Introductory Game Creation: No Programming Required

A.T. Chamillard
Computer Science Department
University of Colorado at Colorado Springs
Colorado Springs, CO 80933-7150
719-262-3150

chamillard@cs.uccs.edu

ABSTRACT

Many incoming college freshmen have accumulated a significant number of hours of experience playing computer games. Extending that experience to actual game creation activities can be highly motivational for these students. Most of these activities require some level of programming expertise, however, making them activities too advanced for the majority of incoming students.

In this paper, we describe a freshman-level course called Problem Solving through Game Creation. Students learn to use a number of drag-and-drop game creation tools to develop both 2D and 3D games, with no programming required in the course. We also cover a variety of other topics and tools related to game development.

Our experience has been that students enjoy the course, but we have more formal course goals as well. Specifically, we hope to motivate students to declare and complete the computer science major and to better prepare students for the initial required computer science courses. We describe these goals in detail and discuss the process we have initiated to continually evaluate achievement of those goals.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, curriculum.*

General Terms

Design, Human Factors.

Keywords

Computer game development, Introductory courses, Student motivation.

1. INTRODUCTION

Many incoming freshman have extensive computer gaming experience, and some of them view themselves as potential independent game developers as well. Given the affinity these and other students feel toward computer gaming, using this highly motivational domain may encourage incoming college students to pursue computer science as a major. Incorporating computer game development in introductory computer science courses could also help retain students struggling with the low-level language issues faced by beginning programmers, especially if the game development content is integrated to provide incremental success for those students. We also believe that integrating such motivational content into introductory courses will contribute to more effective student learning.

Unfortunately, most game development activities require some level of programming expertise, making those activities too advanced for many incoming college students. For the Fall 2003 semester, we developed a new freshman-level course – CS 101: Problem Solving through Game Creation – for developing 2D and 3D computer games using drag-and-drop tools (e.g., no programming is required). Students learn to use a variety of game development tools, but also learn many of the underlying computing concepts associated with game implementation. The high-level goals of the course are to motivate students to declare and complete the computer science major and to better prepare the students for the initial courses in the computer science major. We are currently teaching the course for the third time.

Education researchers have identified the positive motivational effects of incorporating computer game development into introductory computer science courses [2, 3]. Educators have integrated game development activities in courses ranging from introductory computer science courses [9] through upper-level courses [4, 6, 11] and even senior-level capstone courses [8]. Although the majority of computer science educators appear to believe that computer games represent a valuable educational tool, contrary opinions exist as well [12].

Researchers have proposed that one of the key reasons that males predominate in computer science is that most games are marketed toward males [10], though it has also been argued that computer games can be used to interest more girls in computing [5]. We believe that both genders can benefit from the integration of game development activities into computer science education. For example, women appear (in general) to prefer creative games rather than the destructive games preferred by men; providing flexibility in the game assignments included in a course can effectively engage both genders in those assignments. In the Fall 2004 offering of the course described below, the average for the (5) female students was 21% higher than the average for the (27) male students, and the top two students in the class were women.

Although there is strong anecdotal evidence that integrating computer games into computer science courses leads to increased student motivation and improved student learning, the literature does not reflect the results of careful, quantitative evaluation of such integration. We view the course described in this paper, and the evaluation of the course effectiveness, as our first steps in a long-term, quantitative study of these issues.

This paper makes two contributions to the body of knowledge related to the use of game development in freshman-level courses. The first – and primary – contribution is the description of a freshman-level course that is designed to let students create a variety of 2D and 3D games without requiring any programming. This course appears to be a unique approach to having freshman students create interesting games, and as such could be of interest to others interested in introducing these topics early in a computer science (or more general) curriculum. The second contribution is the clear specification of our course goals and, perhaps more importantly, a detailed evaluation plan to evaluate our fulfillment of those goals. Although preliminary results from the first two course offerings are also provided, we believe the general course description and, to a lesser degree, the goal/evaluation plan discussion to be more significant contributions.

The following section provides a brief overview of the course topics and their presentation sequence. Section 3 provides some examples of student course work, and Section 4 describes our two high-level course goals and how we plan to evaluate our achievement of those goals. Section 5 describes some of the ways in which we evolved the course from the first offering to the current offering. Section 6 concludes the paper.

2. COURSE STRUCTURE

The course covers a variety of material, including general game development topics, four drag-and-drop tools for developing 2D and 3D games, and other tools and techniques that support game development. The course is worth 2 credit hours, meeting for a total of 20 lessons (75 minutes each). A condensed course syllabus is provided in Table 1.

