Lucy Christiana

Rahma Ahmed

Kerri-Ann Norton

Programming Nature

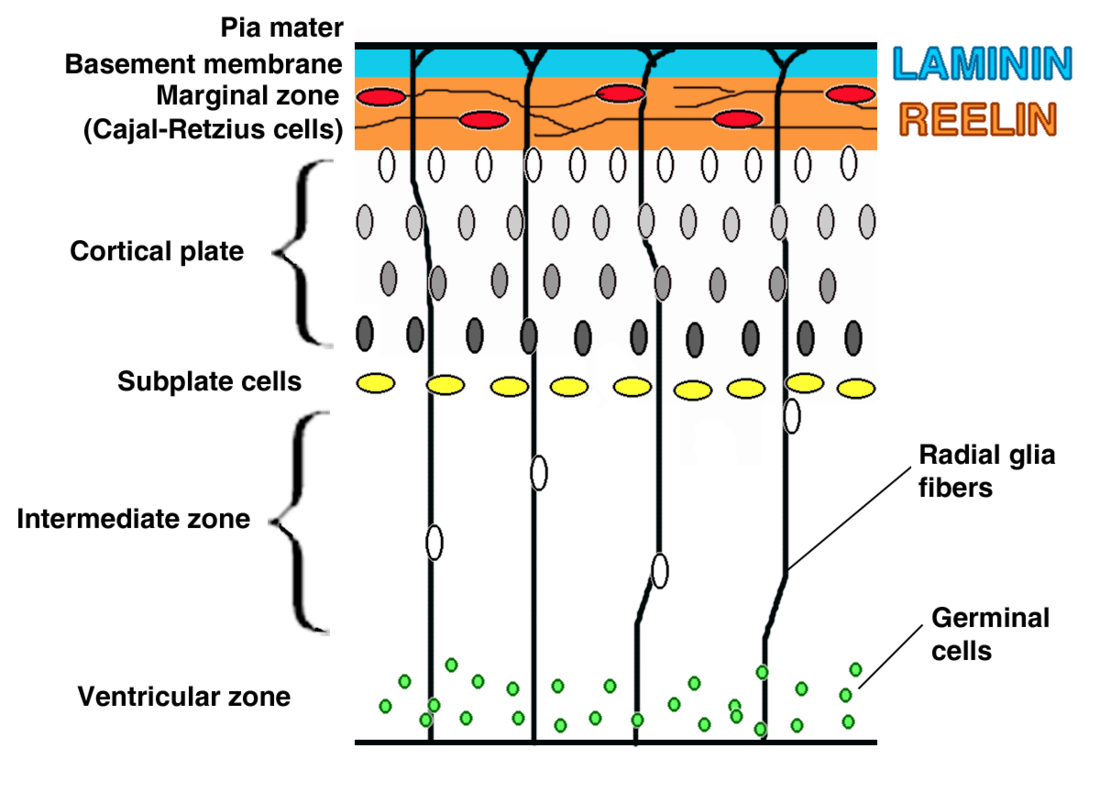
14 December, 2017

The Developing Brain: An Agent-Based

Model for Neural Pathfinding

**Introduction**

Because there are literally billions of neurons in the human brain, one problem neurobiologists are faced with is how did they get there. How does this well-oiled machine that we do not even understand work so efficiently and bring us consciousness? If the exact “home” location were hardcoded into our DNA for every neuron in the brain, our DNA would have to be much longer than the few billion base pairs it is. This means that there is some sort of pathfinding, cell-signaling system in place during this period of development in which the neurons find their homes. Our project examines one piece of this process: corticogenesis, in which the cerebral cortex, or the outermost layer of “gray matter” of the brain is formed.



**Figure 1: Corticogenesis.** Shown here is a diagram of the development process off of which our model is based. The green germinal cells at the bottom are essentially the “bank” of cells which may become neurons. The cells that do become neurons travel up the radial glia fibers, past the subplate cells, towards the reelin layer. These cells then fall into an age-specific order in the cortical plate: youngest to oldest, from the top-down.

