THE UNIVERSITY OF CALGARY

Physically Based Simulation of Growing Surfaces

by

Mark Jeffrey Matthews

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

December, 2002

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Physically Based Simulation of Growing Surfaces" submitted by Mark Jeffrey Matthews in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor, Dr. P. Prusinkiewicz
Department of Computer Science

Dr. Sheelagh Carpendale
Department of Computer Science

Dr. Mario Costa Sousa
Department of Computer Science

Dr. Marcelo Epstein
Department of Engineering

Date

Abstract

Understanding the link between differential growth and form is necessary for understanding the morphogenesis of living organisms. This thesis describes two tools for simulating the growth of surfaces in three dimensions. These tools make it possible to explore the link between differential growth and form, and link growth to pattern formation traditionally considered outside of growth. Surface growth is controlled by diffusible morphogens which can be interactively placed, or predefined. The morphogens can diffuse, decay and interact. Two different techniques, a mass-spring system and finite element analysis, are used to simulate the physics of the surface. A number of test cases are presented. The first example is a model of sea urchin development. Secondly the dorsal petal lobes and leaves of the *Antirrhinum majus* (snapdragon) are examined. Finally, an oscillating genetic network is used to control the growth of a surface, creating some novel forms.

Acknowledgements

I'd first like to thank my supervisor, Dr. Przemyslaw Prusinkiewicz. It's been enjoyable to work with an experienced researcher who seems to have limitless insight, and always kept my research on track. I'd also like to thank our collaborators Dr. Enrico Coen and Anne-Gaelle Rolland for a fruitful collaboration and many penetrating discussions. Special thanks to my examination committee who provided valuable feedback and made the defense experience enjoyable.

This thesis was proofread by a number of people including Dr. P, Richard Lobb, Christine Dooley and the Jungle lab. Thanks guys! Thanks also goes to Tyson Rock and Norm Faulkner of the Alberta College of Art and Design for their glass blowing demonstrations.

I certainly could not have completed this work as quickly as I did without all the programming tips, advice and moral support I received from everyone in the lab. I'd especially like to thank Carla Davidson, Pavol Federl, Mark Fox, Callum Galbraith, Radek Karwowski, Brendan Lane, Robson Lemos, Peter MacMurchy, Lars Mündermann and Jing Yu. You all have made the Jungle lab the best research environment I have ever worked in.

Finally, and most importantly, I'd like to thank my parents who nurtured me to have an inquisitive mind, and have always supported me in whatever I've done. Thank you.

This work dedicated to the glory of God.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Statement of Problem

Since 1968, L-systems [51] have been used to model plants, bacteria and various other organisms. However, L-systems are not an ideal paradigm for modeling isolated plant organs (flowers, leaves, apices, etc.) This is because L-systems are best at modeling branching structures. Although this method is powerful, it has limitations, especially in the study of the morphogenesis of plant organs.

Properly modeling plant organs often involves considering them as a two-dimensional surface deforming in three dimensions. Leaves and flower petals fit this description well. A growth field can be defined on this surface, and the resulting deformation seen as a mechanical problem and simulated as such. Ultimately, morphogenetic models should have self generating growth fields, though models for doing so are relatively unexplored.

In his book "The Art of Genes" [14], Enrico Coen proposed a metaphor for morphogenesis that is particularly well suited for two dimensions: an expanding canvas. This canvas allows diffusible morphogens to interact and control the growth of the canvas. This abstraction is the essence of development and the construction of simulation models should be based on the expanding canvas metaphor.

The understanding of the link between differential growth and form is necessary for developmental models. However, in our day to day encounters, we rarely en-

counter expanding objects[1], making this a particularly non-intuitive problem. To explore the problem of growth and form, I developed a program called Canvas that makes it possible to:

1. interactively explore the link between differential growth and form

2. link growth to pattern formation traditionally considered outside of growth

The user can define a morphogen field over the surface interactively or using an input file. The morphogens can diffuse, decay and interact. A growth field over the surface is derived from the morphogen concentration. Finding a surface that satisfies this growth field can be considered as a purely geometric problem, but here physical simulation is used as a general and robust solution method.

Two different techniques are used to simulate the physics of the surface. The first, a mass-spring system, is a simple method that allows for interactive frame rates. A drawback is that the surface must be discretized in a regular manner, disallowing adaptive subdivision, and limiting the amount of growth possible. The surface shape must also be rectangular, prohibiting curved leaf and petal outlines. The second technique, finite element analysis, is a more general method, based on continuum mechanics, allowing for arbitrary subdivision and hence infinite growth. The benefits of finite element analysis come at the cost of greater complexity and slower simulation.

A number of models are presented. The first is a model of sea urchin development. Sea urchins have been a popular subject of morphologists because of their

[1]Actually, we often encounter expanding objects in almost anything organic. However, this expansion is usually so slow that it is imperceptible.

simple form. Secondly, the dorsal petal lobes and leaves of the *Antirrhinum majus* (snapdragon) are examined. *A. majus* is a good subject for geneticists because of its high mutability and ease of cultivation [14]. This example involved collaboration with *A. majus* researchers, making growth data available to us.

Finally, a more theoretical model is presented. An oscillating genetic network is used to control growth, creating some novel forms.

1.2 Thesis Organization

This thesis has been divided into nine chapters. This first chapter is an introduction to the problem studied. It is followed by the mathematical basis of mechanics and growth (Chapter 2) and pattern formation models (Chapter 3). The numerical methods required to model the theory of Chapters 2 and 3 are presented in Chapter 4. I focus on mass-spring systems and the finite element method. Visualization methods are given in Chapter 5. The operation of the two programs developed for this thesis: Canvas and CanvasLite, are discussed in Chapters 6 and 7. In Chapter 8 I present a number of biological models using Canvas. And finally, in Chapter 9, I summarize contributions, draw conclusions and discuss future work.

Chapter 2

Mathematical Models of Growth

In exploring the link between differential growth and form it is essential to understand the mechanics of how two-dimensional surfaces deform in three dimensions, and how this relates to growth.

I first review previous biological spatial simulation models in Section 2.1. Section 2.2 presents the theory of elasticity, as it relates to surfaces. Sections 2.2.2, 2.2.3 and 2.2.4 define basic continuum mechanics quantities. Section 2.3 presents the concept of the growth tensor and Section 2.4 links growth and continuum mechanics.

2.1 Biological Spatial Simulation Models

Initial work in simulating the spatial structure of biological organisms used a square lattice to represent shape. A state was associated with each square, indicating if the organism occupied that space. Eden [30] simulated accretive growth of a cell in this manner, and Ulam [93] presented a method for generating branching patterns using cellular automata [90].

Branching patterns were considered by Cohen [16], who generated structures influenced by an external density field. A more general framework for generating branching structures, are *L-systems*, which were pioneered by Lindenmayer [51]. L-systems have been used to model numerous aspects of plant growth and architecture, including mechanical interactions [48]. Although L-systems have been used to model

the surface shape of a growing leaf [70], these models did not consider surface mechanics. Map L-systems [52] are an extension of L-systems that are not restricted to branching topology. They have been used to model two-dimensional surfaces incorporating mechanical interactions such as the thallus of the fern *Microsorium linguaforme* [36].

Stevens [82] makes an interesting observation about growth and surfaces. He states that if we consider a hexagon constructed of six triangular segments and one segment is removed, the surface *must* assume a conic shape. Similarly, if a seventh section is added, it will assume a "potato chip" shape. Adding and removing these sections is analogous to growth, and organisms that grow in this manner "know" nothing about how to construct these shapes. They are simply assuming the form that the surface must take, as dictated by mechanical interactions. This is why understanding mechanics is important in understanding spatial developmental models.

