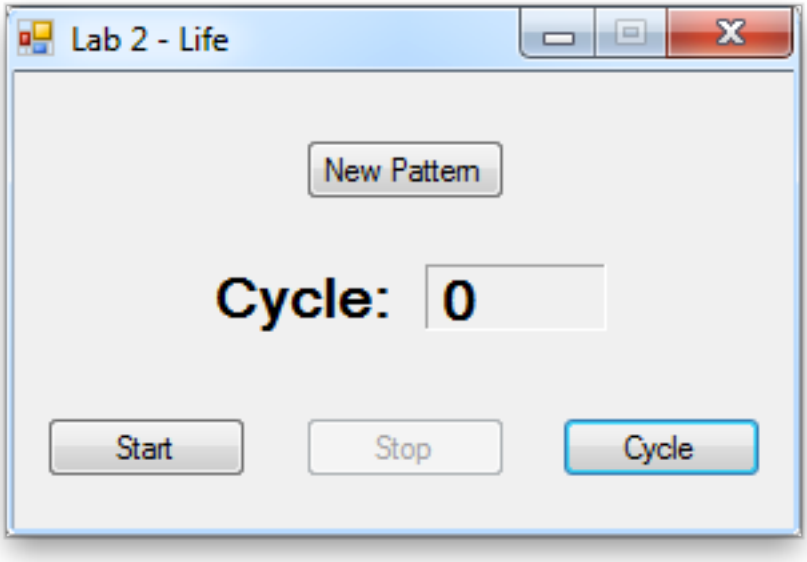
CMPE1600 – Lab 4

The Game of Life was developed as a mathematical puzzle by John Conway in 1970. It is a simulation of the growth of cells. A description of the game, with examples, can be found on Wikipedia at http://en.wikipedia.org/wiki/Conway's_Game_of_Life . The game consists of a two-dimensional array to indicate which locations contain live cells. We will use the GDIDrawer to display the cells to the screen. The main form for the program is shown below:



When the application is started, the controls on the form shall be set as shown above. The FormLoad event will cause a CDrawer window to appear. The drawer will be set to a Scale of 10, so each dot that is drawn will occupy a block that is 10x10 pixels in size. Create **two arrays** of **bytes**, with dimensions of 80x60. Each location in the array will correspond to a location in the drawer window. If a location in the array contains a value of 1, then the cell is alive. A value of 0 is considered to be a dead cell. A location in the array with a live cell is drawn to the Drawer window as a colored pixel; a location with a dead cell remains black.

The two arrays will be used to store the state of the cells. One array, called the **foreground** array, will be used to draw the current state of the cells to the drawer window. The second array, which is called the **background** array, will used to store the results of calculating the new state of the cells. Every cell interacts with its eight *neighbors*, which are the cells that are directly horizontally, vertically, or diagonally adjacent. The new state of the cells after each cycle will be determined by the rules below:

1. Any live cell with fewer than two live neighbors dies, due to under population.

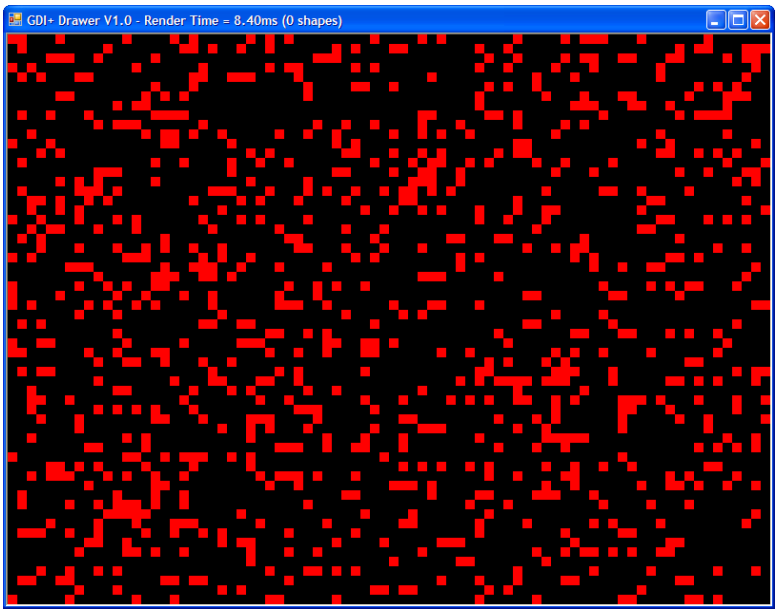
2. Any live cell with more than three live neighbors dies, due to overcrowding.
3. Any live cell with two or three live neighbors lives, unchanged, to the next cycle.
4. Any array location with **exactly** three live neighbors' cells will be populated with a living cell.

