A Games First Approach to Teaching Introductory Programming

Scott Leutenegger
University of Denver
Computer Science Department
2360 South Gaylord Street, Denver Colorado, 80208
303.871.2812
leut@cs.du.edu

Jeffrey Edgington
University of Denver
Computer Science Department
2360 South Gaylord Street, Denver Colorado, 80208
303.871.3297
jedgingt@du.edu

ABSTRACT
In this paper we argue for using a "Game First" approach to teaching introductory programming. We believe that concerns over whether an OO approach or a procedural approach should be used first are secondary to the course assignment and example content. If examples are not compelling, student interest often lags thus making the OO versus procedural argument moot. We believe that game programming motivates most new programmers. Compelling assignments mean that students are far more likely to learn because they are interested, and the visual component allows students to *see* mistakes in their code as manifested in the resultant graphics. We describe our experiences after redesigning and offering a new introductory computer science sequence using 2D game development as a unifying theme. We teach fundamental programming concepts via two dimensional game development in Flash and ActionScript during the first quarter, transition to C++ to solidify concepts and add pointers during the second quarter, then teach a multi-phase project based game approach using C++ and openGL (2D graphics only) during the third quarter. Our surveys show that this approach improved student understanding of all seven basic topics examined.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer Science Education.* D.m [**Software**]: Miscellaneous – *games.*

General Terms
Human Factors, Languages, Theory.

Keywords
Computer Science Education, CS1, Introductory Programming, Game Programming, Game Development.

1.INTRODUCTION
How best to deliver fundamental programming concepts in a CS1 class has been an ongoing discussion since the beginning of computer science education. We believe that recent societal changes have caused both a need and an opportunity for a new approach. The need is caused by plummeting enrollments in computer science. Between 2000 and 2005 there has been a 60 - 70% reduction in incoming freshman computer science majors [18]. This drop makes retention especially important. The opportunity is found in the vast increase in computer/video game use among new students. The majority of people 30 years old and younger either play games occasionally or frequently. Further, contrary to common misperceptions, women make up 45% of all game players [6]. This huge interest in games can be used to entice students toward computer science: introductory programming classes using game creation is one compelling example. In fact, some schools are creating entire programs around game development [2,7,9,12,19]. These programs typically have a strong computer science component.

Our goal is thus to attract and retain majors without "watering down" the technical content of our classes. Thus, to us, the issue is not about objects first versus objects late or procedural versus objects, rather, it is about engaging students with interesting assignments. If the assignments truly interest students then it stands to reason there should be a much higher probability of learning success and retention. We use games for this purpose. The arguments in favor of this approach are similar to the arguments in favor of the media approach used in Python courses [8,14].

Note, many others have been using games as motivating examples. In [1,3,11,17] game programming projects are used as practical examples and motivators in CS1 and/or CS2. A recent SIGCSE panel [16] discussed how games are used to spark interest in general programming and software engineering.

The rise of the game industry and education of students for this industry has given rise to classes aiming to teach the game creation process itself. Perhaps the best source for game curriculum ideas is the International Game Developers Association Education Committee Framework [10]. Again, in a recent SIGCSE panel [16] some panel members discussed courses aimed at teaching the game creation process. In [5] the authors present a framework for integrating game programming into an existing computer science curriculum. In [13] a capstone project course that also helps train student for the industry is described.

As a secondary note, although we have stated that the OO versus procedural approach is secondary to the application content, it does remain an important issue impacting our curriculum. We believe students should learn how to use existing classes of objects before learning how to create new classes [15]. This is reflected in our curriculum as we describe in the following sections.

2.OUR SITUATION
Recently, the University of Denver created a Game Development undergraduate degree. In part to accommodate this new degree we have made changes to our existing computer science curriculum that we believe has also resulted in strengthening our approach for

SIGCSE'07, March 7-10, 2007, Covington, Kentucky, USA.

traditional computer science majors. Two major factors impacted our changes. First our curriculum must accommodate both Game Development majors and "normal" Computer Science majors and allow students to switch between these majors within the first two years. Second, we have a limited number of teaching faculty and cannot offer more than one introductory sequence. Thus, it is necessary to have a unified sequence of courses for all our freshman CS and Game Development students.

Because our institution uses the quarter system and we have a year-long introductory sequence before data structures and algorithms, our model is based on a three quarter freshmen introductory sequence. Our Game Development major is a demanding degree requiring later courses in Operating System, Graphics, and Game Programming, so the freshmen sequence needs to be "solid" and prepare students for the rigors expected in traditional computer science degrees.