Green [38] considers mechanical buckling as a morphogenetic phenomenon. In particular, Green considers the phyllotactic patterns of plant primordia. In contrast to the morphogen theory [26] where chemical substances influence placement of new primordia, Green hypothesizes that primordia are placed in the lowest buckling energy configuration. His work is significant because it considers mechanical stress as a means of local communication for morphogenesis.

Jacobson and Gordon [47] conducted rigorous simulations of the development of newt embryos. In particular they simulated formation of the *neural plate*, one of the first forms of symmetry breaking in an embryo. Their work used a hybrid model called "morphodynamics" to compute the shape change of a form, given a specified growth field.

2.2 Elasticity Theory

Elasticity theory describes how solid objects deform in the presence of external forces. Objects are conceptualized as a continuous medium — not discrete in any way — modeled by differential equations.

Within computer graphics, elasticity theory was first applied to physically based modeling by Terzopoulos *et al* [87]. Their work considered deformable solids, surfaces and curves, of which we consider only surfaces. It is important to note that the theory reproduced here is only loosely applied to the mass spring models discussed later. The concepts are however relevant and are thus presented.

2.2.1 Deformable Surfaces of Terzopoulos *et al.*

A point on a surface can be parameterized by two material coordinates: $\mathbf{a} = [a_1, a_2]$. The vector position of a point within the body is then given by a time varying function $\mathbf{r}(\mathbf{a}, t)$ (Figure 2.1). The undeformed natural rest configuration of the body is similarly defined as $\mathbf{r}^0(\mathbf{a})$.

The motion of a point $\mathbf{r}$ within the surface can be described by an equation given in Lagrange's [37] form:

$$\frac{\partial}{\partial t}\left(\mu\frac{\partial \mathbf{r}}{\partial t}\right) + \gamma\frac{\partial \mathbf{r}}{\partial t} + \frac{\delta \mathcal{E}(\mathbf{r})}{\delta \mathbf{r}} = \mathbf{f}(\mathbf{r}, t) \tag{2.1}$$

where $\mu(\mathbf{a})$ is the mass density of a particle at point $\mathbf{a}$, γ is a damping coefficient and $\mathbf{f}(\mathbf{r}, t)$ the net external force at time t. In the third term $\mathcal{E}(\mathbf{r})$ is an energy functional that measures the net potential elastic energy of the deformed configuration. Essentially, it describes the behavior of a body. The form of this functional for a

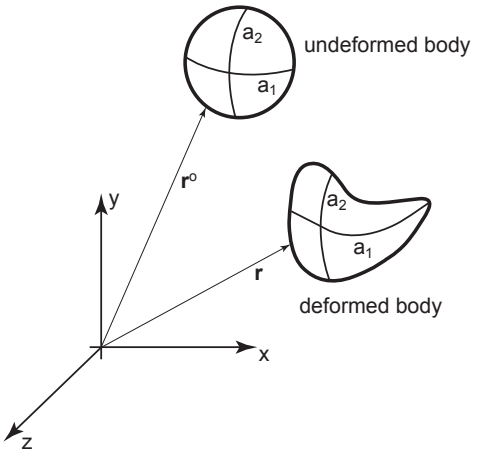


Figure 2.1: Location of a point **r** on a deformed and undeformed configuration of a body. (after [87])

surface is[1]:

$$\mathcal{E}(\mathbf{r}) = \int_{\Omega} \|\mathbf{G} - \mathbf{G}^0\|^2 + \|\mathbf{B} - \mathbf{B}^0\|^2 da_1 da_2 \tag{2.2}$$

G and **B** are the first and second fundamental forms of the surface, given by:

$$G_{ij}(\mathbf{r}(\mathbf{a})) = \frac{\partial \mathbf{r}}{\partial a_i} \cdot \frac{\partial \mathbf{r}}{\partial a_j} \tag{2.3}$$

$$B_{ij}(\mathbf{r}(\mathbf{a})) = \mathbf{n} \cdot \frac{\partial^2 \mathbf{r}}{\partial a_i \partial a_j} \tag{2.4}$$

where **n** is the unit normal of the surface at point **a**. Ω indicates that we are integrating over the whole surface. For a surface, **G** and **B** are 2×2 tensors.

At a more intuitive level, **G** measures stretching and shearing within the surface, while **B** measures out of plane curvature and twisting deformations. $\mathbf{G}^0$ and $\mathbf{B}^0$

[1]It should be noted that in the original work of Terzopoulos *et al*, a *weighted norm* was used which is omitted here for simplicity.

are the fundamental form of $\mathbf{r}^0$ (ie. the natural state to which the surface tends). Equations 2.1 to 2.4 completely describe the behavior of the surface in response to a time varying external force. It should be noted that they have the effect of minimizing $\mathcal{E}(\mathbf{r})$ over time.

This formulation of elasticity is favorable because of its simplicity and its conceptual similarities with mass spring systems. Note that if we consider the mass to exist at discrete points in equation 2.1, then we would then have $\mathbf{f}$ in N and μ would be the weight of each particle in kg. This is the basic formulation of a *particle system.*

2.2.2 Strain Tensor

One can further refine our understanding of the deformed state of a body by defining a strain tensor. We assume that when a body is in a state of stress, it deforms slightly($\mathbf{r}$), and when the stress is removed, it returns to its original rest state($\mathbf{r}^0$).

This defines a displacement field $\delta = [u, v, w]$ which is the displacement experienced by a point in the deformed body from its rest configuration. If $\mathbf{r}$ and $\mathbf{r}^0$ are defined on a Cartesian coordinate system, then u, v, w are defined as:

$$\begin{aligned} u &= \mathbf{r}_x(\mathbf{a}) - \mathbf{r}^0_x(\mathbf{a}) \\ v &= \mathbf{r}_y(\mathbf{a}) - \mathbf{r}^0_y(\mathbf{a}) \\ w &= \mathbf{r}_z(\mathbf{a}) - \mathbf{r}^0_z(\mathbf{a}) \end{aligned} \tag{2.5}$$

We then define the *displacement gradient* tensor as:

$$\mathbf{A} = \nabla\delta = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} \tag{2.6}$$

From this tensor we can derive the *strain vector*, which also defines the shear strains $\gamma_{xy}, \gamma_{yz}, \gamma_{zx}$:

$$\{\epsilon\} = \begin{Bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial w}{\partial z} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \end{Bmatrix} \tag{2.7}$$

2.2.3 Stress Tensor

Similar to the strain tensor, one can also define a stress tensor. If a body is subject to external forces, that body is under stress. The stress at a point within that body can be described by a tensor given as follows:

$$\sigma = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \tag{2.8}$$

Each element within the tensor corresponds to a force shown in Figure 2.2. The normal stresses $\sigma_x, \sigma_y, \sigma_z$ correspond to forces normal to the faces of the cube. The shear forces are $\tau_{xy}, \tau_{yz}, \tau_{zx}$. The first double subscript indicates the plane of ac-

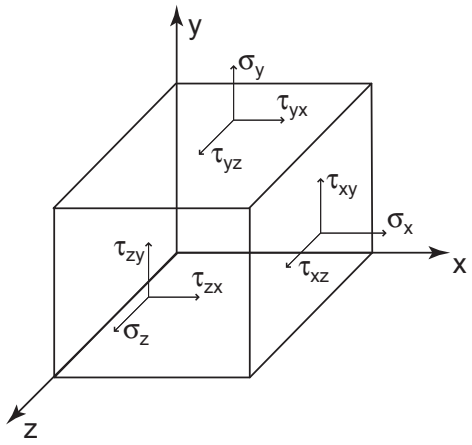


Figure 2.2: Stress forces acting on an infinitesimal body.

tion, and the second subscript indicates the direction of the force. There are only three components of shear stress because it can be shown that in the limit as the volume tends towards zero, $\tau_{xy} = \tau_{yx}, \tau_{yz} = \tau_{zy}, \tau_{xz} = \tau_{zy}$ [45]. Thus, there are six components of stress. These are most commonly represented as a vector:

$$\{\sigma\} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} \tag{2.9}$$

2.2.4 Generalized Hooke's Law

The behavior of an isotropic homogeneous linear elastic medium can be completely defined by two constants: Young's modulus E and Poisson's ratio ν. Young's mod-

ulus is a measure of the stiffness of the material, while Poisson's ratio is a measure of shrinkage perpendicular to strain.