When counting the live cells surrounding a location in the array, there are eight special cases to consider. Locations on each wall of the drawing window (X value of 0 or 79, Y value of 0 or 59) will assume that cells beyond the bounds of the array are dead, and should not be counted. Likewise, the four corners of the drawing window will assume that all locations beyond the bounds of the array are dead, and should not be counted. Write your code such that IndexOutOfRange exceptions are not thrown.

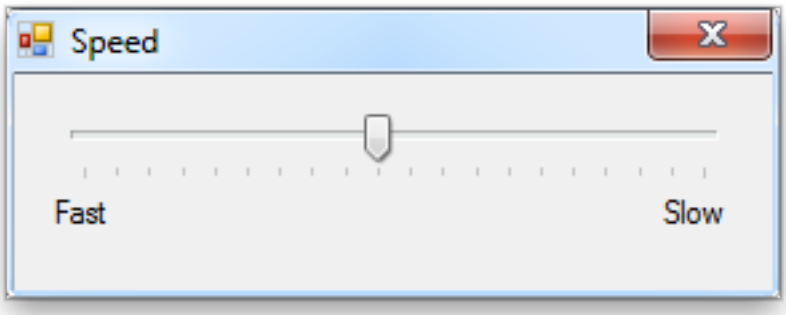
The program will run in cycles, either caused by a timer tick, or by the user pressing the Cycle button. During each cycle, the following actions will take place:

1. Using a double-nested loop, examine all locations in the foreground array. You will assume that all locations beyond the borders of the foreground array, and therefore the drawing window, are dead.
2. At each location in the **foreground** array, count the number of neighbors for the cell, and apply the above rules.
3. Store the cell state result of the above rules in the same location of the **background** array. If the cell is alive, store a 1 at that location; if a cell is dead store a 0 at that location.
4. After all locations of the foreground array have been examined, display the new contents of the **background** array. The background array will now be used as the foreground array for the next cycle.
5. Clear the contents of the **former** foreground array to be ready for the next cycle as the background array.

Initially, when the form is loaded, clear the contents of both arrays, then select one of the arrays to be the current foreground array. Create 1000 live cells in the foreground array at random locations. Use the method Display( ) to display the foreground array to the drawer window by using the CDrawer method SetBBScaledPixel( ). Initially, the **Stop** button will be disabled. Your screen should be similar to the one below:



If the user presses the **Cycle** button, one cycle of Life will take place, and the results will be displayed. If the user presses the **Start** button, the **Stop** button will be enabled; the **Cycle, Start** and **New Pattern** buttons will be disabled. When the **Start** button is pressed, a timer will be enabled with the tick event running one cycle of cell life and the Speed **modeless** dialog will be displayed.

