Teaching Game Programming Using XNA

Joe Linhoff
DePaul University
243 S. Wabash Avenue
Chicago, IL 60604
(312) 362-5861
jlinhoff@cti.depaul.edu

Amber Settle
DePaul University
243 S. Wabash Avenue
Chicago, IL 60604
(312) 362-5324
asettle@cti.depaul.edu

ABSTRACT
As educators work to expand the audience interested in computer science, computer gaming programs have blossomed at a variety of educational institutions. Educators are coming to recognize that gaming is a compelling way to motivate students to learn challenging technical concepts such as programming, software engineering, algorithms, and project management. At the core of many gaming programs are game development courses, which teach technical aspects about software development in a motivating environment. While many game development courses share a common goal, the structure and goals of game development courses can be quite diverse. We describe a game development course that uses the XNA platform to allow a heterogeneous group of students to gain experience in all aspects of console game creation, an approach we believe has some interesting pedagogical benefits.

Categories and Subject Descriptors
K.3.2 [**Computers and Information Science Education**]

General Terms
Design

Keywords
Supporting courses: game development, using emerging instructional technologies: XNA.

1. INTRODUCTION
Game development programs and courses have become increasingly common at universities internationally. Educators are coming to recognize that game development is both a serious occupation and a compelling way to motivate students to learn challenging technical concepts such as programming [1, 4, 7], software engineering [3, 12], algorithms [5], theoretical computer science [6] and project management [14]. The resulting courses have a variety of audiences and goals. Some courses focus on games as a motivator for introductory students [2, 6, 7], while others focus on teaching more complex topics to intermediate or advanced students [3, 8, 12].

ITiCSE'08, June 30–July 2, 2008, Madrid, Spain.

As game development courses and programs have matured, educators have recognized that bringing together diverse student populations in a gaming course is challenging. Some institutions design courses specifically for one type of audience, either on the art and design side [13] or the technical side [3, 12], with the goal of pushing students to develop a set of focused skills. Others purposefully integrate art students and technical students together in the classroom, with the goal of improving the quality of the final product [8]. The two approaches are not mutually exclusive, since art or design and technical students can strengthen their skills while working together on a project. In those situations, however, it is common to grade the two populations using metrics relevant to their areas [8].

More unusual, however, is the approach taken in a course taught at the School of Computer Science, Telecommunications, and Information Systems at DePaul University (DePaul CTI). GAM 380: Console Game Development Environments is a course that can be taken by any student in the Game Development Program, regardless of concentration, as well as other technically-oriented fields such as Computer Science. Students work on individual projects for the first part of the course, gaining skills in all areas of content development, including modeling, animation, sound, and programming. Only at the end of the course are students allowed to form teams to work on creating a larger, comprehensive project. The course uses the XNA platform to allow a heterogeneous group of students to gain experience in all aspects of console game creation, an approach we believe has some interesting pedagogical benefits.

2. CONSOLE GAME DEVELOPMENT
In order to understand the approach taken in GAM 380, it is important to understand the Game Development Program at DePaul CTI, including target audience of the degree and the required development courses.

2.1 Game Development Program
DePaul CTI is one of the largest and most innovative information technology institutions in the United States. Over 2,000 graduate students and nearly 1,000 undergraduates are enrolled in 14 Bachelors and 15 Masters degree programs, ranging from traditional programs like Computer Science to degrees focused on the digital arts such as Digital Cinema and Animation. Such a broad range of degree programs is highly unusual and has attracted large numbers of students during a period of overall decline in technology education enrollments [10]. Benefiting most from the influx of new students with an interest in the digital arts is the Computer Game Development program. DePaul was

one of the first liberal arts universities in the United States to offer a four-year degree in the area. In only 4 years the program has grown to be the second largest at DePaul CTI with 150 majors.

The Game Development Program offers two tracks: the standard concentration, designed to be flexible and to accommodate a variety of interests in game development, design, and production, and a game programming concentration. Both concentrations require students to take a number of development courses, including Game Development I & II that cover 2D and 3D games, Programming in C/C++ I && II, Action Games Programming where students create games using C++ and OpenGL, and the Game Modification Workshop where students modify existing games. Additionally, students in the game programming concentration are required to take more traditional computer science courses such as Data Structures, Computer Systems I & II, and several computer graphics development courses. GAM 380 is not required in either concentration, although it counts as a gaming elective in both. As such, it is designed to appeal to students in both concentrations of the Game Development program, as well as students in other programs such as Computer Science.