Based on Figure 1, our model can be understood as one “glia cell path system.” We have created a model in which a pool of stem cells reproduce neurons. These neurons then travel up a glia path and fill the cortical plate, which we have implemented as a cellular automata, in the order of oldest to youngest. We have focused on the first iteration of this cycle, essentially the “filling” process.

**Methods**

Main

Cortical Plate

Cortical plate is essentially a 2D array of neurons. almost identical to a cellular automata but with a slightly different way of implementing a ruleset. Our version is a grid which holds an age value (if null, that spot is empty). For this class to work, our path had to exist as well as “communicate” with our neuron class, so that the neurons could actually follow the path.

|  |  |
| --- | --- |
| Variables   * Neuron[][] cells * int numNeurons * int columns =2 * int rows = 5 * int cellWidth = width/30 | numNeurons → how many cells are in the array |
| Constructor | Sets numNeurons to 0  Creates new 2D array of of neurons |
| Display | Displays 2D grid  Loops through 2D array and sets positions for each neuron |
| moveCellsDown | Moves cells into position in the cortical plate.  Moves them into whichever spot is open--if both are open, chooses a side randomly.  Cells will move onto whichever one of the columns is less full. |

StemCell

Our stem cell class is our germinal cell pool at the bottom of the screen. This class is loosely based on stem cell classes from past labs.

|  |  |
| --- | --- |
| Variables:   * PVector position * Float radius | Position of cell.  Radius of the cell. |
| Constructor | Initializes the position of the Stem Cell to a random value within the bank at the bottom of the screen. Radius is set to 5. |
| Display function | Displays the StemCell as a green circle. With black stroke. |
| Reproduce function | Makes a new StemCell and returns it. |
| Boolean senses other function | Takes in PVector from c and returns whether there is another stemCell within the given vision range |

Neuron

Our neuron class is a child class of the stem cell class. It has an age value, as well as a lifespan.

|  |  |
| --- | --- |
| Variables   * PVector position, velocity, acceleration * Float maxSpeed, maxForce, age, colorV * Color c * Boolean reachPathStart | Initializes variables for later use. |
| Constructor | Sets age to 0.  Sets velocity to 0.  Sets acceration to (0.01, 0.1)  Sets radius = 10 |
| Display | Draws the actual neuron icon |
| Age function | Adds age value and decreases color value |
| Seek function | Seeks the start of the path using a “desired” PVector. Moves towards the target, which in this case is the glia cell path. |
| Follow function | Allows Neuron to follow path, using path following from the Nature of Code textbook |
| GetNormalPoint function | Takes in PVectors p, a, and b which are used to get the normal point, which allows the neuron to move towards the path that it is following |
| applyForce function | From past labs |
| Update function | From past labs |

Path

Our path is our model of a glia cell, or the “scaffolding” of the neuronal development process. This path was based on the pathfinding section of the Nature of Code book.

|  |  |
| --- | --- |
| Variables:   * float radius * PVector start * PVector end | Start point of the path  End point of the path |
| Constructor | Sets radius to the width of the screen divided by 30. |
| Display | Displays the path as a line within a line that has a width of width of screen divided by 15. |

**Conclusion/Future Steps**

In conclusion, we have created an agent based model of corticogenesis. This model shows a bank of stem cells that become neurons. These neurons then travel up the structural glia cells, in this case a path following implementation. Once they have traveled up the glia, the cells arrange in order from youngest at the surface to oldest, until they have filled the 5-layer cortical plate, which is a 2D array/pseudo-cellular automata of neurons.

Because we have only focused on the “filling” process, future steps could include how this process might continue, if a new neuron arrives, and the oldest one essentially gets “killed off” or moved down. We also have made one glia cell with one age-gradient grid. To expand this model, we can create a wider window with a wider system and drop in different glia cells and age gradients. This way, it is possible to have three different systems (the same structurally), right in front of you, and you can compare how various factors affect the efficiency of this process.

**Acknowledgements**

Daniel Schiffman, *The Nature of Code*

PowerPoints, Kerri-Ann Norton

This project was coded in collaboration with each other. Both of us pseudo-coded/outlined separately and then came together to code.