The relationship between the stress σ and the strain ϵ in an object is then given by generalized Hooke's Law [45]:

$$\{\sigma\} = [\mathbf{C}]\{\epsilon\} \tag{2.10}$$

where C is given by,

$$\mathbf{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{2.11}$$

2.3 Growth Theory

In order to properly model growth, it is necessary to first describe growth. It may be mistakenly assumed that growth is simply a scalar quantity, defined over some object. Growth is not a scalar quantity, and we begin with an intuitive explanation why this is so.

Growth was first rigorously defined as a *tensorial* quantity by Hejnowicz and Romberger [43]. To understand why growth is a tensor, and not a vector or scalar, consider the 2D example shown in Figure 2.3.

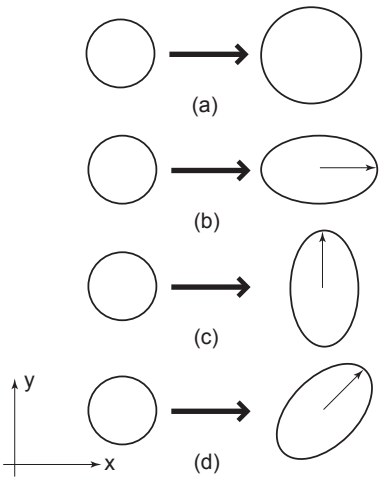


Figure 2.3: Possible resulting shapes from a circle with the same growth at all points.

Suppose that each point on the circle grows in exactly the same way. The simplest type of growth is shown in (a). This is isotropic growth where only a scalar value is needed to describe the growth.

But, the circle could also deform in the ways shown in (b) and (c). In these cases more than one value would be needed to describe growth. Perhaps horizontal growth and vertical growth could be used. One might suggest then that growth is a vector quantity. However, if growth was a vector quantity, then adding growths (b) and (c) together would result in case (d). This is incorrect (Figure 2.4). In actuality, case (a) would result. A simple examination reveals that at least three values are needed to describe planar growth[2]. This rules out the use of scalars and vectors to describe growth.

There are two ways to approach the definition of growth, which will be shown to be related to each other in the end. In each case the velocity field for all points

[2]For example, these values could be the rotation angle for the major axis of growth, the expansion along the major axis of growth, and expansion along the minor (perpendicular) axis of growth

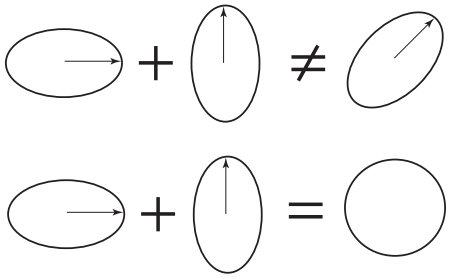


Figure 2.4: Why growth is not a vector.

within the organism needs to be known to define the growth field for the organism.

2.3.1 Growth Tensor

The more general definition of growth is that of the *growth tensor*. If we consider that an object is growing or changing shape, then the points within the object are moving, each with an associated velocity. We therefore have a velocity field $\mathbf{V}$. The growth tensor is then defined as the gradient of this field:

$$\mathbf{G} = \nabla \mathbf{V} \tag{2.12}$$

To understand why this is the case, consider a one-dimensional example. Imagine that you are in the caboose of a magical train. Your friend is the conductor. The cars in between you have had a magical spell cast on them, and multiply while you are moving, so that if perhaps you began with five cars, very soon you have 20 cars. During this process you and your friend have been moving apart, either by the caboose slowing down, or the engine moving faster. The difference between the caboose and engine speed, is a measure of the rate at which the train was growing. The growth tensor does the same thing in three dimensions. It measures the rate of

change in velocity over a distance.

If we refer back to our 2D example shown in Figure 2.3, we would have the following as elements of our growth tensor:

$$\mathbf{G} = \begin{bmatrix} \frac{\partial V_x}{\partial x} & \frac{\partial V_x}{\partial y} \\ \frac{\partial V_y}{\partial x} & \frac{\partial V_y}{\partial y} \end{bmatrix} \tag{2.13}$$

It should be noted from equation 2.13 that there are four degrees of freedom within the growth tensor. Earlier we stated that only three degrees of freedom are needed to describe growth in 2D. The truth is that $\mathbf{G}_{ij}$ also contains a rotational component that does not contribute to growth. As shown by Hejnowicz and Romberger [43], this can be eliminated by taking only the symmetric portion, $\frac{1}{2}(\mathbf{G} + \mathbf{G}^T)$. The same holds for the 3D case where the tensor would have nine quantities, of which only six are relevant.

2.3.2 RERG

Another way to define growth is the Relative Elemental Rate of Growth (RERG), first defined by Richards & Kavanagh [75] in 1943. The essential idea is that it measures the growth at a point p along a segment $\mathbf{s}$ (Figure 2.5). If we let $\mathbf{s}$ deform with the body as it grows, and define s be the length of $\mathbf{s}$ (ie. $s = |\mathbf{s}|$), then we can formally state the RERG as:

$$RERG_s = \lim_{s \to 0, \Delta t \to 0} \frac{\Delta s}{s \Delta t} = \frac{d\mathbf{v}_s}{ds} \tag{2.14}$$

where Δs is the change of length s during time interval Δt, and v_s is the elongation velocity of the segment. The RERG can be calculated from the growth tensor. Given

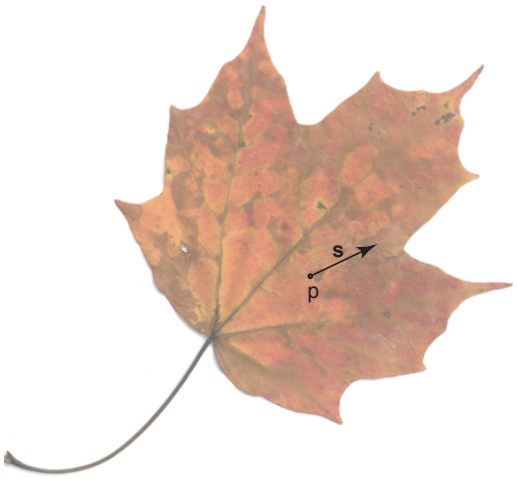


Figure 2.5: RERG definition.

a unit vector $\mathbf{e}$ in the direction of $\mathbf{s}$ and a growth tensor $\mathbf{G}$, we can compute the RERG as follows:

$$RERG_s = \mathbf{e}^T \mathbf{G} \mathbf{e} \tag{2.15}$$

2.4 Linking Growth to Elasticity Theory

A difficult question now arises. How does one integrate growth into elasticity theory? Some theory has been developed on this subject, such as the work of Epstein [33], and Taber [83] who presents a survey of mechanical theories of growth. However, there is no clear single way to approach the subject.

