Week 2 – Game of Life Tutorial

CSYS5010 – INTRODUCTION TO COMPLEX SYSTEMS

1. Launch NetLogo
2. Click Settings in the toolbar of the Interface tab. Change Max-PXCOR and MAX-PYCOR to 25. Keep the wrapping check-boxes checked.
   This changes the size of the world grid from default 32 by 32 grid to 50 by 50 grid.

3. Go to the Code tab. Create a Setup procedure by writing the following code

```
to setup
            clear-all
            ask patches
            ;; create 10% alive patches
            [
                    set pcolor blue ;; blue cells are dead
                    if random 100 < 10
                            [set pcolor green] ;; green cells are alive
            ]
            reset-ticks
end
```

   This code has three sections. The first section is one line, the command `clear-all`, which clears the view (setting the color of all its patches to the default color black) in case there are leftover elements from our previous play of the game. The second section issues commands to all of the patches. In NetLogo, we are polite in our interactions with agents, so to issue commands to the patches we write `ask patches`. There are two commands/requests that patches need to perform. The first asks each patch to set its color to blue. The second request ask each patch select a number between 1 and 100: if the number is less then 10 then set its color to green. The third section is one line `reset-ticks`. This command resets the NetLogo clock.

4. Return to the Interface tab. Create a button and name it SETUP. In the Button dialog, type SETUP as the Display name prompt and `setup` in the Commands prompt.
5. Test the setup button. Does the system initialize correctly? Try changing the setup code, for example, modify the probability of a cell being alive or change the color of the alive cells.
6. Go the Code tab and create a go procedure by inputting the following code

```
to go
        ask patches [           ;; each patch counts its number of green neighboring patches
                                ;; and stores the value in its live-neighbors variable
         set live-neighbors count neighbors with [ pcolor = green]
         ]
         ask patches [
               ;; patches with 3 green neighbors, turn (or stay) green
         if live-neighbors = 3 [ set    pcolor green ]
               ;; patches with 0 or 1 green neighbors turn (or stay) blue
              ;; from isolation
         if (live-neighbors = 0) or (live-neighbors = 1) [ set    pcolor blue ]
               ;; patches with 4 or more green neighbors turn (or stay) blue
              ;; from overcrowding
         if live-neighbors >= 4 [set    pcolor blue]
               ;; patches with exactly 2 green neighbors keep their color
         ]
         tick
end
```

   The first part asks patches to count the number of adjacent green neighbours and store this number in a

variable `live-neighbors`. The second part changes the state of the patch according to the Game of Life. The final part is a one line command `tick`, which advances the clock by one tick.

7. Declare the `live-neighbors` variable by adding the following line at the start of the code (before the setup procedure)
   `patches-own [live-neighbors]`
8. Return to the interface tab and create a button called GO. Set up the GO button to call the go procedure just like we did for the SETUP button. Tick the Forever tickbox if you wish to observe continuously running clock.
9. Run the model. Observe various patterns of the Game of Life