Subject: The Game of LIFE
Posted by chris on Sun, 09 Oct 1994 22:32:39 GMT
View Forum Message <> Reply to Message

WARNING: This post is of no scientific and little educational value.

A friend of mine at stanford recently sent me an IDL version of
the clasic game of LIFE. This is his simple version. He
is presently developing a very nice widgetized version.
He does *NOT* want to be bothered w/ questions about these
activities, please.

(If you are not familiar w/ the game the rules are simple:
A cell dies if it has too many or too few neighboors.
A cell is born (where there wasn't one) if it has exactly 3
neighboors I believe it is)

He requested that it be reproduced exactly so here it is:
(I prefer to put the statement
pro life   ;(No political pun intended)
at the begining to make it a module.)
-Chris

```
 ;------------------------------------------------------------ ----
;LIFE ---demonstrating the power of IDL .   By Ken Ricci, August 1993
;revised March 1994

;This section is just extra setup frills
M=105               ;set your grid size here
N=76
r=500          ;number of generations to run (500 = 5-15 minutes)
world=bytarr(M,N)
window,0,xs=630,ys=456    ;six pixels per grid square

world(45,40)=1 ;Set up the initial organism---
world(45,41)=1 ;This "seed" is good for a nice long run, with gliders
world(45,42)=1
world(44,41)=1
world(43,40)=1

;The entire LIFE program is executed below in only 12 short, sweet IDL
;commands... Ten if you don't count the endif and endfor as separate
;commands...  In fact, if I don't worry about grid-edge overrun or
;stagnation (zero change) conditions, I can implement LIFE in SIX
;lines of IDL (seven including the END statement.)  For a really fast
;version of LIFE, one could use the WHERE command to locate the "hot
;spots" on the grid, so the computer could ignore all those unchanging
;squares in between...
```

```
;********************************************************** ********

 for k=0,r do begin                        ;RUN R GENERATIONS.
  neighbors=world(0:M-3,0:N-3)+world(0:M-3,1:N-2)+ $ ;COUNT NEIGHBORS
     world(0:M-3,2:N-1)+world(1:M-2,0:N-3)+ $ ;BY SHIFTING THE GRID
     world(1:M-2,2:N-1)+world(2:M-1,0:N-3)+ $ ;AND SUMMING.
     world(2:M-1,1:N-2)+world(2:M-1,2:N-1)
   ;USE WHERE TO CREATE A MASK IDENTIFYING THE CHANGES
  changes=where((world(1:M-2,1:N-2) eq 0 and neighbors eq 3) or $
              (world(1:M-2,1:N-2) eq 1 and neighbors/2 ne 1))
if changes(0) ne -1 then begin            ;CHECK FOR ZERO CHANGE.
   temp=world(1:M-2,1:N-2)                ;CUT AWAY EDGES (OVERRUN).
   temp(changes)=temp(changes) XOR 1 ;TOGGLE THE STATUS OF CHANGES.
   world(1:M-2,1:N-2)=temp                ;RESTORE EDGES.
   for i=0,n_elements(changes)-1 do $
    xyouts,6*fix(changes(i) mod (M-2)),6*fix(changes(i)/(M-2)), $
       '*',/device,color=230*temp(changes(i))  ;DISPLAY RESULTS.
   endif
   xyouts,0,0,'*',color=0,/device
    ;(this line is a fudge to reset grapics plotpoint)
  endfor   ;end of k loop

end      ;that's all, folks!
```