Creating a Games Class: A Walkthrough

Shannon Duvall
Elon University
Box 2320
Elon, NC 27244
336-278-6232
sduvall2@elon.edu

ABSTRACT
Adding a games class to a traditional computer science curriculum is becoming a popular way to attract and motivate students to consider computing. However, there are many decisions to be made when adding such a course to an existing Computer Science curriculum. What are the goals of the course? What platform and editor should be used? How should the course be administered? This paper gives a walkthrough of creating a games class that fits in a traditional curriculum, based on experience of creating the CSC 420 course at Elon University.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education, game development, curriculum.*

General Terms
Design, Human Factors.

Keywords
Computer Science education, pedagogy, game development

1. INTRODUCTION
Game development courses are becoming increasingly popular as a way to attract students to computing and motivate existing computer science majors. [1][5][11][12] Video game development is fun, results in a product students are excited about, and gives students experience in a field many aspire to. Academically, creating video games involves both technical and artistic skills, large product management, and teamwork. Games can also provide an excellent context for teaching core concepts such as artificial intelligence, computer graphics, and software design. [3] [14][15]

Creating a game development course involves making many decisions about goals, administration, and evaluation. This paper gives advice on making these decisions based on the experience gained in creating such a class at a liberal arts university: the upper-level elective course has been administered several times, using various platforms and administration techniques. We present our lessons learned in choosing the course goals and tools, administrating the course, and coordinating a student designed game as a final project.

ICFDG 2009, April 26–30, 2009, Orlando, FL, USA.Copyright 2009 ACM 978-1-60558-437-9…$5.00.

2. CHOOSING THE GOALS
Understanding by Design [16] advocates a "backward design" approach which involves first determining the desired results of the course, then identifying evidence that the goals were met, and finally planning the instruction and assignments for the course. Having the goals of the course clearly defined first focuses the rest of the course creation.

Naturally a game development course has the goals of teaching the technical skills required to program a video game and the criteria for what makes a fun and playable game. However, to fit a game development course in a traditional curriculum, it is vital to teach core computer science concepts in addition to the gaming concepts. The following are concepts that can be covered naturally in a gaming context.

Computer Graphics

Naturally, games classes can be a great context for discussing concepts in computer graphics, even those that are not central to games. For example, we teach the ray tracing algorithm in class as an alternative to the feed-forward rendering system we use in our games.

Artificial Intelligence

Video games can also give a context for teaching algorithms from Artificial Intelligence such as path finding, machine learning, heuristic search, and group intelligence.

Project Management

For many students, a video game will be the most code they have written and managed. It may also be the first opportunity they have to work as part of a team. Thus, the game class is a good forum for teaching code version-control management tools like CVS, modeling languages such as UML, and user documentation.

Software Design

Since video games do require managing lots of code, it is natural to discuss software design principles and patterns. In our course keeping code easy to read, maintainable, and extensible is key. Each game is evaluated as much for its software design as its functionality and playability. In addition, one assignment involves taking a game, refactoring it, and then extending its functionality.

Soft Skills

It is becoming increasingly important that computer science majors be equipped with soft skills such as the ability to work in a

team, give presentations, and communicate effectively. [9] These skills are especially important in the video game creation industry and can be practiced in the games course. In our course, student groups work on a semester-long project of creating a game. The groups participate in team-building exercises, give several presentations on their progress throughout the semester, and evaluate one another on their teamwork at the end of the semester.

Social Topics

Video games have had an enormous impact on our society in both positive and negative ways. Some argue that the violence and isolation in game playing is detrimental; others say that video game playing makes us smarter and more dexterous. In addition, video games can be used to make artistic and social statements that can be either positive or negative. Creating video games involves ethical issues from the use of stereotypes to intellectual property rights. The history of video games highlights not only important computer scientists but also interesting legal issues. Exposure to these topics can add a social science element to the class as well as introduce students to the culture of video game development.

3. CHOOSING THE TOOLS

There are several platforms, programming languages, and development environments available for game development. Choosing the best tools for your course involves considering your students' prior experience, course goals, and personal preferences. In this section we will lay out the pros and cons of the platforms we have used in different versions of the course.