In support of the Game Development Program, DePaul CTI was awarded a Microsoft XNA lab grant in June 2007, one of 5 grants awarded that year. The grant included 20 Xbox 360 consoles, 20 Creator's Club licenses, and funds to purchase monitors and upgrade the existing game development labs. As a part of the grant, the faculty at DePaul CTI intends to develop and/or redesign 8 courses in the areas of Computer Game Development and Software Engineering to use the XNA platform. GAM 380 is the first XNA course and was offered for the first time during Fall 2007.

2.2Course Overview

The goal of GAM 380 is to have every student create and use all of the major types of content that go into an Xbox 360 game. The class is designed to take advantage of the content pipeline in XNA, which is discussed in more detail later in this section. The only prerequisite for the course is Game Development I, which is a freshman-level course in game design and development that uses Game Maker and does not require any previous programming experience. Because Game Development I is the only prerequisite for GAM 380, students taking the class may have never taken a traditional programming course. As such, GAM 380 can be taken early in the Game Development Program, something that is unusual for game development courses [2, 7].

The course leads every student through the creation of a font, icons, 3D models, a camera and object animation paths, skeletal animations, sounds, scripts, and other supporting content. All students are also required to edit the program file to change variables and copy and paste code. Programming is intentionally deemphasized, and not directly taught. Instead, students are supplied sample programs, programming references, and also encouraged to cooperate and share code.

Teams are formed near the end of the course. The final project is the creation of at least 30 seconds of game play for a "rail shooter" game in which the camera is moved on path through an environment. More information about assignments and the course structure is found in the following sections.

The unusual approach of requiring every student to become minimally proficient in all areas of content development is a conscious decision. By doing this, we allow students more direct exposure to the content pipeline, which is a significant but little understood part of the development process. By content pipeline we mean the collection of tools, files, and processes required to utilize authored content. Advanced game design courses can be slowed down by the mechanics of creating and moving content into the game, resulting in student frustration and games that under represent their abilities. Creating content that reflects their abilities is crucial for student morale and for student success in employment after graduation [9].

In addition to requiring students to become familiar with many aspects of content development, the course is structured to encourage student sharing of content. After students have turned in individual work to be graded, they are allowed to share that content on subsequent assignments. Both artwork and code are allowed to be shared. Further, students whose content is used by other students in their assignments are awarded extra credit. It is our belief that allowing students to share, and implicitly evaluate, each other's work in this way provides them with valuable experience.

In the remainder of the section we discuss the XNA pipeline in more detail, specify which tools are used in the course and why, and outline the structure of the course including assignments and grading.

2.3 XNA Pipeline

Microsoft's XNA pipeline operates primarily through a visual interface consistent with the rest of the integrated development environment. The visual nature of the interface helps students understand the processes at work.

Content files, such as images or models, are included in the project and treated much like code files. A properties page controls how content is imported and processed. This visibility and control demystifies some of the previously abstruse content pipeline. Students are able to see all of their content files in one place and understand how the files are processed and utilized by their game code. For example, to include a model into a game, the model is exported from Maya into a standard FBX file. The FBX file is then included in the game project. XNA provides a content importer and processor for the FBX format as well as other standard formats. The model is then available for use in game code.

The XNA pipeline also provides the ability to copy files directly to the output directory. This is useful for student script files which, among other things, are used to position objects in the world. For non-standard data formats, there is ample documentation on how to write your own content importer and processor. For example, XNA does not currently provide standard support for skeleton animation data. The course takes advantage of this opportunity to show how to integrate an external pipeline to import, process, and run animations.

2.4 Course Tools

The major components of the development environment include: Microsoft Visual C# 2005 Express Edition with Game Studio

Express, TortoiseSVN, Paint.NET, Microsoft's XACT, Audacity, Maya, and a wiki. All the tools, except Maya, are free for the students to download and install on their own systems. The course also uses an instant messaging system to ease communications.

