Visualizing Morphogenesis with the Processing Programming Language

Avik Patel, Amar Bains, Richard Millet, and Tamira Elul

We used Processing, a visual artists' programming language developed at MIT Media Lab, to simulate cellular mechanisms of morphogenesis – the generation of form and shape in embryonic tissues. Based on observations of in vivo time-lapse image sequences, we created animations of neural cell motility responsible for elongating the spinal cord, and of optic axon branching dynamics that establish primary visual connectivity. These visual models underscore the significance of the computational decomposition of cellular dynamics underlying morphogenesis.

Introduction

Processing is a Java based software language and environment created at MIT Media Laboratory in 2001 for visual modeling (www.processing.org). Processing has grown into a popular tool and online community for programming within the visual arts and data visualization fields. Several artists have used Processing to model or simulate various biological phenomena. One artist animated a matrix of biological cells switching between two states (Polyptic by George Legrady), while another simulated fungal hyphae growth using images as food (Mycelium by Ryan Alexander). A textbook further describes how Processing can animate dynamic processes in nature using diverse underlying theoretical models (Shiffman 2012). However, Processing has yet to make its full mark in modeling of biological phenomena. One potential area of application for Processing is computational visualization of morphogenesis in developmental biology.

Morphogenesis - the generation of shape and form in embryonic tissues - is a critical process in developmental biology (Gilbert 2013). Morphogenesis is responsible for elongating the body axis, forming the organs in the body, and establishing neuronal circuitry in the brain, among other processes. A central issue in the study of morphogenesis is determining the cell behaviors (e.g. motility, rearrangement, shape change, and division) that are responsible for changing tissue shapes. A powerful technique for elucidating the dynamic cellular mechanisms of morphogenesis is time-lapse video-microscopic imaging of cells in embryonic tissues undergoing morphogenesis (Elul and Keller 2000; Elul et al., 1997; Harris et al., 1987). The mechanisms underlying morphogenetic processes can be clarified by observation of cell dynamics from these time-lapse video sequences. Morphometric measurement further defines the quantitative and statistical relationships between the cell dynamics that underlie morphogenesis (Kim and Davidson 2011; Marshak et al., 2007; Elul et al., 1997; Witte et al., 1996). Mathematical decomposition of cell dynamics additionally facilitates the computational modeling of cellular mechanisms that drive morphogenesis (Satulovsky et al., 2008).

Methods

In this paper, we use the Processing programming language to visualize dynamic cell behaviors driving morphogenesis in the developing nervous system. Based on *in vivo* time-lapse image sequences, we created models of cell dynamics underlying two morphogenetic processes in the developing nervous system. We first computationally modeled the cell motility that drives neural tube formation and spinal cord elongation in vertebrate embryos. The terminal branching of optic axonal arbors that establish synaptic connectivity in the primary visual projection was then simulated. These visualizations highlight the process of computational decomposition of the dynamic cell behaviors that shape the developing nervous system, which may be useful for artists seeking to illustrate the development of form in biology.

Results

Modeling of Cell Motility Driving Neural Ectoderm Morphogenesis

Our first Processing model simulated the cell motility underlying neural tube formation and spinal cord elongation. During development, the spinal cord is formed by a series of morphogenetic processes that transform the flat neural plate into the closed neural tube. Disruptions of these morphogenetic processes result in neural tube defects that can severely affect nervous system function. One morphogenetic process critical to neural tube formation is 'convergent extension', in which the neural ectoderm simultaneously extends its antero-posterior axis and narrows its medio-lateral axis. Researchers have used time-lapse imaging and quantitative analysis to study the cellular mechanisms of neural convergent extension in embryos of diverse species, including newt (Burnside and Jacobson 1968), frog (Elul and Keller 2000; Elul et al., 1997), zebrafish (Hong and Brewster 2006), chick (Ezin et al., 2009), and mouse (Williams et al., 2014). The conclusion from these studies is that the embryonic spinal cord tissue elongates by neural cells intercalating (rearranging) along the mediolateral axis (Figure 1a). Some studies further demonstrate that during convergent extension, neural cells express mediolaterally-oriented or randomly-directed dynamic protrusions, which they apply to one another to generate force for their rearrangement (Figures 1a and 1b; Elul and Keller 1999).