3. COURSE SEQUENCE DESCRIPTION

As mentioned before we have not changed the fact that our introductory sequence is comprised of a 3-quarter sequence. What we have changed is the focus or "flavor" to games while retaining every concept offered before the focus shift. In the subsequent sections we describe the curriculum for each of these three classes.

3.1 First Quarter

We assume that students taking this class have had little or no programming experience. We continue to find this to be true for the majority of our incoming freshmen. We start by teaching the Flash development environment and ActionScript syntax. We quickly show how to create and move a graphical object around the screen by using the ActionScript onEnterFrame function. We feel it is important to start with a simple object on the screen as then programming logic and correctness can be viewed by watching the resultant animation. For example, consider the following snippet of code:

```
var xIncrement:Number = 6 ;
onEnterFrame = function() {
   theBall._x += xIncrement ;            // move the ball
   if (theBall._x > Stage.width)   xIncrement *= -1 ;
   if (theBall._x < 0)     xIncrement *= -1 ;
}
```

In this example we demonstrate if statements. Assuming "theBall" is an ActionScript MovieClip object that contains a picture of a ball, then the ball will move back and forth on the screen. Students can instantly see the effect of the if statement and if there is an error they will see that the animation does not perform as expected. This trivial example illustrates a major asset of our approach: instant visual feedback for the student. This same feedback helps students see if their code is correct for the other concepts.

The other concepts taught in the first quarter class include:

- variables
- If/else/switch
- Looping (while and for)
- Arrays (one and two dimensional)
- Using existing objects and classes (String, MovieClip, Date, etc.)
- Creating one's own classes
- Functions and Scope
- Event driven programming including mouse, keyboard, and onEnterFrame events.

Only the last item is non-standard, the rest provide a fairly typical coverage of introductory concepts. Our innovation lies not in the topics covered, but rather in the game focus. We use the pair-programming model and have also found this approach is beneficial. During the quarter, three different game projects are assigned which increase in difficulty and complexity. Students are shown how to create new object classes near the end of the course (if time allows).

3.2 Second Quarter

We switch programming languages at the beginning of the second course and teach C++ and the Unix operating system. The students gain more experience implementing new object classes. We teach pointers (which are not available in ActionScript) and dynamic memory allocation. We also introduce recursion, inheritance, and dynamic data structures (lists and trees). Programming projects are again assigned to teams of two students.

3.3 Third Quarter

This synthesis course continues with C++ programming. The students are introduced to simple graphics programming using the OpenGL API. Again the projects are game and simulation oriented but the teams are larger: three or four students. We introduce UML class diagrams and sequence diagrams as part of the project requirements. The goal of this class are to:

- Create larger projects that solidify skills learned in the first two quarters.
- Learn basic software software engineering concepts such as UML class, sequence, and object diagrams as well as unit testing.
- Learn how to program using an API. In this case the API is openGL.
- Learn how to work in groups.

4. LANGUAGE CHOICE JUSTIFICATION

Language choices for introductory classes are often determined by many factors including both faculty passion for a particular language as well as faculty inertia! In this section we justify our language choices.

4.1Why We Use Flash and ActionScript

There are many reasons we chose Flash/Actionscript. Perhaps the main reasons are that Flash is fun and it is relatively simple to start writing interesting games and applications in ActionScript. The initial learning overhead is quite low. Unlike Java which requires significant "scaffolding" before one can do something interesting, ActionScript programs can be written using very few easy to understand lines of code [4].

Second, using Flash/ActionScript provides immediate graphical feedback. The existence of an error, and often the probably causes, can be determined "visually" by observing the animation or game behavior.

Third, elementary ActionScript syntax is almost identical to C++. This provides a simple language transition from the first quarter to the second quarter.

Fourth, our university also offers degrees in Digital Media Studies and Electronic Media Arts Design. Teaching Flash and ActionScript has attracted a number of these students to our course. In the

upcoming year about one third of our introductory programming students will be drawn from these two other majors.

Fifth, Flash and ActionScript are used in the "real world". Students love the fact that they are learning a language that has immediate applications.

Finally, many simple two dimensional web games are written using Flash and ActionScript. It is important that all Game Development majors learn and understand two-dimensional game programming. This is covered in the Game Development courses (taken in the second or third year of study) but basic concepts should be introduced as soon as possible.