The Microsoft tools, collectively called the XNA Game Studio, provide the integrated development environment. Game code is written and debugged in C#.

TortoiseSVN is a front end to Subversion, the source control system we use. Each student has their own directory and also access to a number of folders for shared content, library files, and other distributions. The source control system is crucial for the course. It is used to setup the class directory structure, provide students with starter kits and updates, and define student sandbox areas. Students also use the system for obtaining help from the instructor, submitting assignments, backing up their work, and for team project coordination.

Paint.NET is a simple image editing program used for font and icon creation, as well as image map manipulation. XACT is used to prepare and package sound data. Audacity is a simple sound editing program which allows students to edit .wav files. Students use Audacity to prepare sounds and setup sound loops.

Autodesk's Maya is an immense 3D package that is the course's primary modeling, world building, and animation tool. Maya is a great tool with significant depth. Because of Maya's complexity, every assignment includes reference to a required Maya book, and also click by click directions, with screen shots, for creating the minimally acceptable level of content.

The use of a wiki was suggested by Alex Seropian of Wideload Games, one of the DePaul CTI Gaming Advisory Board members. The wiki is used for class-wide distribution of documentation, code, and assignments. A few assignments also require students to post to the wiki. For example, they must advertise and explain how to use content they have added to the shared folders in the source control repository. The wiki provides a means for project documentation, collaboration, distribution of code, and as a message board for frequently asked questions.

Instant messaging allows for quick answers to questions, particularly about problems with the environment, gives teams an interactive way to communicate, and provides a way to copy and paste code between classmates. The instant messaging tool also allows for voice communication and group conference calls when appropriate.

2.5 Course Structure

The course is divided into eight modules, which are: fonts and icons; models; design, budget, and 360 deployment; key-framed object and camera paths; skeleton-based animation meshes; projectiles, collision geometry, and "damaged" versions of models; digitized image maps and sounds, shaders; and game presentation. Each milestone focuses on the creation, pipeline, and programming for a specific type of content. The first seven milestones are self contained and do not depend on prior milestones.

The following table lists the topics covered by week and the due dates for each milestone. It should be noted that DePaul University uses the quarter system, with 10-week quarters followed by one week of final exams:

Week	Topics	Milestone due
1	Hello world; conceptual models; tools; content pipeline; creating a using a font; version control	None
2	3D models; Maya; 000ZY; naming; XNA pipeline; 3D painting	Milestone 1
3	Design and budgets; storyboards; list of assets; deploying to the Xbox 360; scripting worlds	Milestone 2
4	Cameras; key-framed paths; DC-pipeline; dispatch	Milestone 3
5	Skeleton-based animation meshes	Milestone 4
6	Projectiles; collision geometry and response; scoring	Milestone 5
7	Capturing sounds; .wav files; looping; streaming; image maps; uv mapping photos; shaders	Milestone 6
8	Integration; input; game loop; loading and unloading levels; starter kit for final game	Milestone 7
9	Open	None
10	Open	None
11	Final project 'pitch'	Final project

Table 1: Topics and milestones for GAM 380 by week

The first seven module milestones are done individually. For the final project, students are given a choice of continuing to work individually or of forming small two to four person teams. This approach is designed to ensure that students gain familiarity with the tools and learn fundamental material without the added pressures arising from team interactions. It also makes it easier to assign grades and evaluate student understanding when the early milestones are completed individually. By postponing team formation, students can discuss design and development ideas with other students, solidify their design, test their ideas and code, experiment with the problem space specific to their design, and select compatible teammates. The authors more fully address this approach to game development assignments in another article [11].

