ES112.02/04
Fall 2003-2004 Term Project
Project #5: “The Game of Life”

1 The Project

The goal of this project is to write a C program that runs the “Game of Life”, and display it on the console screen.

2 Information on the Game of Life

The “Game of Life” is not really a game, there are no players, no winners and no losers. It is called the “Game of Life” because it resembles life in a sense, and it has similar qualities to life itself, like complexity rising out of simplicity.

The Game of Life is “played” on an infinite plane, divided into square cells. Each square is either empty (“dead”) or full (“live”). Every square is considered to have eight neighbours, including the diagonals. The plane evolves in time in steps, according to the rules of the Game of Life. The rules are as below:

- If a square is live in frame n, it will be:
  - Dead in frame $n+1$ if it has 0, 1 (loneliness) or 4 or more (overcrowding) live neighbours in frame n.
  - Live in frame $n+1$ if it has 2 or 3 live neighbours in frame n (survival).
- If a square is dead in frame n, it will be:
  - Live in frame $n+1$ if it has exactly 3 live neighbours in frame n (birth).
  - Dead in frame $n+1$ if it has anything but 3 live neighbours in frame n.

3 Project Requirements

In this project, your goal is to write a C program that simulates the Game of Life on a finite grid of cells. The grid should be 24 by 24 squares large. The grid should initially be filled randomly. Then, each frame should be generated according to the rules of the Game of Life, and displayed on the screen. When displaying, use a space for a dead cell, and an asterisk ('$*$') for a live cell. You should wait about 200 milliseconds between each frame before displaying it. This should go on until the user presses a key on the keyboard, at which point the program should exit.

In order to effectively do the displaying, and checking the keyboard being hit, you need to use platform-specific libraries. In your platform (Turbo C), you should learn about the functions existing in `conio.h` for clearing the screen, and checking the keyboard for being hit.

Last, but not least, since the grid is finite, you will run into trouble when counting the number of live neighbours of a cell at the edge of the grid. In such cases, assume that all cells outside the grid is dead.