As shown in the condensed syllabus, half of the lessons are dedicated to using the four different drag-and-drop tools, although this coverage is spread out over the duration of the course. We designed the course to cover some required introductory material, then quickly have the students develop their first game. Throughout the course, we want students to complete their game using a particular tool before moving on to the next tool, so we interleave the material for the drag-and-drop tools with other game-related topics and course assessments.

Table 1. Condensed Course Syllabus

Topic	Number of Lessons
Introduction	1
Game History and Game Design	1
Game Maker (2D game tool)	3
Graphics and Paint Shop Pro	1
Sound and Music	1
The Games Factory (2D game tool)	3
Mid-Term Exam	1
Pie 3D Game Creation System (3D game tool)	3
Milkshape 3D (3D modeler)	2
3D Gamemaker (3D game tool)	1
Game Implementation through Programming (Java)	2
Final Exam	1

Although half of the course lessons address the use of these tools, it is important to note that student proficiency with these specific tools is not a significant goal for us. As we discuss and work with each tool in class, we find many opportunities to discuss core computer science ideas in the context of that work.

For example, the Game Maker tool uses "objects" as the active entities in the games developed by the tool. We can set basic behavior characteristics for these objects (making an object bounce off the walls, for example) as well as providing more complicated behaviors within the objects. These game objects also have state (speed, location, etc.), providing us an opportunity to introduce the students to some basic object-oriented concepts. In fact, because we can place numerous instances of the object – which all exhibit the same behavior but have unique state information – we can even introduce the idea of instantiation to create objects from a base definition.

Game Maker also provides explicit use of a standard selection construct, letting students specify conditional execution of particular actions using "if" and "else." In addition, the tool implicitly executes a continuous game loop during game execution and students can explicitly identify actions to occur on each step of the loop. We therefore have an opportunity to discuss both selection and iteration control structures as we use the tool.

The Games Factory provides a different model for game development and execution, where some behavior is captured within the objects but most of the game actions are defined separately in an "Event Editor." The Event Editor is used to specify particular events, such as a collision between two objects, and the resulting actions to execute if that event occurs, such as destroying both objects. This lets us discuss the idea of event-driven programming and how it works, as well as providing an opportunity to discuss the differences between the Game Maker and The Games Factory approaches.

It seems to be a very natural progression starting the course with the 2D development tools then moving on to the 3D tools. Unfortunately, the 3D tools we use in the course are much less robust than the 2D tools. It seems that we should then change the presentation order to cover the 3D tools first. This is problematic, however, because we want the students to develop some comfort developing games before moving on to more complicated tools. It also seems to be the case that the students are more impressed with 3D games even if the tools they use to develop those games are more brittle or less flexible.

3. EXAMPLE STUDENT WORK

Game creation assignments comprise 50% of the course grade, with the remaining 50% allocated to a Learning Style Survey and the Mid-Term and Final Exams. Students complete four separate game creation assignments, building games with Game Maker, The Games Factory, Pie 3D Game Creation System, and 3D Gamemaker.

In general, assignment points are allocated in the following way:

- Design Document: 10%
- Game Capabilities: 70%
- Additional Features/Tool Analysis: 20%

The intent of the Design Document is for the students to think about their game design before actually creating the game using the specified tool. Unfortunately, an informal poll of the Fall 2004 students indicated that almost every student created the Design Document after completing their game. Although this is consistent with our experiences with beginning programmers and their use of algorithms in program design, we still believe there should be a way to convince students to design before implementing, even in an introductory games course. We have not yet identified an effective way to accomplish that, however.

The game capabilities points are allocated to the specific requirements for a particular assignment. For the additional features points, we expect students to explore the game creation tools to take advantage of additional tool features not discussed in class. In some cases, we ask that students evaluate strengths and weaknesses of the tool rather than exploring additional features.

Despite the fact that the students use drag-and-drop tools to create the games in this course, some of the resulting games have been quite impressive. For example, a student used Game Maker to create the game shown in Figure 1 in the first few weeks of the semester.