4.2ActionScript Is Not A Perfect Introductory Language

Unfortunately ActionScript has a few problems for use as an introductory language. It was not specifically designed for novice users or teaching, although it is fairly easy to use them for this purpose. Perhaps the biggest drawback is that ActionScript does not require declaration of variables. Because of this, typographical errors can create problems which are sometimes difficult to find. Students get down right angry when they find that a one letter error goes undetected yet makes the program fail to run correctly. Note, if Adobe were to add a compile flag option to do strong type checking this problem would disappear, making the language very attractive as a first language. Finally, despite the syntax similarities to C++, some students still had trouble making the transition from ActionScript to C++. For these students, several fundamental concepts had to be revisited. We are further investigating this last issue in the upcoming year's offering.

4.3Why C++ and OpenGL

We choose C++ for several reasons. First, this is the language that our department has settled on for our CS majors as the "standard" language of most classes. Second, C++ enables meaningful exploration of pointers and dynamic memory allocation. Third, the game industry is heavily dependent on C and C++, thus making the language choice relevant to our graduates. Subsequent classes in our Game Development major require C and C++ also.

We believe that teaching how to program using an API is important given software development's heavy use of APIs. The OpenGL API has the benefit of being relatively easy to use (for 2D applications), a clean design, and again provides the visual feedback we have found so helpful.

5.Women and Games

Given the drop in already low female freshman CS enrollments [18], it is especially imperative that we do not discourage the small number of women who do make it into our classrooms. Some critics of our approach have hypothesized that women are less interested in games than men and that this may drive women away from computer science Although anecdotal evidence affirms that women are less attracted to violent and online multiplayer games such as World of Warcraft, a recent survey by the Entertainment Software Association indicates 45% of all game players are women. Anecdotal evidence indicates women are often more interested in so-called "casual games". If that is true, our approach is then especially relevant as 2D Flash games lend themselves nicely to developing simple games within this medium supposedly preferred by women. Our experience so far has been that women are just as motivated by our game approach as men. See our results section below for numbers which justify this assertion.

6.RESULTS AND CONCLUSIONS

The students completed three informal surveys: one after the first quarter and two separate surveys after the third quarter.

Thirty students participated in the first survey. Of these 11 were female and 19 were male. The high percentage of women (as compared to the current national norm) is due to having a significant number of digital media studies majors in our course. On a scale of 1 to 4, where {1 = "boring"; 2 = "so-so"; 3 = "fun", and 4 = "awesome"}, most of the students reported that the first quarter course was fun giving an average of 2.82 / 4.0. Students were asked if they liked the game approach where {1 = "hated using games"; 2 = "Focus neither hurt nor helped"; 3 = "Game focus was good"; and 4 = "Game focus made it great". The average score for this question was 3.26 / 4.0 thus making it clear the students liked the approach. Since some people have conjectured that a game approach would discourage women we tabulated the same results for women only. The 11 women reported a "fun" average of 2.9 / 4.0 and a game approach good/bad average of 3.18 / 4.0. Not a single woman gave a rating of "1" to either question. Thus, it appears based on this sample that women were as favorably impacted by this approach as men.

Students were also asked to rate their level of understanding of various topics. The rating system was {1 = "no clue" ; 2 = "so-so"; 3 = "Think I understand" ; and 4 = "mastered"}. The first quarter survey results are shown in the first column of Table 1.

As can be seen from the table, after the first quarter, students in general appeared confident in all topics except creating their own classes. This is not surprising as it was taught near the end of the quarter. Both instructors felt that the exam results agree with the students self assessment from the surveys.

The second survey was given at the end of the third quarter. Nineteen students participated in this survey. The results are found in the second column of table 1. We asked the same concept questions as in the first quarter plus additional concept questions. The rating system was the same. The results clearly showed the students felt very confident on basic programming concepts and reasonable comfortable on the C++, inheritance, and pointers.

	Quarter 1	Quarter 3
Variables	3.50	3.84
Input/Output	3.21	3.79
Loops	2.94	3.74
Arrays	2.76	3.53
Functions	2.79	3.63
Using Objects	3.09	3.37
Creating Own Classes	2.21	3.45
Flash / Actionscript environment	2.79	NA
Inheritance	NA	2.95
Vectors	NA	2..79
Pointers	NA	3.00
C++	NA	2.45

Table 1: First Quarter and Third Quarter Survey Averages

The third survey attempted to determine what the students believed they had learned. They were asked whether their knowledge of Objects, Classes, Inheritance, Pointers, C++, and Separate Compilation had stayed the same, increased, or greatly increased during the year. In all categories (except Objects) the average responses were above "improved". The results are shown in Table 2.

