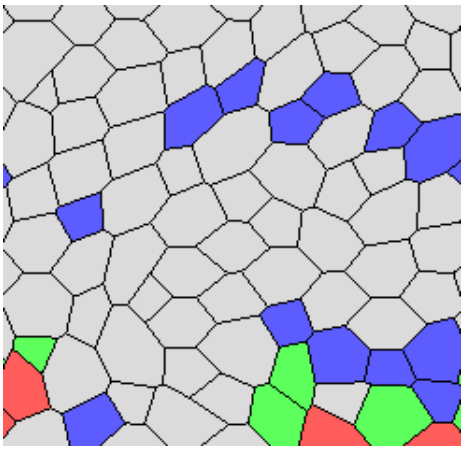
The Ohio State University

Epithelium

The Lightweight, Customizable, Epithelial Tissue Simulator

Author:
Melvyn Ian Drag

Advisor:
Dr. Marcos Manuel Sotomayor
Dr. Edward Overman



March 20, 2015

Contents

Abstract

Epithelial tissue is introduced, current models of epithelial development are described, and then a new implementation of a long existing model is presented. The code for the model is thoroughly examined and the algorithms and data structures underlying it are presented. Some topological insights are made about the nature of epithelial tissue, and some discussion of which numerical methods are used is made.

Chapter 1

Why Study Epithelial Tissue?

INSERT A QUOTE ABOUT EPITHELIAL TISSUE, THE INTERESTING DETAILS IN THE WORLD, ETC.

Imagine a very small and curious bug which alights upon your body and begins to explore. He walks all over your abdomen, chest, and neck and eventually finds your mouth. Being the curious bug he is, he enters and walks past your teeth and down your esophogas and swims through your stomach, strolls along the lining of your intestine and continues until he finds an exit. He may make the trip again, but this time he walks past your teeth and down your trachea and into your lungs, meandering through all of the branching alveoli which are the ends of your bronchi. One might think that this curious bug has gotten to know you and your body quite well by this time, but the fact of the matter is that the bug has learned about nothing more than your epithelial cells. Epithelial tissue forms the boundary between the outside world and the interior of your body, separating your muscles and organs from air, water, food, toxins, dirt, and any other substance. In this way, we can think of the human body to be topologically equivalent to a torus and let the surface of the torus be covered by epithelial tissue.

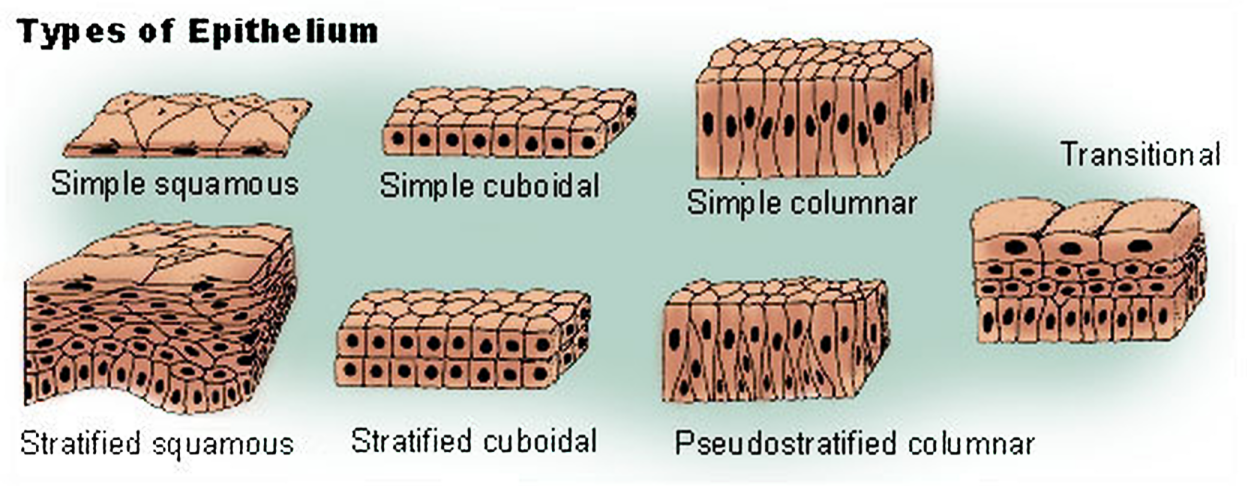


Figure 1.1: The Types of Epithelial Tissue

a new field: Types of epithelial cells. As of now, the tissue is looked at as being different types INSERT graphic.

Chapter 2

Overview of the Model

The world was so recent that many things still lacked names,
and to mention them one had to point with a finger.
- Gabriel Garcia Marquez

2.1 Introduction

A two dimensional **vertex dynamics model** of epithelial tissue is made up of vertices and edges. A cell is represented as a convex hull made up of cells and vertices. This model presupposes that the movement of cells in epithelial tissue can be approximated by the movement of the vertices which make up the cell. Some force is hypothesized to be the guiding force between epithelial cell movement, and this force is applied to all of the vertices in the mesh of cells, producing some final state of the tissue. The three dimensional model is an extension of the two-dimensional model, but the mesh comes to include no only vertices and edges, but also cell faces.

Epithelial vertex dynamics has been a lively field of research since the 1970s because of several heartening resutls in the field. Some researchers have had success modelling the morphogenesis of *Drosophila* wing growth, whereas other researchers have successfully reproduced the dynamics of corneal wound healing. Unfortunately, these results have not come from one standard force, but froma variety of different hypothesized forces. For example, the Weliky-Oster model specifies explicit forces which act act upon the vertices due to tension between neighboring vertices. The Honda-Nagai Model (which I have recreated) instead specifies a potential energy function for the tissue, and the cells move in such a way as to minimize the potential energy.