One of the first decisions to make is the number of dimensions in which to present the games: 2D or 3D. While 3D games may at first seem the obvious choice because they represent most popular console games today, we did not believe that would be appropriate for our course. 2D games are easier to teach and produce, and more freely and readily available tools are available to create content for 2D games. The core Computer Science concepts required for 3D games, such as a rendering pipeline based on matrices to represent geometric transformations, can be taught in 2D to provide a solid foundation for further study. Finally, many popular 2D games are available to motivate students on the web and their cell phones. That said, all the environments discussed below support both 2D and 3D games.

Java

Since our foundational programming courses are taught in Java, we originally taught the games course in Java using the Eclipse IDE. In this way we avoided having to teach a new language or tool. The biggest advantage of using Java is that the games created in the course were platform-independent and the students could publish JAR files on the web for anyone to run. On the other hand, there was little gaming support and the students did not feel they were having a professional game development experience. There are several textbooks available for Java game development. We used *Developing Games in Java*. [2]

XNA

Using Microsoft's XNA platform gives the students the chance to develop games for a gaming console (the Xbox 360) using a professional development tool. The code is written in C# using the Visual Studio editor. Since the content is held in proprietary format, the XACT tool is used to create sounds. These tools are free for download. Our students did not find the move from Java to C# difficult, nor did they find a steep learning curve for XNA and Visual Studio. The biggest advantages of using XNA are the authentic professional development experience and the support for games that the platform provides. On the other hand, the students found the XACT tool difficult to use and they were disappointed that their games would only run on PCs with Visual Studio installed. There are also several good textbooks for teaching XNA depending on the level of programming experience of the students. We used *XNA Game Studio Express: Developing Games for Windows and the Xbox 360*. [8] We are currently in hopes that a new edition of the book is released for more current versions of XNA.

OpenGL

OpenGL is perhaps the most flexible candidate available since it is can be programmed for any platform in its raw form in C, C++, or Java. This means it shares Java's advantages in being cross-platform and a good fit to student's previous programming experience and it shares XNA's advantages in being used to create professional games. However, its major disadvantage is that it has no high-level support specifically for creating games and no suitable game oriented textbooks. Thus choosing OpenGL means either spending much of the course building the abstractions available in the other options or building them ahead of time for the students.

4. THE COURSE AS A GAME

Games are attractive to students because they are fun, provide immediate feedback, and motivate through natural competition. With this in mind, we decided to run the course as a game itself. To give a game feel to the course, the first step was to change the vocabulary of the course. The syllabus and course materials refer to the students as players, the assignments as quests, and the professor as the Game Master. The tests are called challenges, and the classes themselves are called player summits.

The syllabus does not read as a conventional syllabus at all but rather as a game manual. It begins with an invitation to join the game and tells the object of the game (the course goals). It introduces the Game Master (professor information) and the "required technical manuals" (textbooks). It gives instructions for how to play (class and assignment specifics) and how to score points (grading). The game manual concludes with a manufacturer's warning: "Successful completion of the game requires a significant commitment of time, energy, and thought. Since the game requires both individual and team accomplishments, each player should be ready to dedicate himself and his unique talents to his team. While the cost of playing is great, the rewards will be greater still."

In addition to changing the vocabulary of the class, we introduced game elements to the course. All of the quests and challenges are worth large amounts of points to give the feel of a video game. When a player earns enough points through the quests and the challenges, the player levels up and earns coins. (The coins are plastic pirate gold pieces purchased at a party supply store.) The coins can be used to buy power ups such as cheat sheets for the challenges, late days for the quests, a "do over" for a presentation that doesn't go well, or technical help from the Game Master. They also can be bartered among the players for help between groups. For example, if a student is a talented digital artist, she

can decide to make art for groups other than her own in exchange for coins. At the end, coins can be turned in to the Game Master for more points.

Finally, small games are held in class on a routine basis. For example, when an assignment is completed and students show their products to the rest of the class, the class votes on the best one. The winner gets either coins or points. Competitions should highlight different talents in the class, not just technical accomplishments, to ensure that everyone has a chance to win a competition during the semester.

