CS1, Arcade Games and the Free Java Book

Daniel L. Schuster
MCIS Dept
Western State College
Gunnison, CO 81231
970-943-2999
dschuster@western.edu

ABSTRACT

Computer game programming has been adopted by some instructors and schools in an effort to motivate students and make the learning more relevant to the student's world than the console programs many of their instructors learned with. This paper describes the author's experience teaching CS1 using the ACM Java library to write arcade game programs. An online book, the Free Java Book, has been written to support this approach. Experience over the last two years teaching with this approach will be shared and the book will be described.

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
CSI, Games, ACM Java, Free Java Book

1. INTRODUCTION

About eight years ago the author began to struggle with what he felt was an increasing lack of relevance in the CS1 course. The work done in CS1 looked nothing like the computer applications the student grew up with. Many students appeared bored, and certainly the author was bored. Alternatives were explored but all were thought to be lacking in one way or another.

In August of 2006 the ACM Java Task Force released the first version of the ACM Java Library. This library includes a wealth of tools that bring more advanced Java subjects to the introductory level. In particular, the graphics library makes simple graphics and animation accessible at the very beginning of the introductory course. An updated version of the library was released in 2008.

Games are a well known motivator for students learning to program [1]. Studies have indicated that generally students prefer game assignments to non-games [2]. Game programming assignments allow the students to work in a visual, interactive domain that is perhaps more interesting and more real than the console based environment that some other approaches use. Schools have reported success with a gaming approach [1,2,3,4] and some have tried gaming to introduce more advanced topics [5].

The author began teaching with the ACM Java Library in the spring of 2007, and each semester has included more elements of animation and gaming in the CS1 course. Currently CS1 at Western State College of Colorado uses graphics and game programming for about 95% of the course, the remainder being console programming. Note also that at Western essentially all students enter CS1 with no programming experience and sometimes little idea of what programming is, so CS1 must begin at the very beginning.

A textbook has been written to support this approach. The Free Java Book (FJB) is available online at www.freejavabook.org and is available at no charge to educational institutions.

It is important to recognize that it is not the purpose of this CS1 course to teach game programming or the ACM Java library. They are merely a means to an end. The purpose of the course is to teach the usual CS1 topics—loops, decisions, modularity, objects, problem analysis, etc. Game programming provides an interesting application domain and the ACM Java Library provides a thin layer that makes this domain accessible to the beginner.

2. THE ACM JAVA LIBRARY

The ACM Java Library provides several fundamental classes that are the foundation of the arcade game approach.

The GraphicsProgram class and the GObject class provide an application window and a variety of visual shapes to work with. These include rectangles, ovals, lines and labels. These shapes can be collected into a compound object with its own constructors and methods.

The ConsoleProgram class provides simplified input and output for console programs, which are helpful for illustrating some very basic concepts.

A useful random generation capability is also provided by the ACM library.

Standard Java provides the necessary mouse and keyboard interactivity.

All programs run as Java applets, but this is invisible to the student.

The current website for the latest version of the ACM Java Library is www-cs-faculty.stanford.edu/~eroberts/jtf.

3. THE FREE JAVA BOOK

CS1 as taught at Western State College has transitioned from Pascal to C to C++ and most recently Java. For two and a half years the course has been taught increasingly in the arcade game domain.

As is sometimes the case, handouts became more extensive notes which became crude chapters which grew and eventually became the FJB. As of this writing, the FJB is incomplete. Eleven chapters are ready or almost ready for release, with the remainder to written as time allows.

Twenty years of teaching CS1 and five semesters of experience with this approach have led the author to the following topic ordering and general content:

1) Fundamentals – a quick introduction to programming and the basics of the ACM Java library. The student is able to write simple console programs and draw simple static graphic images.
2) ACM Java Graphics – a detailed look at some of the components of the ACM Java Library including rectangles, ovals, lines, labels, graphic images, and the application window. The student is able to characterize the application window, manipulate graphic objects and create simple animations.
3) Making Decisions – logical expressions, the boolean data type, and conditionals are covered in detail. The student is able to write animations with changeable behavior.
4) Loops – for and while loops and the game loop are covered. The student can write more complex animations and learns some loop related algorithms.
5) Methods and Objects I – covers void methods, passing primitive data types as arguments and simple programmer defined objects. Basic UML for objects is introduced. The student can decompose elementary problems into methods and objects.
6) Game Programming I – introduces standard Java applet components, elements of simple games such as object collision, some mouse and keyboard interactivity, and random integers. The student is able to write simple but real games such as Pong, keep score, provide instruction screens and levels of play.
7) Useful Java Objects – covers Java chars, the Character and String classes, the Math library and the ACM Random class. The student can write String based activities (Talk Like a Pirate translations for example) and add significantly to game character with more sophisticated mathematically controlled behaviors and additional random characteristics.
8) Arrays and ArrayLists I – cover the fundamentals of these objects so that the student can conveniently work with large numbers of game elements.
9) Methods II – more about methods, arguments and returned values. The student learns more techniques for dealing with complexity.
10) Objects II – more complete coverage of objects including accessors and mutators, the toString method, etc. UML is revisited. The student can create more fully featured objects.
11) Game Programming II – more sophisticated mouse and keyboard interaction, time dependant object behavior, more complex gaming. The student can write more sophisticated games with more complex interaction and object behaviors.
12) Arrays and ArrayLists II – more about arrays and ArrayLists, searching and sorting algorithms.

