Computation for Science and Engineering

Tanja Magoč and Eric Freudenthal
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968
(tmagoc, efreudenthal} @utep.edu

François Modave
Department of Computer Science
Central Washington University
Ellensburg, WA 98926
modavef@cwu.edu

Abstract—**Computation for Science and Engineering is an introductory computer science course for students majoring in fields other than computer science. This course, which was previously taught as a traditional introductory course focusing on syntax of C, will be offered with its new curriculum in Spring 2010. The new curriculum defocuses syntax and involves students in analyzing dynamic systems, which are frequently not deeply understood by students attending science and engineering courses. This course aims at attracting more non-computer science majors into computing disciplines and providing students with a deeper understanding of dynamism of real-life situations.**

Keywords-dynamic systems, programming, computing fields

I. INTRODUCTION

Motivated by Mark Guzdial's observation that programming is more accessible when examined in the context of interesting and engaging problems [1], we have recently developed at the University of Texas at El Paso (UTEP) a programming course offered to entering college students titled "Introduction to Computational Systems" (ICS) that examines the mathematics underlying familiar physical phenomena such as ballistics and resonance [2].

A typical project in ICS will be a short (4-15 line) program simulating a simple dynamic system whose output is drawn as a raster image. While the plots generated by these programs are essentially equivalent to the display of a graphing calculator, it appears that student definition of the system's evolution as a simple iterated sequence of instructions provides a heightened visceral-level understanding of the mathematics underlying a system's evolution. Furthermore, much like manipulatives used in math courses, the programs are sufficiently simple to be easily analyzed and repaired when a student discovers that their program evolves in an inappropriate manner. The preliminary evaluation of this course indicates that attendees quickly develop competency at basic programming tasks and have favorable attitudes towards learning about the mathematics of dynamic systems using computation in this manner [3].

Driven by the early success of ICS course, we are developing a new curriculum for the introductory computer science class for students not majoring in computer science. Computer Programming for Scientists and Engineers (CPSE) is a course at UTEP intended to teach basic computer programming skills to undergraduate students majoring in STEM disciplines other than computer science. The previous curriculum was a traditional introduction to programming that focused on the syntax and semantics of the C language and did not draw connections to attendees' majors. CPSE attracted too few students to be offered regularly.

We describe a new curriculum for CPSE, now titled "Computation for Science and Engineering" (CompSE), which will be introduced in Spring 2010. CompSE offers a more problem-oriented approach to learning programming concepts and uses the pedagogical approaches of our ICS curriculum to introduce programming as a tool for understanding the dynamism of systems of relevance to attendees' science-technology-engineering-math (STEM) academic majors.

One of the main goals of CompSE is to attract students not majoring in computer science to computational sciences courses and stimulate their interest in continuing study of computation. To achieve these goals, CompSE aims at attaining three main learning objectives.

1. Students will examine and understand the basis of analytical techniques that they have probably seen and memorized in mathematics or science courses but have frequently not been comprehended deeply enough to be able to apply them to real-life problems.
2. Students will learn how computation can be used to analyze problems that are difficult or impossible to understand using only analytical techniques, and
3. Students will realize that real-life problems expose common challenges to simulation such as catastrophic errors due to effects of round-off error or inappropriate simulation interval.

II. MOTIVATION FOR COMPUTATION FOR SCIENCE AND ENGINEERING

While programmed computation is a key tool to understanding the evolution of dynamic systems across the sciences, it is unusual for students in traditional science majors to attend a course that teaches programming skills. Furthermore, traditional introductory programming courses rarely expose the relevance of computation to these disciplines. In order to engage a larger number of students from traditional scientific disciplines in computational sciences, CompSE focuses on the development and examination of programs that simulate dynamic phenomena drawn from these disciplines. These simulations typically compute the sum of finite differences that are computed at each simulated time-step and also provide an alternate learning pathway for understanding underlying mathematical concepts.

Through this approach the course exposes the utility of simulation in examining the evolution of realistic systems whose complexity limits the accessibility of analytical evaluation. CompSE thus motivates the use of computation to the study of compelling realistic situations rather than hypothetical or simplified examples frequently found in coursework.

In order to stimulate students' interest to continue study in computational systems, CompSE aims at improving students' understanding of real-life dynamic systems by use of computer simulations. One of our key objectives is to provide a visceral level understanding of the characteristics of stable, periodic, and unstable systems that include feedback effects such as investment, production-consumer markets, predator-prey models, muscle contractions, impact of environmental conditions on living organisms, and the development of cancer.