Here I choose a simple method, making the following observation. We previously discussed how mechanics relies on a displacement field and how growth relies on a velocity field. We relate growth and strain by defining an interval Δt. That is:

$$\mathbf{G}\Delta t = \begin{bmatrix} \frac{\partial(V_x \Delta t)}{\partial x} & \frac{\partial(V_x \Delta t)}{\partial y} \\ \frac{\partial(V_y \Delta t)}{\partial x} & \frac{\partial(V_y \Delta t)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = \mathbf{A} \tag{2.16}$$

The growth tensor can become a displacement gradient tensor, if we define an interval Δt over which the growth occurs. Realizing that we can construct ϵ from $\mathbf{A}$ this will later allow us to easily integrate growth into the finite element method in Chapter 4.

Chapter 3

Mathematical Models of Pattern Formation

In the previous chapter I laid down the groundwork for how an object changes form in response to growth. This alone does not explain morphogenesis. A complete description of morphogenesis needs to explain the mechanisms that drive growth. Many models have been proposed that can explain aspects of these mechanisms, though a complete model is far from being known. These models are commonly known as *pattern formation* models, and are the subject of this chapter.

Since our work is primarily concerned with deforming surfaces, I focus on models that work for 2D surfaces. I first present the concept of *diffusible morphogens*. Diffusible morphogens are a common way to implement local communication in pattern formation models. *Reaction-diffusion* systems, and *genetic simulation* models incorporate diffusible morphogens into autonomous models, that attempt to conform to biological constraints. One of these constraints is the idea of local action. No one cell within the organism has a complete picture of the whole organism. It can only act on its current state and what exists in the local neighborhood.

Coen [14] outlined a metaphor for development. He likens development to an expanding canvas. The organism is the canvas, and colors on the canvas reflect gene expression patterns. As the organism/canvas grows, there is more and more room for detail, but the details are constrained by what was already placed on the canvas. This metaphor is based on modern biochemistry and is much of the inspiration for the work in this thesis.

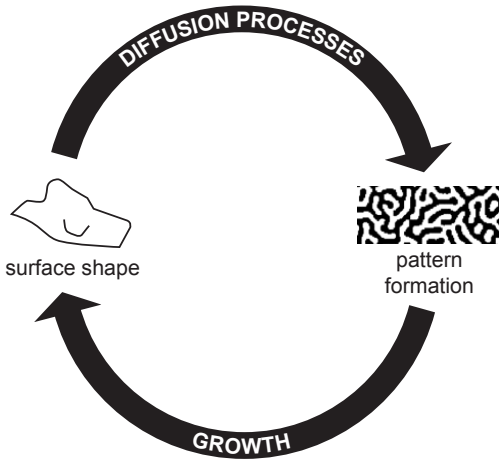


Figure 3.1: Feedback in morphogenetic processes.

In a true biological system feedback is also present (Figure 3.1). If we assume that many of the genetic processes involved in pattern formation use diffusion as a means of communication, then changes in the shape will influence pattern formation. If we also assume that the pattern formation influences the growth of the shape, then we have a complete feedback system. It is this feedback that can make morphogenesis such a complex process.

The above mentioned pattern formation methods have traditionally been applied to static surfaces [91], though some exceptions exist. Meinhardt [55] considered reaction-diffusion patterns on a one-dimensional expanding curve for generating sea shell pigmentation patterns. Crampin *et al* [20, 21] have also specifically considered reaction-diffusion pattern evolution on growing domains. Walter *et al* [95] presented a method for procedural texture generation on expanding surfaces. What is novel about this research is the feedback mechanism between pattern formation and growth.

3.1 Diffusible Morphogens

Diffusible morphogens are a commonly postulated explanation for local communication within organisms. They propose the existence of a chemical that is created by the organism as a means of signaling, can diffuse through the organism, and is received by a more distant portion of the organism.

Within computer graphics diffusible morphogens were first used by Turk [92] and Witkin and Kass [97]. Both their work employed reaction diffusion systems for texture generation, with Turk's working over arbitrary topologies.

Diffusion is a relatively simple process to describe mathematically. If a is the scalar concentration of a chemical over one dimension, x, then the rate of change is given by [29]:

$$\frac{\partial a}{\partial t} = D_a \frac{\partial^2 a}{\partial x^2} \tag{3.1}$$

The rate of change is proportional to the second derivative of the concentration, and a diffusion coefficient, D_a. This can be generalized to multiple dimensions using $\bigtriangledown^2$, the Laplacian operator [49]:

$$\frac{\partial a}{\partial t} = D_a \bigtriangledown^2 a \tag{3.2}$$

However, it is important to note that it is unlikely that such a simple model can account for all types of biological diffusion. It is likely that in many cases *active transport* is used, which causes preferential flow in one direction. For matters of simplicity, I do not consider this mode of transport in this thesis.

3.2 Reaction-Diffusion

Reaction-diffusion systems are one of the more elegant models of pattern formation, first proposed by Turing [91] in 1952. Reaction diffusion systems postulate interacting morphogens that generate spatial patterns. They are successful at recreating a wide variety of pigmentation patterns observed in nature, such as those seen on sea-shells [55], zebras and giraffes [92].

Reaction-diffusion systems typically stipulate the presence of two interacting morphogens. In an *activator-inhibitor* [56, 57] system, one morphogen, a, is a short range activator, while the other, b, is a long range inhibitor. The mathematical formulation of this is as follows [55]:

$$\begin{aligned} \frac{\partial a}{\partial t} &= s\left(\frac{a^2}{a^2+h} + b_a\right) - r_a a + D_a \frac{\partial^2 a}{\partial x^2} \\ \frac{\partial h}{\partial t} &= sa^2 - r_h h + D_h \frac{\partial^2 h}{\partial x^2} + b_h \end{aligned} \tag{3.3}$$

Examining the first equation of 3.3, we see that the production rate a is proportional to a^2, making this an autocatalytic reaction. It is also inversely proportional to h, making h an inhibitor to a. We also note that it has a decay term, $r_a a$, and a diffusion term, $D_a \frac{\partial^2 a}{\partial x^2}$. Since we stated that a is a short range activator, $D_a \ll D_h$. The second equation is similarly constructed except that we note that the production of h is proportional to a^2. Finally we note that b_a and b_h are the basic production rate, and cause production when no other factors would. These are responsible for beginning the pattern formation.

The nature of the reaction can be summarized in Figure 3.2. The activator a promotes the production of itself, but also promotes production of the inhibitor h, which then inhibits a. Figure 3.3 shows some examples of the types of patterns

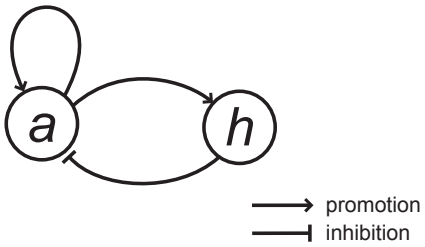


Figure 3.2: Schematic of an activator-inhibitor system.

Figure 3.3: Examples of reaction-diffusion patterns. (Used with permission from [69].)

possible from reaction-diffusion systems. There are also other forms of reaction-diffusion systems such as activator depleted substrate systems and multi-chemical systems [55].

3.3 Gene Expression

All organic life is composed of cells. Cells contain DNA which ultimately characterizes its behavior. Some chemical pathways involving DNA can be abstracted into genetic regulatory networks [54, 53, 42]. Regulatory networks have been described for many processes such as the frog and yeast cell cycle [7, 12], Phage λ [71] (a bacterial virus), and bacterial chemotaxis [4, 73]. It is likely that these networks play a large role in morphogenesis and thus a brief introduction is presented here.

