Nifty Assignments

Nick Parlante (moderator)
Stanford University
nick.parlante@cs.stanford.edu

David Reed
Creighton University
davereed@creighton.edu

Dan Garcia
University of California at Berkeley
ddgarcia@cs.berkeley.edu

John K. Estell
Ohio Northern University
estell@onu.edu

David Levine
St. Bonaventure University
dlevine@cs.sbu.edu

Julie Zelenski
Stanford University
zelenski@cs.stanford.edu

Introduction

Creating assignments is a difficult and time consuming part of teaching Computer Science. Nifty Assignments is a forum, operating at a very practical level, to promote the sharing of assignment ideas and assignment materials.

Each presenter will introduce their assignment, give a quick demo, and describe its niche in the curriculum and its strengths and weaknesses. The presentations (and the descriptions below) merely introduce each assignment. For more detail, each assignment has its own web page with more detailed information and assignment materials such as handouts and data files to aid the adoption of the assignment. Information on participating in Nifty Assignments as well as all the assignment pages are available from our central page...

http://cse.stanford.edu/nifty/

David Reed - Web Programming (CS0)

The growing popularity of the Web and its intuitive graphical interface has opened a new world of programming opportunities for beginning students. Whereas programming used to mean mastering full-featured programming languages and complex programming environments, the introduction of JavaScript as a scripting language for Web pages has placed a free and simple programming environment on every desktop. With only a brief introduction to programming fundamentals, novice programmers can begin to write fun and interesting Web-based programs that utilize the familiar graphical interface of the Web.

In this presentation, I will demonstrate numerous programs that have been developed and used by students in Web-based CS0 courses offered at Dickinson College and Creighton University. These programs combine basic programming skills with the graphical Web interface to produce professional-looking programs that students are proud to display to the world. Some of these programs are provided to the students and used in assignments as tools for experimentation and problem-solving. For example, students use a Web page to generate random letter sequences and use that data to estimate the number of 3-letter words in the English language. Other assignments require students to design, code, and test their own programs. By integrating their code with the already familiar Web interface, even beginning students are able to produce interesting and attractive programs for tasks such as running a slot machine, testing for ESP, simulating random walks, and playing hangman.

Dan Garcia — Shall We Play A Game? (CS1)

This assignment is used in our introductory course in Scheme. Many of our students have never touched a computer in their lives other than to browse the internet. The assignment leverages my earlier "Gamesman" project which allows the programmer to put in the rules, representation, moves and terminating conditions for a small two-player game, like tic-tac-toe.

The system then "solves the game" by playing every game against itself (as in the movie "Wargames") and determines both players' perfect strategy.

For the students' final project they implemented one of about 10 small games we had collected. The students wrote code to represent the board, represent a move, and compute the resulting board of a move.

The assignment stresses good representation and abstraction, the importance of clear specifications, and that just a few lines of code can demonstrate "intelligence". Things which make this assignment nifty...

- The highest-order nifty bit is that the students love these small paper-and-pencil games to which they are introduced. This is recreational mathematics at its finest, and their enthusiasm carries over to the amount of time they spend adding feature upon feature to the finished product.
- The assignment is fairly open-ended in that it allows (but does not require) for the user to implement a graphical output interface.
- The user can easily parameterize some of the rules, which when tweaked, can result in games with vastly different strategies.
- When students are finished with their project they continue to play with it, and teach themselves good winning strategies.
- Students can share their projects with others not in the class.

The project depends on Gamesman currently available only in C and Scheme, and the students need about an hour of very basic combinatorial game theory.

John K. Estell Adventure Games (CS1-CS2)

You are in a twisty maze of nifty assignments, all different... Inspired by the classic Crowther and Woods interactive fiction game, "Adventure" has always been a favorite assignment. By entering simple English command phrases such as GO WEST or TAKE AXE, an adventure game allows the user to explore a simulated world and interact with the objects and characters present within.