2.2 The Nagai-Honda Model

2.2.1 How the Vertices Move

A very basic result from physics is the relationship between force and potential energy, and this is one of the building blocks of the Nagai-Honda Model.

$$\vec{F} = (F_x, F_y, F_z) \tag{2.1}$$

$$W = -\Delta U(\vec{x}) = \int_{x_0}^{x} F_x dx + \int_{y_0}^{y} F_y dy + \int_{z_0}^{z} F_z dz \tag{2.2}$$

Figure 2.1: The Giant's Causeway

$$\nabla(-\Delta U(\vec{x}) = \nabla\left(\int_{x_0}^{x} F_x dx + \int_{y_0}^{y} F_y dy + \int_{z_0}^{z} F_z dz\right) \tag{2.3}$$

$$-\nabla U(\vec{x}) = \vec{F} \tag{2.4}$$

The second building block is the equation of motion. In 1989, K. Kawaski showed that the dynamics of grain growth can be reduced to a first order system given by:

$$\eta \frac{dr_i}{dt} = F_i \tag{2.5}$$

where F_i denotes the force applied to vertex i and the left hand side is the velocity of the vertex multiplied by a positive drag coefficient. In practice, the drag coefficient is absorbed into the step size during the numerical integration.

A very important paper in the field of epithelial tissue, grain growth, and soap froth development modeling is *Soap, Cells, and Statistics*, in which the leading researchers in the field argue that there mush be some natural mechanism underlying the aforementioned phenomena, as well as other, related occurrences. The authors noticed that, some small details aside, the development of epithelial tissue cells, the drying of columnar basalt, the appearance of soap bubbles when pressed between plates, and the development of crystals at high temperature all result in nearly identical geometric structures. For example, consider the images of epithelial tissue presented throughout this paper next to the image of The Giant's Causeway in Northern Ireland. Building upon increasing evidence that the underlying mechanism which produces these polygon meshes is the same or very similar, H. Honda and T. Nagai formulated their model using the previous result about grain growth.

Putting these building blocks together, we have a force acting on the vertices in the mesh, and an equation of motion which explains how the vertices move. The free energy function which provides the apllied force was taken in part from the grain growth model, which contributed the deformation energy and the membrane surface energy. In addition, the model was extended to include a purely biological factor, *differential adhesion*. It is well known that cells have a proclivity to attach tightly to certain cells and do try to let go of other types of cells due to the presence of certain cadherin proteins in the membranes of the cells. This tendency is called differential adhesion.

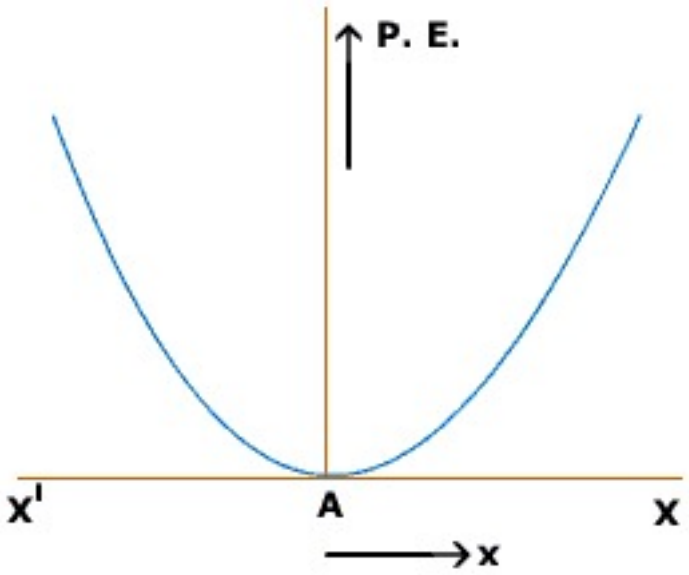


Figure 2.2: Potential Energy as a Function of Distance from Equilibrium.

The first two energy terms assume that the cell is elastic, and that the cell wants to return to a target shape. Therefore, the first two energy terms are of the form:

$$C(x - x_0)^2 \tag{2.6}$$

where x is some physical quantity. The plot of this energy is therefore a parabola with a minimum at $x = x_0$, and the farther x is from x_0, the more potential energy there will be in the cell. The last energy term is an adhesion energy, which is proportional to the amount of surface area that is in contact with another surface. Here are the three energy terms:

1. The deformation energy term
   $U_D = \lambda(A - A_0)^2$
   where A_0 is a target area for a cell, and lambda is some positive constant.

2. The membrane surface energy
   $U_S = \beta(C - C_0)^2$
   where C is the cell perimiter, and C_0 is a target perimeter.

3. The cell-cell adhesion energy
   $U_A = \sum_{j=1}^{n} \gamma_j d_j$
   where n is the number of vertices in the cell, γ is some constant for the boundary in question between one cell and another, and d is the distance between one vertex and the next in a counter clockwise fashion.

As we have already mentioned, the *gradient* of this free energy is what gives us the force acting on each vertex i.

As seen in [**?**], this force on each vertex i is given by:

$$F_i = -\sum_{l \in N_i} (2\lambda(A_l - A_{0_l})\nabla_i A_l + 2\beta(C_l - C_{0_l})(\nabla_i d_{l,I_l-1} + \nabla_i d_{l,I_l}) + \gamma_{l,I_l-1}\nabla_i d_{l,I_l-1} + \gamma_{l,I_l}\nabla_i d_{l,I_l} \tag{2.7}$$

where l is the lth cell containing vertex i, given a counter clockwise orientation. I_l is the local index of node i in element l.

