

## **CM151 – Project 2**

**Due: December 14<sup>th</sup>, 11:55pm**

In this project, you are encouraged to work in groups of 2 or 3. You can choose from one of the below suggestions, or you can create your own project.

As in project 1, you may decide to select your own topic. You are strongly recommended to check with Matt before starting to make sure it's not too hard. The size of the project should be roughly the same as project 1.

Submit your program through Moodle. Grading will be based on:

- Correctness of the program - does it perform correctly for each specific feature?
- Functionality of the program - which of the desired features have been implemented?
- Composition of the program - are there appropriate functions or is everything put together in one main program?
- Your ~15 minute discussion with Matt about your project.

### **Suggestion 1: The Game of Life**

The game of life was created by John Conway in the late 60's and described in Martin Gardner's Scientific American column. For this game you will create a text interface that will control a graphics to run The Game of Life. The game of life is described here:

[http://en.wikipedia.org/wiki/Conway's\\_Game\\_of\\_Life](http://en.wikipedia.org/wiki/Conway's_Game_of_Life)

and Matt can provide more materials if you're interested. The text interface must provide the options described below, and the graphical interface should allow you to click on it to turn pixels on and off.

#### *Graphical Interface:*

The graphical interface is an array of squares (or pixels) that form a grid, that have one of two values: on or off. When a pixel is on, it will have a bright color, like: white, yellow, etc. When a pixel is off, it will have a dull color, like: black, gray, etc. When the mouse clicks on a certain pixel the color should be set to "on". This mouse clicking behavior should be activated by the text window, and turned off when clicking outside of the array of squares.

### *Text Interface:*

The text interface allows you to control the application and should have the following features provided as a numbered menu. Users can choose which feature they want by entering a number. After each feature is performed the menu should be redrawn.

- An about option that will explain your program and who created it.
- A run once option, which will run the simulation once.
- A run n times option, which will ask the user for an integer value and run the game that many times.
- A clear option that set all the pixels to “off”.
- A random option that will randomly set pixels to “on or off”.
- A load glider option that will create a glider at a specific location in the grid of pixels. Note, the location will be provided by the user and must be a valid position. If the position is invalid, your program should request a new position.
- Activate mouse update. This will repeatedly look for a mouse click on the graphics window, activating the pixel that is being clicked on. If the mouse click is outside of the pixel grid, this option will finish and the control will be given back to the window.
- Exit program.

### **Suggestion 2**

Use one of the USB missile launchers to locate a large red square of paper in its field of view and fire a missile at it. This project involves interfacing with a physical device, similar to an instrument you’d find in your lab, but more fun.

### **Suggestion 3**

Write a program that does something with the stock market. For instance, you could read in a stock’s historical data from the web using Python’s HTML parser

<http://docs.python.org/3.3/library/html.parser.html?highlight=html%20parser>

from a public website

<http://finance.yahoo.com/q/hp?s=MSFT+Historical+Prices>

Given this data you could

1. Graph the company’s price over time.
2. Allow the user to select a date and stock amount and tell them how much they would have made/lost.
3. Compare this investment to a different stock in the same industry.
4. Try to come up with an algorithm for when to buy stock in this company. Then, show that your algorithm does / does not work for a specific time range.