



www.jupiter-ace.co.uk

[Home](#)

[What is ...](#)

[Hardware](#)

[Software](#)

[Documents](#)

[Programming](#)

[FAQs](#)

[Emulators](#)

[Links](#)

[Previous Page](#) > [Listings Index](#) > [Life listing](#)



Life

by
Julian Skidmore

see [here](#) for the download files, wave, Ace and Tap.

```
64 constant width
40 constant height
width height * constant gridSize

create grid0 gridSize allot
create grid1 gridSize allot

: showgrid ( grid -- )
  gridSize + ( end of grid )
  width - width dup + swap
  height 0 do
    width 0 do
      i j 3 pick c@ plot
      1+
    loop
  over -
  loop
drop drop
;
( 8 * 4096 = 4s, 4.37 slow. 2.6s fast, 12.6kips )

( different show grid. )

: hex 16 base c! ;

hex 2c00 gChrSet decimal
16 constant gUdg0

( each cell is 0,0,0,0 for off, 6,9,9,6 for on
  32
  10
)

create cellpats
6 c, 9 c, 9 c, 6 c,

: GenCellChar ( n )
  dup gUdg0 + 8 * gChrSet +
  GenCellSemiChar
  drop
  swap 4 *
  GenCellSemiChar
;

: GenCellSemiChar
  cellPats ( n chrPtr cellpats )
  3 0 do
    3 pick 4 and if
```

```

        dup c@
    else
        0
    then ( n chrPtr cellPats val )
    4 pick 8 and if
        over c@ 16 *
    else
        0
    then
        + ( n chrPtr cellPats val )
    3 pick c!
    1+ swap 1+ ( n chrPtr+1 cellPats+1 )
loop
;

: showgrid2
width 2 / swap
height 2 / 0 do
    3 pick 0 do ( width / 2 )
        dup c@ dup + over 1+ c@ + dup +
        over width + c@ + dup +
        over +br + c@ + 65 + emit
        1+
    loop ( 27 words * 608, 1..2s )
    width dup + + ( next row )
loop
;

: Calcpop (grid -- )
0
gridSize 0 do
    over c@ +
    swap 1+ swap
loop
0 20 at ." Pop=" . space space
;

0 variable seed

( rand in range 0..64k)
: rand ( -- rand )
seed @ 1+ 75 * dup seed !
;

: gengrid ( level grid -- )
height 0 do
    width 0 do
        over rand < over c!
        1+
    loop
loop
drop drop
;

( 123 On top, 123 are +gridsize-1,
+gridsize-width and +gridsize-width+1
4x5 That's +ftfr +ft +ftr
678 4 and 6 are the same as
calcCellL.)

: calcCell-1t ( gridloc -- )
dup +ftfr + c@ ( t1 )
over +ft + c@ + ( +t )
over +ftr + c@ + ( +tr )
over +bl + c@ + ( +l )
over 1+ c@ + ( +r )
over +bfr + c@ + ( +bl )
over width + c@ + ( +b )
over +br + c@ + ( +br )
;

( DONE )

( 123 On top, 123 are +gridsize-width-1,
+gridsize-width and +gridsize-width+1
4x5 That's +ftl +ft +ftr 678 )
: calcCellT ( gridloc -- )
dup +ftl + c@ ( t1 )
over +ft + c@ + ( +t )
over +ftr + c@ + ( +tr )
over 1- c@ + ( +l )
over 1+ c@ + ( +r )
over +bl + c@ + ( +bl )
over width + c@ + ( +b )
over +br + c@ + ( +br )
;

( DONE )

( 123 at rhs, 3 5 8 are: +gridsize-width-1,
+gridsize-width, +gridSize-width*2+1
4x5 That's +ftl +ft +fbfl
678 Otherwise, like CalcCell-r, but )