Furthemore, we have to specify the area and vertex-distance gradients,since we will have to use the chain rule in calculating derivatives. The area of a cell is given by Gauss's Shoelace Formula:

$$A = \frac{1}{2}\Big|\sum_{i=1}^{N}\Big(x_i y_{i+1} - x_{i+1}y_i\Big)\Big| \tag{2.8}$$

where N+1 = 1. The gradient is given by:

$$\nabla_i A_l = \frac{1}{2}\Big\langle y^l_{I+1} - y^l_{I-1},\ \ x^l_{I-1} - x^l_{I+1}\Big\rangle \tag{2.9}$$

where the superscrpits l denote that x, y are in cell l. The subscripts are local indices in the cell l, and the orientation of vertices is counterclockwise. The circumference is given by:

$$C = \sum_{j=1}^{N} d_j = \sum_{j=1}^{N}\sqrt{(x_{j+1}-x_j)^2 + (y_{j+1}-y_j)^2} \tag{2.10}$$

Therefore

$$\nabla_i C = \nabla_i d_{i-1} + \nabla_i d_i \tag{2.11}$$

and

$$\nabla_i d_{l,j} = \frac{1}{d_{l,j}}\Big\langle x_{j+1} - x_j,\ \ y_{j+1} - y_j\Big\rangle \tag{2.12}$$

2.2.2 Topological Changes to the Mesh

It is an empirical truth that three cells nearly always meet at any vertex in an epithelial tissue. This has led some researchers to look for some relationship between Voronoi diagrams or Dirichelt domains and epithelial tissue, since these interesting diagrams also form convex polygons which meet at vertices in groups of three. Since the coordination number of the average vertex is three, the model is also intested in what sorts of topological changes can occur to the tissue without changing the topology. As it turns out, there are three.

Epithelial tissue typically undergo three types of toplogical changes. The first is called a T1 swap and in illustrated in Figure **??**. This is also called a "neighbor exchanging swap" because, as you can see two cells cease to be neighbors and the other two cells become neighbors. The T1 swap occurs when two vertices become critically close to each other, and instead of allowing the force to drive the vertices into each other we rotate their edge by 90 degree. The second topological change is the T2 swap, which is also known as the "cell removal swap." This occurs when a triangular cell becomes too small and is deleted and replaced by a single vertex. The last swap is cell division, occasionally referred to as the T3 swap.

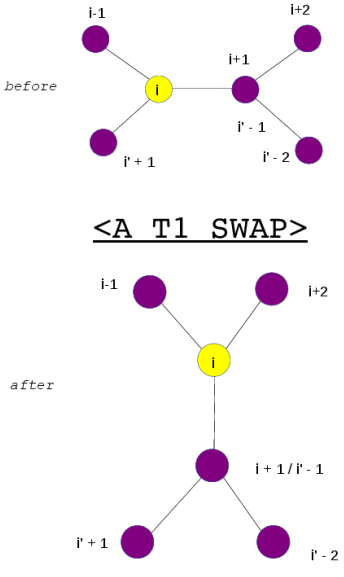


Figure 2.3: A T1 Swap

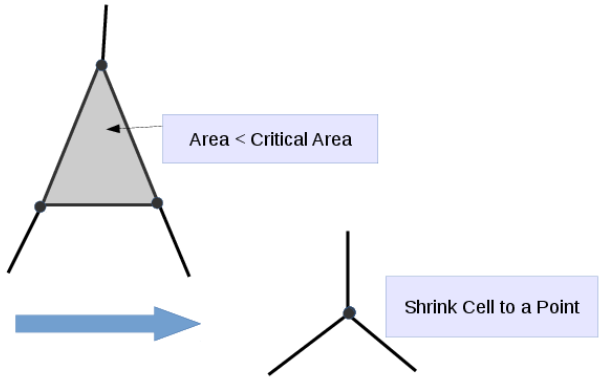


Figure 2.4: A T2 Swap

Cell division was not a part of the original Honda-Nagai Model, but it is now a very hot topic in computational tissue morphogenesis and deserves mention. The challeng with implementing the T3 swap is that there are infinitely (within the bounds of floating point arithmetic) many choices about where to divide a cell, and there are several competing opinions (though no unanimouslly accepted theory) about how the dvision is oriented. Cell division occurs by placing two new vertices in the interior of opposite edges and then connecting them by a new edge. It is not clear which edges ought to have vertice implanted, or where to insert these vertices. There will be more discussion of cell division in the sectin of the paper dealing with the technical details of the implementation of *Epithelium*.

2.3 Why the Specified Topology Is Not Perfect

The choice to impose degree three on each vertex is not one hundred percent consistent with nature, but has been a part of most models of epithelial tissue. This is because empirical evidence shows that the *majority* of vertices in a tissue will have out degree three. It is a simplifying assumption that *rosettes* (epithelial cells organized radially about one vertex which has degree greater than or equal to 4) do not change the global dynamics of the development of an epithelial tissue. Recent computer vision developments [**?**] have made it easier to detect rosettes in epithelial tissue samples and may in the future provide information about the number of these formations, or evidence that rosettes are an important feature of epithelial tissue.

