Nifty Assignments

Nick Parlante
Julie Zelenski
Stanford University
nick.parlante@cs.stanford.edu
zelenski@cs.stanford.edu

Eric S. Roberts
Jed Rembold
Willamette University
eroberts@cs.stanford.edu
jjrembold@willamette.edu

Ben Stephenson
Jonathan Hudson
University of Calgary
ben.stephenson@ucalgary.ca
jwhudson@ucalgary.ca

Stephanie Valentine
University of Nebraska-Lincoln
valentine@unl.edu

Juliette Woodrow
Kathleen Creel
Nick Bowman
Stanford University
jwoodrow@stanford.edu
kcreel@stanford.edu
nbowman@stanford.edu

Larry "Joshua" Crotts
Andrew Matzureff
University of North Carolina at Greensboro
Independent
ljcrotts@uncg.edu
andrewmatzureff@gmail.com

Mike Izbicki
Claremont McKenna College
mike@izbicki.me

ABSTRACT

The Nifty Assignments special session is about sharing the ideas and ready-to-use materials of successful assignments.

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 the assignment. A key part of Nifty Assignments is the mundane but vital role of distributing the materials – handouts, data files, starter code, rubrics, autograders – that make each assignment ready to adopt. Each assignment presented has complete materials freely available on the Nifty Assignments home page `nifty.stanford.edu`.

If you have an assignment that works well and would be of interest to the CSE community, please consider applying to present at Nifty Assignments.

ACM Reference format:

Nick Parlante, Julie Zelenski, Eric S. Roberts, Jed Rembold, Ben Stephenson, Jonathan Hudson, Stephanie Valentine, Juliette Woodrow, Kathleen Creel, Nick Bowman Larry "Joshua" Crotts, Andrew Matzureff, Mike Izbicki. 2022. Nifty Assignments. In the *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2022), March 3–5, 2022, Providence, Rhode Island, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3478432.3499268

SIGCSE 2022, March 3–5, 2022, Providence, RI, USA.

ACM ISBN 978-1-4503-9071-2/21/03.
https://doi.org/10.1145/3478432.3499268

Keywords:
Education; assignments; homeworks; examples; repository; library; nifty; pedagogy

Spelling Bee (CS1) — Eric Roberts, Jed Rembold

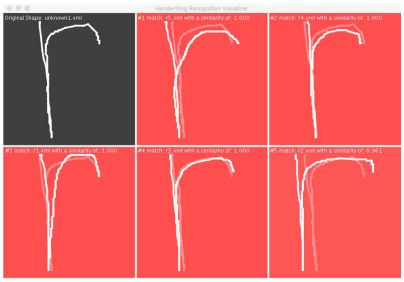
The SpellingBee assignment comes near the middle of a CS1 course and emphasizes string, file, and processing in an application that has been a significant revenue generator for the nation's leading newspaper. The problem is to find all the English words that can be formed from a beehive arrangement of seven letters, where each word must contain at least four letters and include the letter in the center hexagon. All SpellingBee puzzles include a ***pangram*** that includes all seven letters. For example, the puzzle shown at right includes the pangram **wizardry**, along with 36 other acceptable words. Students have found this project to be exciting without being overwhelming. They also quickly appreciate just how much better computers are at solving search problems than humans are.

Food Webs/Zombie Apocalypse (CS1) — Ben Stephenson, Jonathan Hudson

At first glance it might appear that food webs and the zombie apocalypse have little in common, but both subjects include relationships between entities that are naturally represented as directed graphs. We used this similarity to create two versions of essentially the same assignment whose primary difference was the backstory students were given about the problems they were asked to solve. Both versions asked students to represent a directed graph using Python's list and dictionary data structures and then analyze the directed graph to identify particular nodes within it. For example, in the food web version, students were asked to identify the animal that preys on the largest number of other organisms while the zombie apocalypse version asked students to (equivalently) identify the individual who has spread the zombie virus to the greatest number of people. While the code required to solve these problems is identical, the different "window dressing" used for each version of the problem obscured this similarity. This allowed us to extend the life of the ideas on which the assignment was built and reuse previously developed materials such as grading schemes in subsequent terms.

