Assignment 1: The Game of Life – effects of surroundings and contacts

Due February 1, 5pm.

Many processes in biology can be simulated by a set of primitive objects along with rules that dictate their behavior (e.g., when they are formed, when they die, how they move, how they interact with other objects). The Game of Life is a standard programming project for learning how to use arrays and perform simulations.

One starts with (generally) a 2-D array where some locations are occupied and others are empty. This could be done randomly, via some explicit patterns, or read in from a file. This is generation 1 of your "habitat".

You then create a second array to hold the results of the next generation, which is created by applying rules to each location in the array. For example, if location (i,j) is empty in the current generation, but it has at least 2 neighbors that are not empty, then in the next generation we would set that location to occupied (a birth!). Similarly, if a location is occupied, but all of its neighbors are empty, it might die of loneliness, or if all its neighbors are occupied it might die of overcrowding. You get the idea.

For this project, create a basic 2-D Game of Life using graphics to show the state of each location for a given generation. You should have at least 4 rules for birth and death of cells. Run the simulation for a large number of generations, stopping if the number of changes between generations goes to zero. You can either have the user click a key to go to the next generation or let each display stay on the screen for 1 or 2 seconds before advancing. Again, if you find code for this on the web, please indicate the source and what changes you made to it to make it your own work. Generally, you will learn more and gain a better sense of accomplishment if you develop this from scratch. These are the minimum requirements.

Remember that in the assignments part of your grade depends on going beyond the minimum requirements. There are many opportunities to move beyond the basics. For example:

- You might have more than one type of object (predator/prey), with different sets of rules.
- You could integrate moving objects, such as migrating cells in a developing organism moving into unoccupied cells (careful about collisions!).
- You could simulate the propagation of a disease based on contacting an infected cell (especially cool to watch when the objects are moving).
- You could add interaction to allow users to set parameters or interact with the grid while it is running or paused.

- You could build separate views that show the simulation in a different way, like a bar chart showing different object types in the current simulation step, a line chart to show changes over time.

Don't restrict yourself to these examples! The goal is to develop and implement your own ideas.

Resources

If you're using Processing, there are many simulation examples on the web. One especially relevant resource is Daniel Shiffman's online book, The Nature of Code. The code examples use Processing, and part of it deals with agent-based modeling.

If you want to add things like sliders and text input, see the controlP5 library.

If you use Sublime Text for coding, here is a tool that will allow you to run Processing from within your editor.

Writeup Requirements

Your README should:

1. If you are using outside code for the core of your project:
    - Identify the source(s) of the code.
    - Tell me how you modified it to make it your own work.
2. Include descriptions of rules that you applied.
3. Indicate how you generate the first generation, e.g. randomly, via some explicit patterns, or read in from a file.
4. For the free-form (beyond the requirements) component:
    - Describe the biological significance of the additions you make.
    - Describe the technical significance of the additions you make.
    - (Note: depending on what you come up with, there may be some overlap, e.g. both biological and technical significance within the same feature. This is fine, just be sure to talk the feature from both perspectives.)
5. Provide explicit instructions on running your program.

Turning in the project

Submit a zip or tar file containing:

- All of the source code
- A README file containing the elements listed in the Requirements section

The source code must run on a Mac or Linux machine.

Submissions should be made using myWPI. Either:

1. (Recommended) Upload a .txt file with a link to your GitHub repo, or
2. Zip your files and README and upload to MyWPI

Grading

Each homework assignment is graded on a 100 point scale:

- 85 points will be based on whether the program fulfills the minimum requirements.
- 15 points will be based the additions you made:
  - 7.5 Bio-related additions
  - 7.5 Technical additions

Total – 100

(0 will be assigned if the code can't be compiled or run.)