5. PROJECT ADMINISTRATION

We decided that our course would include a major project (known in the class as "The Ultimate Mission") that includes creating a game engine for a specific genre of game and a 2D video game based on the genre engine. This choice serves dual purposes by emphasizing the software's design and giving the student as much latitude as possible to demonstrate what they have learned about game design during the course. [13] We certainly do not expect students to produce professional grade games, or even very original games, during a semester, but the best ones have been quite creative, using student created resources and school related themes. While the best projects have been approximately equal in quality despite the tools used, the overall student satisfaction and the quality of the lower end projects was better using XNA.

At first the major project was a four-week endeavor, and then we made it a six-week duration. While these shorter times ensured that students had a grasp of the course material before starting the final project, it seemed to put too much pressure on them to produce a quality game at the end of the semester. This semester, we decided that the project would last the entire semester. Although the students do not have the technical skills to start programming right away, they can spend the first part of the semester designing a game and team building. The majority of the programming still occurred at the end of the semester, but this extra time allowed the students to get some of the basic work done earlier and gave us extra opportunities to teach more project management concepts.

The administration of group projects, especially semester-long projects, is a tricky issue. We decided that the groups would use class time to report both formally and informally on their progress weekly. This format helps groups make progress throughout the semester, allows them to help each other brainstorm game ideas, get over technical stumbling blocks, and review each other's code. It also provides the students with accountability to one another, which is often more motivating than being accountable to the professor only.

Students begin by writing a profile of themselves, giving their strengths and weaknesses in both hard and soft skills. They also include their preferences in terms of what genre of game they would like to work on. These profiles are posted on the course website and help students self-select their teams. The selection of teams is complete within the first two weeks of class. The transparency of this process helped emphasize the importance of each students' role on the team and the need for a common vision for the final game and on average produced better teams than when we simply let students pick their own teams based on their own private criteria.

Next, students enter the design phase of the project, culminating in a game proposal. We invited several staff and faculty members to the game proposals as an "expert" panel, although they were not all game development experts. The panel judged the proposals based on the presentation style, game idea, and how thoroughly the game design was fleshed out. Having an expert panel created a feeling of importance to the proposal process and forced the groups to describe every aspect of the game, despite the fact that the rest of the class was familiar with the group's game concept by the time of the proposal. Later the panel was invited back for the presentations of the final products.

Once the teams are formed they create team names and roles within the team. They make a website for their team on which they keep the current release of their games as well as the supplemental reports that are required such as design and user documents. We do not expect that individuals will necessarily keep the same roles throughout the semester, since they will want to try different aspects of making the game and discover their own strengths. During the first part of the course, students generally chose from a "pre-production" job such as artist or sound director, and later in the semester had "in-production" jobs like level creator or engine programmer.

By the halfway point of the semester the game concept has been fully formed, the content of the game is in place, the engine design is decided upon, and the screens can be drawn and animated, even if all of the user interaction is not in place. The teams create for themselves a development schedule and define their weekly goals for completing the engine and the game, and continue to report to the rest of the class regularly. During the second half of the semester the expectations for project progress are intensified and the number of other assignments is cut back extensively so that students have time to complete the implementation of their projects.

In the last two weeks of the course the teams enlist students outside the course to play test their games. They are not allowed to guide the testers unless they are asked for help. The testers then give feedback to the team as to which aspects of the game they liked and what things can be made better. During the final presentations of the projects they report on the testers' comments and their corresponding changes to the game. Sample projects are given in Figures 1 and 2.

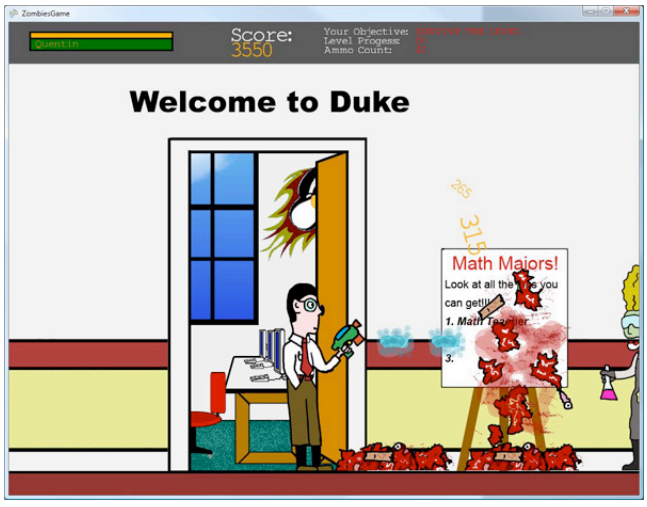

Figure 1: "Elon Zombies" Zombies have taken over campus. In addition to killing them, the player must try to find out why there are zombies everywhere.

