Integrating a Science Perspective into an Introductory Computer Science Course

John J. Magee and Li Han
Clark University, jmagee, lhan@clarku.edu

Abstract – **Experiences of including a Science Perspective in our university-level introductory computer science course are presented. The Science Perspective has helped in recruitment of students who otherwise might not have considered taking a computer science course. The assignments offered as part of the Science Perspective have been an integral part of our approach to teaching computer science in a way that is interesting and engaging to students by inviting them to explore scientific applications in computing. Our approach uses the Alice 3D animation-based programming environment followed by an event-based graphical Java programming approach. The interactive and visual nature of these approaches has increased student engagement, which has led to better retention in follow-up computer science courses. This approach could also be applicable in teaching computer science in secondary education.**

Index Terms – Introductory Computer Science, Science Perspective, Alice and Java programming, recruitment and retention, STEM synergies.

INTRODUCTION

Recruitment and retention of computer science students is a well-documented problem. At Clark University, Computer Science enrollments had fallen to a level where it was difficult to justify the number of faculty necessary to sustain the major program. Through a re-design of our Introductory Computer Science course, we were able to address recruitment, and further, achieve higher enrollments in the more advanced courses.

Recruitment was addressed by adding a Science Perspective to the Introduction to Computer Science course. The Science Perspective enables the course to fulfill one of the liberal arts requirements in the sciences, as an alternative to traditional natural science courses such as Introduction to Biology or Chemistry. We also reserved one section of the course for incoming first-year students who would have the time left to enroll in more computer science courses or choose it as a major, rather than allow the seats to be filled by students fulfilling the requirement toward the end of their college career.

Filing seats with students only taking a course as a requirement could be a dangerous proposition: students might not care about the material which could lead to poor performance and low retention in future courses. We needed an approach that would teach computing concepts and techniques in a way that was accessible and interesting to students with diverse backgrounds and interests, and thus encourage more students to continue with CS education. We also wanted to include topics that demonstrate the social impact of computing in an effort to attract and retain more women to the major.

Through these efforts, we have been able to recruit more students into the introductory computer science course, and we have enrolled many more students in the more advanced courses. The remainder of this paper is laid out as follows: First we discuss our pedagogical approach using Alice and Java; we then discuss the requirements of the Science Perspective and an Alice and a Java assignment that fulfills those requirements. We conclude by discussing the responses students have had to our approach.

ALICE + JAVA

Our pedagogical approach starts with Alice [1], a 3D animation computing environment, and continues with graphically-based Java programming environment focusing on objects and event programming [2].

These approaches have several positive features in common that have been shown to help introductory students overcome some of the common difficulties faced when learning programming concepts and techniques. First, they are both visual. The computer instructions that you write cause some visible change on the computer screen starting from Day 1. Second, they are object-first approaches: students learn to interact with objects by "sending messages" to them before they have to deal with the details of datatypes, values, and methods. In both cases, students can jump right in to creating impressive looking animations and interactive programs. This early engagement is key to attracting and retaining students' interest and engagement throughout the semester. We chose this approach after evaluating many options and coming across a similar course offering by Lisa Ballesteros at Mount Holyoke College [3].

The Alice environment offers many positive aspects as a learning platform for the start of an introductory computer science course. Programs are created by drag-and-drop, rather than typing. By making it impossible to make a typing mistake, students automatically create syntactically accurate programs. This alleviates many frustrating experiences that new programmers often face and frees them to focus on the conceptual topics and algorithmic thinking. For example,

students can learn about procedural abstraction by writing methods and using parameters easily since the environment forces correct usage of method calls. The Alice topics culminate with event-driven programming, allowing students to create interactive environments and games.

Our Java approach picks up where Alice leaves off: event programming. Students begin creating graphical, interactive programs from the first day with Java. We are able to quickly recap the concepts covered in Alice: methods, functions, abstractions, values, datatypes, and object-oriented programming. Since students have already become familiar with the concepts, they are simply implementing them in a new language. We observed that even though students now had to type their programs, they were more easily able to deal with syntactical errors since they were not also struggling with new concepts.

