EXPERIENCES BUILDING A COLLEGE VIDEO GAME DESIGN COURSE

Paul E. Dickson
Hampshire College
School of Cognitive Science
893 West St.
Amherst, MA 01002, USA
(413) 559-5861
pdickson@hampshire.edu

ABSTRACT

This paper details some trials and tribulations of the design and implementation of a video game design course that not only enables students to create video games but also to learn standard computer science skills. The decisions that led to the platform decision of the iPhone/iTouch are discussed along with the consequences of that choice. The project-based and dynamic curriculum that encompasses real-world problem solving is described. Initial successes and failures of this course and the lessons learned are addressed as well as the level of student excitement surrounding it.

INTRODUCTION

In Spring 2009 we decided that a video game design course would be a good elective within our computer science program. As our program does not have a set major with a required number of courses and electives, our electives must have a broad appeal in order to attract enough students to make them feasible. Video game design was decided upon because it has a definite "cool factor" among students and because it would give our students the ability to show their peers what they have learned in a way not often possible in computer science.

It was decided that computer game design could be a valid addition to our curriculum if the course focused on design from the ground up and did not just use prewritten game development software. We envisioned that video game design could be used to teach students about real-time processing and development, resource management, real-world environments, and device limitations as well as give them good practice programming large chunks of code and working in teams. Huang [5] describes some of the advantages of game-related assignments in relation to student engagement, teamwork, development of algorithms, etc. All of our choices about the design of this course appear to fall within the framework of good use of computer games creation within a computer science curriculum as described by Sung [12]. Sung's review was published near the end of the semester this course was taught.

RELATED WORK

Courses in computer game design generally are taught in schools that offer degrees in game design or as single elective courses offered within an established computer science or engineering major. While some traditional universities like DePaul University [3] offer a complete major in video game design, most schools

that offer these courses tend toward career education, like ITT Technical Institute [6]. In either case the courses taught do not relate to the one we chose to offer because no single course covered the breadth of material we hoped to include.

Single courses in game design such as those taught Bruce Maxwell [10] and Kevin G. Stanley [11] were also discovered. These courses focused on game design from start to finish and helped lead to our course.

A new area where games are appearing in computer science curricula is in CS1/2 courses, where games are used to bring students into a major [12]. This type of integration is not applicable to our course, which is intended for more advanced students.

PLATFORM DECISIONS

The first decision to be made about his course was what platform to use for development and we considered Microsoft's Xbox, a toy game development platform, Flash, OpenGL, and the iPhone/iTouch. Our decision was based on the best combination of feasibility, ease of entry, and students being able to show off their work.

We ruled out using a toy game development platform from the start since such platforms would likely never be used again by students after the end of the semester. We decided against Flash because while it is ubiquitous and would have made it easy for students to show off their final games, it does not have the cool factor that would really motivate students. In addition, every person who has played a Flash-based game has played at least one really bad Flash-based game, and therefore it would be less exciting because students would not have sufficient time to create a really good game. Creating games that relied on OpenGL on the computer was eliminated because students would have had to make sure that OpenGL was able to run on a given machine before they could show off a game. The largest factor in disregarding the computer-based games was that the platform did not lend itself to enabling students to show what they had created.

Our initial preference was to develop on the Xbox since Microsoft had previously given out Xboxes to universities that wanted to build courses around them. Also, Microsoft made the development software available without charge and students would be able to show off their final games. Discussions with Stanley [11] and a look at the fine print disabused us of these ideas. First, Stanley described how only a small portion of the student projects in his game development class had actually been ported to the Xbox because of the large hassle involved with the device compilation, defeating the purpose of developing on the Xbox. Second, Microsoft does not allow you do burn discs of games, which means that downloading a game to a console requires a yearly paid subscription. These barriers to students showing their work, in addition to the fact that we are primarily an Apple campus and Microsoft's game development suite only runs on Windows, dissuaded us from using the Xbox.

The platform we decided on was the iPhone/iTouch. Grissom [4] does a good job of describing the excitement and possibilities for using iPhones in courses. Apple has a university developer program [1] that makes it free for schools to enable students to download their created software to devices. In addition, many students

own iPhones/iTouches, making it easy from them to show off what they have created for at least the time they are in school and under the school's development license. Developing for a hand-held device also has the cool factor as students can show off what they have created anywhere they go and to anyone they meet. A final advantage is that it is a platform with limited input capabilities, which limits what students may attempt to create, a good thing in a course where students will only have limited time to create games.