Figure 2: "Shipwrecked" The player is shipwrecked on a deserted island. To get off the island, he must collect items to build a raft such as bamboo, rope, and canvas for a sail. In order to pick up each piece, however, he must win a mini-game such as tic-tac-toe or a memory match.

6. OTHER DETAILS

6.1 Opening Day Activities

The first day is a great opportunity to get students excited about the course and thinking about video games in a new way. The following are possible exercises for the first day of class.

Play and Analyze

Ask the students to play a video game they have never played before. With free online casual game sites offering thousands of games it is not hard for everyone to find a new game. Ask students to play a game outside their favorite genre. For example, we will often ask men in the class to play a game marketed for women. Then engage the students in a discussion about what makes a game fun. This discussion is a great time to introduce vocabulary about game play such as emergence, non-linearity, and feedback.

Dream a Game

Gather students into groups and let them brainstorm concepts for a new video game. Provide props such as toys, scratch paper and markers, and flashcards for inspiration. Many brainstorming activities are given in the book *Game Design Workshop.* [7] One such idea is to give each team cards with random words written on them. The team must then come up with as many game concepts based on the words as possible. Have each group settle on the one concept they think is most promising and flesh out the idea in as much detail as possible, including sketches. Let each group present their game to the class and the class can vote on the best of all. Discuss what elements go into designing a game, introducing vocabulary such as non-player character, resources, and storyline. We have used this exercise in the past and found that students have often come up with a game idea that later becomes the final project game.

Team Building

Because our project takes the entire semester, it is vital that the project teams be created early, and also that the teams are well-balanced in terms of personality and talent. The students need to get to know one another's strengths as well as discover their own strengths. Therefore, it is often good to start class with some team-building exercises. One such exercise is known as "the floating stick" and videos of groups doing this exercise can be found online. Each group should form two lines facing one another and each person should hold out one finger. The group collectively holds up a long dowel. The object is to lower the dowel to the floor while keeping each person's finger on the dowel. Someone should be appointed as overseer, who calls out those people who are no longer touching the stick. The game is called "floating stick" because the stick naturally gets raised rather than lowered because everyone is concentrating on holding the stick. Only when the entire team moves as one can the task be accomplished.

Many other team-building activities are easily found or created, such as leading one another blind-folded around an obstacle course or solving a logic puzzle together. Ask the students to write about what they learned about themselves. Were they a leader or a follower? How did they communicate to the group? How creative were they in finding solutions? Ask each student to identify their personal strengths and what they would like to work to improve during the semester. This may also be a good time to introduce the various roles of team members such as team leader or art director.

6.2 Evaluation

How should games be graded? In computer science we are often used to grading projects based on code functionality and design. A good game, however, should also be fun, easy to learn, and challenging. In our course we ask the students to define what makes a game fun. Of course, different students will have different ideas about which games are fun and what properties make a game fun, but we encourage the students to find a few qualities of good games that everyone can agree upon. Next, we challenge the students to give objective ways that they can tell that a game meets that criterion. For example, students agreed that a good game would be fair. They then said that a fair game is not too hard. "Too hard" is not an objective measure, however, so the students had to refine what was meant by a game being too hard to be fair. They then decided that if the average person loses the game within the first thirty seconds of play the game is too hard. Now there is an objective criterion that all the games must meet. The students themselves created the standards for their work.

In addition to programming projects, our course includes some theoretical content from graphics and some social content from the history and impact of video games. Our tests, (or "challenges") cover a variety of topics. For each test, the students were given several questions in various categories and were allowed to choose a number of questions from each category to answer. The following are some sample questions from tests:

- *On Gameplay:* Choose one of the following games and discuss the game's economy.
- *On Graphics:* Consider the following two images. They are rendered using two different shading techniques. Name and describe each.
- *On Society and Games:* Discuss the controversy surrounding the Digital Rights Management of the game "Spore".
- *On XNA:* Describe the sections of the game loop and what code should go into each section.