2.4 Feedback Mechanisms and Proliferation

This model is limited in that includes neither mechanical nor chemical feedback, and does not specify how cells proliferate. These are two extensions which can and have been made to the model by certain researchers. On the other hand, the dynamics of some successful models such as [**?**] have the patterning of cells specified entirely by morphogens. In these models, chemicals called 'morphogens' are what drive cells to proliferate, and the low concentrations of the morphogen on the fringe of the tissue explain why the tissue eventually ceases to grow as the tissue reaches a certain size. Other models, such as those described in [**?**] are based upon the vertex dynamics models such as the Honda-Nagai model, but include additional mechanical feedback terms which

limit the growth of cells.

2.5 Additional Curiosities

The vast majority of models assume that exactly three cells meet at any vertex, except vertices on the boundary. In the language of topology, one would say that all interior vertices in the tissue have degree three. Empirical observations show, however, that more more than three cells can meet at any junction under certain circumstances [**?**]. This means that models ought to be extended to include the formation of these 'rosettes'.

Chapter 3

Further Remarks About Epithelial Tissue and the Honda-Nagai Model

3.1 The Euler Characeteristic and Its Implications

In this section I will expand upon some observations made in [**?**]. **Euler's Formula** is an equation which relates the number of edges, faces, and vertices in a graph or polyhedron. An invariant χ relates the faces, edges, and vertices as follows:

$$\chi = V - F + E \tag{3.1}$$

The invariant depends upon the graph or polyhedron in question. We will ignore the exact value of χ and comfort ourselves with the fact that it is a constant. The majority or current vertex dynamics models assume that vertices will have a coordination number of 3, and are based upon empirical results from biology. While there are certain models which consider *rosettes*, for now we will assume that all vertices connect to exactly three other vertices. Then we notice that all edges have two vertices, and that all vertices are connected to three edges. Inititally, our intuition tells us that there should be three times as many edges as vertices, which leads us to the incorrect:

$$3V = E \tag{3.2}$$

But then we notice that if we consider all of the vertices in the mesh, we count each edge twice, so we divide the conjectured number of edges by two and then simplify to get:

$$3V = 2E \tag{3.3}$$

Similarly, if we consider how to relate the number of edges to the faces in the mesh, we conjecture that the number of edges in equal to the number of cells with 3 sides plus the number of cells with 4 sides plus the number of cells with 5 sides plus . . . each multiplied by the number of edges per cell. THis gives us:

$$\sum_{k=3}^{N} kF_k = E \tag{3.4}$$

Where N is the highest number of edges encountered by any cell in the mesh. But in this way we have again counted all of the edges twice, so the true number of edges must be the summation

above divided by 2. We simplify the equation to :

$$\sum_{k=3}^{N} kF_k = 2E \tag{3.5}$$

Now, we are able to reduce Euler's Forula to one variable using the relationships given above.

$$V - F + E = \chi \tag{3.6}$$

$$\frac{2E}{3} - F + E = \chi \tag{3.7}$$

$$\frac{5E}{3} - F = \chi \tag{3.8}$$

$$\frac{\sum_{k=3}^{N} kF_k}{6} - F = \chi \tag{3.9}$$

$$(\frac{\sum_{k=3}^{N} kF_k}{F} - 6)F = 6\chi \tag{3.10}$$

Biological cells are very small, and an epithelial tissue is composed of many [NUMBER?] cells, so we assume that $F \to \infty$ and then immediately notice that the expression in parentheses must tend to zero as F goes to infinity, or else the left hand side of the above equation will not approach the constant 6χ So we know that the algebraic mean of the number of vertices per face must be 6. Of course we have no reason at this point to assume a distribution which guarantees that the majority of cells in the tissue will have exactly six edges and will not, say consist of a mixture of 5 and 7 sided cells. Nevertheless, empirical evidence shows a strong central tendency in the distribution of cell shapes. Whenever a cell tries to stray from the mean, there are means (such as a T3 swap, described in the next section) of recentering the distribution at 6 sides.

Chapter 4

Epithelium

4.1 About Epithelium

For my Master's Thesis I have implemented the Nagai-Honda Model for epithelial tissue development. My software is easy to install, takes up less than 50Mb, comes in parallel and non-parallel versions (Version 1.0.0, Version 1.0.1) and has very few dependencies which are likely already installed on mac and linux computers.

My code allows users to specify all parameters of interest in a couple of well commented configuration files, can handle simulations of arbitrarily large size, and can generate beautiful animations and plots of epithelial tissue development as well as useful plots of various quantities which are of interest to the researcher.

On top of all of this, the source code is highly modularized and allows for ambitious users to easily extend the code to meet their needs. For example, alternate numerical integrators can easily replace the existing one, new mesh generators can replace the square mesh I have developed, and all data is output in very simple formats(which users with scriting experience can transform to fit into the graphical utilities of their choice). In addition, the cell and vertex classes are well documented and can be extended to output new data, as users may need. What follow are a few graphs to give some flavor of what *Epithelium* can do.

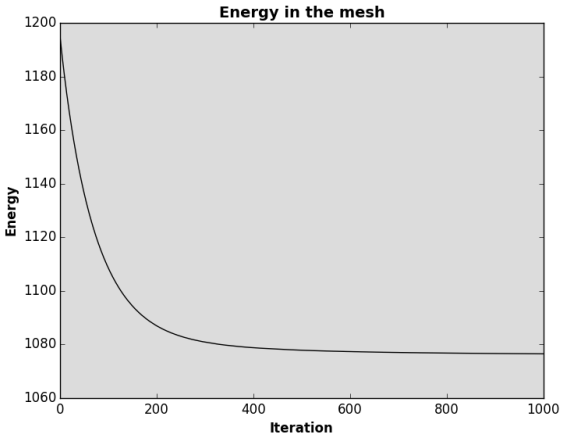


Figure 4.1: A plot of the energy in the system as a function of iteration.

