CX 4230 Project 1
Checkpoint 1

Team Members:
Brian Glowniak
Joel Katz

Project Description and Approaches:
The Game of Life is a model proposed by John Conway in which the universe of states is described by a two-dimensional grid of square cells in which each cell may be either alive or dead. In the standard form of this model, the state of a cell is a deterministic function of its own state and the states of its neighbors. A neighbor is defined using the Moore neighborhood, which is the adjacent cells in the rectilinear plane (left, right, up, and down) and the diagonally adjacent cells (up-left, up-right, down-left, down-right). If the cell is alive and there are 0 or 1 live neighbors, the cell dies as is by underpopulation. If the cell is alive and there are 2 or 3 live neighbors, the cell stays alive. If a live cell has 4 or more neighbors, that cell dies as if by overpopulation. If a cell is dead and has exactly 3 live neighbors, it becomes a live cell as if by reproduction.

This model has many interesting properties that make it a popular topic to study, including the many interesting conglomerations of living cells that can form. Some formations are stationary (still lifes), some cycle between two or more configurations (oscillators), some translate across the grid space continually (spaceships), and some emit spaceships (guns). Using these formations, the "Game-of-Life is a universal computer" (Jarkko). More specifically, since spaceships can transmit information infinitely we can create signals, and the other formations can be used to process them, providing the necessary components to perform computing operations.

For our tutorial on this topic, we will first go over cellular automata, and how to implement one using the Game of Life as an example. From this, we will build upon the model with interesting expansions including: asynchronous updates (whether they are sequential or random), adding a time-to-live element (nodes die off after a certain amount of time, represented by a a color change as it progresses), and variations on the birth and death rules (e.g. B1/S12 - born if 1 live neighbor, survive if 1 or 2 live neighbors, die otherwise -- Sierpinski Triangle/ Fractals). We will also explore the idea of simulating logic gates and transmitting information as well. Each of these variations/experiments will be accompanied with analysis and will provide a platform to build knowledge of cellular automata and of the various capabilities it has, beginning with the basics and iteratively adding more complex features.

With regards to our platform of development, we are planning on creating this tutorial with Python embedded within Jupyter notebooks. This platform allows us to isolate sections of text and code, and have the user interactively run the code segments and see the results. Code segments will be accompanied by analysis, visuals, and explanations of what is being done.

Project Progress

To become familiar with the model we have chosen at its most basic level, we have implemented Conway's Game of Life using Python. This script can be found in our Github repository titled as *conway_basic.py*. A summary of the code is provided below. Those interested in viewing the full source can find our Github link located at the end of this document.

In terms of implementation details, we have chosen to represent the state of the world as a 2D array populated with 1s and 0s, which correspond to the states ALIVE and DEAD. We have chosen to represent states as integers rather than booleans to allow us to easily expand our initial implementation later on (i.e. when we implement a "time-to-live" for nodes). The script first runs *init_world(N, M, X, threshold)*, which generates an N x N grid with X clusters of size M x M that are populated according to the threshold value (i.e. higher threshold means these clusters will be more densely populated with living entities). The *timestep(world)* function takes in a world-state and iterates through each cell in order to perform an update according to the rules of the game. This is currently a brute-force implementation, and we are going to explore routes to optimize this algorithm as we move forward in the project. The *timeseries(world, Y)* function takes in a world and performs Y timesteps on it, and we visualize the results using pcolor plots from the Matplotlib Python library. An example is provided below:

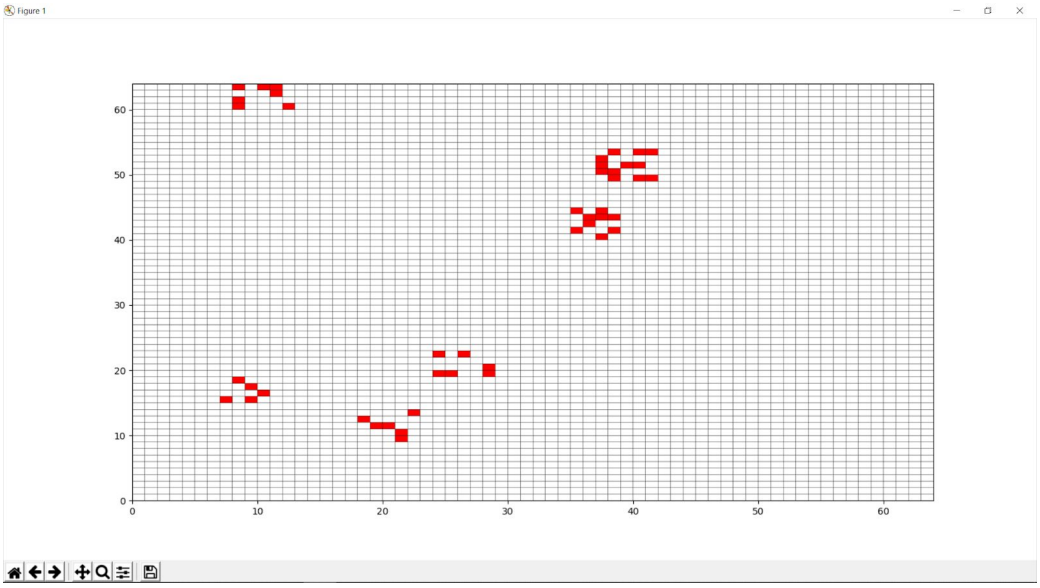

Figure 1: An initial 64 x 64 world state generated with 6 5 x 5 clusters

Having created this basic implementation, it will act as a foundation for us to build upon as we explore the system in further depth. This first script will be translated to a Jupyter notebook and will act as our initial tutorial on cellular automata.

Division of Labor:
As a group of two, it is very easy to keep each other accountable in terms of output and overall effort. With regards to this first checkpoint, Joel performed research and generated some documentation on the Game of Life model, while Brian programmed the initial model in a standard Python script. As we move forward, we first plan to divide up the elements and experiments we want to cover into discrete tutorials/segments (e.g. asynchronous updates). Then, we will divide responsibilities by evenly assigning these to ourselves, fully implementing and integrating that feature into the existing codebase, then adding this element to the tutorial in a Jupyter notebook. If any snags arise, we will work together closely to resolve them, and we will of course also continually review each other's code to ensure consistency and accuracy.

Github Repository Link:
https://github.gatech.edu/bglowniak3/CX-4230-Project-1

This repository contains our initial coding efforts, our Checkpoint 1 Report, and a README. It has been shared with all instructors as required.

Project References:
Kari, Jarkko. "Cellular Automata: Tutorial." TUCS (Turku Centre for Computer Science), 2008. (http://grammars.grlmc.com/LATA2008/slides/lata2008_cellular_automata.pdf)

"Life Universal Computer." Igblan. (www.igblan.free-online.co.uk/igblan/ca/)

Rennard, Jean-Philippe. "Implementation of Logical Functions in the Game of Life." (https://www.rennard.org/alife/CollisionBasedRennard.pdf)