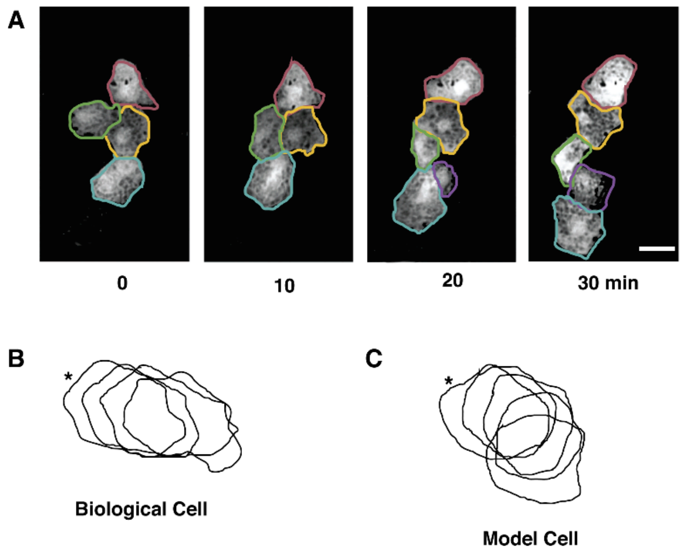


Figure 1. *(A)Time series of still images depicts a small group of fluorescently-labeled neural cells intercalating in a neural ectoderm explant. In the images depicted, the green cell moves between the yellow and blue cells, and the purple cell then moves between the green and blue cells. This results in elongation of the group of cells along the antero-posterior axis of the tissue, which is oriented vertically. Scale Bar – 10 µm.*
(B)Contours of a biological neural ectoderm cell shows how its shape (protrusions) and position change over time, during 2 minute intervals (represents cell at 0 min).*
(C) Contours of the model motile cell during iterations of the program, shows how its shape and position changes as it moves towards the cursor (represents cell at start of program).*

The computational model was based on a time-lapse image sequence of a small group of fluorescently-labeled motile neural cells intercalating in an explant undergoing convergent extension (Figure 1a; Supplementary Video 1; Elul and Keller 1999). Our Processing program focused on animating the motility of a single two-dimensional neural cell within this group (Figures 1b and 1c; Supplementary Video 1). Future versions of the model will attempt to simulate the more complex rearrangements and interactions of a group of motile neural cells. To model the dynamics of an individual neural cell, we incorporated several shape and motility parameters (Figures 1b and 1c; Supplementary Code 1). We determined that a slightly off-circle polygon, with eight nodes or vertices, was a suitable representation of neural cell shape. Using the 'mouse' command in Processing, we set the cursor to be an external force, attracting the cell. We then modeled the change in position of the cell center and nodes in response to the cursor position. The 'acceleration' of the cell was linearly proportional to the distance between the cell and the cursor, becoming stronger the farther away the cell was from the cursor. 'Springing' and 'damping' parameters were added to positively and negatively modulate the acceleration of the cell, respectively. In each loop of the program, the new cell center position was determined as a function of the acceleration, while new node positions were established based both on the acceleration, and on a specific random number or 'frequency,' assigned initially to each node. This resulted in each of the nodes moving somewhat independently from the other nodes of the cell. In addition, during movement of the cell, the nodes rotated through an angle, mimicking the rotational motion of the neural ectoderm cells in the time-lapse sequence. Finally, we established an organic constant parameter which was also dependent on the acceleration (distance between cursor and cell center). The organic constant modulated the curvature or convexity of the polygon sides as the cell moved towards the cursor. We created analogous shape and motility parameters for the nucleus of our model cell, which in the future, could allow us to explore nuclear-cellular motility relationships.

Visualization of Optic Axon Branching in the Developing Visual System

We also modeled the terminal branching of optic axonal arbors that form the primary visual connection and establish visual function. During development of the nervous system, axons extend to their target tissues in the brain, where they branch into arbors that contain synaptic junctions. Axonal arbors are dynamic structures, with remodeling of branches and synapses occurring throughout life, in correlation with cognitive processes such as learning and memory (Meyer and Smith 2006). Optic axonal arbors that establish visual connectivity are peripherally located in the brain, and can be imaged in intact, living animals, including lower vertebrates such as tadpoles of frogs or fish (Figure 2a; Marshak et al., 2009; Witte et al., 1996; Harris et al., 1987). From *in vivo* time-lapse videos of optic axonal arbors, previous researchers quantified the rates of branch additions and retractions, and the mean lifetimes of branches that underlie the remodeling of these arbors (Figure 2a; Marshak et al., 2007; Witte et al., 1996). Another study measured and modeled the mean branching angle of axon arbors *in vitro* (Shefi et al., 2005). Studies showed that molecular perturbations can misshape optic axonal arbors by altering the rates of addition, retraction and lifetime of arbor branches (Wiley et al., 2008; Elul et al., 2003). Some of these molecular perturbations of the optic axonal arbor dynamics have also been associated with disruptions of visual synaptic function (Mannitt et al., 2009; Ruthazer et al., 2006; Alsina et al., 2001).