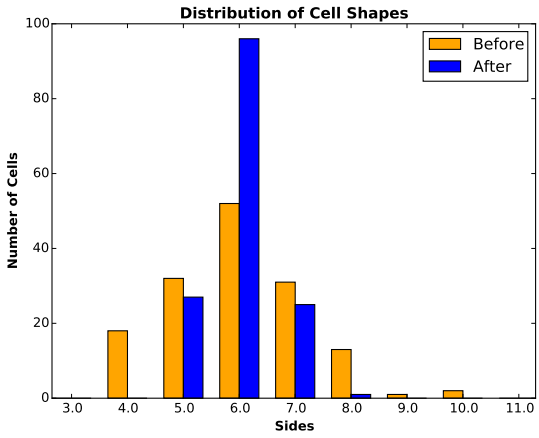


Figure 4.2: A plot of the number of sides cells have.

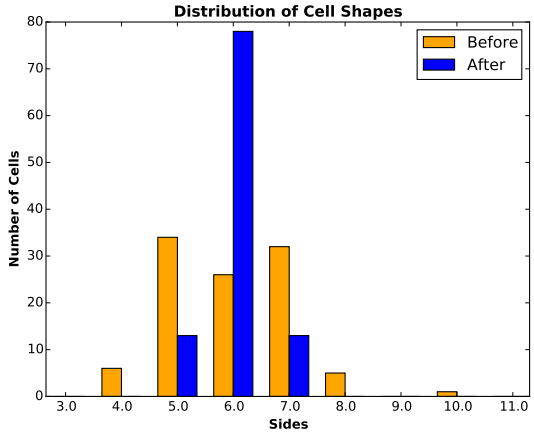


Figure 4.3: The equilibrium distribution of cell shapes given a different initial condition.

4.2 A Comparison of Epithelium to Other Simulators

IN HERE PUT IMAGES FROM OTHER SOFTWARES -¿ Okuda, Chaste, the Paper Yoshi sent.

Mention how Chaste was a step in the right direction, but the dependencies are too many and the it is difficult to get started. Epithelium is lightwiehgt and the code is easy to read, a better introductory tool.

TIGHTEN UP THE GRAPHS.THEY ARE TOO SPREAD OUT. INCLUDE GRAPHS OF INTERIOR ANGLES.

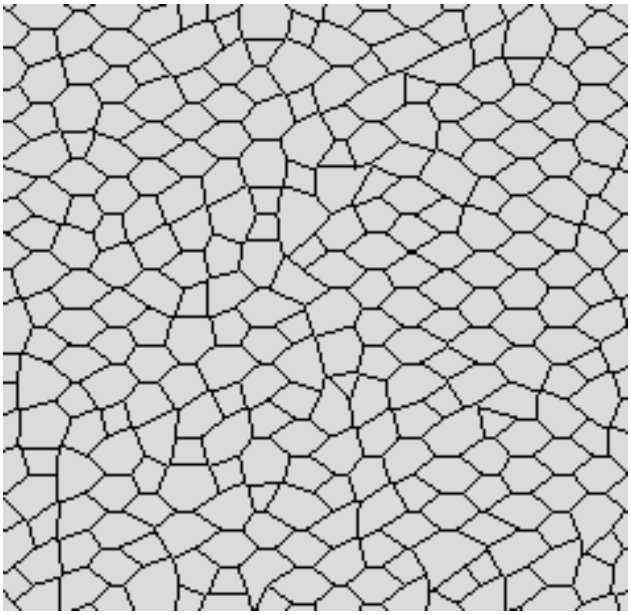


Figure 4.4: A mesh of cells before the application of forces.

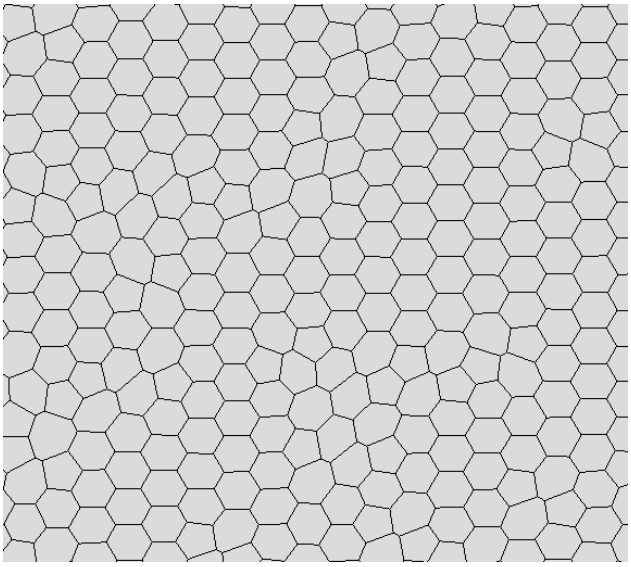


Figure 4.5: A mesh after the application of forces.

Chapter 5

The Design of *Epithelium*

In this chapter I will explain the layout of the code, including the classes used, the data structure and the algorithms which I developed to calculate all of the inter- and intra- cellular forces governing vertex motion. A number of ideas were scrapped in the beginning of the modeling process because of the complicated relationship. The model I have implemented is a vertex dynamics model, and as such, one would think that the object of interest should be a vertex. Vertices require information about the cells which contain them in order to be moved. With this as a starting idea, I developed a sophisticated vertex class which contained cells as member variables. Apart from the software bloat coming from many vertices containing massive amounts of cell information, another issue was that cells are made of vertices, and, as such, cells out to contain vertices as member variables! Unfortunately this bilateral inclusion is not supported by C++.