- *On Code Design:* Consider the following pseudocode. Tell what code smells are present and how the code could be refactored. [6]

6.3 Assignments

The various assignments ("quests") given in the course reflect the goals for our course: learning to program games, being able to analyze gameplay, and utilizing good code design. The following are sample assignments given in the course:

- The students are asked to create an animation. There should be no user interaction, but the animation should have a story rather than simply being "pretty". They must use at least two of Lasseter's principles of animation to tell their story. [10]
- Arcade games are good beginning programming assignments. Students are told to re-create the arcade game Nibble, but they must put their own spin on the game in some way. For example, one team created a game where a Phoenix (our mascot) chased political candidates around the board (see Figure 3). The students' games must fit the criteria of a "good game" that the students created at the beginning of the semester. They also had to write an argument that the game was fun according to the Theory of Funativity. [4]

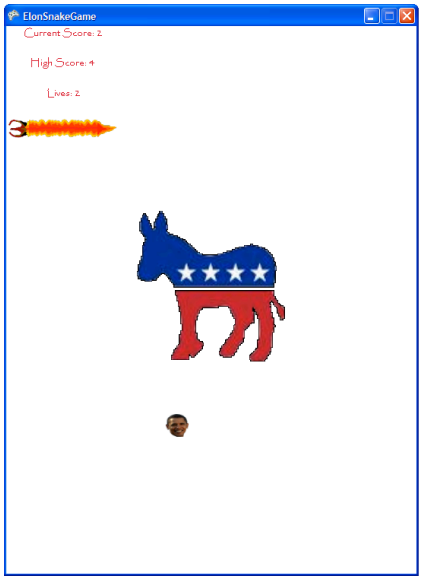

Figure 3: "Elon Snake" This take on the classic game Nibble includes the Elon Phoenix logo as the snake and political figures from the 2008 Presidential election (Barack Obama's head is the fruit to be eaten).

- After the arcade games are turned in, we conduct a code review and the students get a chance to refactor their games. They then add three new features to the game. If the code is well designed and open to extension, the new features should be added very easily.
- During class students play various games from different genres and different time periods, from original arcade games, early console games, casual games, serious games, and modern PC and console games. The students must then write a review of the games analyzing the game play, learning curve, and artistic and social aspects of the game.

7. CONCLUSION

The video game programming course is one of the most popular upper-level electives at Elon. We are very happy with the transition to XNA and plan to continue using that platform for the course in the future. Students were surveyed about their experience learning XNA and the Visual Studio platform. As the data in Table 1 shows, the students did not find the transition from Java and Eclipse to XNA, C# and Visual Studio very difficult.

Question	Average Student Response
On a scale of 1 to 5, with 1 being "very easy" and 5 being "very difficult" how was your experience with learning XNA?	2.8
On a scale of 1 to 5, with 1 being "very easy" and 5 being "very difficult" how was your experience with learning to use Visual Studio?	2.2

Table 1: Results on students' perception of learning XNA and Visual Studio

We also had some initial concern about the varied nature of the course goals. Would students see the course as only about game implementation? We surveyed students about how much they felt they learned about graphics and code design and found that they did indeed recognize their accomplishments in these areas as well. Results of these questions are given in Table 2.

Question	Average Student Response
On a scale of 1 to 5, with 1 being "none" and 5 being "lots" how much did you learn about code design this semester?	3.9
On a scale of 1 to 5, with 1 being "none" and 5 being "lots" how much did you learn about graphics this semester?	4.0

Table 2: Results on students' perception of learning code design and graphics

Finally, we found that the students really enjoyed the fact that the class was administered as a game. Many said it should be even more integral to the class. One student commented that the mini-competitions helped to define the goals of the assignments and the class. It is also interesting to note that one-third of the class was female, and they enjoyed the competitive nature of the in-class games as much as the males did.

To create a games course, start by defining the goals for the course, which can certainly include several core computer science concepts as well as the implementation and game play of the video games. Then decide on the tools that work best, given the background of the students and the desired final result of the game. We have found that the XNA platform is easy to learn and use and gives the students experience using a professional tool. Giving a games course a game-like feel made class fun and helped to immerse the students in the class experience. A major project in

the class works best when it is begun early and plenty of time is given to the design of the game before any work is done to actually implement it. Team members should be invited to discover and improve their leadership and communication skills through team building exercises. Let the students define the criteria of good games and evaluate them accordingly. The games course can be one of the most motivating contexts for learning and one of the most popular courses in your curriculum.