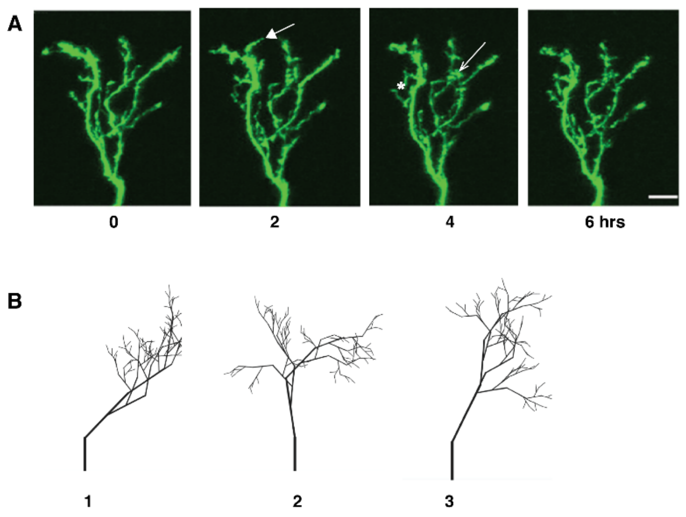


Figure 2. *(A) Time series of images of an optic axonal arbor in the brain of an intact, living tadpole. In the images shown, new branches are added (arrow), existing branches elongate (open arrow), and branching angles change (*). In addition, parent branches are thicker than daughter branches. Scale Bar – 10 µm (printed with permission from Marshak et al., 2007). (B) Sequential images show remodeling of model arbor in three subsequent iterations of the program (number of daughter branches set at two, maximal branching angle set at 60°, based on images of biological arbor shown in A). As the arbor grows, new daughter branches are added, angles between the branches are changed, and parent branches grow longer and thicker.*

The model of axonal arborization was based on a time series of images of an optic axon branching in the tectum of an intact tadpole (Figure 2; Supplementary Code 2, Marhsak et al., 2007). The Processing sketch began with a single vertically oriented 'optic axon' (Figure 2b). In each loop of the program, the terminal branches split into additional daughter branches (Figure 2b; Supplementary Video 2). The parameters in this model included the number of daughter branches a branch bifurcated into, the maximal angle the daughter branches made with the parent branch, and the lengths and thicknesses of the parent and daughter branches. We also observed that as the arbor grew its' branching angles changed. Therefore, in each loop of the program, the model branching angle was assigned a random number between 0 and the maximal branching angle. This provided a stochastic component to an otherwise deterministic model. In agreement with the images of optic axon arbors, model parent branches increased in thickness over time, and newly formed daughter branches were shorter and thinner than the parent branches (Figure 2). An additional feature of this model was the incorporation of a dialogue box that allowed the user to modify the values of the number of daughter branches and the maximal branching angle of the arbor. This allowed the user to easily visualize, for example, how changing the number of daughter branches from two to three, or the maximal branching angle from 60° to 30°, would alter the overall arbor morphology (Figure 2b). Such user modification of the program could help researchers visually depict the effects of molecular perturbations on particular features of optic axon arbor development.

Discussion

The spatial and temporal resolution we depicted in our Processing animations of cell and arbor dynamics were determined, in part, by the resolution of the underlying time-lapse image sequences. We sketched the motile neural cell in real time (at a frame rate of 25), based on time-lapse image sequences captured at relatively high temporal resolution (30 second intervals; Figure 1a; Supplementary Video 1; Elul and Keller 2000; Elul et al., 1997). In contrast, we modeled axonal branching at a more discrete time scale (frame rate of 1), based on a series of images of optic axons captured at two hour intervals (Figure 2a; Supplementary Video 2, Marhsak et al., 2007). Over the last decade, many new microscopic imaging technologies with greatly increased temporal and spatial resolution have been developed (Krzic et al., 2012). Researchers have used one of these new technologies - light sheet microscopy - to capture 4D images of embryonic development in Drosophila and other small organisms *in toto* (Chen et al., 2014). Having such higher-resolution, multi-dimensional time-lapse images as the basis for our Processing simulations would likely increase the temporal and spatial scale of the animations.

Other researchers have developed rules-based models of cell motility and branching dynamics that may also be useful to artists seeking to illustrate morphogenesis. One study modeled cell movement by defining local protrusion and global retraction factors that modify shape and motility parameters of a model cell (Satulovsky et al., 2008). Another group incorporated adhesive, drag, repulsive, and directed migration forces to change the acceleration of individual cells within an epithelial sheet (Vitorino et al., 2011). Computational models and quantitative descriptions of branching morphogenesis in biology are also prevalent. The textbook, "The Algorithmic Beauty of Plants", which is popular within the Processing community, describes 'L-systems'- a fractal-based, recursive model of plant branching and development (Prusinkiewicz and Lindenmayer 1990). In addition, the NeuroMorph database is a repository for thousands of digitized arbor images together with associated morphometric parameters (www.neuromorph.org). Finally, one chapter of the book "Computational Neuroanatomy," focuses on determining the minimum number of parameters necessary to accurately model geometry of neuronal arbors (Burke and Marks 2002).