Additional coverage on Text Files, Multi-dimension Arrays and ArrayLists may be required for courses at some schools and is planned.

Quick Reference and Setting Up Your Computer appendices have been written. Additional appendices on compilation errors and the standard Java language are planned. A support forum exists.

The Free Java Book has some characteristics that are beneficial to the beginning student.

- language focus – there is no attempt to cover a large subset of the Java language with a massive book. Instead focus is only on those elements that form a foundation for arcade game programming and further study as a CS student.
- application domain focus – the application domain is almost completely restricted to arcade game programming. The student works constantly on animation and games, while remaining in Java.

The result is that the student is surrounded and immersed in what they need, but distractions and possibly extraneous material are removed from their environment.

4. SAMPLE ASSIGNMENTS

The arcade game application domain provides a wealth of assignments at every point in the curriculum and every level of difficulty. A few representative problems are described.

Robot Assignment – the student produces a robot.

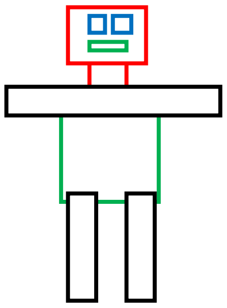

Figure 1. Robot

The assignment teaches sequence, familiarity with the coordinate system and working with objects from the ACM Library.

Drop And Roll assignment – the student produces an animation of a ball falling off and rolling away from a block, snapshots of which are shown below.

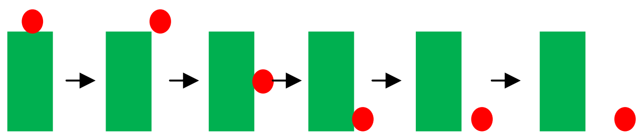

Figure 2. Rolling Ball

The assignment teaches boolean expressions, loops and animation basics.

Crash assignment – the student flies a UFO across the application window, with a flickering exhaust or flashing lights. When the UFO hits a space mine it blows up into several pieces but the alien parachutes to the ground. The program concludes with the message *The End* which scrolls right to left, starting at the right side.

In this assignment the student works with multiple objects, multiple loops, detects collision by comparing coordinates and produces differentiated movement of objects. Some students have produced elaborate versions with more action.

This assignment was inspired by a student who took a simpler version of it and added the parachuting alien.

You Choose 1 assignment – the student creates an interactive game of their choice, with the instructor's approval.

Allowing the student to select their own task increases their engagement. The instructor's approval is required to insure that they are taking on a task they can handle at this point in the course, and that the assignment requires enough of the student.

Students have produced good, straightforward versions of Pong, Helicopter in a Cave, Dodgeball, various shooting/clicking games, a juggling game, BreakOut, Lunar Lander and Frogger as seen in Figure 3.

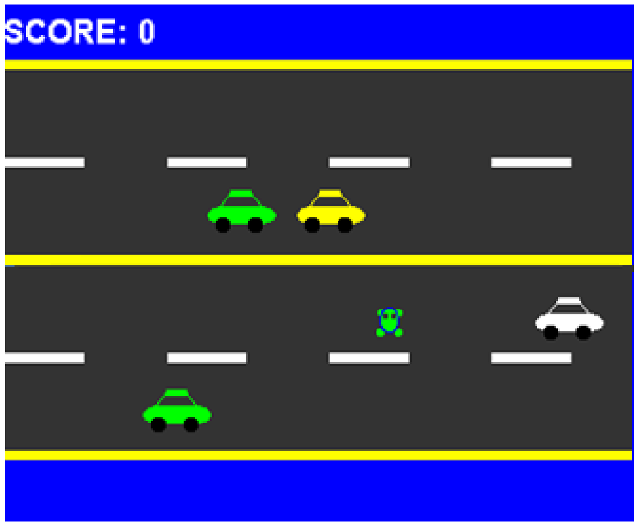

Figure 3. Frogger by Curtis Prock

The assignment requires programmer created objects and methods, mouse or keyboard control, scoring and levels.

