Design Document for “Game of Life”

INFT2100 - 702, PROG2100 - 702

Authors	Ben Allen 1111111, Isaac Benoit 2222222, Jamie Curnew 3333333, Will Matheson 4444444
Initiated	November 14th, 2012
Current Rev.	0.8
Revision Date	November 28th, 2012

Key:

	y →					
x ↓	0,0	0,1	0,2	0,3	…	0,59
	1,0	1,1	1,2	1,3	…	1,59
	2,0	2,1	2,2	2,3	…	2,59
	3,0	3,1	3,2	3,3	…	3,59
	…	…	…	…	…	…
	19,0	19,1	19,2	19,3	…	19,59

	y →		
x ↓	x-1,y-1	x-1,y	x-1,y+1
	x,y-1	**x,y**	x,y+1
	x+1,y-1	x+1,y	x+1,y+1

Program begins. A grid of 60 columns by 20 rows is initialized. (This could be implemented as a two-dimensional Boolean array: `bool lifeGrid[20][60];` .) Each space represents a cell. All cells are initially dead. The grid does not wrap – the edge of the grid is the edge of the world.

User is asked which cells are initially alive, one cell at a time, by entering the cell coordinates in (x,y) format. (The user enters coordinates as if the grid is 1-indexed; the actual coordinates on the grid are 0-indexed and therefore should be -1 from the user-selected values.) The chosen cells will be made alive. If a cell has already been marked to be made alive, consider displaying a message like “Cell already alive.” If a row or column choice is out of range, display an appropriate error message.

User indicates being finished with special pair -1, -1. The initial configuration is displayed to the user.

Loop through the grid:

For each cell, count its living neighbours (including diagonals). (This can be done using +/- 1 in x and/or y as shown in the second x/y table.)

If a cell is alive…

… and has *n* living neighbours…	… this occurs:
0-1	Cell dies (loneliness)
2-3	Cell remains alive
4 or more	Cell dies (overcrowding)

If a cell is dead…

… and has *n* living neighbours…	… this occurs:
2 or fewer	Cell remains dead
3	Cell becomes alive
4 or more	Cell remains dead

The actual becoming alive or dead happens all at once after having stepped through the whole grid. If we were to change the states of the cells on the initial step-through, it would affect later results. We'll need to store a collection of x,y coordinates to be activated by another routine. (Try using parallel vectors of integers for storing coordinates, for example: `vector<int> makeDeadRow; vector<int> makeDeadCol;` Be sure to clear them after the changes are made.)

The results are displayed. The user will be prompted as to whether they would like to continue – i.e., run and display another generation ('y' or 'Y'), or quit ('n' or 'N').

Project Closure Report Version Control

Version	Date	Printed By	Change Description
0.1	Nov 14 2012	WM	Initial draft
0.5	Nov 19 2012	WM	Noted that life/death changes occur all at once
0.6	Nov 20 2012	WM	Noted that user uses grid as if it is 1-indexed
0.7	Nov 24 2012	WM	Should display initial configuration to user; note about vectors to store coordinates to change
0.8	Nov 28 2012	WM	We've abandoned this design, but I came back to this document to note that the grid does not wrap.