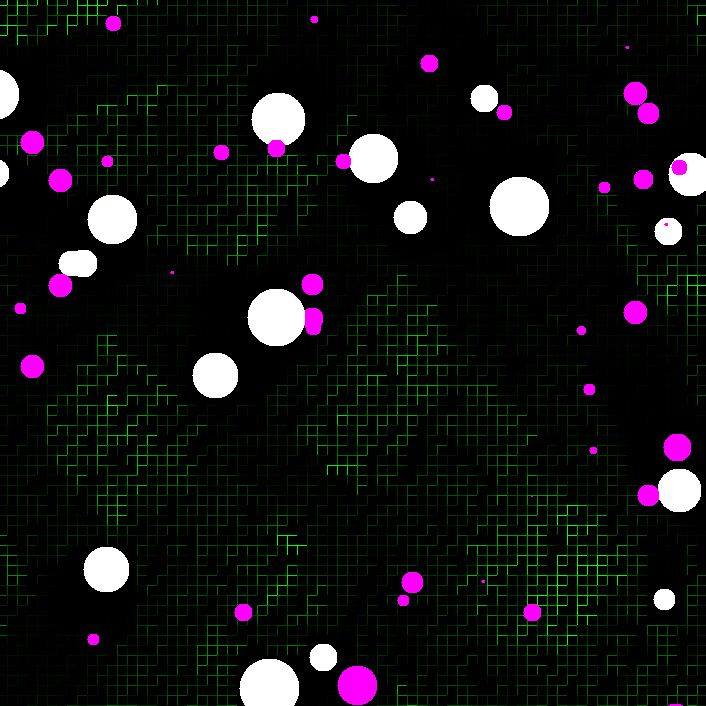
**Cell Evolution Simulator**

Wilson Wu

**Description**

Biological evolution is the change in heritable traits when a biological population passes on its genetic information it its successor population. The idea is that while this change could either negatively or positively impact that population, the positive and more favourable traits would begin to dominate in succeeding generations. This program simulates evolution by generating a multitude of cells with randomized initial traits, such as speed, size, and splitting size. The cells will have to survive in an environment that is filled with various hazards and food sources that they would need in order to survive. At a certain age, the cells will split into separate cells that have similar genes, but a few randomized mutations. Eventually, cells will become more efficient, and surviving generations will be able to better navigate through the environment.



There are two major types of cells:

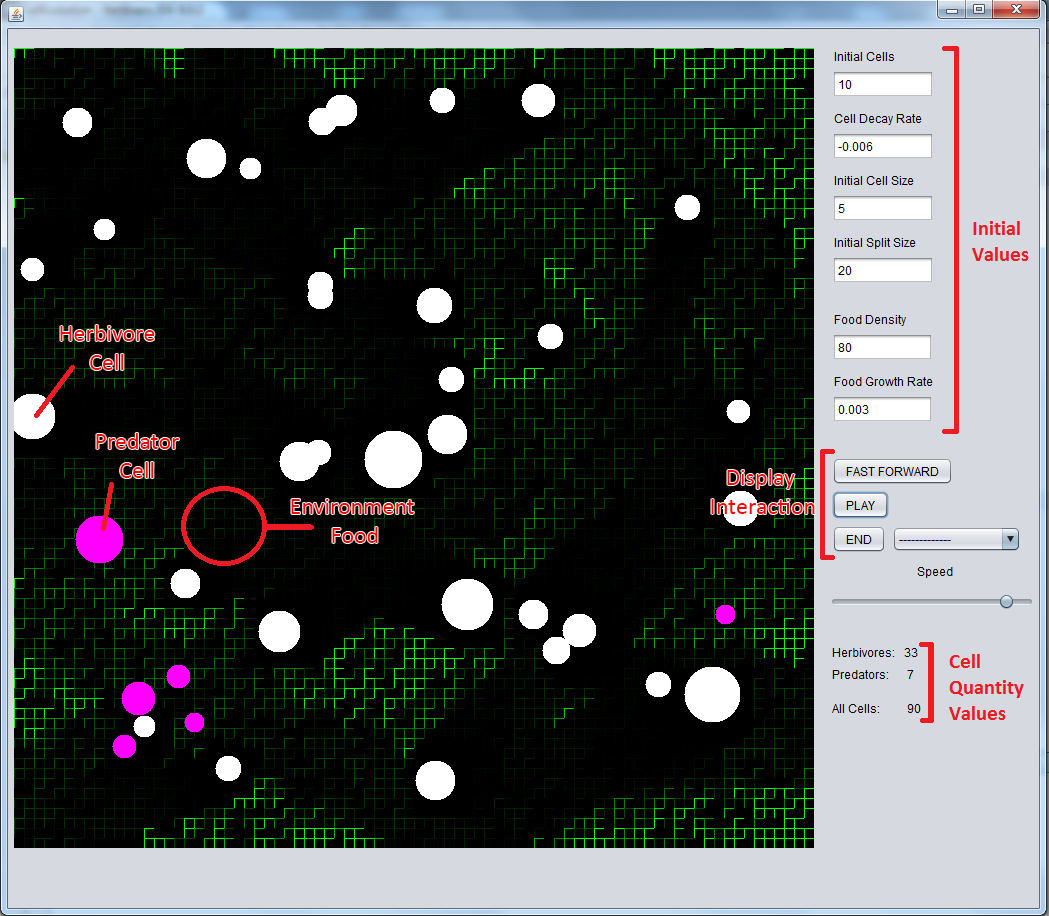
**Predator cells**: Purple cells that feed exclusively on smaller herbivore cells.