Families of dynamic systems are examined as threads of inquiry drawn from various disciplines in which layers of complexity are progressively added to increase the realism of the computational model. We begin with a financial modeling thread, and continue with examining threads in other fields including environmental science and molecular biology.

Besides the mastery of basic programming concepts, anticipated learning outcomes of this course also include an understanding of the power and limitations of simulation, visceral understandings that simple systems can have highly dynamic behavior, and interest in continued study of computational systems.

Our planned evaluation for the success of this curriculum will examine both the effectiveness of the course in achieving student learning objectives, and student interest in continuing multi-disciplinary studies that include computation and programming.

III. Computation for Science and Engineering

The new curriculum will be offered for the first time in the Spring of 2010. By this curriculum, like ICS, rather than initially focusing on syntax, CompSE will instead immerse students in problem solving and incrementally introduce language features in an as-needed manner. Python is selected as the initial programming language due to its simplicity to convey major programming skills, and C is introduced later in the semester to demonstrate to students the ease of learning a new programming language once they are comfortable using Python.

The new curriculum still examines foundational programming concepts such as input and output, branching and loop statements, and objects. However, the presentation of these concepts is driven by the needs of the projects being examined rather than following a more traditional approach in which programming concepts to be taught are determined first, and the examples to demonstrate these notions are chosen based on the concepts that are to be explained.

Dynamic systems are examined in modules chosen from various disciplines including physics, finance, biology, and environmental science. Each dynamic system is originally presented under simplifying assumptions, which make it easier for students to understand the basic principles of the system at hand. After the initial comprehension of the underlying dynamic system, layers of complexity are progressively added to modeling of the system to increase the realism of the computational model.

Below, we describe several modules of CompSE. Each examines an evolving sequence of challenges that permit students to engage in discovery-learning as they first understand and implement simplistic models, discover their naïve assumptions, and then discover and apply successive refinements that reflect deepening understandings of the problem.

A. Investment Module

We begin with a financial modeling thread that first implements simple interest that might be offered by a bank. It is clear that, using this approach, the amount of money in a bank account is represented by a linear function. However, the students are asked to simulate the balance in consequent years, for a given initial investment and interest rate, without finding the exact equation of the line. Only after experimenting with different initial amounts and interest rates, students become familiar with expected behavior of the investment system in the case of a simple interest rate. Moreover, students are asked to relate the behavior of the system to known mathematical functions, which is expected to deepen their understanding of simple mathematical functions, such as linear functions.

The next layer of complexity is added by considering compound interest, which yields an exponential function. Students are led through similar steps as in the case of a simple interest in order to understand a more realistic investment system and, at the same time, understand the behavior of exponential functions.

Subsequent labs examine the selection of alternative investment instruments such as stocks and real estate. We first consider two criteria for investing in stocks – return and risk, for which we can define a stable (possibly linear) relationship: if the return of a stock increases, the risk of investing in this stock increases as well [4]. However, in reality, when the risk reaches a certain high point, no one will invest money in the stock even if it offers high return, and therefore the established (linear) relationship between return and risk becomes invalid. The course explores how this new phenomenon could be modeled. Naïve models are demonstrated to be inadequate when the resulting system reaches an absurd state. Various approaches to refining the models are examined such as piece-wise corrections, higher-order approximations, or alternate models.

In the first case, boundaries of the regular behavior are established (i.e., until which point the relationship between risk and return is linear), and the branching condition is placed at this point. The already built model is used until the boundary condition is reached, and a new model is built for the remaining part of the problem.

In the second method, the existing model is thrown away and a new model is built to address the new situation. Instead of using a linear relationship between return and risk, possibility of using mathematical functions that tend towards some horizontal asymptote are examined, such as exponential and logarithmic functions.

Furthermore, the course explores models of how a financial system can continue to evolve including the risks of oscillatory behavior, market crash, and various approaches that can lead to stable recovery.

The effect of simulation granularity and utility of mathematics in modeling these systems is also explored. Market crashes could result in losing money, thus the amount of money in investment instruments decreases rapidly in this situation. If appropriate/sufficient/enough attention is not taken during the simulation, this amount might become negative, which is not an adequate representation of the real-life situation. In order to repair the simulation, students need to determine when exactly the value becomes zero. However, it might not always be possible to find the exact value due to granularity of the simulation. Thus, students are forced to accept an approximate value as the solution, therefore facing the round-off error in this particular problem.