2.6 Grading

Every type of content produced for the course, as well as each task and project has its own set of requirements. Each module of the course includes grading points for conforming to the requirements of that content type. As an example, the table below shows some of the requirements for a selection of content types:

Content type	Requirements
Model geometry	• Created at (0,0,0) with Z forward, Y up • 1 unit = 1 foot • Triangles, or quads where not animated
Images	• Targa format, RLE • 24 bit, or 32 bit if alpha needed • Width and height a power of 2
Sounds	• Mono for effects • Stereo for tunes • Sampling rate appropriate to sound
Animations	• Duration, key first and last frame • Correctly tagged
All Content	• Follows naming conventions internally • File type, name, and directory correct • Submitted to source control properly • Assignment grading requirements • Quality

Table 2: Course requirements by content type

The students also design their own game level and set budgets for the different types of assets. The budgets include polygon counts, sound sizes, image map sizes, and memory usage.

Every milestone is divided into a 90% and a 10% list of requirements. The 90% requirements are the basic, mechanical requirements. The 10% section lists a number of possible tasks that go beyond the basics. The 10% section includes additional programming or high quality art for students to work on at their option. For example, the set of requirements for the first milestone is:

90% (all are required)

- fonta.tga (font image file)
  - committed to SVN w/comment
  - alpha setup correctly
  - characters in full white
  - width & height powers of 2, <= 2048
  - width & height smallest size for image
  - format: TGA, 32 bit, RLE
- s_font.cs (C# code file)
  - committed to SVN w/comment
  - compiles w/o warnings
  - builds and runs
  - displays “Hello NAME”

10% (choose any of the following)

- stylized font (art)
- color tinted display of letters (programming)
- proportional font supported (programming)
- animated display of letters (programming)
- icons (as characters) for game (art)

2.7 Results

Student response to the GAM 380 course has been positive. Twenty five students completed the class in the Fall 2007 quarter. The class was made up of 1 freshman, 3 sophomores, 10 juniors, and 11 seniors. Of these, 16 (64%) indicated they were game development majors, 3 (12%) indicated they were computer science majors, and the remaining did not indicate a major.

Overall, the results show those with a programming background, either self declared or computer science majors, did better in the class. The division between programmers and non-programmers was not straightforward. Of the 25 students who completed the class, 3 initially self reported as being more of an artist or designer and 14 as programmers. The remaining 8 students were classified based on instructor evaluation. The final count was 16 programmers and 9 non-programmers. The average class grade for the 16 programmers was 86% compared to 79% for the 9 non-programmers. The average class grade for the 3 computer science majors was also 86% versus 79% for the 16 game development majors.

Of the major assignment types -- 7 individual milestones, 3 quizzes, and one final team project -- the quizzes showed the biggest difference in scores between programmers and non-programmers. Programmers scored an average of 80%, while the non-programmers scored 71%. The milestones and final projects showed less difference in grades: 79% for programmers versus 74% for non-programmers and 91% for programmers versus 88% for non-programmers.

3. CONCLUSIONS AND FUTURE WORK

Based on the class goals, the grade data suggests more work needs to be done to minimize grade results affected by programmer versus non-programmer skill sets. This is especially applicable to the impact of the quizzes. However, the results are encouraging. One possible reading of the data could be that even though game development is at its foundation a highly technical field there exists opportunity to mix programmers and non-programmers on equal footing in a game development class. The programmers were capable of creating functional models, animations, and sounds. Likewise, the non-programmers were able to write scripts, modify code, and shepherd data through the complexities of the content pipeline. Both groups created successful projects and solved many problems.

GAM 380 provides benefits to a variety of students. For non-technical game developers and artists, GAM 380 provides a healthy insight into some of the inner workings of the content pipeline. For example, non-technical developers learn the importance of properly naming objects, how those names are used by the programmers at the far end of the pipeline, and the why behind some of the arcane rules intrinsic to game development. The course gives programmers glimpse into the world of artists and helps them develop a better understanding of the whole game development space. This understanding will hopefully help them write code better suited to the problem space, and, possibly, easier to use tools.

While the Fall 2007 section of GAM 380 had many junior and seniors since it was the first undergraduate XNA course to be taught at DePaul CTI, the focus on content makes it a good second-year course. It allows students to get a taste of every aspect of game creation and doesn't depend on programming or quality art. In short, it allows students to get a taste of all the disciplines. Hopefully, some of the lessons from this course will aid students as they progress through modeling, animation, and programming courses. One of the curricular goals is the improvement of content quality in the more advanced courses, which can hopefully be achieved if the students do not have to expend all their energy on the mechanics of the content pipeline.