SCIENCE PERSPECTIVE

The Clark University Program of Liberal Studies includes six Perspectives that offer students an opportunity to experience learning and knowledge in a breadth of disciplines: Aesthetic, Global Comparative, Historical, Language and Culture, Natural Scientific, and Values.

The description of the Science Perspective offers the following: "Scientific Perspective courses teach the principal methods and results of the study of the natural world. Courses focus on the knowledge and theoretical bases of science. They also include laboratories or similar components to introduce you to the observation of natural phenomena and the nature of scientific study."

To meet the goals of the Science Perspective, we designed assignments that would engage the students in interesting scientific problems within the framework of our Alice and Java pedagogical approaches.

CHAOS SIMULATION ASSIGNMENT

I. Background

Toward the end of the Alice topics we give an assignment to investigate and visualize Chaos Theory through simulation.

We motivate student investigation of the problem by describing some of its practical applications. Students simulate a chaos model and observe the chaotic behavior in the animation. The assignment page contains general information on chaos theory using excerpts from Wikipedia and other internet sources.

We introduce the problem with the more familiar term "butterfly effect" – the idea that a butterfly flapping its wings creates a tiny change in the world's condition, but that the effect could be multiplied in unpredictable ways to change the weather on the whole planet weeks later. The butterfly effect actually got its name from computer simulation of weather systems, where a meteorologist noticed that a small change in parameters, such as truncating a measurement to four decimal places, created an entirely different forecast. This is one reason why we can only predict the weather about a week ahead of time.

More precisely Chaos theory is a field of Mathematics that studies the dynamics of systems that are highly sensitive to initial conditions. Chaos theory has applications in several disciplines including: physics, economics, biology, engineering, finance, meteorology, politics, demography, and philosophy. We provide students with references to more information about these topics.

II. Problem Specification

The chaos model for the simulation is given in (1). It is highly sensitive to the initial value of a.

$$a_{n+1} = 3.9a_n(1 - a_n), \quad 0 < a_n < 1 \tag{1}$$

We provide the a_n values of the first 9 iterations of this model for the initial values of $a_0 = 0.25$ and $a_0 = 0.26$, which shows significant differences after iteration 5:

a_n:	0.2500	0.7312	0.7664	0.6981	0.8219
	0.5709	0.9554	0.1662	0.5404	
a_n:	0.2600	0.7504	0.7305	0.7677	0.6955
	0.8259	0.5607	0.9606	0.1474	

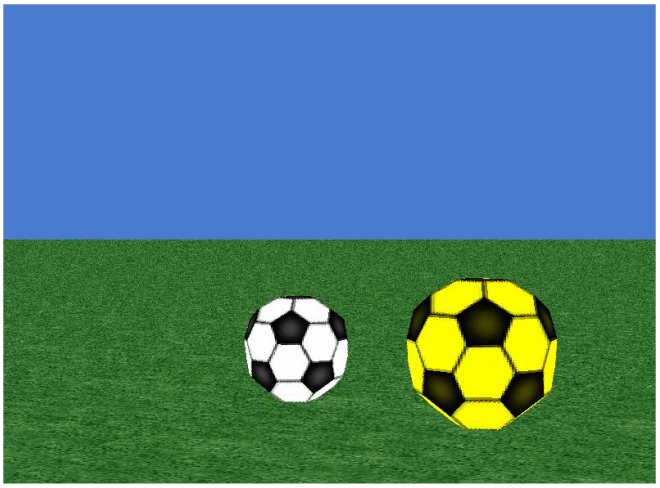

FIGURE 1
SCREENSHOT OF A CHAOS SIMULATION EXAMPLE SOLUTION USING THE SIZE OF 3D ANIMATED SOCCER BALLS TO DEMONSTRATE DIVERGENT BEHAVIOR.