The nifty thing about an adventure game assignment is that it exercises a broad range of programming constructs that tie many pertinent concepts together, but in a way that allows for creativity and latitude for personalization on the part of the programmer. It is this freedom to finally craft something of their own design, plus the challenge of a "large" assignment, that excites and motivates the students; they realize that one must have a good grasp of all of the concepts covered in class in order to accomplish this task, which makes for an appropriate capstone experience. This assignment has worked well on various levels.

There is great latitude in how an instructor can present this assignment. Normally, a minimum set of operations (such as movement and object manipulation) and rooms are specified. From this basis point one can present a skeletal program specifying both the data structures and function prototypes, and ask for the implementation of the functions that manipulate the structures; provide detailed map and room descriptions as a framework and ask that it be implemented; or just specify a theme and some sample interactions and leave the students to their own designs. See the Nifty web page (above) for instructional resources, including skeletal code, data files, scenarios, and fully implemented programs to make it easy to adopt this assignment.

David Levine — Sort Detective (CS2)

The sort detective is meant to test student's understanding of sorting algorithms and their behaviors on particular kinds of data sets. It can be used to compare and contrast any number of (comparison-based) sorting algorithms.

A common laboratory for studying sorts and asymptotic run-time behavior has the students run various sorting algorithms on particular sets of data and draw particular conclusions from the results. Such labs are sometimes praised as exemplifying the scientific method, but they are often subject to the criticism that they are very cookbookish.

The SortDetective turns this assignment on its head. Students are presented with a complete program containing various sorting algorithms, data input capabilities, and measurements that can be made. Unfortunately, all of the algorithms are "anonymous". Rather than being told what sorts to run, the students must design their own experiments so as to match the behaviors with the algorithms. As a result, it would be very odd for any two independent groups to do the same thing!

The students must make the same measurements as in the more traditional lab, but then they must actively associate those measurements with a theory rather than seeing "if they fit". They must exhibit more judgment about what constitutes "good enough". Finally, they must write up their conclusions in a relatively free format.

Nifty things about this assignment: Students must design their own experiments, requiring a higher degree of engagement than the traditional lab.

NO CODING! Computer science is not programming, but many of our lab exercises give this impression. This one does not.

It is easy to change the assignment from semester to semester by changing the mapping of the sorts and/or by reimplementing one of the algorithms to make it match the textbook's version. Source code for the project can serve as a small case study for design.

Julie Zelenski — Boggle (CS2)

(Thanks to Owen Astrachan for helping with this section.) Boggle stands out in our minds as one of our biggest success stories in CS2 assignments. Stanford and Duke have both been using Boggle in the intro courses since the early 1990s. With our 10 years of experience, we have found lots of neat ways to put the game to work in teaching our students nifty things.

A Boggle board contains 16 letter cubes randomly arranged in a 4x4 grid and the goal of the game is to form words by tracing paths through adjacent cubes without re-using letters. The game can be written to pit a single human player versus a computer or a multi-player version, with or without networking.

At Stanford, Julie introduced Boggle because of its lovely recursive properties. At the heart of the game are the searches: the backtracking exploration to find the human player's word on the board and the exhaustive traversal to find all remaining words for the computer player. Each has an elegant recursive solution and both raise interesting issues such how to properly mark/unmark the cubes to avoid duplication, how to efficiently prune dead end paths, and so on. The students love the game and find it especially satisfying to write a program that so soundly whips them in round after round of play.

At Duke, Owen was intrigued by the differing ways to construct a computer Boggle player. A trie-based dictionary can efficiently support a prefix-pruned search to find words on the board. But surprisingly, you can also solve the problem by trying every word in the dictionary to see if exists on the board rather than the other way around, exploiting the speed of the cpu to solve the problem in a opposite-of-how-humans-do-it way. There are interesting alternatives to consider in terms of the board data structure such as constructing an explicit graph linking the cubes versus using row,col adjacency information.

Overall, we have found Boggle to be a great source of nifty assignments that exercise recursion and data structures nicely while creating something fun and appealing for the students.