In the future, we plan to split this course into two courses. One of the courses will be a slightly more introductory course, possibly skipping animated meshes and spending more time on path-based animations. The other will be a more advanced course, run later in the curriculum, to allow students to create and also program more advanced content. The goal is to retain the benefits of an early course in console development as embodied in the introductory course, while responding to the requests of the students for more in-depth coverage of content development in the more advanced course.

While the support provided by the XNA lab grant awarded to DePaul CTI is both helpful and unusual, the course described in this paper is accessible to a variety of institutions. Microsoft announced in February 2008 that XNA Game Studio will be made available free of charge to students. Further, games created by students can be made available on Xbox Live for download. Both large and small institutions can take advantage of the opportunities for educating students about the console development pipeline that XNA provides.

4. ACKNOWLEDGMENTS

We are indebted to André Berthiaume for his feedback on this article. We thank the DePaul CTI gaming faculty and students for their work on the XNA Lab Grant. This course was supported by an XNA Lab Grant provided by Microsoft Corporation.

5. REFERENCES

[1] Adams, J.C. 1998. An Object-oriented Capstone Project for CS-1. In Proceedings of the SIGCSE Technical Symposium on Computer Science Education, (Atlanta, Georgia, February 1998).

[2] Chamillard, A. 2006. Introductory Game Creation: No Programming Required. In Proceedings of the SIGCSE Technical Symposium on Computer Science Education, (Houston, Texas, March 2006).

[3] Claypool, K. and Claypool, M. 2005. Teaching Software Engineering Through Game Design. In Proceedings of the Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, (Monte De Caparica, Portugual, June 2005).

[4] Feldman. T.J., and Zelenski, J.D. 1996. The Quest for Excellence in Designing CS1/CS2 Assignments. In Proceedings of the SIGCSE Technical Symposium on Computer Science Education, (Philadelphia, Pennsylvania, February 1996).

[5] Faltin, N. 1999. Designing Courseware on Algorithms for Active Learning with Virtual Board Games. In Proceedings of the Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, (New Orleans, Louisiana, 1999).

[6] Korte, L., Anderson, S., Good, J., and Pain, H. 2007. Learning by Game-Building: A Novel Approach to Theoretical Computer Science Education. In Proceedings of the Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, (Dundee, Scotland, UK, June 2007).

[7] Leutenegger, S. and Edgington, J. 2007. A Games First Approach to Teaching Introductory Programming. In Proceedings of the SIGCSE Technical Symposium on Computer Science Education, (Covington, Kentucky, March 2007).

[8] Parberry, I., Kazemzadeh, M., and Roden, T. 2006. The Art and Science of Game Programming. In Proceedings of the SIGCSE Technical Symposium on Computer Science Education, (Houston, Texas, March 2006).

[9] Parberry, I., Roden, T., and Kazemzadeh, M. 2005. Experience with an Industry-Driven Capstone Course on Game Programming. In Proceedings of the SIGCSE Technical Symposium on Computer Science Education, (St. Louis, Missouri, February 2005).

[10] Perkovic, L. and Settle, A. 2007. Computing Branches Out: On Revitalizing Computing Education. In Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering, (Las Vegas, Nevada, June 2007).

[11] Settle, A., Linhoff, J., and Berthiaume, A. A Hybrid Approach to Projects in Gaming Courses. In GDCSE 2008: Microsoft Academic Days Conference on Game Development in Computer Science, (Celebrity Century, February 28 – March 2, 2008).

[12] Sweedyk, E. and Keller, R. 2005. Fun and Games: A New Software Engineering Course. In Proceedings of the Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, (Monte De Caparica, Portugal, June 2005).

[13] Tsai, M., Huang, C., and Zeng, J. 2006. Game Programming Courses for Non Programmers. In Proceedings of the International Conference on Game Research and Development, (Perth, Australia, 2006).

[14] Wolz, U. and Pulimood, S. M. 2007. An Integrated Approach to Project Management through Classic CS III and Video Game Development. In Proceedings of the SIGCSE Technical Symposium on Computer Science Education, (Covington, Kentucky, March 2007).