DEVELOPING COMPUTER SIMULATIONS USING OBJECT-ORIENTED PROGRAMMING. THE THREE-BODY PROBLEM: A CASE STUDY

Mike O'Leary[*]
Department of Mathematics
Towson University
Towson, MD 21252
moleary@towson.edu

Shiva Azadegan[*]
Department of Computer and Information Science
Towson University
Towson, MD 21252
azadegan@towson.edu

Since 1999 the authors have taught an interdisciplinary course in modeling, numerical methods, and computer programming. The primary audience for the course is junior and senior majors in mathematics and computer science. The prerequisites for the course are two semesters of calculus and one semester of computer programming, in C++.

The goal of the course is to teach students how to take a realistic problem, model it, choose an appropriate numerical method, implement that method using modern object-oriented computer programming and modern computer graphics, interpret the results, and write a report that summarizes their investigation.

The course is team taught by a mathematics professor and a computer science professor. The course is driven by a sequence of three realistic project problems. These problems have included the three-body problem, the motion of a baseball under air resistance, the double pendulum, heat flow, and traffic flow. We integrate the teaching of the mathematics and computer programming as closely as possible, and introduce topics as they are needed to solve problems.

We teach the students numerical methods appropriate for evaluating integrals (Trapezoidal rule, Simpson's rule), for solving single and systems of ordinary differential equations (Euler's method, Implicit Euler, Heun's method, Runge-Kutta methods, Runge-Kutta-Fehlberg methods) and finite difference methods for solving partial differential equations (including von Neumann analysis of stability).

We teach students modern, object-oriented programming using Microsoft Visual C++ and the Microsoft Foundation Classes. In particular, we show students the benefits of using classes in their programs, like organization, encapsulation, data hiding, inheritance, and reusability. We teach students event-driven programming, and the use of modern computer graphics

To illustrate how a course such as this can be taught, we shall describe in some detail how we taught the course in Spring 2001, where we began the course by studying the three-body problem. Because there is no textbook for this course, we created a sequence

[*] The authors' work had been supported by the National Science Foundation, through grant NSF-DUE-9952625.

of worksheets that were given out in class. These worksheets correspond roughly to one 75-minute class, with modifications made when extra time was needed in the computer laboratory. We shall simply summarize the worksheets; the originals are available at http://www.towson.edu/~moleary/Simulation.htm.

Worksheet 1:

- Mathematics
  - Trapezoidal Rule
  - Simpson's Rule
- Programming
  - Review of functions
  - Function prototypes
  - Passing functions
- Assignment
  - Write a console program to implement Trapezoidal Rule & Simpson's Rule
  - Implement and analyze Simpson's 3/8 Rule

Worksheet 2

- Programming
  - Classes, objects & instances
  - Private & public data & methods
  - Constructors & destructors; setters & getters
  - Overloading
  - Assignment
- Create a class for points, vectors, and line segments.
  - Use inheritance

Worksheet 3

- Programming
  - Classes, continued
  - Pointers to functions
- Assignment
  - Write a program with a class called Integral.
  - It should have a public method that take as input the number of subintervals and return the trapezoidal rule approximation of the integral.
  - It should have another method that return's the Simpson's rule approximation

Worksheet 4

- Programming
  - Introduction to dialog-based programming
  - Introduction to event-driven programming
- Assignment
  - Write a dialog-based version of the integral program

Worksheet 5

- Programming
  - Create a dialog based program with graphics.
  - Classes for balls, for control dialog and for graphics dialog.

Worksheet 6

- Modeling
  - Newton's law of gravity.
  - Model for the three body problem $\frac{d}{dt}\begin{pmatrix}\vec{r}_i\\ \vec{v}_i\end{pmatrix}=\begin{pmatrix}\vec{v}_i\\ \vec{a}_i\end{pmatrix}$ where

    $$\vec{a}_i = G\sum_{i\neq j} m_j \frac{\vec{r}_i-\vec{r}_j}{\left\|\vec{r}_i-\vec{r}_j\right\|^3} \text{ for } i=1,2,3.$$
- Mathematics
  - Euler's method
  - Implicit Euler
  - Heun's method (Improved Euler)
  - Applications to single equations
- Assignment
  - Write a dialog based program to solve a particular initial-value problem.