The finance module of CompSE is accessible to students with a variety of academic backgrounds. The course assumes that students have only a superficial understanding of investment and familiarity with high-school algebra. As in ICS, each type of dynamism and its effects are incrementally taught and examined through incorporation into students' simulated computational systems.

B. Environmental Science Module

CompSE continues by examining dynamic systems from other disciplines including environmental science and the impact of environment on living organisms.

Through the simulation of a simple agricultural ecosystem, the lab sequence examines the nature of random processes, their effect on dependent stateful dynamic systems, and the Monte-Carlo method. At each time step, the evolved health (state) of the plant population is modeled as a function of current health and current climate conditions. For example, a simplistic model of a healthy plant will transition through several states of diminished health when subjected to drought, and its recovery to a healthy state will similarly require a prolonged period of adequate irrigation.

Lab exercises will examine the nature of randomness in climate models. For example, in an early lab, the simulation of irrigation water availability is naively simulated as an independent random daily event with fixed probability. Students can run multiple Monte-Carlo experiments and determine the relationship between expected availability of irrigation water and probability that the crop will fail.

Subsequent exercises challenge students to develop more realistic climate models and to examine their effect on plant resilience. In addition, projects can model or simulate more complex models of ecosystems such as ones that include reproduction or competition.

C. Molecular Biology Module

In contrast to the familiar finance and environmental science examples described above, it is unusual for students not studying biology to be familiar with the processes of molecular biology. This section offers most students the opportunity to understand unfamiliar biological processes and approaches to simulating uncertainties and constraints.

One of the problems that we model in CompSE is development of cancer. For example, four types of cells' mutations are understood to contribute to development of cancer [5]. These four types of mutations are explained to students with enough details to offer understanding of the problem and the opportunity to develop at least a starting point in building a meaningful model for prediction of cancer given a current state of a patient.

Since the prediction of cancer is an ongoing long term project in medicine, the students are not expected to build the entire model, but rather brainstorm about possible approaches, and consider limitations of particular techniques through simulation of the system on a smaller scale.

Since many cell mutations remain unexplained nowadays, students are faced with a somewhat random generation of mutations. Based on the existing literature on cell mutations, it is not known when a mutation will occur. However, the rate at which each mutation occurs is well-studied and documented. Thus, even though the mutations are random, the students need to simulate these mutations taking into consideration the estimated rates of each mutation.

While students' models of cancer growth will necessarily be simplistic, students will be exposed to key concepts and will be familiar with the dynamism of such systems and how they are studied.

IV. Comparison to Old Curriculum

In the past, CompSE (then titled CPSE) was taught as a traditional introductory computer science course, which focused on teaching students the syntax and semantics of C programming language. While learning basic concepts of programming is of great importance, it also could be boring for students who do not plan to major in computer science.

Because of the focus on syntax and semantics, traditional programming courses are organized around programming concepts such as input and output, branching and loop statements, and often objects. Lab assignments for these courses are selected in accordance to the topic that was just learned, and are often hypothetical or very simplified real-life problems. Students are often faced with using computer programming to solve problems that could easily be solved with pen and paper, a calculator, or a simple spread sheet, thus not getting the right feel for the importance of programming and computer science in general.

Moreover, programming is often taught without mentioning limitations of certain programming language or limitations of programming in general. Students are only introduced to viable concepts and asked to solve problems whose solutions could be obtained in a straightforward manner.

In contrast to the previously taught curriculum of a traditional introductory programming course, which focuses on syntax of a language, the new curriculum for CompSE offers a more problem-oriented approach to learning programming concepts. As described in the previous section, the focus of the new curriculum of CompSE is on introducing students to behavior of dynamic systems. A system is studied in phases starting from its simplified versions to more complex and more realistic behaviors. Besides learning programming tools, students also learn different dynamic systems, some of which come from their fields of study. Students are introduced to a variety of dynamic behaviors, some more stable than others, and are expected to transfer this knowledge and experience to other non-programming courses that they take.

Unlike traditional methods for teaching programming, students in the CompSE course face limitations of programming and simulation techniques. They realize that not everything could be simulated exactly as it happens in reality, and learn how to set reasonable assumptions. In the second half of the semester, students are also forced to determine which of the learned techniques suits best a given problem.