**Herbivore cells**: White cells that feed exclusively on the green environment “grass”.

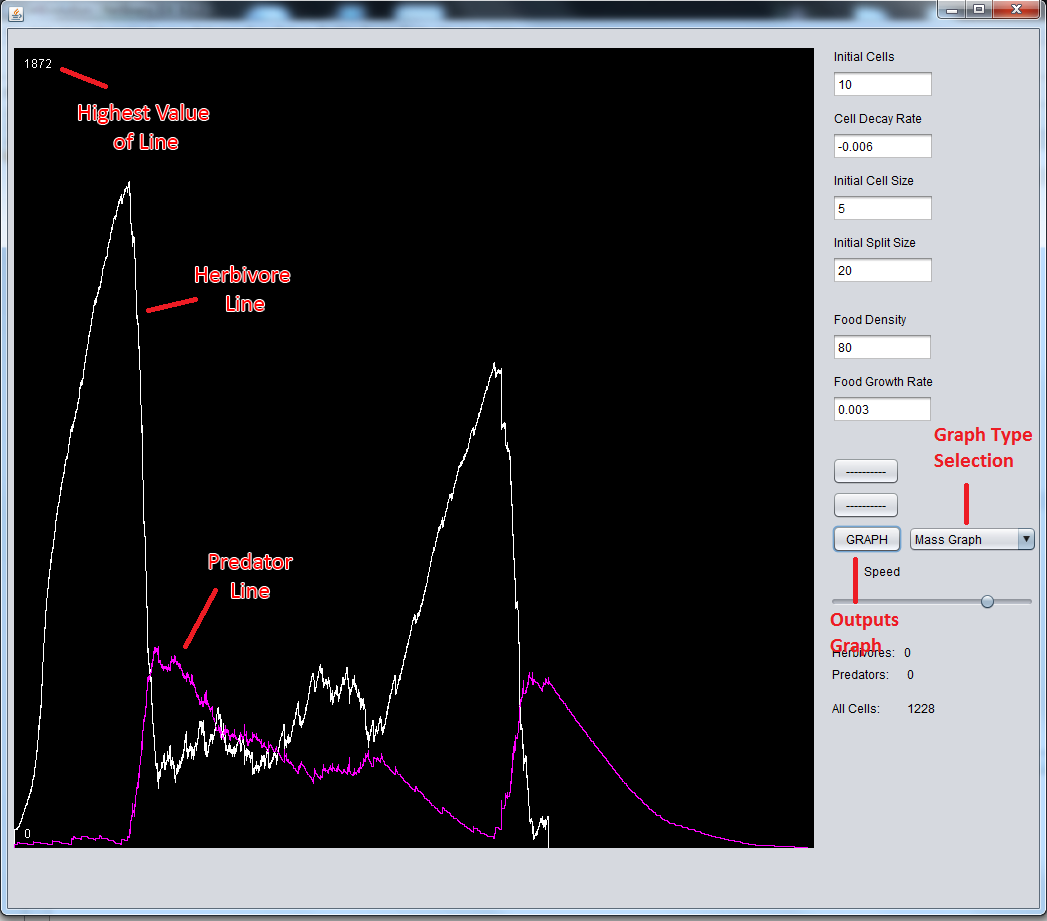
**Inputs**

* **Initial Values**
* **Cell Decay Rate**
* **Initial Cell Size**
* **Initial Split Size**
* **Food Density**
* **Food Growth Rate**
* **Start Button** - 3 states
  + Start: Begins simulation
  + End: Ends simulation
  + Graph: graphs cell Data
* **Pause Button**: pauses simulation
* **Fast Forward Button**: accelerates run speed by not drawing data to screen
* **Speed of Simulation:** sets duration the simulation pauses for between each frame

**Outputs**



Once the start button is hit, the simulation begins with the variables set. The speed can be modified through the slider, pause button, and fast forward button. The simulation will run continuously until the “END” button is pressed. Once the “END” button is hit, a graph button will be brought up (see below).



A graph button will be available once the simulation is ended, with options of various data graphs. Hitting the graph button will bring up the graph chosen. Graphs show line graphs of trends, comparing predator cells with herbivore cells.

**Classes**

**Cell Class**

Class that stores information about cells.

|  |  |
| --- | --- |
| **Fields** |  |
| int **xPos, yPos** | Location of the cell midpoint. |
| int **xSpeed, ySpeed** | Direction and speed the cell is moving at. |
| double **cellSize** | Radius of a cell. It is 1:1 scale with the visual graphics. |
| double **splitSize** | The radius size a cell has to reach, before it dies and spawns two new cells. |
| boolean **predator** | Sets if a cell is a predator or prey. Predator when the variable is true, prey when the variable is false. |
| boolean **cellAlive** | Decides if a cell is alive. True for alive, false for dead. |
| Random **rand** | Random number generator used for generating random cell mutations |
| **Methods** |  |
| public **Cell**() | Creates a randomized cell, that sets all its variable traits pseudo-randomly (ie. xPos yPos, predator) in a limited range. CellAlive will always be set to true. |