```

```

: calcCellTR ( gridloc -- )
  dup +ftl + c@ ( t1 )
  over ft + c@ + ( +t )
  over +fbfl + c@ + ( +tr )
  over 1- c@ + ( +l )
  over +bl - c@ + ( +r )
  over +bl + c@ + ( +bl )
  over width + c@ + ( +b )
  over 1+ c@ + ( +br )
;

( DONE )

: calcCellL ( gridloc -- )
  dup 1- c@ ( far top right )
  over width - c@ + ( +t )
  over +bl - c@ + ( +tr )
  over +bl + c@ + ( far right )
  over 1+ c@ + ( +r )
  over +bfr + c@ + ( +b far right )
  over width + c@ + ( +b )
  over +br + c@ + ( +br )
;

( 38 - DONE )

: calcCell ( gridloc -- gridloc sum )
  dup +br - c@ ( t1 )
  over width - c@ + ( +t )
  over +bl - c@ + ( +tr )
  over 1- c@ + ( +l )
  over 1+ c@ + ( +r )
  over +bl + c@ + ( +bl )
  over width + c@ + ( +b )
  over +br + c@ + ( +br )
;

( 38 words - DONE )

( 123 at rhs, 3 5 8 are: -127, -63, +1
4x5 That's +bfr +bl -
678 Otherwise, like Central.)

: calcCellR ( gridloc -- )
  dup +br - c@ ( t1 )
  over width - c@ + ( +t )
  over +bfr - c@ + ( +tr )
  over 1- c@ + ( +l )
  over +bl - c@ + ( +r )
  over +bl + c@ + ( +bl )
  over width + c@ + ( +b )
  over 1+ c@ + ( +br )
;

( DONE Tested )

( 123 On bl, 78 are like b, 1, 4 are
like calcCell-1. 6 is -FBFL.
4x5 That's - +ftl - +ft - +ftr
678 Like bottom, but with these changes )

: calcCellBL ( gridloc -- )
  dup 1- c@ ( far top right )
  over width - c@ + ( +t )
  over +bl - c@ + ( +tr )
  over +bl + c@ + ( far right )
  over 1+ c@ + ( +r )
  over +fbfl - c@ + ( +bl )
  over +ft - c@ + ( +b )
  over +ftr - c@ + ( +br )
;

( 123 On bot, 678 are -(+gridsize-width-1),
-(+gridsize-width) and -(+gridsize-width+1)
4x5 That's - +ftl - +ft - +ftr 678
Like central, but with these changes )

: calcCellB ( gridloc -- )
  dup +br - c@ ( t1 )
  over width - c@ + ( +t )
  over +bl - c@ + ( +tr )
  over 1- c@ + ( +l )
  over 1+ c@ + ( +r )
  over +ftl - c@ + ( +bl )
  over +ft - c@ + ( +b )
  over +ftr - c@ + ( +br )
; ( DONE )

```

```

( 123 On br, 2,5 are like calcell-r,
  67 are like bot, 8 is -ftfr.
4x5 That's - +ftl - +ft - +ftr 678
Like central, but with these changes )

: calcCellBR ( gridloc -- )
  dup +br - c@ ( t1 )
  over width - c@ + ( +t )
  over +bfr - c@ + ( +tr )
  over 1- c@ + ( +l )
  over +bl - c@ + ( +r )
  over +ftr - c@ + ( +bl )
  over +ft - c@ + ( +b )
  over +ftfr - c@ + ( +br )
;

: updateCell ( dst src sum -- dst src )
  dup 2 < over 3 > or if ( dst src sum -- )
    drop over 0 swap c! ( dst src -- )
  else
    3 = if ( dst src )
      over 1 swap c! ( dst src )
    else
      over over c@ swap c! ( dst src )
    then
  then
; ( 14: die, 16: survive, 15: born )

: nextCell
  1+ swap 1+ swap
;

: calcCellsT ( dst src -- dst src )
  calcCellTL UpdateCell nextCell
  width 1- 1 do
    calcCellT UpdateCell nextCell
  loop
  calcCellTR UpdateCell nextCell
;

: ShowRow ( row -- )
  0 10 at ." Row=" . space
;

: ShowGen ( gen -- )
  0 0 at ." Gen=" .
;

: calcCellsMid
  height 1- 1 do
    calcCellL UpdateCell nextCell
  i ShowRow
  width 1- 1 do
    calcCell UpdateCell
    1+ swap 1+ swap
  loop
  calcCellR UpdateCell nextCell
  loop
;

: calcCellsB
  calcCellBL UpdateCell nextCell
  width 1- 1 do
    calcCellB UpdateCell nextCell
  loop
  calcCellBR UpdateCell nextCell
  drop drop
;

( each loop, 38 + 14 + 4 => 56 words,
  * 2560 = 143360 => 17.9s to calc
  in slow mode, 11.9s in fast mode.
  That's 4 generations / min
  with the column method, each loop 34 words
    + 14 + 4 => 52 words * 2560 16.64
  or 11.09s)

: calcCells ( dst src -- )
  calcCellsT
  calcCellsMid
  calcCellsB
;

0 variable gen

: fill ( chr dst len -- )
  0 do
    over over c!
    1+
  loop
;

```

```
: Life ( level -- )
  grid0 GenGrid
  cls
  grid0 showGrid
  0 gen ! ( generation )
  gen @ showGen
  grid1 grid0
  begin
  calcCells ( update for next generation )
  gen @ 1+ dup showGen gen !
  swap dup ShowGrid
  inkey
  until
;
```

(Runs at 29s per generation)
(including display update).
Still, since the zx81 version in Basic
took 15 minutes per generation,
this version is about 31x faster.