III. Assignment

Students are directed to write an Alice program that produces a similar animation. Here are the requirements:

1. The animation should have two objects, each having a visible attribute, such as its size or its x (or y or z) position, whose value is determined by the a value in the chaos model. For each object, different a values should cause some visible changes to the object.
2. The animation should let the user specify initial values for the two objects and the number of iterations, and then simulate the chaos model for the given initial values and number of iterations.

3. Programming wise, develop one object and write a method for it called *chaosSimulation* that simulates the chaos model and changes the object accordingly, e.g. by changing its size or location. The *chaosSimulation* method should have two parameters: *initialValue* and *numberOfIterarations*. It should use a loop to simulate the chaos model *numberOfIterations* times.
4. In the *world.myFirstMethod*, first use some constant arguments, such as 0.25 for *initialValue* and 10 for *numberOfIterations*, to test and debug the chaos simulation method. After making the simulation method work, generalize *world.myFirstMethod* to make it ask for and get user input for initial size and number of iterations. Use while loops to keep asking for user inputs, with useful prompts to the user, until getting valid input values, before passing these values to the chaos simulation method.
5. After making the simulation and world methods work for the first object, create the second object as a copy of the first object and position both objects properly in the scene. Then add statements to *world.myFirstMethod* to get user input for the initial value of the second object and make two objects do chaos simulation together.

In completing this assignment, the students learn about iterative processes, repetition, variable updates, and mathematical programing. Most important of all, the choice of how to visualize the process is up to them. We observed that most students chose to visualize using object resizing rather than position, possibly because our demonstration video shows objects resizing. The built-in resize method of Alice objects treats its parameter as a percentage change of its current size. Students need to design and implement an appropriate procedure to determine viable parameters to use in the resizing or repositioning of objects.

EMERGENCE SIMULATION ASSIGNMENT

I. Background

The semester-culminating assignment in Java is designed to introduce students to scientific investigation and reinforce to demonstrate their advanced programming skills.

The assignment begins by describing the goal of discovering the rules behind a given phenomenon:

- Observe and analyze relevant data.
- Develop hypothesis.
- Test hypothesis through more experiments, analysis, and simulation.
- Compare the results from the tests to the observed phenomenon.
- If the results are not consistent, go back, make more observations, revise hypothesis, do more tests.

Emergence is the phenomenon of global, unexpected patterns emerged out of local, simple instructions. Like the Chaos Simulation assignment, we describe motivating examples of where emergence appears in the real world. For example: the synchronization of hundreds or thousands of fireflies. First they flash randomly but after some time and influencing each other, they flash in sync. There is no leader control. Simple rules behind this: all fireflies have nearly the same frequency for their flashing, but their phase is shifted. If a firefly receives a flash of a neighbor firefly, it flashes slightly earlier.

Other examples of Emergence appear in nature: In living, biological systems such as flocks of birds or schools of fish or in non-living physical systems such as snowflake formation or the stones of Northern Ireland's Giant's Causeway. Emergence also appears in the social sciences: economics, culture, and political philosophy and in the organization and structure of the internet and cities. We provide links to internet sources for the students to explore the topic in more detail on their own.

II. Problem Specification

The assignment provides students with a picture of a 2D array of red and green dots as the initial condition. We also provide subsequent pictures of the array at several iterations (after 7 iterations of the example array, the whole array is green and remains green).

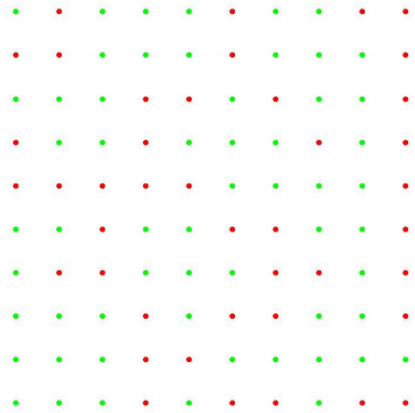

FIGURE 2
2D ARRAY OF DOTS, INITIALLY WITH A RANDOM DISTRIBUTION OF RED AND GREEN COLORS.

