This is MazeDemo version 000. I wrote it (except for a tiny bit in the middle, but more about that later), my name is David S. Joerg, I can be contacted at joerg@alliant.mcs.anl.gov.
It is now yours; use the source code in any way you wish, have lots of fun with it. However, having toyed around with a lot, I know some particularly good ways to have fun with it:

I) Make the window small (80 by 80 is good) and set "Spacing" to 2. Set "Tick Period" to 1, and then "Solve Maze".

II) Set "Spacing" to 20, set "Tick Period" to 1, and make the window as big as the screen, then "Solve Maze". This may be a nice example for someone who's trying to get a mental grasp of recursive algorithms for the first time; it would be a much better instructional device with more stop-start-rewind control.

III) Set "Spacing" to 2, set "Tick Period" to 1000, and make the window as big as the screen. It will take a while for the maze to draw, but watching the maze get solved is fun -- the patterns look almost like lightning sometimes, and other times like a river delta or some coastline.

Obviously, there are zillions of ways to improve this program. First and foremost, the sliders really really should be SliderDualActing's, but I couldn't find the SliderDualActing Distribution at cc.purdue.edu or cs.orst.edu! Am I blind? If someone knows where SliderDualActing is hiding, please tell me because I would really like to use it.

The maze-drawing and solution-drawing code could probably be sped up a bit, although some substantial work has already gone into optimizing those.

The maze-solving code is deterministic; it will solve the same maze the same way every time. This is boring. It would be neat to modify the algorithm to randomly pick which order it will check the four directions in.

The graphics are bare-bones; some pre-prepared bitmaps judiciously applied could make everything look much nicer but still draw just as fast.

I'm not going to do anything about the above problems; I've got other fun things to do, and so little time. However, I am in the process of making MazeDemo v.001, which will do its work *in parallel*, and you will be able to choose from a variety of parallel maze-solving algorithms. You will be able to see each algorithm in **vibrant**

life, writhing and squirming on the screen in *parallel.* What fun! This will be done using the C-Threads package, which is wonderfully documented in **/NextLibrary/Documentation/NeXT/SysRefMan/16_Mach.wn**.
I will also be writing an article discussing C Threads and what it offers to Appkit programmers, which will be in the July issue of the NeXT Users' Journal.

MazeDemo uses a very strange algorithm to generate its mazes; I still don't understand it myself. I first saw this wonderful little gem of a C program a few months ago, and have been entranced by it for a long time. By slowly and painfully modifying variable names and replacing compact expressions with more verbose and explicit equivalents, I was able to distill and control the algorithm within. Now I can control the precise width and height of the maze, as well as the random factor. But don't ask me how it manages to generate mazes with no isolated points. That's right, folks; look at the mazes in MazeDemo, and you'll see that betweeen any two arbitrary points, there is a path between the two that crosses no lines.

By the way, if you're going to compile "mazemaze.c" below, you'll need to use the "-

fwritable-strings" flag.

Path: uunet!wyse!vsi1!ames!mailrus!tut.cis.ohio-state.edu!cwjcc!hal!ncoast!allbery
From: tromp@cwi.nl.UUCP (John Tromp)
Newsgroups: comp.sources.misc
Subject: v04i110: An amazingly small C-program
Message-ID: <7656@boring.cwi.nl>
Date: 30 Sep 88 23:32:22 GMT
Sender: allbery@ncoast.UUCP
Reply-To: tromp@cwi.nl.UUCP (John Tromp)
Organization: CWI, Amsterdam
Lines: 16
Approved: allbery@ncoast.UUCP

Posting-number: Volume 4, Issue 110
Submitted-by: "John Tromp" <tromp@cwi.nl.UUCP>
Archive-name: mazemaze

[This one should have been entered in the Obfusticated C contest. ++bsa]

Run this C-program and enter a number. Ain't that amazing ?
John Tromp (tromp@piring.cwi.nl)

```
char*M,A,Z,E=40,J[40],T[40];main(C){for(*J=A=scanf(M="%d",&C);
--                E;                J[                E]                =T
[E    ]=  E)    printf("._");   for(;(A-=Z=!Z)   ||   (printf("\n|"
)     ,    A     =                 39                ,C                --
)     ;    Z     ||     printf    (M    ))M[Z]=Z[A-(E    =A[J-Z])&&!C
&     A    ==                T[                                        A]
|6<<27<rand()||!C&!Z?J[T[E]=T[A]]=E,J[T[A]=A-Z]=A,"_.":" |"];}
```

Bon Appetit,
Dave