5.1 [Highly Simplified] Pseudocode

5.2 Classes

Before going into how this issue was resolved in its entirely, I will describe the cell and coordinate classes.

5.2.1 The Cell Class

The cell class contains a number of useful functions and pertinent data members to make the code easy to read and understand. All cells know their index, they know which vertices make them up, they are able to calulate their area and perimeter, can modify their constituent vertices, can tell you whether or not ther contain a vertex, and can print out a graphical, color coded representation of themselves to an OFF file.

```
public:
        cell(int index, std::vector<int> AssociatedVertices, double target_area = 
        {
                assert(index >= 0);
                m_AssociatedVertices = AssociatedVertices;
                m_index = index;
                m_target_area = target_area;
                m_target_perimeter = std::sqrt(pi * m_target_area);
```

```
                m_gamma = gamma;
        }

        cell(){}

        std::vector<int> GetVertices(){return m_AssociatedVertices;};
        int GetIndex(){return m_index;};
        void SetIndex(int index){m_index = index;};
    void SetTargetArea(double area){m_target_area = area;};
        double GetTargetArea(){return m_target_area;};
        double GetTargetPerimeter(){return m_target_perimeter;};
        double ComputeArea(double * X, double * Y);
        double ComputePerimeter(double * X, double * Y);
        void PrintCell(std::ofstream &OffFile);
        int ContainsVertex(int index);
        void SetGamma(double gamma){m_gamma = gamma;};
        double GetGamma(){return m_gamma;};
        void InsertVert(int v1, int v2);
        void EraseVert(int index)
        {
                std::vector<int>::iterator it = std::find(m_AssociatedVertices.begi
                m_AssociatedVertices.erase(it);
        };
        void ReplaceVert(int before, int after)
        {
                std::vector<int>::iterator it = std::find(m_AssociatedVertices.begi
                *it = after;
        };
        void SetVertices(std::vector<int> vertices){m_AssociatedVertices = vertices
        int GetNumSides(){return m_AssociatedVertices.size();};
private:
        std::vector<int> m_AssociatedVertices; //Counterclockwise
        int m_index;
        double m_target_area;
        double m_target_perimeter;
        double m_gamma; // We can change this later.
};
```

5.2.2 The Coordinate Class

The coordinate class stores the index of a coordinate, and whether or not the vertex will move during the integration. My code will run with two types of meshes: meshes with border, and meshes without. A mesh with border is a mesh with fixed border elements. A mesh without border is a mesh which was generated with periodic boundary conditions and which maintains the periodicity throughout the calculations done by the program. Border vertices will not move, whereas interior vertices will be able to move; the coordinate class allows us to secify whether or not a vertex is interior or exterior. Another benefit of this class is that is allows use to easily extend the code to include forces acting on individual vertices, and to specify other types of vertices besides

interior and exterior. This code was developed with eyes to the future.

```
class coordinate
{
public:
        coordinate(int idx, bool t) : index(idx), IsInner(t){};
        coordinate(){index = -1; IsInner = 0;};
        int index;
        bool IsInner;
        inline bool operator==(const coordinate& rhs){return index == rhs.index;};
};
```

5.3 A Relational Database

A popular way to store data since the 1970's is to store data in group of tables which are connected via *keys*. This database is popular in business for reasons which will become apparent by means of a simple example.

Consider a business which sells a number of products, and wants to keep track of their customers, the customers orders, the customers addresses, and information about the products ordered. A wasteful way to store this data is to store a large table with a column for customer. Next to every customer's name is the customers address. Next to the customers address is the customer's order number. Next to the customer's order number is a column of items ordered. And, next to each item ordered is the item information. This method of storing data is terribly redundant, because for every customer's order which contains item X, the description of item X is stored in the table next to X. Furthermore, one customer may make several orders, so the customer's information will be stored multiple times for each order number.

A better idea is to break this data up into several tables which together define a *schema* for the data. In one table we store customer and customer infromation. In the second table we store a list of order numbers, and for each order number a set of identifying tags for the products ordered. Then, in another table we store the company's inventory as a table in which item information can be accessed via the tag stored in the order table. Now we have three nonredundant tables from which information can be extracted via *join* operations. A join operation extracts the rows from two tables which contain matching keys.

The code I have written implements this relational database model. I have six tables, including the simulationCells, coordinateList, X, Y, tempX and tempY tables. The cell and coordinate tables are implemented as 1d vectors of cell and coordinate objects, whereas the position tables are implemented as 1d arrays for ease of passing these structures to Cuda C functions which will speed up the computation. We will talk more about Cuda C and the parallelization of this code in a later section. The cells can extract coordinate information from the coordinateList table via the *index* key, and the coordinateList can access the position information from the X and Y arrays via their own *index*. The temporary X and Y arrays store temporary position information about the vertices before the mesh positions. My algorithm is a fault tolerant algorithm in that in checks for computational flaws in the temporary location arrays before updating the actual X and Y arrays, which would invalidate the entire computation.

5.4 Calculating the Gradient

5.5 Moving the Vertices

Version 1.0.0, the latest stable release of*Epithelium* loops over the vertices in the mesh and, for each one, computes the force applied to each verted and tehn computes a displacement due to the force using the Euler Method. In their original paper, H. Honda and T. Nagai described the use of a Modified Runge Kutta Method to move the vertices, but this method would result in extra unnecessary computations at each time step. For our purposes we only need the precision offered by the Euler Method because our model is a qualitative model and not a quantitive model. So long as the parameters in the code do not have physical significance, there is no need to worry about rounding errors in our code. This was the decision made by the research group at Oxford that developed CHASTE, the *other* leading software for implementing the Nagai-Honda Model. Of course, the integrator has been implemented as a function for as this model develops, other, more sophisicated integrators can be included.