| public **Cell**(list of traits) | Creates a specific cell, that sets all variable traits to specific values that is specified in the parameters. CellAlive will always be set to true. |
| public **Cell**(Cell other) | Creates a cell, based on a parent cell “other”. Location is set to the parent cell, while other variable traits are set to the parent cell’s with a small range of variance. CellAlive will always be set to true. |
| public static double **distance**(double x1, double y1, double x2, double y2) | Calculates the distance between two points. |
| public boolean **edible**(Cell other) | Checks if a cell is in range to eat another cell “other”. A cell can eat another cell if it is completely encompassing “other” (cell has a bigger cell radius, and distance of midpoints is less than the cell’s radius). |
| public void **updateCellLocation** () | Updates the location of a cell. Adds on xSpeed and ySpeed to xPos and yPos, respectively. Reverses xSpeed or ySpeed, if cell’s edge location is at the edge of the screen. |
| public String **toString**() | Returns all the cell’s variables and information in a string. |

**Environment Class**

Superclass that stores data on the environment.

|  |  |
| --- | --- |
| **Fields** |  |
| int **blockNum** | Number of “environment” grid items per row and column. |
| double[][] **envioFill** | Array that stores values of environment background items. |
| **Methods** |  |
| public **Environment** (int BN) | Initiates variables. |
| public void **setup**() | Fills up envioFill as an empty array. |

**Food Class**

Subclass that stores data about the environment's food. Extends to Environment.

|  |  |
| --- | --- |
| **Fields** |  |
| double **growthFactor** | Multiplier constant for how fast the food will grow at. |
| double[][] **foodGrowth** | Array that stores how fast each food element will grow. |
| Random **dice** | Random number generator used to generate random growing speeds. |
| **Methods** |  |
| public void **setup**() | Calls superclass, and fills up growthFactor with growth values |
| public void **update**() | Updates envioFill with its respective growth factor. |

**Hazard Class**

Unused subclass that stores data about the environment's hazards. Extends to Environment.