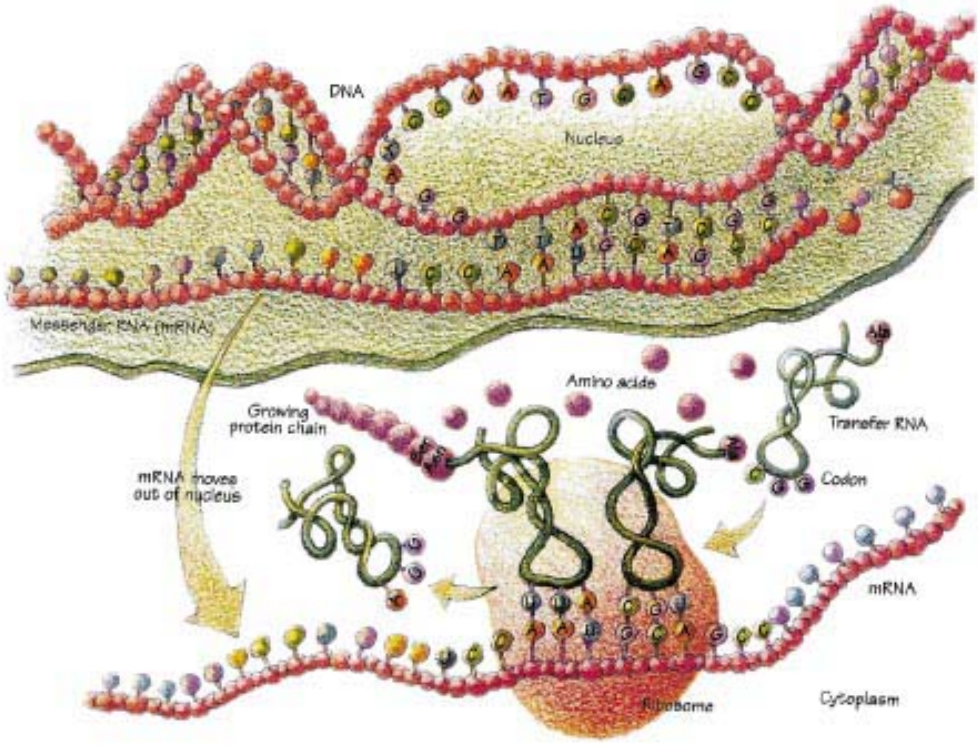


Figure 3.4: Genetic transcription and translation in eukaryotes. (Used with permission from [35].)

Genes are sections of DNA that code for proteins. The process of turning a gene into a protein is a two step process [2, 80] (Figure 3.4). First, a copy of the DNA is made with RNA by RNA polymerase. This is known as *transcription*. Then, this messenger RNA (mRNA) can be interpreted by a ribosome to create a polypeptide chain. Polypeptides then fold into a functional protein. This process is known as *translation*. Proteins are used by the cell for many purposes. When a gene is actively being transcribed and translated, it is said to be *expressed*.

DNA contains sequences called operators, to which some *regulatory proteins* can directly bind. When regulatory proteins bind to these sequences they influence the

rate at which genes are being expressed. The expression of these genes could in turn, influence the expression of other genes, forming networks. Networks that serve to regulate behavior of the cell are know as genetic regulatory networks.

Modeling these regulatory networks has been the subject of recent research [73]. A number of continuous models have been proposed, but betray the complexity of gene expression [35]. The problem is that gene expression is quite a "messy" process [2]. mRNA decays quite rapidly and there may be several copies of mRNA coming off of one piece of DNA at a time. The mRNA may code for several proteins at once (polycistronic mRNA). Also, since cells are so small, there may only be a small number of proteins present to promote or inhibit genes. This makes the process highly stochastic, and difficult to model with traditional continuous reaction rate chemistry.

There are, however, many successes. A system that is well understood is Phage λ [71], a virus that attacks bacteria. The regulatory network that controls if the virus remains dormant or not has been successfully modelled [35]. Another success in the understanding of gene expression is the construction of synthetic gene networks. Elowitz [32] was successful in incorporating a synthetic gene network, called the "repressilator" into *Escheria coli* to produce oscillating production of green fluorescent protein (GFP).

3.4 Connectionist Model

Mjolsness *et al.* [60] outlined a mathematical framework for the simulation of genetic activity within cells. They use a differential formulation of gene expression based on

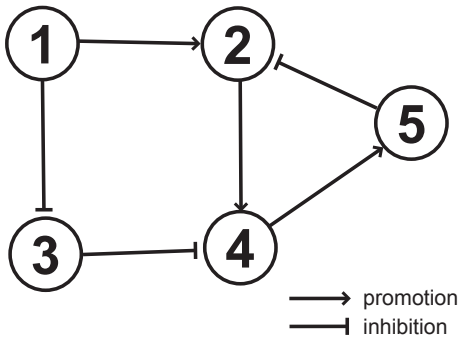


Figure 3.5: An example genetic network. Nodes represent the concentrations of gene products T_1 to T_5. Arrows represent a negative or positive influence of one gene product on the production of another.

a *connection matrix*. An element W_{ij} of the matrix is the amount by which which gene product T_j promotes or inhibits gene i.

Consider the following gene interaction diagram shown in Figure 3.5. The genetic interactions in the diagram be represented with the following connection matrix:

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.4}$$

Given a suitable gene expression thresholding function g_i, the time development of cellular activity can be formulated as follows:

$$\frac{dT_i}{dt} = R_i g_i \left(\sum_{j=1}^{N} W_{ij} T_j + h_i \right) + D_i \nabla^2 T_i - \gamma_i T_i \tag{3.5}$$

where R_i is the production constant of gene i, h_i is the threshold value for gene

i. and D_i is a diffusion coefficient. The diffusion term has been modified from its original form in [60] to accommodate multidimensional diffusion. The function g_i is typically sigmoidal because gene expression levels reach a maximum value in real systems. Finally, γ_i is a decay rate for gene product T_i.

Chapter 4

Numerical Methods

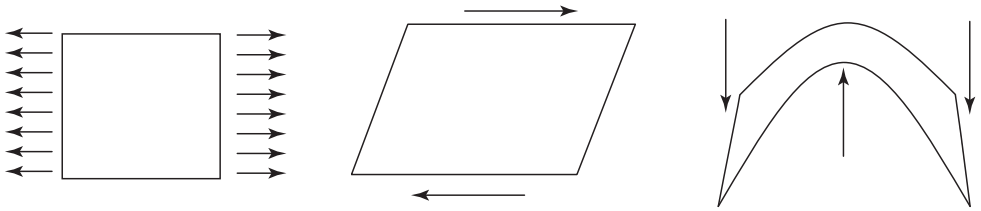
While describing the mathematical models in the previous two chapters we have assumed continuous media. However, simulating these models with a computer requires us to consider our media in discrete units and the use of numerical techniques for finding solutions. This is the subject of this chapter.

In this chapter, I present two different approaches to simulating 2D surfaces deforming in 3D. The first method, a *mass-spring system*, has the advantages of speed and simplicity at the expense of a fixed discretization, limited growth and non-physical behavior. The second method, the *finite element method* (FEM) is a more general, more physically accurate technique, which allows for arbitrary discretization, but at the expense of complexity and speed.

4.1 Mass-Spring System

A mass-spring system is a system of idealized point masses (particles) interconnected by force producing Hookean springs, for the purpose of simulating physical behavior. Within plant modeling, mass-spring systems were first used by Fracchia *et al.* [36] for examining the growth of cellular structures. Mass-spring systems have been used extensively in computer graphics to model cloth [8]. This thesis draws strongly on this body of work[1]. This thesis is also based on the work of Dimian [24] who modeled

[1]Consider the fact that silk is often used for fake flower petals.



(a) stretching (b) shear (c) out of plane bending

Figure 4.1: Modes of deformation of a 2D surface in 3D.

deforming surfaces.