BugBot assignment – the student creates a bug robot that wanders a playing field, bumps into objects and searches for a goal. The assignment does not require interactivity but makes significant use of methods and objects.

Asteroid Escape assignment – the student creates a version of the classic Asteroids game. The game features user created objects for the asteroids, the spaceship, black holes, etc. The spaceship is a complex object with several mutators and accessors. The keyboard is used to control the spaceship. Instructions are displayed, current spaceship status (fuel, etc.) are displayed, scoring is kept and there are multiple levels.

The student works with more sophisticated objects and significant additional complexity.

You Choose 2 assignment – the student creates an interactive game of their choice, with the instructor's approval. Suggested games include Missile Command, Helicopter Rescue or various puzzles. Students have also turned in excellent versions of Frogger, Asteroids and shoot-em-ups.

At Western State College, this assignment is the final project of the CS1 course. The student builds on their existing tools and skills and adds the capability to handle large amounts of data and implement more sophisticated behavior. It is expected that resultant game (or activity, because some students opt for puzzles, etc.) will be at least reasonably professional. And some of them are. Figure 4 is a screen capture from an elaborate version of Asteroids written by a student with little previous programming experience.

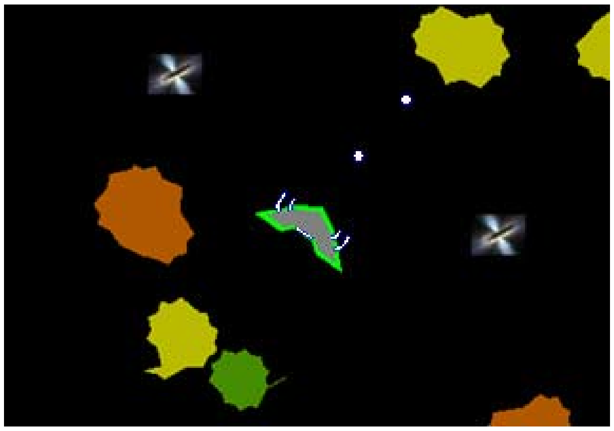

Figure 4. Asteroids by Peter Lewis

5. COMMENTS ABOUT THIS APPROACH

Most topics of the ACM library are taught as "black magic", with as little explanation as possible. The student learns about the ACM objects and methods as tools, not as interesting in themselves. For the sake of brevity, some obvious topics such as GPolygons are simply left out. The author's experience is that some more advanced students will ask for more powerful tools and a reference to online documentation on GPolygons has been sufficient.

Much of the course is taught Just In Time (JIT). Thus a topic is taught when it is needed and when the student is likely to be ready for it, and not before.

Consider the introduction to objects. At this point in the course students are writing programs of perhaps 200 lines of code, with a fair amount of this being repeated code. Void methods have been introduced to encapsulate tasks such as displaying instructions.

Students are also writing programs that move a multi-part graphic object, perhaps 3-component UFO as seen in Figure 5.

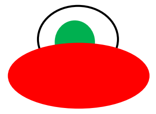

Figure 5. UFO

To move this UFO using what the student already knows at this point requires moving each of the three components. The student realizes, as one of mine said, "there must be a better way." And there is—objects. Gathering these three components into one object makes the student's work simpler. The student learns the default and non-default constructors. Objects are embraced by the students because they are easy to grasp and they are necessary. Later in the course, when the student's increasing sophistication demands more powerful objects, more thorough coverage is appropriate.

In this approach game playing is also introduced JIT. The student learns basic mouse and keyboard interactivity, object collision, randomizing the action, how to keep time and score and play levels early in the course. This allows them to write good, basic, playable games just a few weeks into the course. More sophisticated interactivity and time dependant object behavior should be delayed until the student is ready for it.

The majority of students will turn in simple but playable and fun games. These students will have accomplished the goals of a CS1 course, but not created works of the gaming art. They might produce little better than a good version of Space Invaders or Break Out. On the other hand a few will produce stunningly good programs. Students in this course have written wonderful, colorful, complex versions of Missile Command and Asteroids and other games created with this course as their only programming experience. They've been challenged, had a lot of fun and learned what they needed to learn.

6. OTHER APPROACHES WITH GAMING

The ACM Library provides a thin layer over standard Java and might be viewed as the *minimal* toolkit needed to accomplish graphics game programming at the CS1 level, while still keeping the student immersed in Java. The ACM Library works with any programming editor capable of working with Java. Thus this approach involves as much standard Java and standard tools as possible.

Greenfoot is, to the author's understanding, a thicker layer above standard Java that allows more sophisticated simulation and game creation. The student writes Java code that runs in the Greenfoot framework. Development takes place in the Greenfoot application, with significant tools for abstraction and visualization [6].