	Average (max of 3.0)
Objects	1.79
Classes	2.05
Inheritance	2.05
Pointers	2.18
C++	2.05
Separate Compilation	2.16

Table 2: Average Increase of Understanding, where a value of 1.0 means "no change", 2.0 means "increased", and 3.0 means "greatly increased"

Overall our surveys indicate that the students did learn effectively and that the second and third quarter classes helped solidify earlier concepts.

Enrollment in our courses has increased for several reasons: the new Game Development degree draws new students to the university, the Media students are interested in learning Flash and ActionScript, and the courses have a new reputation of being fun and interesting. In Fall 2005 we started with 36 students in the introductory sequence. In Fall 2006 we have 60 students starting in the introductory sequence. Given the continuing declines in national CS enrollments and our doubling in enrollments, perhaps the game approach is something that should be seriously considered by additional schools.

Retention through the sequence has improved considerably over past years. Of the intended majors who started in the Fall, 85% took all three classes. In addition, a few non-majors ended up taking all three classes and at least one has switched to majoring in Game Development.

Finally, despite some claims that a game-oriented approach would cause women to be discouraged, our results show that women are equally encouraged by this approach as men.

In summary, it appears that our new approach has resulted in higher retention, increased attraction of new students, and that women seem to be positively influenced just as men. Furthermore, it appears that we have achieved these goals without sacrificing technical depth.

7.ACKNOWLEDGMENTS

We thank our teaching assistants who have been especially helpful in delivering this new curriculum during the past year.

8.REFERENCES

[1] Adams, J.C., Chance-it: an OO capstone project for cs-1, SIGCSE'98, 10-14, 1998.

[2] Argent, L., Depper, B., Fajardo, R., Ghertson, S., Leutenegger, S., Lopez, M. and Rutenbeck, J, Building a Game Development Program, IEEE Computer, Vol 39, no 2, 52-61, 2006.

[3] Becker, K., Teaching with games: the minesweeper and asteroids experience, J. Comput. Small Coll, 17(2), 23-33, 2001.

[4] Crawford, S., Boese, E. ActionScript: A Gentle Introduction to Programming. Journal of Computing Sciences in Colleges, Volume 21, Issue 3 (February 2006), pp. 156-168.

[5] Coleman, R., Krembs, M., Labouseur, A., and Weir, J., Game design & programming concentration within the computer science curriculum, SIGCSE'05, 545-550, 2005.

[6] ESA, "Essential Facts About the Computer and Video Game Industry. Entertainment Software Association, 2005. http://www.theesa.com/files/2005EssentialFacts.pdf

[7] Fullerton, T., Play-Centric Games Education, IEEE Computer, Vol 39, no 2, 36-42, 2006.

[8] Guzdial, M. A Media Computation Course for Non-Majors, ITiCSE'03, June 30 - July 2, 2003, Thessaloniki, Greece, pp. 104-108.

[9] Horswill, I., and Noval, M., Evolving the Artist-Technologist, IEEE Computer, Vol 39, no 2, 53-62, 2006.

[10] International Game Developers Association Education Committee, The curriculum framework, 2003

[11] Lorenzen, T., Heilman, W., Cs1 and cs2: write computer games in java! SIGCSE Bull., 34(4), 99-100, 2002.

[12] Murray, J., Bogost, I., Mataes, M., and Nitsche, M., Game Design Education: Integrating Computation and Culture , IEEE Computer, Vol 39, no 2, 3-51, 2006.

[13] Parberry, I., Roden, T., Kazemzadeh, M.B., Experience with an industry-driven capstone course on game programming, SIGCSE'05, 91-95, 2005.

[14] Ranum, D., Miller, B., Zelle, J., Guzdial, M. Successful Approaches to Teaching Introductory Computer Science Courses with Python, Special Session, SIGCSE'06, March 1-5, 2006, Houston, Texas, USA.

[15] Roumani, H. Practice What You Preach: Full Separation of Concerns in CS1/CS2. SIGCSE'06, March 1-5, 2006, Houston, Texas, USA.

[16] Sweedyk, E., deLaet, M., Slattery, M.C., Kuffner, J., Computer games and cs education: why and how. SIGCSE'05, 256-257, 2005

[17] Trono, J.A., Taxman revisited, SIGCSE Bull., 26(4), 56-58, 1994.

[18] Vegso, J, Drop in CS Bachelor's Degree Production. Computing Research News, Vol 18, No 2, March 2006, http://www.cra.org/CRN/articles/march06/vegso.html

[19] Zyda, M., Educating the Next Generation of Game Developers, IEEE Computer, Vol 39, no 2, 30-35, 2006.