For now, a displacement is calculated and stored in a temporary X vector. No vertex is permitted to move more than one half of the minimum delta separating vertices (the δ under which a T1 swap will occur). In this way we can guarantee that we will never miss the occurence of two vertices coming critically close to each other. If a vertex were allowed to move more than this distance, it is possible that two vertices located at a distance of slightly over δ might move past each other and invalidate the mesh. We cope with this issue by employing an adaptive time step to the code. The integration starts with a user defined maximum time step and begins to loop over all of the vertices in the mesh. The results of the integrstion are stored in the temporary X array. If the vertex happens to have moved too much, then the entire set of integrations for that time step is invalidated, the tiem step is halved, and the procedure starts over.

5.6 Embarassing Parallelism and CUDA

The numerical integrations and the updating vertex locations steps of the code exhibit what MATLAB co-founder Cleve Moler describes as "embarassing parallelism". That is, these steps involve absolutely no data dependency across the elements being visited in the for loop. That is to say, computing the displacement of vertex a does not depend upon the computation of the displacement of vertex b. Similarly, the vertex locations can all be updated in parallel since the update is simply a vector sum operation.

A popular hardware choice for parallel programming in last decade has been the NVIDIA GPU. Thanks to the generous support of the Sotomayor Lab at Ohio State where this paper and software have been written, I was able to work on a computer which has a very powerful CUDA capable GPU. CUDA is the NVIDIA extension of the C programming language which can specifiy which part of a piece of code to run on the CPU, and which parts to run on the GPU. The CUDA capable GPU features many simple microprocessors which can perform only rudimentary arithmetic operations. This is an idea piece of hardware for a piece of code which has to perform many arithmetic operations which do not have inhibiting data dependencies.

INSERT IMAGE of a CUDA CPU ALONGSIDE SOME ARRAYS TO MAKE IT CLEAR HOW THIS HAPPENS

Version 1.0.1 of the program exploits this data parallelism to update the locations of the vertices

in the mesh in one time step. This is simply a vector sum, and the cuda GPU is well suited to this task. I am now exploring the means of extending the code to use this parallel hardware for the integration steps as well, but due to the complexity of the force calculations and the number of diffeent values needed to be fetched to compute each integration, this may not turn out to be feasabile. The CUDA hardware favors simple arithmetic operations which can be applied simultaneously to many elements situated adjacent to one another in memory. The single most expensive task for a GPU code is to transfer data to the GPU effectively. This involves sending elements which are adjacent in memory and using them multiple times before sending a result back to the cpu. Version 2.0.0 will likely include this modification at least for some part of the computation of force.

5.7 Computing Topological changes.

The algorithm for performing a T1 swap goes as follows:

5.7.1 Eight Cases For Coordinate Placement in a T1 Swap

1. **Given:**
   I.x ¡ Ip1.x AND I.y == Ip1.y **Then:**
   I $\mapsto$ mp + (0, -dy) AND Ip1 $\mapsto$ mp + (0, dy)

2. **Given:**
   I.x ¿ Ip1.x AND I.y == Ip1.y **Then:**
   I $\mapsto$ mp + (0, dy) AND Ip1 $\mapsto$ mp + (0, -dy)

3. **Given:**
   I.x ¡ Ip1.x AND I.y ¡ Ip1.y **Then:**
   I $\mapsto$ mp + (dx, -dy) AND Ip1 $\mapsto$ mp + (-dx, dy)

4. **Given:**
   I.x ¿ Ip1.x AND I.y ¿ Ip1.y **Then:**
   I $\mapsto$ mp + (-dx, dy) AND Ip1 $\mapsto$ mp + (dx, -dy)

5. **Given:**
   I.x ¡ Ip1.x AND I.y ¿ Ip1.y **Then:**
   I $\mapsto$ mp + (-dx, -dy) AND Ip1 $\mapsto$ mp + (dx, dy)

6. **Given:**
   I.x ¿ Ip1.x AND I.y ¡ Ip1.y **Then:**
   I $\mapsto$ mp + (dx, dy) AND Ip1 $\mapsto$ mp + (-dx, -dy)

7. **Given:**
   I.x == Ip1.x AND I.y ¿ Ip1.y **Then:**
   I $\mapsto$ mp + (dx, -dy) AND Ip1 $\mapsto$ mp + (-dx, dy)

8. **Given:**
   I.x == Ip1.x AND I.y ¡ Ip1.y **Then:**
   I $\mapsto$ mp + (dx, -dy) AND Ip1 $\mapsto$ mp + (-dx, dy)

5.8 The C++ Standard Library, and Use of the New Library.

5.9 Why Epithelium is a good piece of software

To close our discussion of *Epithelium* I would like to discuss why it is a great piece of software for *you*. This bit of code builds easily and has only a couple of dependencies which are likely already installed on you computer if you are interested in computational science. You will need a compiler supporting at least the C++11 standard, python2.7, the matplotlib plotting library for python, the numpy python library, and - the only packages which will likely not be installed on your computer - the geomview geometric visualization software and the imagemagick image conversion program. Geomview and ImageMagick are free, very stable and have been around for more than 20 years. Of course, all of the animation, energy, and cell shape information is outputed from this software in file formats that are easy to adapt to your choice of plotting and animation software. The energy and histogram files are space separated lists and the mesh information is outputted in the OFF file format, which features a list of coordinates followed by a list of polygons which are defined counterclockwise as a list of vertex indices from the preceding list of vertices. A sample mesh file is show below to convince you of how simple this file format is, and to suggest that you could write a smal script to change this file to a different format that will be acceptable to your choice of visualization software.