Additional morphogenetic processes could be modeled using Processing or other visual programming languages. Morphogenetic processes involving cell movement include gastrulation (Skoglund et al., 2008), eye development (Chauhan et al., 2015), and metastasis of cancer cells (Spanjaard et al., 2015). Branching morphogenetic processes that could be modeled encompass lung morphogenesis (Menshykau et al., 2014), establishment of blood vascular networks (Boas and Merks 2015), and branching of tumors (Miura 2015; Takaki 2005). Blender (www.blender.org), an open source 3D animation and graphics software, could also be used to model morphogenesis. Blender would be especially useful for modelling three-dimensional projections of cells and arbors generated with confocal and light-sheet microscopy techniques (Chen et al., 2014; Elul et al., 2003). In more general terms, tools from the arts and technology fields could help biologists and artists create effective visualizations that communicate the dynamics of morphogenesis to a broader audience.

Acknowledgements

The authors thank the artist who suggested they use the Processing language for animating morphogenesis.

References

Alsina, B., Vu, T., and Cohen-Cory, S. 2001. Visualizing synapse formation in arborizing optic axons *in vivo*: dynamics and modulation by BDNF. *Nature Neuroscience,* 4(11):1093-1101.

Boas, S.E., and Merks, R.M. 2015. Tip cell overtaking occurs as a side effect of sprouting in computational models of angiogenesis. *BMC Systems Biology*, 21(9):86.

Burke, R.E., and Marks, W.B. 2002. Some approaches to quantitative dendritic morphology: Computational Neuroanatomy. Totowa, New Jersey: Humana Press.

Burnside, M.B., and Jacobson, A.G. 1968. Analysis of morphogenetic movements in the neural plate of the newt *Taricha torosa*. *Developmental Biology*, 18(6):537-552.

Chauhan, B., Plageman, T., Lou, M., and Lang, R. 2015. Epithelial morphogenesis: the mouse eye as a model system. *Current Topics in Developmental Biology*, 111:375-399.

Chen, B.C., Legant, W.R., Wang, K., Shao, L., Milkie, D.E., Davidson, M.W., Janetopoulos, C., Wu, X.S., Hammer, J.A. 3rd, Liu, Z., English, B.P., Mimori-Kiyosue, Y., Romero, D.P., Ritter, A.T., Lippincott-Schwartz, J., Fritz-Laylin, L., Mullins, R.D., Mitchell, D.M., Bembenek, J.N., Reymann, A.C., Böhme, R., Grill, S.W., Wang, J.T., Seydoux, G., Tulu, U.S., Kiehart,

D.P., and Betzig, E. 2014. Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution. *Science*, 346(6208): 1257998.

Elul, T.M., Kimes, N.E., Kohwi, M., and Reichardt, L.F. 2003. N- and C-terminal domains of beta-catenin, respectively, are required to initiate and shape axon arbors of retinal ganglion cells *in vivo*. *Journal of Neuroscience,* 23(16):6567-6575.

Elul, T., and Keller, R. 2000. Monopolar protrusive activity: a new morphogenic cell behavior in the neural plate dependent on vertical interactions with the mesoderm in Xenopus. *Developmental Biology*, 224(1): 3-19.

Elul, T., Koehl, M.A., and Keller, R. 1997. Cellular mechanism underlying neural convergent extension in *Xenopus laevis* embryos. *Developmental Biology*, 91(2):243-258.

Ezin, A.M., Fraser, S.E., and Bronner-Fraser, M. 2009. Fate map and morphogenesis of presumptive neural crest and dorsal neural tube. *Developmental Biology*, 330(2):221-236.

Gilbert, S. 2013. Developmental Biology, 10th edition. Sunderland, MA: Sinauer Associates, Inc.

Harris, W.A., Holt, C.E., and Bonhoeffer, F. 1987. Retinal axons with and without their somata, growing to and arborizing in the tectum of *Xenopus* embryos: a time-lapse video study of single fibres *in vivo*. *Development*, (1):123-133.

Hong, E., and Brewster, R. 2006. N-cadherin is required for the polarized cell behaviors that drive neurulation in the zebrafish. *Development*, 133(19):3895-3905.

Kim, H.Y., and Davidson, L.A. 2011. Punctuated actin contractions during convergent extension and their permissive regulation by the noncanonical Wnt-signaling pathway. *Journal of Cell Science*, 124(Pt 4):635-646.

