through a partnership with Nintendo and Freescale, our students develop games for the Nintendo Gamecube. 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 computer hardware. We are currently running a pilot of the console course with five students.

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.

2. 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 Charlotte[4] 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.

3. 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, we are expecting our students to 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).

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

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.

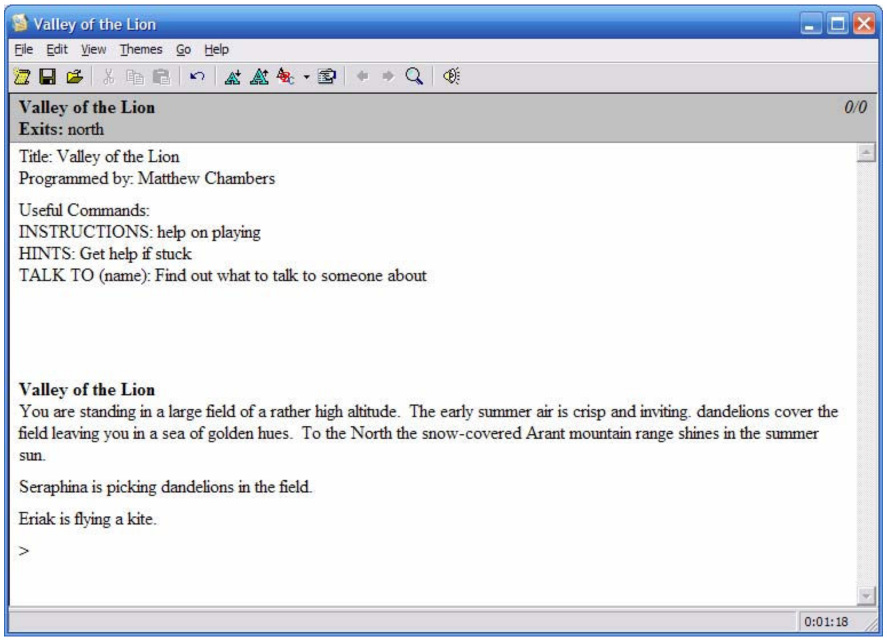
Finally, by their junior year, all the students have taken physics (statics, dynamics and kinematics), three courses in calculus, and linear algebra. Since students need to create all the physics in their games (e.g., how a ball will bounce off the end of the screen), they need to be familiar with physics. 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 DirectX and code from the books we use in 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.

3.1.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 an interactive fiction (IF) game. We start with IF, because it allows students to quickly develop games that rely on great stories and characters, and does not require complex graphics. Thus, they get immediate experience with game play. See Figure 1 for a screenshot from a student IF game.



Valley of the Lion 0/0
Exits: north

Title: Valley of the Lion
Programmed by: Matthew Chambers

Useful Commands:
INSTRUCTIONS: help on playing
HINTS: Get help if stuck
TALK TO (name): Find out what to talk to someone about

Valley of the Lion
You are standing in a large field of a rather high altitude. The early summer air is crisp and inviting. dandelions cover the field leaving you in a sea of golden hues. To the North the snow-covered Arant mountain range shines in the summer sun.

Seraphina is picking dandelions in the field.

Eriak is flying a kite.

>

0:01:18

Figure 2: Student IF game (written by Matthew Chambers).

The second project is an arcade-style game (e.g., see Figure 3) that has simple rules and scoring with minimal sound effects. This project is programmed using the Windows API. The final project requires students to develop an arcade game, complete with bitmapped graphics (optional for the second project), music, sound effects, scoring and a splash screen. For all three projects, students choose the game to implement and are required to write specifications.

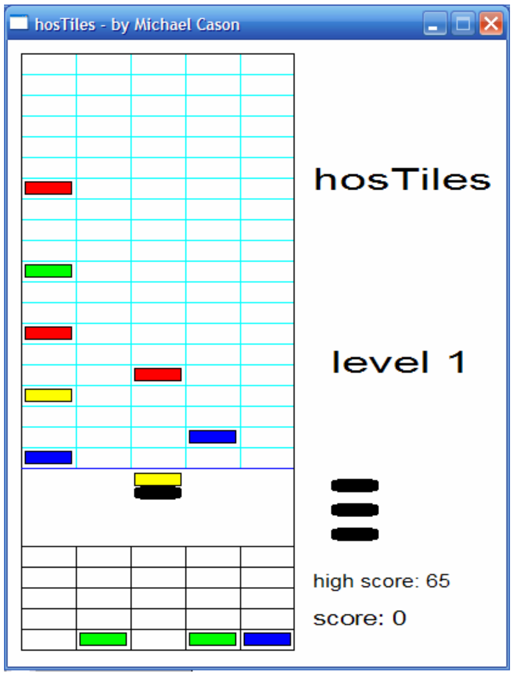


Figure 3: Example arcade-style game (written by Michael Cason).

We move from Windows to DirectX to ease the move into DirectX technology. Our students get some experience with the Windows's API in various classes, but they need to have a refresher. In addition, Windows has a graphics engine, which is lacking in DirectX. So, it is much easier to program a graphics game in Windows; this helps them to concentrate on game play while gradually introducing more complex programming.