COURSE FORMAT

The goal of this course is to teach computer science through the creation of video games. The semester is divided roughly into 3 parts. The first third of the course is devoted to students learning specifics of the platform. Students learn how to use Apple's Xcode development environment [2] to build interfaces that include buttons, text fields, labels, multi-screen applications, and other aspects of the iPhone SDK. They are also introduced to Objective-C (the development language used by Apple) and the steps involved with downloading applications to devices. Each lecture takes place in a computer lab, and students work through examples as they are presented and then are given time to modify them in class before new material is introduced.

The second third of the course is devoted to an introduction to OpenGL ES, the graphics library that runs on the iPhone. The goal of this section is not to comprehensively teach students OpenGL ES but instead to teach them just enough to get started building games with the reasoning that anything specific that they need and are not been taught they can pick up on their own as needed. Therefore, basic point, line, triangle, coloring, translation, rotation, touch recognition, and collision detection are covered. Again, students follow along with material, typing it in while it is presented and playing with it during class. In many cases, work created during class is presented by students to the entire class to show off how the material can by used in ways well beyond that fathomed by the lecturer.

The final third of the course relates to their final project games. To begin this section each student presents an idea for a final project game to the class. Students then decide which game they want to work on and form teams (this semester, 5 teams for the 17 students in the class). The rest of the semester is spent working on these final projects. Each week one lecture is devoted to going over some minutia of game design or drawing (e.g., texture integration into Open GL) that may be of use to the design teams in general. The other lecture of the week is devoted to in-class work on the games. This lecture often begins with mini-lectures given by students about interesting points and solutions that they have discovered over the course of their work on the final project games.

Course References

Finding an appropriate set of reference materials for the course turned out to be one of the greatest difficulties associated with it. Few books exist about iPhone development, and the only iPhone game development books with code were published during the semester the class was given. The textbook settled on was Beginning iPhone 3 Development [8] because it gave good worked-through

examples that showed off both how to use Xcode and how to write programs that utilize the iPhone SDK. Even the choice of this book illustrates one of the issues associated with iPhone and other cutting edge classroom development. It was discovered 3 weeks into the semester that students had a newer version of the text than did the professor as the professor's copy was bought 3 months before and was hence out of date and had been updated. Quickly outdated source material proved a recurring theme throughout the course.

The best references found for OpenGL use on the iPhone were online tutorials [7, 9]. These tutorials both give good concrete examples of how to use OpenGL ES 1.0 on the iPhone and proved invaluable. The downside was that they were 4 months old when the class began and were out of date since Apple upgraded the iPhone to OpenGL ES 2.0. This would not have been a problem except that all tutorials were written using a template in Xcode that no longer exists. No written texts that have concrete examples of OpenGL on the iPhone have yet been found by the authors of this paper.

The iPhone is such a new device that its standards and code keep getting massive changes and upgrades and there is limited platform stability. Over the first 5 weeks of the semester, we had to install 3 different versions of Xcode because Apple kept updating the iPhone software and Xcode. The upgrades were required since any device that had been updated (done without thinking by many students to their own devices and by the instructor as well) could not have software installed on them without a software upgrade (free but over 5GB to download each time). These updates fortunately did not appear to include any updates to the templates used in Xcode but often changed the look of menus and made small changes to software functionality.

Assignments

The course includes 4 substantive assignments, one of which is the final project. The first assignment is intended as an introduction and to let students have some fun. They have just learned how to build a basic interface and are asked to use buttons, labels, and text fields to build an application that specifically tailors insults to the name that was typed into the application. Students are given the opportunity to use what they have learned but also to express themselves and have fun with the assignment.

For the second assignment students are asked to build an application that has the user enter a name and date of birth and then click a button to enable the application to supposedly search the web for images of that person. When the button is pressed the application returns an image of a monkey or something similar. This assignment familiarizes students with how to build multi-screen applications and flip between the screens, something important if they wish to have a settings page for their eventual games. Again, creativity was encouraged and many students made their applications with special provisions to show images of themselves when their own information was entered.

The third assignment is to build Tic-Tac-Toe or a similar game of their choosing. This assignment is intended to give them both a first opportunity to build an application that uses OpenGL to draw and to create basic opponent logic into an

assignment. This was the first assignment where pair programming was allowed and encouraged. As with all previous assignments, students were given a week to complete it but unlike previous assignments students ran into a lot of trouble. The task of building the interface, graphics, and handling user touches proved the limit of what any students were able to accomplish. The hurdle of getting graphics up on their own for the first time when none had experience prior to the class with OpenGL or with handing user touches to a device proved so high that none had enough time to begin game logic. In future offerings of this course, this assignment will probably be broken up into two parts, the first related to interface design and the second to program logic.