Similarly to the old curriculum of the CPSE course, CompSE covers all basic programming concepts. However, these concepts are not taught to just show to students that these particular tools exist, which is often the case in traditionally taught curriculum. Computer programming concepts in CompSE are taught when needed to solve a particular problem, and thus clearly show to students the need, usage, and implementation of these concepts.

V. Expected Learning Outcomes

Anticipated learning outcomes of CompSE include mastery of basic programming concepts, deeper understanding of the dynamics of real-life systems, understanding of common simulation techniques, and interest in continued study of computational systems.

Since it is currently almost impossible to study, understand, and analyze dynamic systems without the aid of computers, one of the main goals of CompSE is to involve a larger number of students from various disciplines in computational sciences. By providing a more problem-oriented approach to teaching introductory programming, CompSE examines examples that are more related to attendees' field of study. Thus, the expectation is that the new curriculum will not only attract a larger number of students to study the first course in programming, but also increase the interest of these students to continue enrolling in computing classes.

Moreover, CompSE aims at improving students' understanding of real-life dynamic systems by the use of computer simulations. Since simulations allow students to quickly evaluate the impact of different parameters in a model, the behavior of dynamic systems is better understood by experiments. Furthermore, students are challenged to develop these simulations on their own, thus a thorough understanding of the dynamic system at hand is needed.

Even though the focus of CompSE is on modeling dynamic systems, basic computer programming skills are covered in this course. They are introduced through simulation of dynamic systems under reasonable simplifying assumptions. The complexity of computer programming is gradually increased through addition of layers of complexity of the dynamic system at hand.

Students are exposed to different modeling and simulation techniques. As an outcome of this course, students should have a good understanding of the advantages and disadvantages of different modeling techniques with respect to their accuracy, simplicity, and speed. Students are also expected to understand the benefits and drawbacks of small scale simulations of large dynamic systems as well as limitations faced during simulation phase.

VI. Evaluation

Our planned evaluation will examine both the effectiveness of the course in achieving student learning objectives, and student interest in continuing multi-disciplinary studies that include computation and programming.

The main student learning objectives include mastery of basic programming skills, understanding underlying concepts of dynamic systems, learning simulation techniques along with their limitations, and ability to use simulations to solve problems that are hard or even impossible to solve using only analytical techniques. The achievement of these objectives will be measured by applying in-class examinations as well as several lab assignments in which the students will be challenged to work through problems that have not seen yet and which are selected from different disciplines.

Students' interest in continued study of computation will be measured by pre- and post- class online surveys. These surveys will measure:

- Students' beliefs that computation is necessary in their field.
- Students' confidence that they have enough programming skills to write programs relevant to their fields.
- Students' confidence that they can learn (if not already possess) programming skills relevant to their fields.
- Students' confidence about understanding dynamic systems, and
- Whether students intend to take more computing courses.

The comparison of students' answers at the beginning and the end of the semester will determine the change of students' attitudes towards CompSE and other computation courses. These surveys and analysis of surveys will be prepared and applied in collaboration with an external evaluator.

Moreover, we will track down the students from CompSE classes to find out whether they have taken any additional computation classes after completion of our CompSE course.

VII. Synopsis

CompSE is an introductory computer science class for non-computer science majors. Unlike the previous curriculum it replaced, CompSE uses a problem-oriented discovery learning approach to programming and understanding dynamic systems. This approach is expected to attract students from various majors and stimulate their interest in continuing the study of computation. Moreover, the students are expected to learn basic concepts of programming, and understand the basic principles of dynamic systems and simulation techniques.

References

[1] M. Guzdial, "Design process for non-majors computing courses," *Proc. 36th ACM Technical Symposium on Computer Science Education (SIGCSE)*, ACM, 2005.

[2] E. Freudenthal, M. K. Roy, A. Ogre., T. Magoc, and A. Siegel, "Computational introduction to computer science," *Proc. Annual Symposium of the Special Interest Group on Computer Science Education (ACM SIGCSE)*, 2010.

[3] M. Suskavcevic, O. Kosheleva, A. Gates, and E., Freudenthal, Preliminary assessment of attitudes towards Mathematics for non-STEM section of Computational Computer Science Zero, UTEP CS Technical Report, May 2009.

[4] M. Joshi, *The Concepts and Practice of Mathematical Finance*. Cambrige, 2003.

[5] S. L. Spencer, M. J. Berryman, J. A. Gracia, and D. Abbott, "An ordinary differential equation model for the multistep transformation to cancer," *Journal of Theoretical Biology*, no. 231, pp. 515-524, 2004.