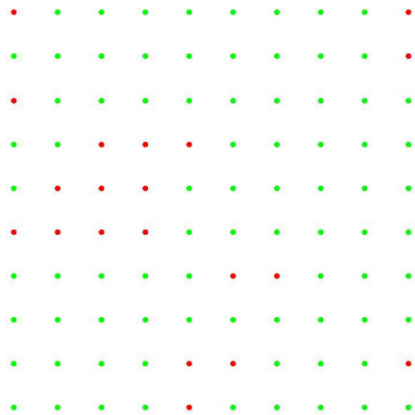

FIGURE 3
THE DOT ARRAY AFTER ONE ITERATION STEP.

At each step, each dot checks its neighbors' colors and determines what will be its own color at the next step. The challenge is that we do not tell the students what the color update rules are: How should they define the dot's "neighbors" and how does the dot decide its color based on the neighbors? The students are directed to use the scientific investigation process to try to discover the rules.

III. Assignment

Students are asked to create a simulation that implements the rules that they observe in the example dot patterns. We provide some guidance and starter code that helps them lay out grids of dots of various sizes. The simulation should run a step every time the mouse is clicked.

Many programming concepts taught in the course are needed to complete this assignment: multi-dimensional arrays, object references, nested loops, conditionals and selective execution, and accumulator patterns. Of particular practical interest is dealing with the "edge cases" and "corner cases", which in this problem are visibly present at the edges and corners of the array.

More technically, what the students are creating is an example of a cellular automaton, a concept they will see in much more depth in advanced courses if they continue in computer science. By postponing discussion of this concept to those more advanced courses, we are able to foster a constructivist approach to solving the problem.

STUDENT RESPONSE

We have conducted informal surveys of our students who have taken the redesigned course. Most students felt that the Alice + Java approach was useful for them to learn concepts without having to worry about the possibility of making typing mistakes or other syntax errors. Some students felt that the drag-and-drop nature of Alice made it too time consuming to construct complex statements, and others felt that software stability issues were an annoyance. A large majority of students responded that we should keep the same approach in future offerings of the course.

Due to increased retention, we offered two sections of the second-semester course in the program, Data Structures, for the first time in years. Our Algorithms course, typically taken as a 3rd course in the program, was also full this year for the first time in many years.

CONCLUSIONS

We have presented our approach to including a Science Perspective as a primary objective of our introductory computer science course. Our approach uses the Alice 3D animation-based programming environment followed by an event-based, graphical Java programing approach. The interactive and visual nature of these approaches has increased student engagement, which has led to better retention in follow-up computer science courses. The Science Perspective has helped in recruitment of students who otherwise might not have considered taking a computer science course. The assignments offered as part of the Science Perspective have been an integral part of our approach to teaching computer science in a way that is interesting and engaging to students. This approach would be applicable in either secondary education or higher education.

ACKNOWLEDGMENT

The authors would like to thank their colleagues and student teaching assistants for aiding in the success of the redesigned introductory course. The course also owes its success to the textbooks and supplemental material provided by Alice in Action: Computing through Animation by Joel Adams and JAVA: An Eventful Approach by Kim B. Bruce, Andrea P. Danyluk, and Thomas P. Murtagh.

REFERENCES

[1] Cooper, Stephen; Dann, Wanda and Pausch, Randy. 2003. Teaching objects-first in introductory computer science. In Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03). ACM, New York, NY, USA, 191-195.

[2] Bruce, Kim B.; Danyluk, Andrea and Murtagh, Thomas. 2001. A library to support a graphics-based object-first approach to CS 1. In Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education (SIGCSE '01). ACM, New York, NY, USA, 6-10.

[3] CS101 Course Page - Lisa Ballesteros, Mount Holyoke College. 2009. https://www.mtholyoke.edu/courses/lballest/cs101_s01/ Accessed: September 1, 2011.

AUTHOR INFORMATION

John J. Magee, Visiting Assistant Professor, Department of Math and Computer Science, Clark University.

Li Han, Associate Professor, Department of Math and Computer Science, Clark University.