The final assignment is the final project. For this the students work in groups on games the students have envisioned and chosen to work on. Students are given class time to work as described above and a lot of leeway to design and work on the games as they choose. Students have 5 weeks to take a game from an idea to a fully functional, if simple, game. This first semester the 17 students in the course broke up into 5 groups of 3 or 4 members each. The groups were formed based on interest and therefore some were stronger than others. By the end of the semester, 2 groups had succeeded in creating games, 2 groups had made significant progress, and 1 group barely made any progress. This latter group had only 3 students, 1 of whom stopped participating in the course and failed.

CONCLUSIONS

At the conclusion of the course, a class meeting was held to discuss the course in general and improvements that could be made. The instructor raised the issue of chaos caused by software updates, out-of-date references, and hardware problems. Student reactions were extremely positive: being able to show their work on an iTouch far outweighed these concerns. The disorganization of the course did not bother them primarily because they knew this was its first offering.

Working with the iPhone gave students a better look at more real-world programming problems. Software development in a classroom does not translate well to dealing with poorly documented technology. The students learned a lot from dealing with hardware and under-development documentation. They suggested decreasing the time spent on the introduction to the iPhone and increasing the time spent on the final project , which was the part of the course that taught them the most. Working in groups on their final project pushed them to learn more than did the individual smaller assignments earlier in the semester. They commented that a larger final project with more evaluation during the process would be an improvement. They also commented that because there was no available expert on iPhone development, they had to learn more on their own and thus now better understand the device and their code.

Though only 2 of the 5 final project games were completed, the project itself was a success. All 4 of the groups (14 of 17 students) that made real progress towards final games learned a lot from the experience. They were able to apply object-oriented design principles, data structures, archiving schemes, etc. from other classes to this project and learned a significant amount from doing it. All students had more confidence in their ability to program and handle large projects

at the end of the semester. The only downside of the final project, from the student perspective, is that they did not have enough time. At least 2 of the groups have continued to work on their game after the semester ended.

Overall, this course was a success. The work students do in computer science often lacks a "wow" factor and therefore we sometimes have trouble motivating them to invest in their work. This course reverses that trend and as one student was overheard to say, "All my friends want a copy of this game once we get it finished". If friends are this interested, students will go above and beyond to create a great game and will learn a lot more from the process.

REFERENCES

[1] Apple Inc., iPhone Developer University Program, http://developer.apple.com/iphone/program/university.html, retrieved November 16, 2009.

[2] Apple Inc., Tools, Xcode, http://developer.apple.com/TOOLS/Xcode/, retrieved November 16, 2009.

[3] DePaul University's College of Computing and Digital Media, DePaul Game Dev: Game Design, Programming, Production & Animation Education, http://gamedev.depaul.edu/, retrieved November 16, 2009.

[4] Grissom, S., iPhone application development across the curriculum, *Journal of Computing in Small Colleges,* 24 (1), 40-46, 2008.

[5] Huang, T., Strategy game programming projects, *CCSC '01: Proceedings of the sixth annual CCSC northeastern conference, Journal of Computing in Small Colleges*, 16 (4), 205-213, 2001.

[6] ITT Technical Institute, ITT Tech Offers an Education For The Future, http://itt-tech.edu/teach/list/degd.cfm, retrieved November 18, 2009.

[7] LaMarche, J., iPhone Development: OpenGL ES from the Ground Up: Table of Contents, http://iphonedevelopment.blogspot.com/2009/05/opengl-es-from-ground-up-table-of.html, retrieved November 16, 2009.

[8] Mark, D., LaMarche, J., *Beginning iPhone 3 Development: Exploring the iPhone SDK*, New York, NY: Apress, 2009.

[9] Maurice, S., iPhone OpenGL ES Tutorial Series, http://web.me.com/smaurice/AppleCoder/iPhone_OpenGL/Archive.html, retrieved November 16, 2009.

[10] Maxwell, B. A., CS 269/369: Computer Game Design, http://www.cs.colby.edu/maxwell/courses/cs369/J09/, retrieved November 16, 2009.

[11] Stanley, K., College of Arts and Science, University of Saskatchewan, http://artsandscience.usask.ca/college/directory/display.php?bioid=1184, retrieved November 16, 2009.

[12] Sung, K., Computer games and traditional CS courses, *Communications of the ACM*, 52 (12), 74-78, 2009.