Alice is another level (or several) up the abstraction ladder, with the student creating a program using a drag-and-drop interface, with visual elements corresponding to standard programming language statements. Developers work within the Alice application [7].

7. PROS AND CONS OF THIS APPROACH

In five semesters of increased reliance on the arcade game approach, including three that use essentially only gaming, the author has perceived several important advantages of an arcade gaming approach:

- **increased motivation and interest** – it is very clear that students are far more interested. Students ask more questions in class, come in for more help, take on more extra challenges. A problem that could be solved in 200 lines of code is likely to have an extra 300 lines because of extra features that students has added.
- **real java programming** – the ACM Java Library is a thin layer, so the student is writing real Java programs from the beginning.
- **various student levels are easy to accommodate** – the nature of games allow the more advanced student to add extra features. Some choose to add extra characteristics for more visually interesting game elements. Others add more elaborate play such as a BugBot with more intelligence or an Asteroids game where the spaceship has shields. Not all of these innovations are successful however.
- **minimal knowledge of specialized techniques** – time dependant object behavior is the only game specific technique used in this approach. The student is always learning real and useful Java.
- **natural progression to objects** – there's no question of objects early or objects later. They are introduced when needed and further developed when required.
- **enhanced opportunities for artistic expression and creativity** – many students will create only very basic visual objects, but some will go to considerable trouble to produce attractive, accurate or complicated game elements and backgrounds.
- **flexible problem domain** – the approach builds non game activities such as slide show software or puzzles easily.
- **infinite number of varying assignments** – almost any arcade game or puzzle is a candidate for an assignment. However care is required to be sure that the student has the appropriate skills at the time of assignment.
- **minimal set up and cost** – the student need only download the acm.jar file containing the library, configure the IDE to find it, and they are ready to go. The games run successfully on older computers and the ACM library is available at no cost. No special IDE is required—the author has used both TextPad and Dr. Java.
- **reliability of the ACM Java Library** – the author experienced very few problems, odd behaviors or unexplained crashes.

In the author's experience, there is one minor downside to the gaming approach. Some learning overhead is required—the student has to become moderately fluent with a subset of the ACM Java library, and some very basic gaming techniques. The trade-off is well worth it though.

There is also one considerable downside—the lack of support materials. The Free Java Book is the author's attempt to overcome this problem.

8. TRANSITIONING TO STANDARD JAVA

The transition to pure standard Java is painless, and is handled by one example program that covers the structure of a standard program and console input and output, with perhaps some assigned reading.

9. CONCLUSIONS

The ACM Java library successfully brings graphics into a Java based CS1 class. Creating, modifying and moving graphic objects is very simple. Working with programmer defined graphic objects is naturally more complex, but still easily accessible. Interactions between objects is available to the beginning student. The ACM library, combined with standard Java, provides all the tools needed for the beginning student to work in the arcade game application domain.

Using the ACM library as a starting point the beginning student is able to write basic interactive arcade games within a few weeks, and much more refined games as the course progresses. At the end of the course strong students are able to write excellent arcade games that fully explore the topics, skills and techniques that a CS1 course should cover.

The arcade game application domain is suitable for CS1 and provides a wealth of possible assignments. Students enjoy creating animations and games, and student enthusiasm is notable.

The Free Java Book provides support for instructors wishing to undertake this approach to the CS1 course.

10. REFERENCES

[1] deLact, M., Kuffner, J., Slattery, M., and Sweedyk, E. Panel Session:Computer Games and CS Education: Why and How. In *Proceedings of the Thirty-Sixth SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2005)*, February 23-27, St. Louis, Missouri, 2005, 256-257.

[2] Cliburn, D. The Effectiveness of Games as Assignments in an Introductory Programming Course. In *Proceedings of the Thirty-Sixth ASEE/IEEE Frontiers in Education Conference (FIE 2006)*, October 28-31, San Diego, California, 2006.

[3] Bayliss, J. Using Games in Introductory Courses: Tips from the Trenches. In *Proceedings of the Fortieth SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2009)*, March 4-7, Chattanooga, Tennessee, 337-341, 2009.

[4] Morrison, B. Engagement: Gaming throughout the Curriculum. In *Proceedings of the Fortieth SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2009)*, March 4-7, Chattanooga, Tennessee, 342-346, 2009.

[5] Kote, L., Anderson, S., Good, J., Pain, H. Learning by Game-Building. In *Conference Proceedings of the 12th SIGCSE Annual Conference on Innovation and Technology in Computer Science Education (ITCSE 2007)*, June 25-27, Dundee, Scotland, UK, 2007

[6] Henriksend, P., Kölling, M. Greenfoot Invent – Program – Share, http://www.greenfoot.org

[7] Carnegie Mellon University, Alice An Education Software that teaches students computer programming in a 3D environment, http://www.alice.org