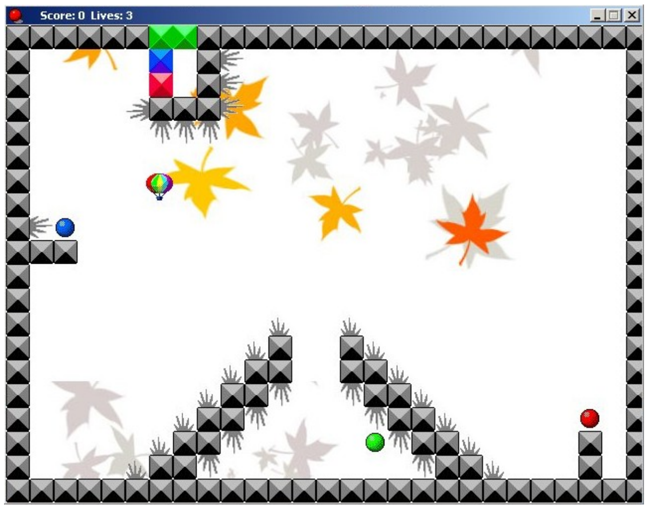

Figure 1. Student Game Maker Game

The background slowly scrolls downward, giving the impression of falling leaves. The balloon is controlled using the arrow keys, with acceleration and gravity affecting the behavior of the balloon in reasonable ways. If the balloon strikes a spike, the game plays an interesting sound, deducts a life, and returns the balloon to its starting position. Finally, "collecting" a ball of a particular color removes the block(s) of that color from the portal at the top left of the screen. When all the blocks have been removed, the balloon can pass through the portal and enter the next level of the game.

Figure 2 shows a game created by a student using The Games Factory. In this game, the player controls the plane shown in the upper center of the screen using the arrow keys. The terrain scrolls with the plane as it flies around, yielding a play area much larger than that shown in the figure. The silver planes chase the player's plane trying to shoot it down, and the player's plane can of course shoot back as well.

Finally, Figure 3 shows a game created by a student using Pie 3D Game Creation System. In this First Person Shooter game, the player walks through an environment trying to escape the guards and reach the goal area. In addition to guards and walls, the environment also contains weapons that can be picked up and used by the player, as well as doors that require the player to locate and pick up a key before the door will open.

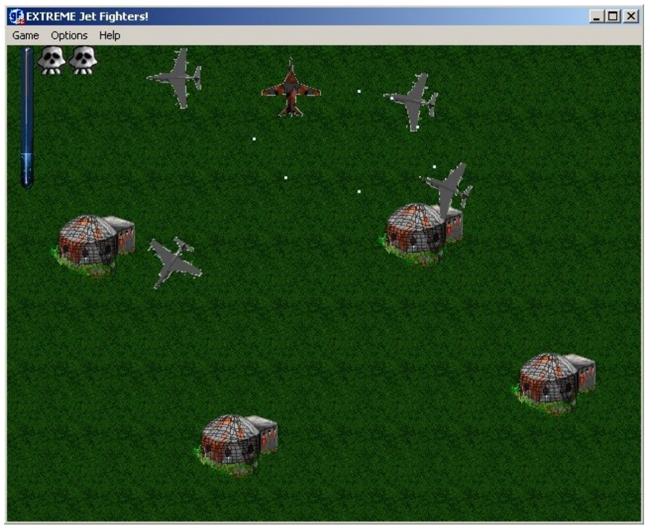

Figure 2. Student The Games Factory Game

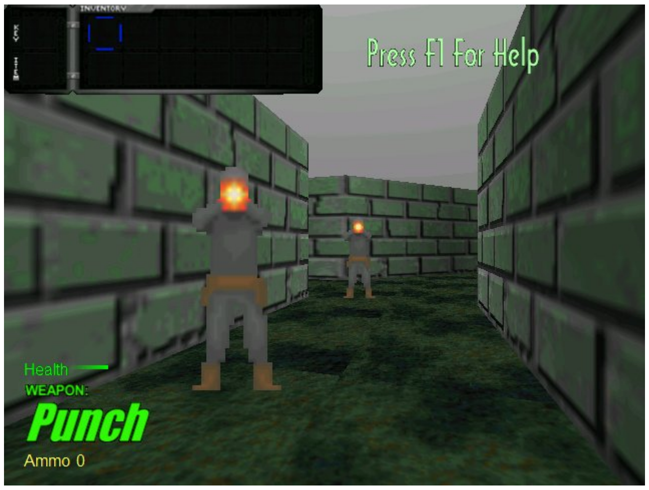

Figure 3. Student Pie 3D Game Creation System Game

The fourth tool – 3D Gamemaker – is very constraining in terms of game creation capability. In fact, we include this tool (while only dedicating a single lesson to its use) so that students can critically analyze the tool in comparison to the other 3 tools in the course. We discuss their analysis from the perspective of both a game developer and a user of a game created by the tool. In addition, we also show how to import a 3D model generated using Milkshape 3D into the game, giving us an opportunity to demonstrate how different game assets are typically generated using a variety of tools.