Krzic, U., Gunther, S., Saunders, T.E., Streichan, S.J., and Hufnagel, L. 2012. Multiview light-sheet microscope for rapid *in toto* imaging. *Nature Methods*, 9(7):730-733.

Manitt, C., Nikolakopoulou, A.M., Almario, D.R., Nguyen, S.A., and Cohen-Cory, S. 2009. Netrin participates in the development of retinotectal synaptic connectivity by modulating axon arborization and synapse formation in the developing brain. *Journal of Neuroscience*, 29(36):11065-11077.

Marshak, S., Nikolakopoulou, A.M., Dirks, R., Martens, G.J., and Cohen-Cory, S. 2007. Cell-autonomous TrkB signaling in presynaptic retinal ganglion cells mediates axon arbor growth and synapse maturation during the establishment of retinotectal synaptic connectivity. *Journal of Neuroscience*, 27(10):2444-2456.

Menshykau, D., Blanc, P., Unal, E., Sapin, V., and Iber, D. 2014. An interplay of geometry and signaling enables robust lung branching morphogenesis. *Development*, 141(23):4526-4536.

Meyer, M.P., and Smith, S.J. 2006. Evidence from *in vivo* imaging that synaptogenesis guides the growth and branching of axonal arbors by two distinct mechanisms. *Journal of Neuroscience*, 26(13):3604-3614.

Miura, T. 2015. Models of lung branching morphogenesis. *Journal of Biochemistry*, 157(3):121-127.

Prusinkiewicz, P., and Lindenmayer, A. 1990. The Algorithmic Beauty of Plants. New York: Springer-Verlag.

Ruthazer, E.S., Li, J., and Cline, H.T. 2006. Stabilization of axon branch dynamics by synaptic maturation. *Journal of Neuroscience,* 26(13):3594-3603.

Satulovsky, J., Lui, R., and Wang, Y.L. 2008. Exploring the control circuit of cell migration by mathematical modeling. *Biophysical Journal*, 94(9):3671-3683.

Shefi, O., Golebowicz, S., Ben-Jacob, E., and Ayali, A. 2005. A two-phase growth strategy in cultured neuronal networks as reflected by the distribution of neurite branching angles. *Journal of Neurobiology,* 62(3):361-368.

Shiffman, D., Fry, S., and Marsh, Z. 2012. The Nature of Code. Simulating Natural Systems with Processing. Mountain View: Daniel Shiffman.

Skoglund, P., Rolo, A., Chen, X., Gumbiner, B.M., and Keller, R. 2008. Convergence and extension at gastrulation require a myosin IIB-dependent cortical actin network. *Development*, 135(14):2435-2444.

Spanjaard, E., Smal, I., Angelopoulos, N., Verlaan, I., Matov, A., Meijering, E., Wessels, L., Bos, H., and de Rooij, J. 2015. Quantitative imaging of focal adhesion dynamics and their regulation by HGF and Rap1 signaling. *Experimental Cell Research*, 330(2):382-397.

Takaki, R. 2005. Can morphogenesis be understood in terms of physical rules? *Journal of Bioscience*, 30(1):87-92.

Vitorino, P., Hammer, M., Kim, J., and Meyer, T. 2011. A steering model of endothelial sheet migration recapitulates monolayer integrity and directed collective migration. *Molecular Cell Biology*, 31(2):342-350.

Wiley, A., Edalat, K., Chiang, P., Mora, M., Mirro, K., Lee, M., Muhr, H., and Elul, T. 2008. GSK-3beta and alpha-catenin binding regions of beta-catenin exert opposing effects on the terminal ventral optic axonal projection. *Developmental Dynamics*, 237(5):1434-1441.

Williams, M., Yen, W., Lu, X., and Sutherland, A. 2014. Distinct apical and basolateral mechanisms drive planar cell polarity-dependent convergent extension of the mouse neural plate. *Developmental Cell,* 29(1):34-46.

Witte, S., Stier, H., and Cline, H.T. 1996. *In vivo* observations of timecourse and distribution of morphological dynamics in Xenopus retinotectal axon arbors. *Journal of Neurobiology,* 31(2):219-234.

Authors

Avik Patel, BSc, is a third-year medical student at Touro University California. He received his Bachelor of Science in Neurobiology from UC Davis. His interests include visualization of anatomy and biochemical processes. Avik was excited to learn Processing and apply it to the modeling of optic axonal arbors *in vivo*.

Amar Bains, BA, recently graduated from the University of California, Berkeley with a biochemistry degree. He is pursuing a Master's degree post-graduation, and plans to eventually attend medical school. In his free time, he practices the tuba and dances Bhangra for UC Berkeley's official team.

Richard Millet is a programmer based in the San Francisco Bay Area. He has worked for Apple, and in the Research Information Technology group at UC Berkeley. Currently, he is the technical lead for CollectionSpace, an open source software for Museum Databases.