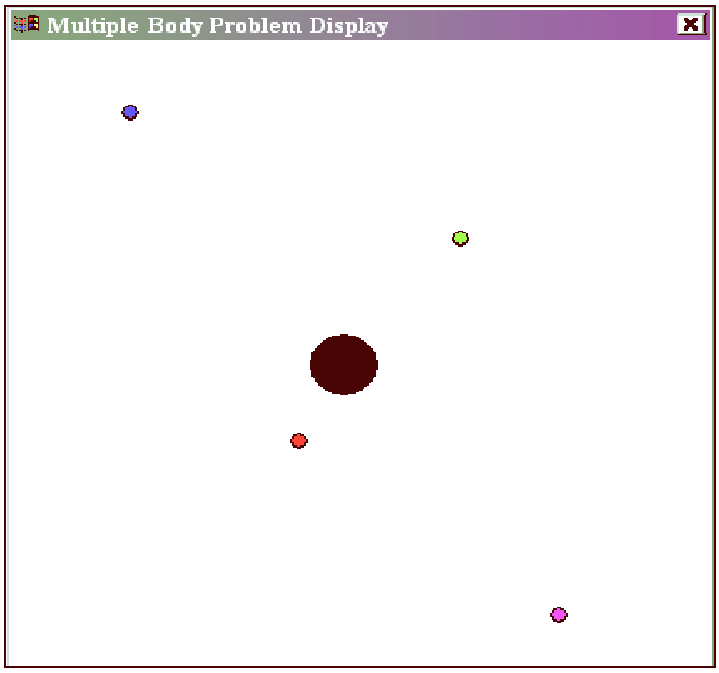
Worksheet 7

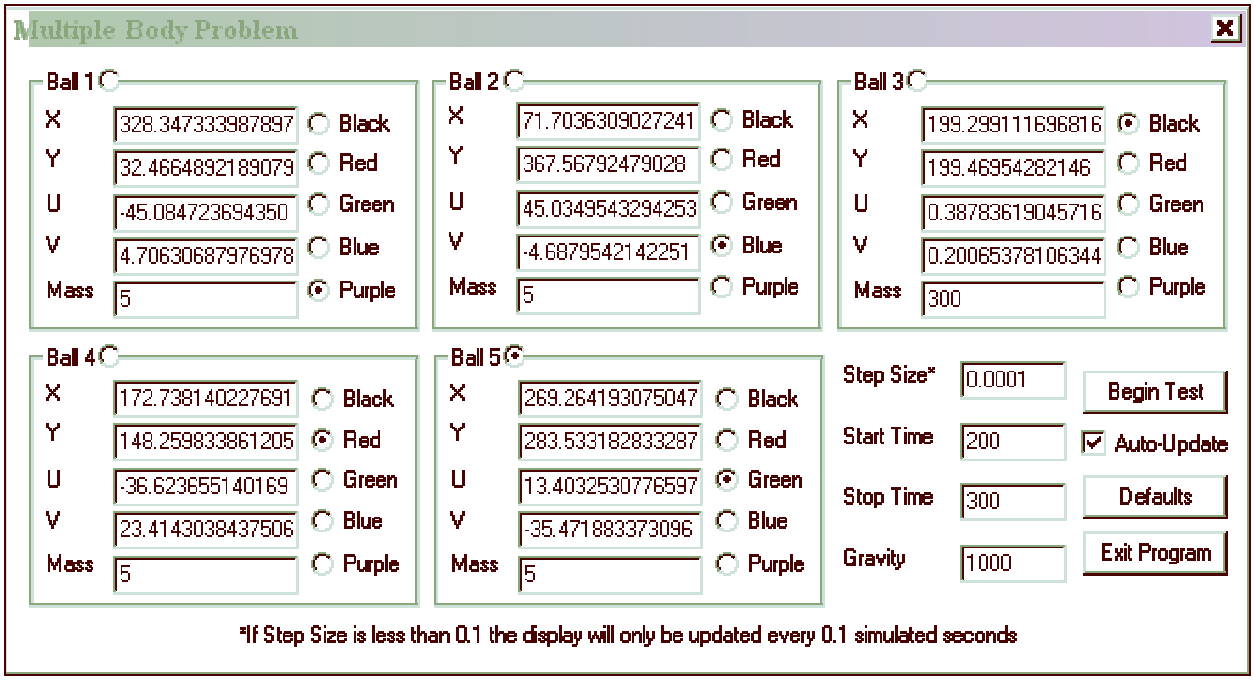
- Mathematics
  - Fourth order Runge-Kutta method
  - Applications to systems of differential equations
- Assignment
  - Modify the previous program to also implement the Runge-Kutta method
  - Write a program that solves a system using the Runge-Kutta method

At this point, we assigned students their first project, to write a C++ program that simulates the motion of three bodies under the influence of gravity. We asked the students to complete a report which describers the model, the numerical method that they used to solve the problem, and the structure of their program. We also asked them to describe the range of possible behaviors for the system In particular, we asked them to consider the questions:

- Is it possible for the system to eject one of the bodies to infinity?
- Is it possible for the system to return to its initial state?
- Is it possible for the system to return to a state close to, but not the same as its initial state?

A screenshot of the program that one of out students developed to solve this problem is on the next page.





Once this project was completed, we continued by analyzing the double pendulum and the problem of one-dimensional heat flow.

We have taught this course three times so far, and have learned a number of valuable lessons. When we first taught the course, we would have a common due date for the project program and the project report. However, our experience has shown this to be a

problem. In particular, students waited until the last possible moment to complete the program. Consequently, the quality of their reports suffered, as they did not have sufficient time to work on them. To combat this, we have a due date for the program code, and a separate, later, due date for the project. This split has aided in grading, as we have our computer science professor grade the code, and the mathematics professor grade the report.

Another significant issue was the difficulty of the open-ended project questions, like the one we gave for the tree-body problem. The original idea was to let students experiment with the simulations that they created and to draw some conclusions. In the event, students did not know what they should do, and thus did very little. We have found it more effective if we ask a number of very specific and focused questions which they answer in their report.

The general reaction from the students has been positive. In particular, they enjoyed the way the programming and the mathematics were integrated. They also enjoyed the fact that they were writing code with a purpose- rather than solving a homework exercise, they had a specific purpose for their program.

On the negative side, students were occasionally intimidated by the projects' difficulty. In particular, students had difficulty looking at one problem on so many different levels. In general, the mathematics students had more difficulty with the programming, and the computer science students had more difficulty with the mathematics. We used this an an opportunity to encourage teamwork.