

Instructors looking to emphasize problem solving may remove some comments, give a brief physics lecture, and have the students figure out the equations for themselves. Instructors may also challenge advanced students to

Add animation (e.g., rotating land masses or explosive collisions), or
Add dynamic class loading.

The instructions and supporting materials can be downloaded from <http://www.cis.gvsu.edu/~kurmasz/NiftyAssignments/PlanetLab>.

SUDOKU*

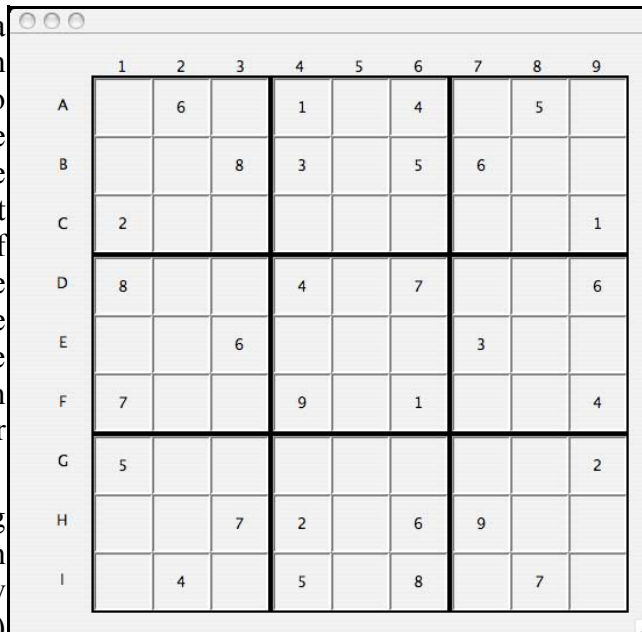
NIFTY TOOLS

Zachary Kurmas, Hans Dulimarta, and Yonglei Tao
Dept. of Computer Science
Grand Valley State University
kurmasz@gvsu.edu

This “Nifty Assignment” is a series of ten small to medium assignments that allow students to incrementally build a complete Sudoku application. The students are initially given a functional, but feature-poor, implementation. Each of the assignments is a feature for the students to add. We have limited the dependencies between the assignments so that an instructor can choose the subset that best fits his or her course.

One primary goal in designing these assignments was to develop an ordered set of tasks that would allow students in CS 1 to (almost) completely implement a fun program (as opposed to having them implement only a few methods or implement a “toy” program that they’ll never use again).

The motivation in providing a complete working application as a starting point was to help students understand the application’s entire design (i.e., “the big picture”) as early in the project as possible. In contrast, many multi-part projects are organized in a bottom-up



	1	2	3	4	5	6	7	8	9
A		6		1		4		5	
B			8	3		5	6		
C	2								1
D	8			4		7			6
E			6				3		
F	7			9		1			4
G	5								2
H			7	2		6	9		
I		4		5		8		7	

* Copyright is held by the author/owner.

fashion, which can make it difficult for students to understand how the code they are writing will be later incorporated into the larger project.

The usefulness of this project extends to CS 2. Instructors can use this project to
Illustrate the separation of model view, and controller
Provide practice writing GUI-based input, and
Examine refactoring.

The instructions and all supporting materials can be found at
<http://www.cis.gvsu.edu/~kurmasz/NiftyAssignments/Sudoku>.

RAPTOR*

NIFTY TOOLS

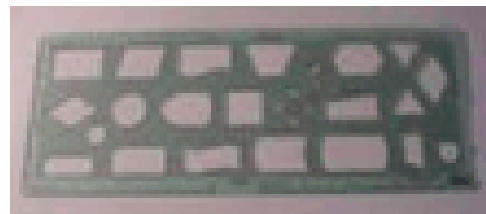
Mark S. Hall

Assistant Professor of Computer Science

University of Wisconsin-Marathon County (UWMC)

Mark.Hall@uwc.edu

I wonder what visual picture comes to mind for any 40+ year-old Computer Science professors who were “forced” to use flowcharts for their introductory Computer Science courses. For me, the visual image is not a pretty one. Visions of crumpled pieces of paper flying in the air and landing in the closest trash can. Even at that young age, I could understand that flowcharts could be a very useful design tool, but the tool, the legendary flowchart template, did not have an acceptable man-machine interface.



Any change to an algorithm meant erasing and modifying the flowchart symbols that were “squished” onto a sheet of paper. After several changes, the paper was no longer useable, and the complete algorithm had to be re-drawn from the beginning. But nevertheless, syllabi specified that the flowchart template was a mandatory purchase at the local campus bookstore.

Studies have shown that flowcharting can be very effective for visual learners to write and comprehend algorithms. But using the flowchart template was very primitive and very time-consuming. Is there a better way to use the same concept but make it electronic and interactive which would aid students to develop successful algorithms without increasing their frustration levels and their blood pressure?

* Copyright is held by the author/owner.