Tamira Elul, PhD, is a scientist and educator. She is an Associate Professor at Touro University, California where she teaches medical students histology and directs an experimental and theoretical research program focusing on neuronal morphogenesis. In recent years, she has used art and artistic tools to disseminate morphogenesis to a broader community. Tamira Elul can be contacted at tamira.elul@tu.edu.

Supplementary Video

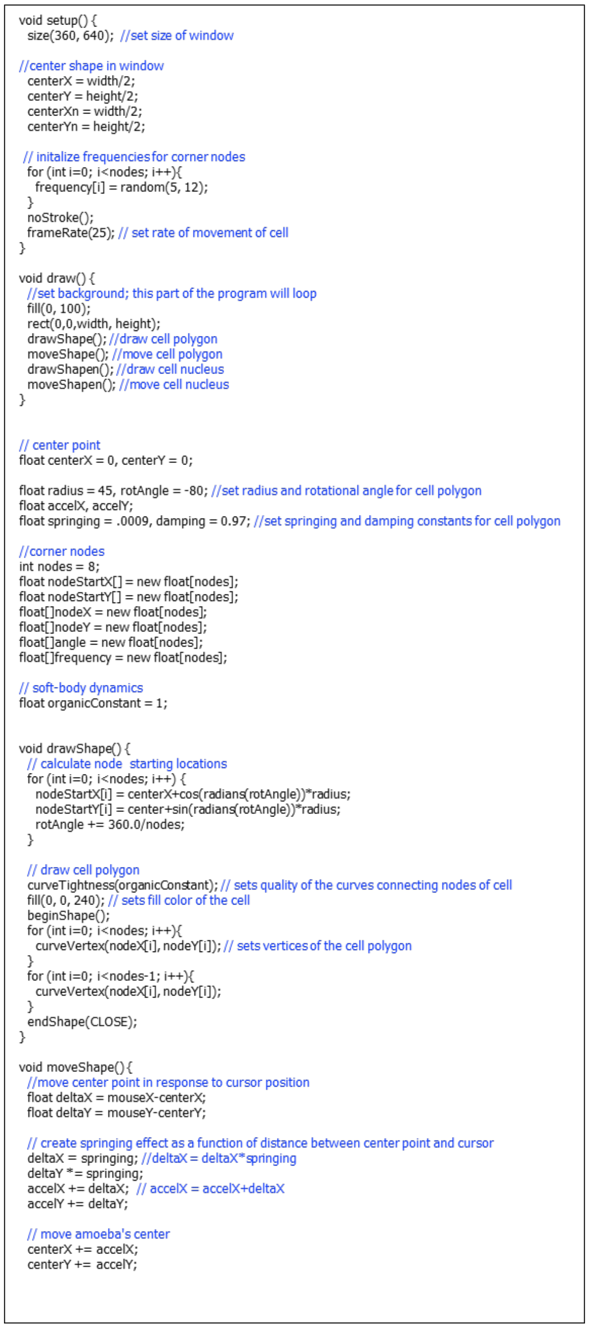
Video 1 (video link also available in the HTML)
Animation shows computational cell protruding and deforming as it moves towards the cursor as an 'attractive force' (left panel). Note change in curvature of sides of the model cell as it moves towards the cursor. A video (which loops 5 times) depicting biological neural ectoderm cells moving and intercalating is also shown (right panel, time interval between frames is 30 seconds).

Video 2 (video link available in the HTML)