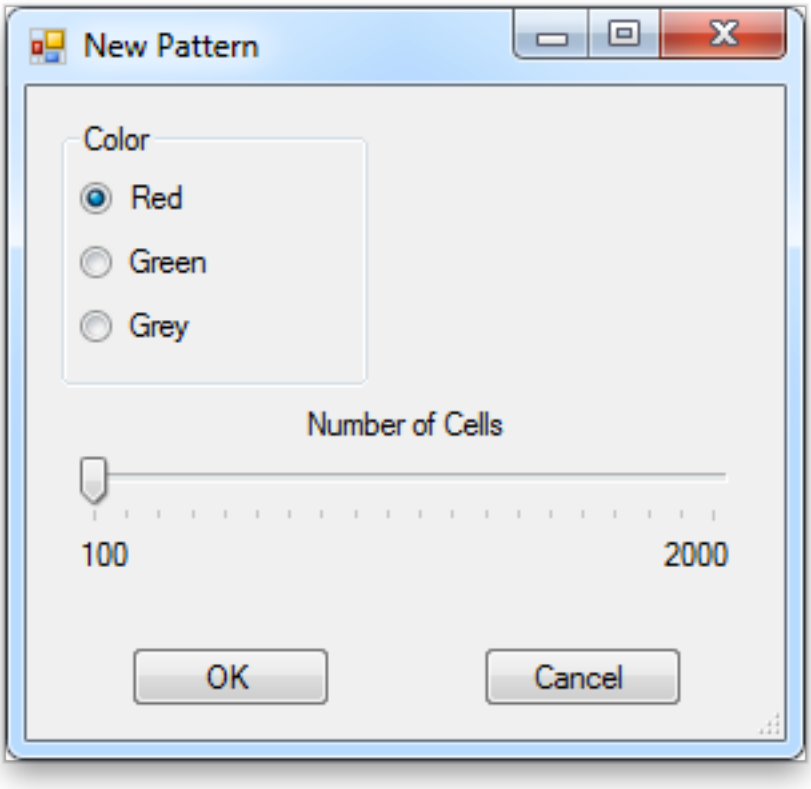


The value of the trackbar will in the Speed modeless dialog be used to set the animation speed. Initially, when the dialog appears, the trackbar will be set to the current animation speed. The fast setting will cause a timer tick every 200 milliseconds, and the slow setting will cause a tick every 2 seconds. When the Speed dialog is closed (using the Close button) the dialog will be hidden, animation will stop, on main form the Start button will be enabled, and the Stop button will be disabled.

On the main form he **Stop** button will disable the timer, and enable the **Start**, **Cycle**, and **New Pattern** buttons.

Your program will count and display the number of life cycles that have occurred for that pattern.

If the user presses the **New Pattern** button the New Pattern **modal** dialog will be displayed. The dialog will allow the user to select the color to be used for the pattern and the number of cells to be drawn. Initially, when the dialog is shown, the controls will display the currently selected color and number of cells. When the dialog is closed using the OK button, a new random pattern of cells will be generated and displayed, and the cycle counter will reset to zero.



Testing your program may require you to try initializing the array to some known patterns from the Wikipedia web page, and observe the results as your program runs life cycles. By way of example, a line of three pixels is a stable pattern that oscillates from horizontal to vertical.

Methods

ClearArray( )

Clears the 2D array of bytes passed to it to zeros.

LifeCycle( )

The method **LifeCycle** will be passed two 2D arrays of bytes, one of which will be used as the current foreground array, and second will be used as the current background array. You will examine each location of the foreground array, and apply the rules of cell growth to calculate the life or death of cells in the background array.

DisplayArray( )

Clears the drawing window, and then displays the cells found in the 2D byte array passed to the method. Live cells shall be displayed using a color, and dead cells will be black. Hint: to clear the drawing window use:

```
m_Canvas.BBColour = Color.Black;
```

Grading

Requirement	Grade
Pressing the Cycle button displays one cycle of cell growth following the rules of Life.	20
Start button begins the animation of Life, and displays the Speed modeless dialog.	10
Trackbar in Speed modeless dialog adjusts the animation speed. Trackbar is initially set to the current animation speed when dialog appears.	10
Animation stops and main form controls correctly set when Speed dialog is closed.	10
New Pattern button displays the New Pattern modal dialog with controls initialized to current color and number of cells.	10
A new pattern of cells is correctly displayed when the New Pattern dialog is closed.	10
Documenation	30