Lab 6

11 Oct 2007

The content of this lab will be eerily familiar to many of you: the first portion of it is another "comprehend the code" reading exercise whose main content is a mild rehash of one of my CS 142 projects. From there, we'll move on to a more AI-ful version, which will be assigned as homework.

The setup: Path detector

The problem this piece of code is to solve is that of reading in a maze/grid/terrain file—with a start and finish point marked—and determine whether the finish is reachable from the start.

The navigable space

The "board" will be a rectangular grid of arbitrary size; at each cell of the grid, you can have one of four possible things: the start, the finish, an impassable wall, or a plain old open space.[1]

The format for files containing a grid requires the first line to consist of two integers, representing the width and height of the grid respectively. Each subsequent line represents one row of the grid, with each cell represented by a single char: a lowercase 'O' for the start space, an asterisk for the finish, a hash mark ('`#`') for a wall, or a period for an open space. For example:

```
6 5
......
...*..
..##..
......
..o...
```

You can assume that every correctly-input grid will have exactly one start and exactly one finish. Extra lines in the file are ignored (and thus are a convenient place to write comments about that particular grid).

[1] In the project, there will actually be one more cell type, the teleporter.

The display thereof

To display the actual grid (and eventually paths on the grid), there needs to be a GUI; an applet is just a quick-and-dirty solution here. Walls are displayed as solid black squares, open squares should be white, and the start and target squares are clearly marked. During the solving process, squares that have been explored turn grey.

The control widgets let you specify a grid file and load it in; a way to start the solver; and a way to either take a single step or to start animating the search process.

Task 1: Grok the code

In the course directory, in the subdirectory `lab6`, I've put a solution for the above problem description. This will be your starting point for future work, so copy it into your own directory. Note that the `-R` option to `cp` copies recursively (i.e. including subdirectories), which will be useful to you here.

The code is inside the `pathfinder` package. To run the applet, compile it and then type

```
appletviewer pathfinder.html
```

from within the `lab6` directory.

Read through this code and make sure you understand it fairly thoroughly.

Task 2: Paths

As written, the task and program above were written only to determine whether a path existed; they do not do any of the work of actually recovering that path, and in particular, of recovering the *shortest* such path.

Modify the code so that once a path is found, all the grid locations along that path are marked in the middle with a black dot. Don't forget that since this is an event-based GUI, you won't be able to simply draw a dot once, but rather, you'll have to store the path information in such a way that every time `paint()` gets called, the path will be drawn.