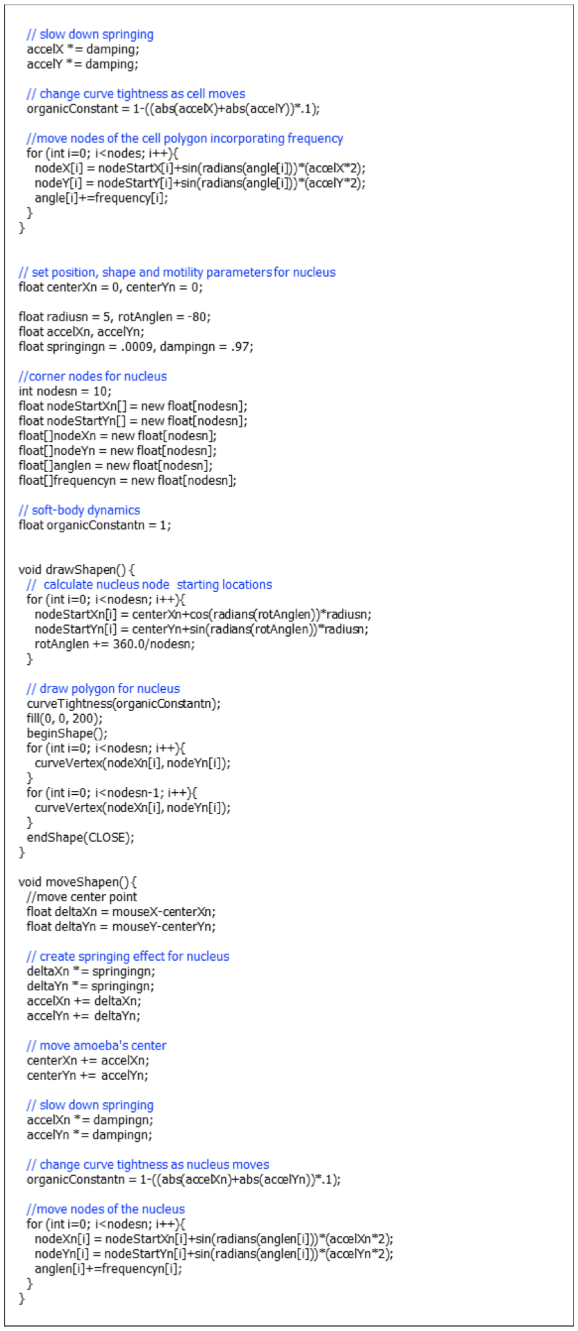
Animation shows model axonal arbor branching and growing, with two new daughter branches being added to each parent branch in each iteration of the program, and a maximal branching angle of 60°. These parameters are based on analysis of biological arbors, as shown in Figure 2A (Marshak et al., 2007).

Supplementary Code 1

Model Neural Cell Motility



```
void setup() {
  size(360, 640);  //set size of window

//center shape in window
  centerX = width/2;
  centerY = height/2;
  centerXn = width/2;
  centerYn = height/2;

 // initalize frequencies for corner nodes
  for (int i=0; i<nodes; i++){
    frequency[i] = random(5, 12);
  }
  noStroke();
  frameRate(25); // set rate of movement of cell
}

void draw() {
  //set background; this part of the program will loop
  fill(0, 100);
  rect(0,0,width, height);
  drawShape(); //draw cell polygon
  moveShape(); //move cell polygon
  drawShapen(); //draw cell nucleus
  moveShapen(); //move cell nucleus
}


// center point
float centerX = 0, centerY = 0;

float radius = 45, rotAngle = -80; //set radius and rotational angle for cell polygon
float accelX, accelY;
float springing = .0009, damping = 0.97; //set springing and damping constants for cell polygon

//corner nodes
int nodes = 8;
float nodeStartX[] = new float[nodes];
float nodeStartY[] = new float[nodes];
float[]nodeX = new float[nodes];
float[]nodeY = new float[nodes];
float[]angle = new float[nodes];
float[]frequency = new float[nodes];

// soft-body dynamics
float organicConstant = 1;


void drawShape() {
  // calculate node  starting locations
  for (int i=0; i<nodes; i++) {
    nodeStartX[i] = centerX+cos(radians(rotAngle))*radius;
    nodeStartY[i] = center+sin(radians(rotAngle))*radius;
    rotAngle += 360.0/nodes;
  }

  // draw cell polygon
  curveTightness(organicConstant); // sets quality of the curves connecting nodes of cell
  fill(0, 0, 240); // sets fill color of the cell
  beginShape();
  for (int i=0; i<nodes; i++){
    curveVertex(nodeX[i], nodeY[i]); // sets vertices of the cell polygon
  }
  for (int i=0; i<nodes-1; i++){
    curveVertex(nodeX[i], nodeY[i]);
  }
  endShape(CLOSE);
}

void moveShape() {
  //move center point in response to cursor position
  float deltaX = mouseX-centerX;
  float deltaY = mouseY-centerY;

  // create springing effect as a function of distance between center point and cursor
  deltaX = springing; //deltaX = deltaX*springing
  deltaY *= springing;
  accelX += deltaX;  // accelX = accelX+deltaX
  accelY += deltaY;

  // move amoeba's center
  centerX += accelX;
  centerY += accelY;
```



```
  // slow down springing
  accelX *= damping;
  accelY *= damping;

  // change curve tightness as cell moves
  organicConstant = 1-((abs(accelX)+abs(accelY))*.1);

  //move nodes of the cell polygon incorporating frequency
  for (int i=0; i<nodes; i++){
    nodeX[i] = nodeStartX[i]+sin(radians(angle[i]))*(accelX*2);
    nodeY[i] = nodeStartY[i]+sin(radians(angle[i]))*(accelY*2);
    angle[i]+=frequency[i];
  }
}


// set position, shape and motility parameters for nucleus
float centerXn = 0, centerYn = 0;

float radiusn = 5, rotAnglen = -80;
float accelXn, accelYn;
float springingn = .0009, dampingn = .97;

//corner nodes for nucleus
int nodesn = 10;
float nodeStartXn[] = new float[nodesn];
float nodeStartYn[] = new float[nodesn];
float[]nodeXn = new float[nodesn];
float[]nodeYn = new float[nodesn];
float[]anglen = new float[nodesn];
float[]frequencyn = new float[nodesn];

// soft-body dynamics
float organicConstantn = 1;


void drawShapen() {
  //  calculate nucleus node  starting locations
  for (int i=0; i<nodesn; i++){
    nodeStartXn[i] = centerXn+cos(radians(rotAnglen))*radiusn;
    nodeStartYn[i] = centerYn+sin(radians(rotAnglen))*radiusn;
    rotAnglen += 360.0/nodesn;
  }

  // draw polygon for nucleus
  curveTightness(organicConstantn);
  fill(0, 0, 200);
  beginShape();
  for (int i=0; i<nodesn; i++){
    curveVertex(nodeXn[i], nodeYn[i]);
  }
  for (int i=0; i<nodesn-1; i++){
    curveVertex(nodeXn[i], nodeYn[i]);
  }
  endShape(CLOSE);
}

void moveShapen() {
  //move center point
  float deltaXn = mouseX-centerXn;
  float deltaYn = mouseY-centerYn;

  // create springing effect for nucleus
  deltaXn *= springingn;
  deltaYn *= springingn;
  accelXn += deltaXn;
  accelYn += deltaYn;

  // move amoeba's center
  centerXn += accelXn;
  centerYn += accelYn;

  // slow down springing
  accelXn *= dampingn;
  accelYn *= dampingn;

  // change curve tightness as nucleus moves
  organicConstantn = 1-((abs(accelXn)+abs(accelYn))*.1);

  //move nodes of the nucleus
  for (int i=0; i<nodesn; i++){
    nodeXn[i] = nodeStartXn[i]+sin(radians(anglen[i]))*(accelXn*2);
    nodeYn[i] = nodeStartYn[i]+sin(radians(anglen[i]))*(accelYn*2);
    anglen[i]+=frequencyn[i];
  }
}
```