**Drawer Class**

Class that updates the screen’s visual display.

|  |  |
| --- | --- |
| **Fields** |  |
| int **width, height** | Size of visual display screen. |
| int **gridSize** | Visual size of each environment fill item. |
| Color **tempColor** | Variable used to temporarily store a new color. |
| JPanel **drawingPanel** | Panel where frames are displayed on. |
| **Methods** |  |
| public void **paintImage**(ArrayList <Cell> cellArray, double[][] foodEnvioFill) | Displays a buffered image onto the screen canvas. |
| private Image **createImage**(ArrayList <Cell> cellArray, double[][] foodEnvioFill) | Creates a buffered image which will be used for displaying. Draws up the cells, environment, and background. |
| public void **startUp**() | Initiates canvas window. |
| public void **paintCellGraph**(ArrayList <Integer> herbArray, ArrayList <Integer> predArray) | Paints a line graph onto the canvas. Graph data is taken in two separate arrays. |

**CellEvolutionGUI Class**

Main class that runs and controls the simulation.

|  |  |
| --- | --- |
| **Fields** |  |
| int **numInitCells** | Number of initial cells. |
| int **blockNum** | Number of “environment” grid items per row and column. |
| double **cellDecay** | Rate the cells shrink by every frame. |
| double **growthFactor** | Multiplier constant for how fast the environment food will grow at. |
| int **initCellSize** | Constant for the range the initial cells will spawn at. |
| int **initSplitSize** | Constant for the range the initial cells will split at. |
| boolean **programRunning** | Variable that decides if the program will run |
| boolean **programStarted** | Variable that decides if the program will start. |
| boolean **paintScreen** | Variable that decides if the program will paint the generated frames onto the screen. |
| Random **dice** | Random number generator used to decide if certain cell mutations will take place. |
| Food **myFood** | Food object that stores data for the background “grass”. |
| Drawer **myCanvas** | Object that is used to generate the visual frames. |
| int **blockSize** | Variable used to store graphical size of each environment item. |
| ArrayList <Cell> **oldCellArray, newCellArray, deadCellArray** | Array of cell objects that store the data for all the cells. oldCellArray is used to apply calculations for newCellArray, which is used to store data for cells that will be displayed. deadCellArray stores information about every cell that dies. |
| ArrayList <Integer> **herbPopArray, predPopArray, herbMassArray, predMassArray** | Various arrayLists of integers that store data, which is used for graph generating. Data is stored about population, and overall cell mass of herbivores and predators. |
| int **herbTot, predTot, herbMassTot, predMassTot** | Various integers that store data about the current cell generation, which is used for graph generating. |
| **Methods** |  |
| int **leftBound, rightBound, topBound, bottomBound** | Ranges that a herbivore can reach on the environment. |
| double **foodTotal** | Total amount of food that a herbivore cell is in range of to eat. |
| public void **setSimulation**() | Grabs the initial values from the GUI, and initiates all the object items. |
| public void **updateImage**() | Calls for screen to be visually refreshed. |
| public void **updateCells**() | Updates data about each cell object, based on their interaction with other cells and the environment. Interaction updates include cells eating, cells dying, and cells splitting. Also stores data about the cell population into arrayLists. |
| public static void **sleep**(int duration) | Temporarily pauses the simulation. |
| private void **startButtonActionPerformed**(java.awt.event.ActionEvent evt) | Button used for starting and ending the simulation. Once the simulation ends, it becomes the button that calls for data to be graphed |
| private void **pauseButtonActionPerformed**(java.awt.event.ActionEvent evt) | Button that calls for the game to be paused/unpaused. |
| private void **fastForwardButtonActionPerformed**(java.awt.event.ActionEvent evt) | Button that sets the game on fast forward, by stopping visual animations. |
| public static void **main**(String args[]) | Controls the flow of the program. It manages the functions that have to refresh every frame, and runs based on various boolean variables. |