In Comp 441, we also set the stage for later classes by having students develop a graphics engine and a game engine, based on the La Mothe[5] book we use. The game engine introduces basic 2D graphics as they build an engine for DirectX. 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):

- 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 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.

3.2.Comp 446

In Comp 446, the students will apply their game development knowledge to creating 3D games, using another LaMothe book[7]. 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 item collection, multiple styles of player movement, physics simulations, more advanced collision detection, and the possibilities for multiplayer network extensions. The course lectures will build the example project demonstrating concepts with code that can be adopted and transformed by the students in their own projects adding pieces and highlighting the changes that occur throughout the progression.

Some of the topics covered will 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 sprites 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 will be based on their final project, but regular project demonstrations will be used.. The demonstration allows the class to see the progress of their peers and learn from the experiences other students are encountering on their own projects. Since we can not sacrifice the lecture time to have all the students demonstrate their projects every week, we instead will use a rotating schedule that allows us to see the progress of each student at least once every three weeks, a method we have used successfully in other classes. The final project will be graded using the same criteria as that used for Comp 441.

This class will differ a little from the initial game course, because there is a much greater need for the hardware that the students are working on to be as current and up to date as possible for some elements of their projects. In these cases, we have lab machines with high-end graphics capabilities and a few high-end laptop computers for demonstration days.

3.3.Comp 447

Writing console games is demanding, as consoles do not have the same types of runtime environments enjoyed by a PCs programmer and more of the hardware interfaces are visible to the programmer. However, the consoles provide significant power, with hardware and software support specifically for gaming. In addition, consoles have a certain *panache* that makes them very attractive to program.

The console games course will be organized in a similar way to Comp 441, in that there will be a series of projects of increasing difficulty. The graphics and game engine knowledge developed in the first two classes will be applied to this course, with changes made appropriate to new hardware (Nintendo GameCube) and a new IDE, Freescale CodeWarrior. The SDK for the GameCube is different from the DirectX SDK and console games are developed on special hardware called TDEVS. These are GameCubes with special features for loading and debugging games.

Since the programming environment is different from the PC-based systems, we need to spend a good part of the term teaching students about the systems. Of course, the principles of good game play still apply, so we do not need to spend as much time on those topics. By the third course, the students are familiar with game design and development. Our plan for the projects is as follows:

- The first game is an arcade game, essentially re-implementing the DirectX-based game from Comp 441.
- Similarly, the second game is a re-implementation of the Comp 446 game, with some reduction in scope (e.g., fewer levels and fewer bosses).
- Finally, we expect the students to develop a game that makes good use of the console features, such as superior controller features, particle systems, and high-quality sound.

By re-implementing games from previous classes, the students do not have to be concerned with design issues and can concentrate more on programming and exploring what the console can offer. This sets them up well to write their final game.

4. 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 Comp 441 and portions of Comp 446. Thus, computer hardware is not a major concern, except as we noted earlier (and even in those cases, the student machines are fine with reduced frame rate).

Developing games for PC requires DirectX, and that requires Visual Studio (VS). We have used VS 6 and VS 2005, and now use only VS 2005 development. In general, VS 2005 is a very good environment for game development. Microsoft makes the DirectX SDK (software development kit) freely available, as well as a re-distributable runtime environment.

Creating bitmaps and other artwork is done with various programs. In class, we use the VS 2005 bitmap editor, Autodesk Maya, and Microsoft Paint. For 3D work, a full-feature system, like Maya, is needed. Our students use Maya and a variety of other programs for artwork creation.

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.

The same is not true for console games, however. Consoles have traditionally been closed systems with only licensed developers able to get the SDKs and hardware needed for development. For our classes, we purchased TDEVs from Freescale after getting licenses from Nintendo. Development is done entirely with CodeWarrior, which is also an excellent development environment. The version of CodeWarrior we use is targeted to the TDEVS and requires a Nintendo-supplied SDK.

5. Experience

We have offered the first games course, Comp 441, twice as a formal class and twice as a special topics course (which is one way we prototype courses). Both offerings 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, about thirty students have taken all four offerings of the class. As a point of comparison, our senior class size is about twenty students. Game programming is very popular, and, interestingly, we are now enrolling 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 parameter tuning (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 in normal format, 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, 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.

Examples of the students games can be found at http://www2.gcc.edu/dept/comp/index.shtml, and can be downloaded.

We are offering a combination of Comp 446 and 447 this term (Spring, 2007) as a special topics course. Since the equipment necessary for Comp 446 arrived on campus recently, we were unable to offer the course until now. The current offering has five students. The materials the student use are manuals and tutorials produced by Freescale and Nintendo. The students are getting first-hand experience in working with professional documentation and are learning how to setup equipment from the ground up (rather than simply use equipment we setup for them).

Working with consoles has some interesting legal issues. As we mentioned, consoles are not open systems, and working on them requires executing a non-disclosure agreement.(NDA) requiring the signer to keep all hardware and software information confidential. All students, faculty, and staff working with the consoles must execute the NDA.

Furthermore, we must keep all the software, hardware and manuals associated with the TDEVs private. This means that the students are not allowed to install the SDKs on their computers, and they can work on the TDEVs only with faculty supervision. We also work on the equipment using rooms where other students and faculty cannot see any proprietary items.

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 texts. 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.

6. 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 (an engineering teaching institution), 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.