Supplementary Code 2
Model Axonal Branching

```
float x=0;
float q=255;
float w=1;
float e=255;
import javax.swing.JOptionPane; //Dialog boxes;
int a;
int b;
void setup(){
 size(600,800); //Set canvas size;
 String input1 = JOptionPane.showInputDialog("input initial branch number (a), a={1,2,3,4}");
 a = Integer.parseInt(input1); //Allow user to choose the number of initial branches;
 String input2 = JOptionPane.showInputDialog("input maximum branching angle (b), (PI/b)");
 b = Integer.parseInt(input2); //Allow user to define the absolute maximum branching angle;
}
void draw(){
  smooth();
  frameRate(1); //Rate of arbor remodeling is set;
  loop(); //Program will loop continuously;
  newProjection();
}
void mousePressed(){
  noLoop();
  redraw(); //Allow user to 'pause' with a cursor click;
}
void newProjection(){
  background(10);
  q=q/10;
  w=1.5*w;
  e=e/10;
  stroke(q,w,e); //Color formatting. Color is dynamic;
  text("click to pause", 10, height-10);
  translate(width/2,height);
  branch(x); //Branch() is a function that packages multiple 2D transformations;
  x=x+2;
  if(x>height/6){
    x=height/6; //Set vertical growth boundary. Keeps visualization in the frame;

    }
}
void branch(float h){
  float sw=map(h,1,200,1,5);
  strokeWeight(sw); //Line thickness increases over time but daughter branches are thinner than
parent branches;
  float theta3=random(-PI/b,PI/b); //Randomly generate values for angles in real-time.
Represented by fractions of Pi;
  line(0,0,0,-h); //Represent the initial projection as a single line;
  translate(0,-h); //Translate to the end of the line;
  h *=0.7; //Daughter branch's length will be a fraction of the parent branch's length;
  if (h>10){ //Once a certain length is achieved, the following transformations occur...;
    if (a==1){
    pushMatrix(); //Save the transformation. Display what has been done so far;
    rotate(theta3); //Rotate branches by a random angle;
    branch(h); //Draw rotated branches at the distal ends of the parent branches;
    popMatrix();}; //Restore the transformation matrix. Ready to start process over;
    if (a==2){
    pushMatrix();
    rotate(theta3);
    branch(h);
    branch(h);
    popMatrix();};
    if (a==3){
    pushMatrix();
    rotate(theta3);
    branch(h);
    branch(h);
    branch(h);
    popMatrix();};
    if (a==4){
    pushMatrix();
    rotate(theta3);
    branch(h);
    branch(h);
    branch(h);
    branch(h);
    popMatrix();};
  }
}
```

Licensing