If a typical surface is considered, three resistive forces are observed (Figure 4.1). These deformations can be quantified by the first and second fundamental forms **G** and **B** presented in Section 2.2.1. Stretching deformations (a) are given by the diagonal diagonal elements of **G**, while shear deformations (b) are given by the off diagonal elements. Bending deformations (c), are given by the elements of **B**. We wish to develop forces that will resist these types of deformations. The method used by Terzopolous *et al.* was to define a scalar deformation energy based on **G** and **B** (equation 2.2), and use variational derivatives to find a force (equation 2.1).

Mass-spring systems don't use such an elaborate system. The primary appeal of mass-spring systems are their speed (explicit integration schemes can often be used) and simplicity. Within computer graphics, where visual appeal is valued over physical exactness, they have been used to model a variety of physical phenomena simply and rapidly.

Mass-spring systems use a network of Hookean springs to provide similar re-

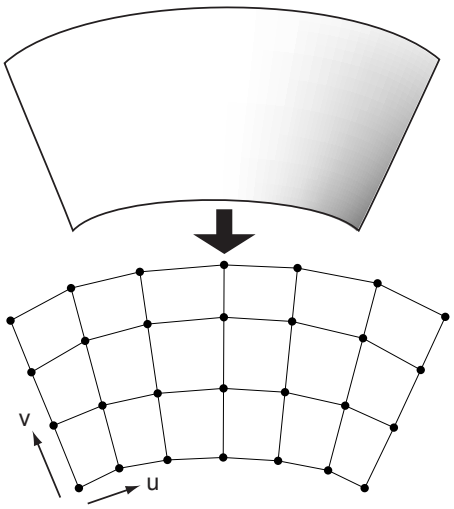


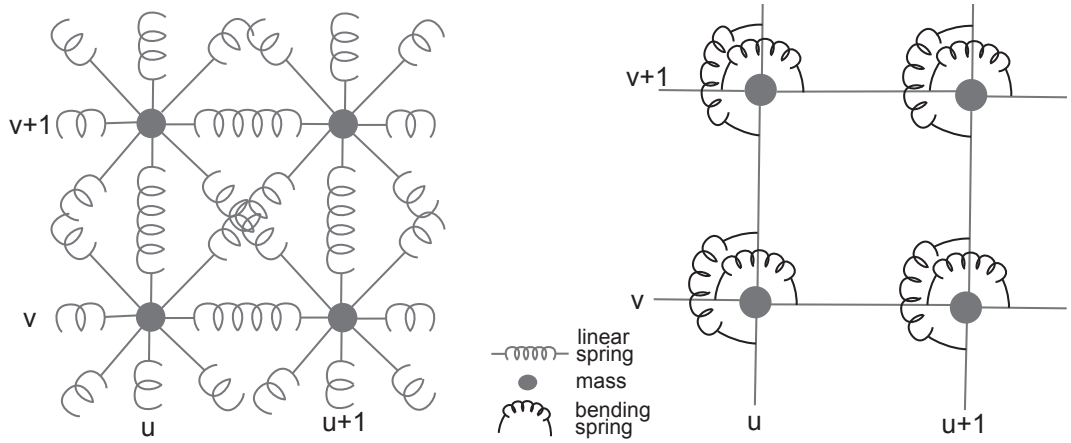
Figure 4.2: Canvas discretization.

sistance to deformation types (a),(b) and (c). It is assumed that the mass of a continuous media can be concentrated at discrete mass points (Figure 4.2) on which these springs act.

Resistance to stretching is accomplished by linking springs across every mass in the u and v direction (Figure 4.3(a)). Resistance to shearing is accomplished by cross-linking springs. Resistance to bending is added by special bending springs (Figure 4.3(b)). These bending springs are not 1D springs, but instead respond to out of plane bending. They are described in Section 4.1.2.

By varying the spring constants for stretch, shearing and bending deformations, the characteristics of the surface are changed. Objects like leaves and paper resist stretch and shearing, while easily allowing bending. Other thicker materials like plywood, resist bending more strongly.

Mass-spring models should not be thought of as a rigorous discretization of continuous surfaces, but as a discrete model unto itself that only *approximates* the



(a) Linear springs.

(b) Bending springs.

Figure 4.3: Springs used to approximate the resistance of a continuous surface.

behaviour of continuous surfaces. This however has done nothing to inhibit their use within computer graphics, where their simple formulation and speed has made them popular.

4.1.1 General Formulation

We enumerate all particles in the system using the index $i = 1..N_m$, where N_m is the number of particles in the system. Each particle has an associated state, described by a position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$. Particles within a mass-spring system obey Newton's second law: $\mathbf{f} = m\mathbf{a}$ [64]. If we defer the formulation of spring forces for a moment, we can state the acceleration $\mathbf{a}_i$ of particle i as follows:

$$\mathbf{a}_i = \frac{1}{m}\left(\sum_k \mathbf{f}_k - k_d \mathbf{v}_i\right) \tag{4.1}$$

where m is the mass of all particles, and k_d a damping coefficient. $\mathbf{f}_k$ is the k^{th} force

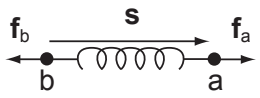


Figure 4.4: Linear Spring

exerted on particle i by a spring. We sum over all forces acting on particle i (the number of forces can vary for each particle). The damping force $k_d\mathbf{v}_i$ has been added so that the system will reach a rest state.

To determine the position $\mathbf{x}_i^t$ and velocity $\mathbf{v}_i^t$ of particle i at timestep t, we use Euler's method of numerical integration:

$$\begin{aligned} \mathbf{v}_i^t &= \mathbf{v}_i^{t-1} + \mathbf{a}_i^{t-1}\Delta t \\ \mathbf{x}_i^t &= \mathbf{x}_i^{t-1} + \mathbf{v}_i^{t-1}\Delta t \end{aligned} \tag{4.2}$$

where Δt is the length of time between each time step.

4.1.2 Spring Formulation

Two force-producing springs are constructed as follows. The first is a *linear* spring that obeys Hooke's law. We define a spring as existing between two particles a and b (Figure 4.4). Each spring has an associated rest length s_0. We define a vector $\mathbf{s} = \mathbf{x}_a - \mathbf{x}_b$ between particles a and b. The magnitude of force exerted by a spring is $k_s(s_o - |\mathbf{s}|)$, which is positive for compression. This force will be directed along the direction of the spring $\frac{\mathbf{s}}{|\mathbf{s}|}$. Combining these we obtain $\mathbf{f}_a$, the force on particle a as:

$$\mathbf{f}_a = k_s(s_o - |\mathbf{s}|)\frac{\mathbf{s}}{|\mathbf{s}|} \tag{4.3}$$

Newton's third law states that for every action, there must be an equal and

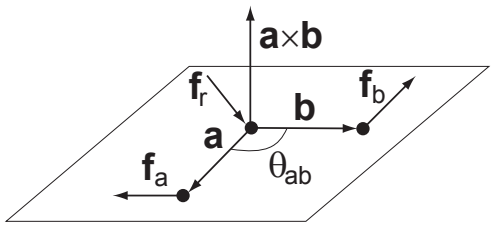


Figure 4.5: Bending Spring

opposite reaction [64], so the force $\mathbf{f}_b$ on particle b must be equal and opposite to $\mathbf{f}_a$:

$$\mathbf{f}_b = -\mathbf{f}_a = -k_s(s_o - |\mathbf{s}|)\frac{\mathbf{s}}{|\mathbf{s}|} \tag{4.4}$$

