Robert Swartz
www.mathapplets.net
Towers of Chicago

This program solves the Towers of Chicago puzzle. In this problem, a set of discs are to be moved from one peg to another while observing the following 4 rules: (1) Only one disc can be moved at a time, (2) A larger disc can't be placed on top of a smaller one, (3) Auxiliary pegs may be used for the temporary placement of the discs, and (4) The task should be completed in a minimum number of moves.

On a 4k screen, this applet can display up to 85 discs x 20 pegs (where the dimensions are maximized in Windows 10). It can also print the moves for up to 1000 discs x 100 pegs. The applet can handle up to 20 rotating colors for both the discs and the pegs; the initial and final pegs are displayed in green and red. The discs are displayed either round or sharp. When the applet is paused, the move button can be used to step through the moves.

The traditional version of the Towers of Chicago makes use of only 3 pegs: the initial peg, the final peg, and 1 auxiliary peg. However, the multipeg version makes use of any number of pegs: the initial peg, the final peg, and 2 or more auxiliary pegs. As the number of pegs is increased, while keeping the number of discs constant, the number of moves required usually decreases.

The optimal solution for the n disc by 3 peg problem is well known: recursively move n-1 discs from the initial peg to the auxiliary peg, then move the remaining disc to the final peg, then move the first n-1 discs from the auxiliary peg to the final peg. This algorithm results in 2^n-1 moves.

The presumed optimal Frame-Stewart algorithm that this program uses for n discs and p pegs is as follows: recursively move k discs from the initial peg to an available auxiliary peg using the p peg subalgorithm, then move the remaining n-k discs from the initial peg to the final peg using the p-1 peg subalgorithm, then move the first k discs from the auxiliary peg to the final peg using the p peg subalgorithm. Dynamic programming is used to find the k values. The k values are calculated to make the number of moves minimal.

This program can find different optimal solutions for each combination of discs and pegs (where pegs are greater than 3). This is accomplished through 2 kinds of randomization, Scramble k Values and Scramble Discs. Each of these uses a random number seed (an integer from 0 to 999). Scramble k Values uses a different k value for each recursive instance of a particular subalgorithm; if this isn't selected, the Random k Seed sets constant k values for each subalgorithm. Scramble Discs simply shuffles the discs around the auxiliary pegs.