Chapter 6

Numerical Methods for the Project

6.1 The Forward Euler Method

Adaptive Time Stepping

6.2 A Modified Runge Kutta Method

Why We Don't Use This Method

Appendix A

Valuable Programming Lessons Learned

A.1 GitHub

When you are working on a large project such as this one, it is a good idea to have some sort of version control system which tracks the changes you have made to your code, and to return to an earlier working version in case something gets terrribly broken. Many people who do not know about version control will do just this, except they will save as' every couple of days. Unfortunately, this method is very space inefficient, as each time you 'save as', you save your entire project. Roughly speaking, git saves only the small changes you have made between versions. Git is a popular version control tool, and github is a popular place to store you files online.

Get Your Files on GitHub

- Go to github.com and sign up for an account
- Make a new repo on github
- cd to the directory of your project on your machine.
- git init
- git add .
- git commit -m "some message"
- git remote add origin YOUR URL HERE (This url is given to you from github when you make the repo.)
- git pull origin master
- git push origin master

Access And Modify These Files Somewhere Else

-

Appendix B

How Several Programming Languages Were Used In Harmony

Bibliography

[1] Buchmann, A. Alber, M., Zartman, J. *Sizing it up: The mechanical feedback hypothesis of organ growth regulation.* Seminars in Cell and Developmental Biology, 2014.

[2] *Chaste Tutorial* `https://chaste.cs.ox.ac.uk/chaste/tutorials/release_2.1/UserTutorials.html`

[3] Drasdo, D. *Bucking Instabilities of One Layered Growing Tissues.* Physical Review Letters 84.

[4] Durand, M., Stone, H. Relaxation Time of the Topological T1 process in a Two Dimensional Foam. arXiv.0

[5] *Epithelial Tissues* `http://www.botany.uwc.ac.za/sci_ed/grade10/mammal/epithelial.htm`

[6] Farhadifar, R., Roper, J., Algouy, B., Eaton, S., Jullcher, F. The Influence of Cell Mechanics, Cell-Cell Interactions, and Proliferation on Epithelial Packing. *Current Biology* **17** 2095-2104. (2007)

[7] Fletcher, A. , Osterfield, M., Baker, R., Shvartsman, S. *Vertex Models of E[ithelial Morphogenesis.* Biophysical Journal 106, June 2014.

[8] Fletcher, A., Osborne, J., Maini, P, Gavaghan, D. *Implementing vertex dynamics models of cell populations in biology within a consitent computational framework.* Prog. Biophys. Mol. Biol. 113: 299 - 326.

[9] Gibson, W., Gibson, M. *Cell Topology, Geometry, and Morphogenesis in Proliferating Epithelia.* Current Topics in Developmental Biology **89** 87-114. (2009)

[10] P. Grassia, C. Oguey, R. Satomi. Relaxation of the topological T1 process in a two-dimensional foam. *The European Physical Journal E* **35** (2012)

[11] H. Honda. Description of Cellular Patterns by Dirichlet Domains: The Two-Dimensional Case. *Journal of Theoretical Biology* **72** 523-543. (1978)

[12] H. Honda, H. Yamanaka, M. Dan-Sohkawa. A Computer Simulation of Geometrical Configurations During Cell Division. *Journal of Theoretical Biology* **106** 423-435. (1984)

[13] Honda, H. Essence of Shape Formation in Animals. *Forma*, **27** S1-S8. (2012)

[14] K. Kawasaki, T. Nagai, K. Nakashima. Vertex models for two-dimensional grain growth. *Philisophical Magazine Part B* **60** 399 - 421. (1989)

[15] K. Liu, S. Ernst, V. Lecaudey, O. Ronneberger. Epithelial Rosette Detection in Microscopic Images.

[16] Marshall et. al. What Determines Cell Size? *BMC Biology* **10** (2012)

[17] R. Nagpal, A. Patel, M. Gibson. Epithelial Topology. *BioEssays* **30** 260-266. (2008)

[18] K. Nakashima, T. Nagai, K. Kawasaki. Scaling Behavior of Two-Dimensional Domain Growth: Computer Simulation of Vertex Models. *Journal of Statistical Physics* **57** 759-787. (1989)

[19] Ohlenbusch, H.M., Aste, T. Dubertret, B., Rivier, N. The Topological Structure of 2D Disordered Cellular Structures. arXiv.

[20] S.Okuda et. al. Reversible Network Reconnection Model for Simulating Large Deformation in Dynamic Tissue Morphogenesis. *Biomech Model Mechanobiol* **12** 627-644 (2013)

[21] `http://www.millerplace.k12.ny.us/webpages/lmiller/photos/636532/Epithelial\%20TissueTypes.bmp`

[22] Potential Energy Graph `http://images.tutorvista.com/content/work-energy-power/potential-energy-variation.gif`

[23] Weaire, D. and Rivier, N. *Soap, Cells, and Statistics*

[24] Luebke, D. and Owens, J. *Intro to Parallel Programming* `https://www.udacity.com/course/cs344` 2013.

[25] Nagai, T. Honda, H. *A dynamic model for the formation of epithelial tissues.* Philisohpical Magazine, Pt. B. Vol 81, Issue 7, 2001.

[26] Restrepo, S., Zartman, J. , and Basler, K. *Coordination of Patterning and Growth by the Morphogen DPP* Current Biology 24, 245- 255

[27] R. Thom. Topological Models in Biology. *Topology* **8** 313- 315 (1969)