The other type of spring used is a *bending* spring. It is similar to the linear spring except that instead of responding to changes in length, it responds to changes in angle θ_{ab}. It exerts a moment proportional to displacements from a rest angle θ_0:

$$|\mathbf{f}_a| = \frac{k_s(\theta_o - \theta_{ab})}{|\mathbf{a}|} \tag{4.5}$$

The bending spring requires three points to define it (Figure 4.5). Vectors $\mathbf{a}$ and $\mathbf{b}$ are defined as going from the central mass to the outer masses. The restoring force must exert a moment, so $\mathbf{f}_a$ acts perpendicular to $\mathbf{a}$ in the plane of $\mathbf{a}$ and $\mathbf{b}$. This direction can be caluculated by:

$$\frac{\mathbf{f}_a}{|\mathbf{f}_a|} = \frac{\mathbf{a} \times (\mathbf{a} \times \mathbf{b})}{|\mathbf{a} \times (\mathbf{a} \times \mathbf{b})|} \tag{4.6}$$

The case for $\mathbf{f}_b$ can be determined similarly. From the dot product definition we can determine θ_{ab}:

$$\theta_{ab} = \cos^{-1}\left(\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right) \tag{4.7}$$

Combining 4.5, 4.6, and 4.7, we obtain:

$$\mathbf{f}_a = k_s\left[\theta_o - \cos^{-1}\left(\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right)\right]\frac{\mathbf{a}\times(\mathbf{a}\times\mathbf{b})}{|\mathbf{a}||\mathbf{a}\times(\mathbf{a}\times\mathbf{b})|} \tag{4.8}$$

and,

$$\mathbf{f}_b = k_s\left[\theta_o - \cos^{-1}\left(\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right)\right]\frac{(\mathbf{a}\times\mathbf{b})\times\mathbf{b}}{|\mathbf{b}||(\mathbf{a}\times\mathbf{b})\times\mathbf{b}|} \tag{4.9}$$

Since the forces must all be balanced, the reaction force on the central particle is:

$$\mathbf{f}_r = -(\mathbf{f}_a + \mathbf{f}_b) \tag{4.10}$$

This constitutes all the theory required for the physical simulation of a surface using a mass-spring system.

4.1.3 Morphogen Distribution Model

Diffusible morphogens are included in this mass-spring model, as described in Section 3.1. The initial morphogen distribution is user defined (implementation details are discussed in Chapter 6), and the effect of morphogens on the surface is described in the next section.

Associated with each particle i is a morphogen concentration p_i. Each particle can also be referenced by a set of uv coordinates (Figure 4.2), which I denote p^{uv}.

Each particle i has one set of unique uv coordinates. We begin by reformulating the pure diffusion formulation of equation 3.2 to include a decay term $-k_{decay}p$:

$$\frac{dp}{dt} = k_{diff} \bigtriangledown^2 p - k_{decay}p \tag{4.11}$$

where the morphogen diffusion and decay rates are specified by k_{diff} and k_{decay}. Without the decay term morphogen concentrations would not return to zero when sources are removed. To apply this to our discretized system, we use a finite-differencing explicit integration approach [68]:

$$p^{uv}_{new} = p^{uv} + k_{diff}\left(p^{u+1\,v} + p^{u-1\,v} + p^{u\,v+1} + p^{u\,v-1} - 4p^{uv}\right)\Delta t - k_{decay}p^{uv}\Delta t \tag{4.12}$$

Note that this assumes that the distances between particles is constant.

This is generalized to multiple morphogens by using a different symbol for other morphogens, such as q_i, r_i, s_i. In this model, no interactions exist between morphogens, such as those described for reaction-diffusion or the connectionist model.

4.1.4 Growth Formulation

An advantage to mass-spring systems is the way that growth is easily integrated. To grow the surface, we simply extend the rest length s_o of the spring. Isotropic growth is implemented by letting morphogens influence spring rest length as follows:

$$\frac{ds_o}{dt} = k_{sens}ps_o \tag{4.13}$$

Since morphogen concentrations are only defined at particles, the average of the two connected particles is used to determine its influence on a given spring.

Morphogen concentrations can also be used to modify curvature by changing the rest angle of bending springs:

$$\frac{d\theta_o}{dt} = k_{curl}p \tag{4.14}$$

where bending springs are influenced by the morphogen concentration of their central particle.

Finally, we can also implement anisotropic growth by modifying equation 4.13 to include a growth tensor:

$$\frac{ds_o}{dt} = k_{tsens}\mathbf{e}^T\mathbf{G}\mathbf{e}ps_o \tag{4.15}$$

where $\mathbf{G}$ is a 2D growth tensor, and $\mathbf{e}$ is a unit vector of the spring direction in uv space. This is the same RERG calculation presented in Section 2.3.2. There is no "obvious" way to determine how a tensor should diffuse over a surface, but in this work, each element G_{ij} of the growth tensor is treated as a separate morphogen. For example, if we simulate four morphogens p, q, r, s over the surface, then we could give $\mathbf{G}$ by:

$$\mathbf{G} = \begin{bmatrix} p & q \\ r & s \end{bmatrix} \tag{4.16}$$

This is our morphogenetic model for our mass-spring model in its entirety. The look and feel of this model is described in Chapter 6.

4.2 Finite Element Elasticity

The Finite Element Method (FEM) is a commonly used technique for solving differential equations over arbitrarily discretized domains. Its origins date back to the development of the digital computer in the 1940's. Courant [19] first used triangles for discretizing solution spaces in 1943. However, the term *finite element method* did not appear later until its first use by Clough [13] in 1960. The technique was later popularized by Zienkiewicz [98]. A more complete account of the history of the finite element method is given by Huebner *et al* [45].

FEM is commonly used by engineers for elasticity problems, such as calculating stresses and strains within structures. FEM is however applicable to a wide range of problems, including plastic deformation [65], heat distribution, and fluid dynamics [45].

Within computer graphics FEM has been used to model deformable objects [58], brittle objects [66], cloth, and inflating balloons [39]. This thesis is also based on the work of Federl [34], who used FEM to model crack formation on expanding surfaces.

A primary advantage of the finite element method is that it treats the solution space as continuous. In principle this allows for arbitrary discretization, not the regular discretization needed for finite differencing methods.

The process of Finite Element Analysis can be broken down into six basic steps [45]:

- Discretization
- Interpolation
- Elemental formulation

- Assembly
- Constraints
- Solution

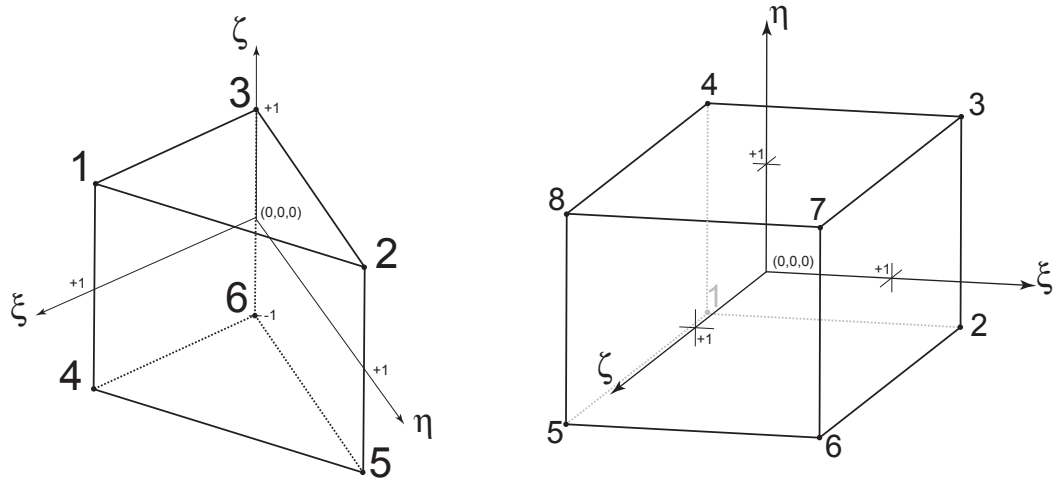
Each step is discussed in the following sections.