Students clearly view the game creation assignments as very motivational opportunities to exercise their creativity. During the Fall 2004 and Fall 2005 offerings of the course, we have also included a class demonstration of the top 5 games for each assignment. Students seem to find that demonstration very motivational, and often applaud the games as they see the creativity that their fellow students have demonstrated.

4. COURSE GOALS

This section describes the two main goals we have identified for the course and discusses how we will continually determine whether or not we are achieving those goals.

4.1 To motivate students to declare and complete the computer science major

Computer gaming is an inherently motivational topic, particularly for those students who are comfortable with computers and are thinking about majoring in computer science. To supplement this inherent motivation, the course is structured to provide additional motivation through each assignment, where students exercise their creativity in developing numerous complete games during the course.

Note that this goal is concerned both with *selection* of the computer science major and *completion* of that major. Although having students select the computer science major certainly provides the department with short-term benefits, we are also interested in having students successfully complete the major once they have selected it. The course described in this paper will clearly have a stronger effect on selection of the major, however, because many factors affect completion of the major, including student motivation and performance in the many required and optional computer science courses that follow this one.

To determine whether or not we are meeting this goal, we need to quantify how much this course has motivated students to declare the computer science major and how many of the students who take this course complete the major. Data about motivation is collected through exit surveys for the course, while data about completion of the major will be available from the Registrar (in several years).

Although we will collect this data on a continuing basis, we do have student motivation results from the Fall 2003 and Fall 2004 exit surveys. One of the exit survey questions asks the students to rate, on a scale from 1 to 5, whether they are much less motivated to major in computer science (1) to much more motivated to major in computer science (5). The mean of these ratings in Fall 2003 was 3.27 (N=15), indicating a slight growth in motivation to declare computer science as a major. The mean of these ratings in Fall 2004 was 3.78 (N=27), indicating stronger growth. We believe that the more positive results in the second offering were a result of the course changes discussed below.

4.2 To better prepare students for the initial required courses in computer science

Students entering the initial required computer science courses in the computer science major do not always have the minimal set of problem-solving and other skills expected of them by the instructors of those courses. This course includes materials and activities to try to improve the skill set of the students entering those courses. In many cases, students are taking both this course and our introductory programming course concurrently, so our ability to affect their incoming skills for that course is certainly limited.

In any case, we plan to determine whether or not we are meeting this goal through evaluations in the two required freshman computer science courses. We plan to administer a skills test during the first week of the semester in those courses, and we will also analyze student performance on the course assessments in those courses. By comparing the incoming skills and course performance in those courses for students who have and have not taken the course described in this paper, we hope to determine whether or not taking this course helps prepare them for those other courses. We are currently working to develop that skills test and administer it in the freshman computer science courses.

5. COURSE EVOLUTION

We are just beginning the third offering of this course, and we are committed to a continual process of course evolution and improvement. We expect a large portion of our course evolution efforts to be guided by the results of our analysis of the data we collect to evaluate course goal achievement. Because it will take some time for that data to be useful – particularly in terms of completion of the computer science major and preparation for later courses – we are also using student exit surveys and instructor observations to effect changes in the course.

One of the changes we made in the course was removing the Personal Software Process (PSP) [7] from the course content. We included the time management aspects of the PSP in the initial course offering, reasoning that incoming freshmen could benefit from learning these skills. Instructor observations indicated, however, that many students disliked this component of the course. These observations were supported by exit survey data, in which 7 of the 15 respondents identified the PSP as one of the worst parts of the course. Even more importantly, we note that following computer science courses do not require the use of PSP. It seems futile to force incoming freshman students to use the PSP and argue for its value when these students will no longer use it after this course. Given these observations, we removed this topic from the course content.