8. ACKNOWLEDGMENTS

Many thanks go to Dr. Tiffany Barnes and Robert Duvall for their help in getting the games course at Elon started and successful. This experience report includes lessons learned that they have generously passed on.

9. REFERENCES

[1] Bayliss, J. and Strout, S. 2006. Games as a ”flavor” of cs1. In SIGCSE ’06: Proceedings of the 37th SIGCSE technical symposium on Computer science education. ACM, New York, NY, 500–504.

[2] Brackeen, D. 2004. Developing Games in Java. New Riders Publishing, Indianapolis, IN.

[3] Claypool, K. and Claypool, M. 2005. Teaching software engineering through game design. SIGCSE Bulletin, Volume 37. ACM, New York, NY, 123–127.

[4] Falstein, N. 2004. Natural Funativity. Gamasutra. http://www.gamasutra.com/features/20041110/falstein_01.shtml

[5] Feldman, T. J. and Zelenski, J. D. 1996. The quest for excellence in designing CS1/CS2 assignments. Proceedings of the Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education. ACM, New York, NY, 319-323.

[6] Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, D. 1999. Refactoring: Improving the Design of Existing Code. Addison-Wesley, Boston, MA.

[7] Fullerton. T. 2008. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. Elsevier, Burlington, MA.

[8] Hall, J. 2008. XNA Game Studio Express: Developing Games for Windows and the Xbox 360. Thomson Course Technology, Boston, MA.

[9] Joseph, D., Ang, S., and Slaughter, S. 1999. Soft Skills and Creativity in IS Professionals. In *Proceedings of the Thirty-Second Annual Hawaii international Conference on System Sciences-Volume 7 - Volume 7*. HICSS. IEEE Computer Society, Washington, DC.

[10] Lasseter, J. 1987. Principles of traditional animation applied to 3D computer animation. Proceedings of the 14th Annual Conference on Computer Graphics and interactive Techniques. SIGGRAPH '87. ACM, New York, NY, 35-44.

[11] Leutenegger, S. and Edgington, J. A Games First Approach to Teaching Introductory Programming. 2007. SIGCSE ’07: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education. ACM, New York, NY, 115–118.

[12] Linhoff, J. and Settle, A. 2008. Teaching game programming using XNA. ITiCSE ’08: Proceedings of the 13th annual Conference on Innovation and Technology in Computer Science Education. ACM, New York, NY, 250–254.

[13] Newman, I., Daniels, M., and Faulkner, X. 2003. Open ended group projects a 'tool' for more effective teaching. Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20. Conferences in Research and Practice in Information Technology Series, vol. 140. Australian Computer Society, Darlinghurst, Australia, 95-103.

[14] Rankin, Y., Gooch, B., and Gooch, A. 2007. Interweaving Game Design into Core CS Curriculum. Academic Days on Game Development in Computer Science Education. Nassau, Bahamas.

[15] Settle, A., Linhoff, J., and Berthiaume, A. 2008. A Hybrid Approach to Projects in Gaming Courses. Proceedings of the 3rd international Conference on Game Development in Computer Science Education. GDCSE '08. ACM, New York, NY, 36-40.

[16] Sumner, R. W., Thuerey, N., and Gross, M. 2008. The ETH Game Programming Laboratory: a Capstone for Computer Science and Visual Computing. Proceedings of the 3rd International Conference on Game Development in Computer Science Education. GDCSE '08. ACM, New York, NY, 46-50.

[17] Sung, K., Panitz, M., Wallace, S., Anderson, R., and Nordlinger, J. 2008. Game-themed Programming Assignments: the Faculty Perspective. SIGCSE ’08: Proceedings of the 39th SIGCSE technical symposium on Computer Science Education. ACM, New York, NY, 300–304.

[18] Wiggins, G. and McTighe, J. 2005. Understanding By Design. Association for Supervision and Curriculum Development.

[19] Yee, B., Sturman, D., and Feiner, S. 2007. Integrating Video Game Development Experience in an Academic Framework. Academic Days on Game Development in Computer Science Education. Nassau, Bahamas.