Handwriting Recognizer (CS1) — Stephanie Valentine

In this assignment, students explore template matching, an often-employed sketch recognition technique. Template-matching systems maintain a library of sketched examples called templates. In a system that recognizes letters, the system would include some sample As, some sample Bs, etc. When attempting to recognize an unknown shape, the system mathematically compares the unknown shape with every template in the library. The unknown shape is identified as the template to which it is the most similar.



This assignment focuses on searching, sorting, code reading, and computer graphics (drawing on a canvas) but touches many other topics: strings, arrays, loops, functions, objects, file I/O, and unit testing. The assignment specification provides a simplistic algorithm to implement, but students are encouraged to experiment with the algorithm to improve its accuracy. The handwriting recognition domain gives students agency to go beyond a single right answer, to iteratively tweak and improve their algorithms to achieve better results.

Bias Bars (CS1) — Juliette Woodrow, Kathleen Creel, Nick Bowman

In this assignment, students parse and store a complex dataset of college professor reviews. They use this structured data to build an interactive graphical tool that plots word frequency data across review quality and professor gender to reveal interesting trends about biases in language usage. In the process, students will practice working with dictionaries, string parsing, nested data structures, normalization algorithms, and graphical programming. Our hope is that students can use this exercise in data visualization and data analysis to think critically about the biases that exist in online datasets. The end product of the assignment is a complete tool that can be used to dig deep into the provided dataset while considering its social and ethical implications. It is part of the Embedded Ethics curriculum, which encourages computer science students to consider ethical issues from the outset integrating skills and habits of ethical analysis throughout the Computer Science curriculum.

Reddit Bot (CS1/2) — Mike Izbicki

In this assignment, students build simple chatbots that post messages to the popular social media website Reddit. The student's code uses a Reddit-official API where the posts are separate from the normal Reddit web content.

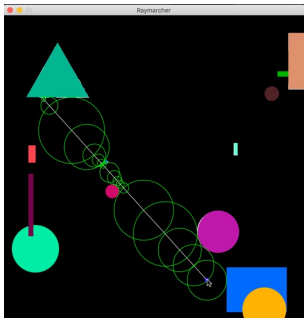
The assignment introduces several interesting topics including navigating tree structures, dealing with non-determinism, and basic devops. Reddit stores comments as a large tree, and students must navigate this tree to reply to the correct comments. The results of running a student's program will depend on other messages posted to Reddit, and so students must use good debugging techniques to ensure their programs are correct. They must run their programs continuously over a week period and respond to rare bugs that cause their program to crash only intermittently. Students also learn how to manage API secrets. (They receive a major penalty if I ever discover their login credentials!)

The real benefits of this assignment are non-technical. The assignment forces students to wrestle with important ethical and legal dilemmas about online bots and freedom of speech, and it gives them the technical knowledge necessary to meaningfully participate in these debates. Because of these non-technical applications, this assignment is particularly well suited for getting humanities students excited about computer science.

Ray Marching (CS1/2) — Joshua Crotts, Andrew Matzureff

In this assignment, students develop a small application that incorporates topics from computer graphics and simple computational geometry: ray marching. Ray marching is a rendering and collision detection technique that bears similar resemblance to the old-school ray casting rendering method used in games such as Wolfenstein 3-D by Id Software. Students create and implement shapes, rays, collision detection, as well as user interaction via keyboard or mouse with event listeners.

Students seem to generally crave practicality in computer science. There is always a presence of theory, but most enjoy understanding how their work is useful in the real world. Plus, graphics are naturally more appealing than console applications; they pull in students and allow those that joined computer



science for game development/design to discover the intuition behind classic games that innovated three-dimensional environments in a time when rendering "true" 3D graphics was painfully slow and infeasible for home computers.