We held the first offering of this course in a computer lab so that students could follow along with the tool demonstrations. This environment caused a number of problems, however. The worst of these problems was exhibited by regular interruptions to the class discussions requesting that we back up and reiterate sequences of actions so that particular students could "catch back up" to the class. We found this to be particularly disruptive to the flow of the class, and many students also appeared to find these interruptions irritating. A lesser problem was that a number of the students simply used the computers in the lab to surf the web or play games. This did not affect the class as a whole, but we believe those students would have been better served by paying attention to and participating in the class lecture. The Fall 2004 offering of the course was held in a traditional classroom, largely because the 32 students enrolled in the class exceed the size of any of our labs. It is interesting to note that 15% of the Fall 2004 students said they would prefer that the class be taught in a lab rather than a classroom, but we will continue delivering the lectures in a traditional classroom environment.

A significant surprise in the Fall 2004 course offering was that 48% of the students indicated that they had problems either installing or using one or more of the game creation tools on their personal computers. We did not experience these problems in the initial course offering, nor have we had any similar problems on the computers in our computer lab. Given this feedback, we are investigating other 2D and 3D drag-and-drop tools to use in the course. The current tools are bundled with the course textbook [1], however, so any replacement tools we identify will need to be free. This limitation may, of course, lead us to continue using the current tools. In that case, we will take the approach we took in the Fall 2004 semester; if students are unable to get particular tools to work on their own computers, we send them to the computer lab to develop their games.

6. CONCLUSIONS AND FUTURE WORK

This paper describes a freshman-level course in which students create 2D and 3D games using drag-and-drop tools. Our intent was to make game development activities available to incoming freshmen without programming skills; most students do very well on the game assignments using the game development tools. We have also taught the majority of the course content to a group of homeschooled students ranging in age from 12 to 16; those students were also able to successfully complete the game development activities with no programming experience.

One interesting side effect of this course is that it drove a change to our standard computer lab policies. Specifically, those policies prohibit students from playing games on the lab computers. This was clearly an inappropriate policy for students in this course, since they would essentially be required to create their games without being allowed to test them!

Reactions to the course from our fellow faculty have been generally positive. We have carefully made the point that this is not a "content free" course in which students simply play games; that the course does in fact provide educational value to both computer science majors and non-majors.

True support for the course will be evident soon, as we have recently been discussing whether the course should be required as part of our computer science curriculum. Initial reactions to this idea have ranged from generally strong support for the idea to an argument that we shouldn't even recommend that our incoming students take this course over other freshman free electives. In addition to providing guidance for continual course improvement, we hope the data collected to evaluate our course goal achievement will also help us quantify and support the value of the course to computer science majors.

The course described in this paper represents our first step in a long-term, quantitative study of the effectiveness of integrating computer games into the computer science curriculum. We plan to continue this work both through continued evaluation of the course described here, through integration of computer game development activities into other early courses in the computer science major, and through evaluation of the new courses we have developed to support our new Game Design and Development Minor.

7. REFERENCES

[1] Ahearn, L. and Crooks, C.E. III. *Awesome Game Creation: No Programming Required*, 2nd ed, Charles River Media, 2002.

[2] Becker, K. Teaching with games: the Minesweeper and Asteroids experience. *Journal of Computing Sciences in Colleges, 17(2)*:23-33, 2001.

[3] Chamillard, A.T. and Braun, K. Evaluating programming ability in an introductory computer science course. In *Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education*, Austin, Texas, March 2000, pp. 212-215.

[4] Claypool, K. and Claypool, M. Teaching software engineering through game design. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education,* Monte de Caparica, Portugal, June 2005, pp. 123-127.

[5] Gorriz, C.M. and Medina, C. Engaging girls with computers through software games. *Communications of the ACM, 43(1)*:42-49, 2000.

[6] Huang, T. Strategy game programming projects. *Journal of Computing Sciences in Colleges, 16(4)*:205-213, 2001.

[7] Humphrey, W.S. *Introduction to the Personal Software Process*, Addison Wesley Longman, 1997.

[8] Jones, R.M. Design and implementation of computer games: A capstone course for undergraduate computer science education. In *Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education*, Austin, Texas, March 2000, pp. 260-264.

[9] Lorenzen, T. and Heilman, W. CS1 and CS2: write computer games in Java! *ACM SIGCSE Bulletin, 34(4)*:99-100, 2002.

[10] Natale, M.J. The effect of a male-oriented computer gaming culture on careers in the computing industry. *ACM SIGCAS Computers and Society, 32(2)*:24-31, 2002.

[11] Sweedyk, E. and Keller, Robert M. (2005). Fun and games: A new software engineering course. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 138-142), Monte de Caparica, Portugal.

[12] Walker, H.M. Do computer games have a role in the computing classroom? *ACM SIGCSE Bulletin, 35(4)*:18-20, 2003.