4.2.1 Discretization

For every type of problem considered with FEM, a state variable is considered over a domain. For thermal analysis, the variable would be temperature; for elasticity, this would be object deformation vectors (u, v, w as defined in Section 2.2.2). FEM allows for arbitrary discretization of this domain, so long as the unit of discretization, or *element*, can be defined with some type of interpolating function. Two example elements are the wedge or pentahedral element (Figure 4.6(a)), and the brick or hexahedral element (Figure 4.6(b)).

It is important to note that these elements describe a volume, and not a flat surface as this thesis proposes. FE methods exist to describe a surface using flat elements, known as *plate* models [27]. However, they are mathematically more complex than modeling a volume. I have chosen therefore to model the surface as a volume, with thickness, so that simpler methods could be used. Surfaces are discretized using wedge elements (Figure 4.7). It can be seen that the surface has a length and width, but also thickness.

We can define a matrix $[\mathbf{X}]^{(e)}$ that specifies the coordinates of each node in global space. For a wedge element, this is a 6×3 matrix:



(a) A 6-node wedge (pentahedral) element

(b) An 8-node brick (hexahedral) element

Figure 4.6: Two different types of elements. Bold numbers indicate local node numbering. Small numbers indicate auxiliary (ξ, η, ζ) coordinates.

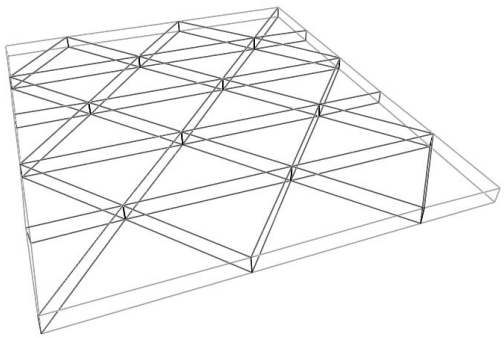


Figure 4.7: A surface discretized with wedge elements.

$$[\mathbf{X}]^{(e)} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ x_6 & y_6 & z_6 \end{bmatrix} \tag{4.17}$$

The subscripts indicate the local node numbers and x, y, z are the coordinates of the node in a global Cartesian coordinate system. We use the superscript $^{(e)}$ to indicate local *elemental* matrices that use local node enumeration as opposed to a global enumeration method, introduced in the following sections.

It can be shown for elasticity problems that as the number of elements used to discretize a domain tends towards infinity, the finite element solution will converge on the continuous solution [98].

4.2.2 Interpolation

FEM defines state variables that are associated with each node. These state variable can be interpolated between nodes through the use of interpolation or *shape* functions. A variable V can be interpolated at a point $\mathbf{p}$ within an element as the sum of a number of shape functions N_i belonging to that element:

$$V = \sum_{i=1}^{r} N_i(\mathbf{p}) V_i \tag{4.18}$$

where V_i are the nodal values and r is the number of nodes in the element. The brick and wedge elements in Figures 4.6(a) and 4.6(b) use auxiliary or local coordinates

ξ, η, ζ, which are used to specify the point $\mathbf{p}$. Elements that use auxiliary coordinates are known as isoparametric elements [18, p.202].

The set of six linear shape functions for the wedge element are as follows [45]:

$$\begin{aligned} N_1(\xi,\eta,\zeta) &= \tfrac{1}{2}\xi(1+\zeta) \\ N_2(\xi,\eta,\zeta) &= \tfrac{1}{2}\eta(1+\zeta) \\ N_3(\xi,\eta,\zeta) &= \tfrac{1}{2}(1-\xi-\eta)(1+\zeta) \\ N_4(\xi,\eta,\zeta) &= \tfrac{1}{2}\xi(1-\zeta) \\ N_5(\xi,\eta,\zeta) &= \tfrac{1}{2}\eta(1-\zeta) \\ N_6(\xi,\eta,\zeta) &= \tfrac{1}{2}(1-\xi-\eta)(1-\zeta) \end{aligned} \tag{4.19}$$

The above shape functions have the property of evaluating to $N_i = 1$ at node i, and 0 for all other nodes.

The nodal values for FEM elasticity are vectors defining a displacement field of the deformed state from the rest state (as defined in Section 2.2.2). The displacement $\tilde{\delta}$ at a point (ξ, η, ζ) within the element can be determined, if the displacements of all the nodes δ_i are known:

$$\tilde{\delta}(\xi,\eta,\zeta) = \sum_{i=1}^{N} N_i(\xi,\eta,\zeta)\delta_i \tag{4.20}$$

This can also be written in matrix notation:

$$\{\tilde{\delta}\}^{(e)} = [\mathbf{N}]\{\delta\}^{(e)} \tag{4.21}$$

if we define:

$$\{\delta\}^{(e)} = \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \\ \vdots \\ u_r \\ v_r \\ w_r \end{Bmatrix} \tag{4.22}$$

where u_i, v_i, w_i are the deformation of node i as defined in Section 2.2.2, and r is the number of nodes in the element ($r = 6$ for wedge elements). I also introduce $[\mathbf{N}]$ the *interpolation function matrix*, defined as:

$$[\mathbf{N}] = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & ... & N_r & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & ... & 0 & N_r & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & ... & 0 & 0 & N_r \end{bmatrix} \tag{4.23}$$

Note that since the shape functions N_i are a function of $\xi\eta\zeta$, the matrix $[\mathbf{N}]$ is also a function of $\xi\eta\zeta$, and should correctly be written $[\mathbf{N}(\xi, \eta, \zeta)]$. However this notation is cumbersome and (ξ, η, ζ) is here omitted for $[\mathbf{N}]$ and all other matrices that contain shape functions N_i.

I also introduce here $[\mathbf{J}]$, the *Jacobian matrix*, defined as:

$$[\mathbf{J}] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \tag{4.24}$$

where x, y, z are the global coordinates. The Jacobian can be thought of as a scaling matrix, that relates physical lengths x, y, z to reference lengths ξ, η, ζ. Note that $[\mathbf{J}]$ is not constant throughout wedge elements, and can be calculated as follows for any six-node element as:

$$[\mathbf{J}] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} & \frac{\partial N_5}{\partial \xi} & \frac{\partial N_6}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} & \frac{\partial N_5}{\partial \eta} & \frac{\partial N_6}{\partial \eta} \\ \frac{\partial N_1}{\partial \zeta} & \frac{\partial N_2}{\partial \zeta} & \frac{\partial N_3}{\partial \zeta} & \frac{\partial N_4}{\partial \zeta} & \frac{\partial N_5}{\partial \zeta} & \frac{\partial N_6}{\partial \zeta} \end{bmatrix} [\mathbf{X}^{(e)}] \tag{4.25}$$

4.2.3 Elemental Formulation

Until now I have discussed a framework for representing state variables, giving both a way to discretize the solution space, and a method to interpolate state variables. The next step is to define the behaviour of an element.

I begin by defining the elemental stiffness matrix and give an explanation of what it represents. The general idea is that we wish to determine the deformation of an object given the forces acting on it, or the forces given the deformation. We therefore have two sets of nodal variables: deformations and forces. I have already introduced $\{\delta\}^{(e)}$ as the vector of nodal deformations. We can similarly define $\{\mathbf{F}\}^{(e)}$, a vector of nodal forces. This can conceptually be thought of as the forces acting at the nodes. Finite Element elasticity relates these nodal deformations and forces through a *stiffness matrix* $[\mathbf{K}]^{(e)}$:

$$[\mathbf{K}]^{(e)}\{\delta\}^{(e)} = \{\mathbf{F}\}^{(e)} \tag{4.26}$$

An intuitive explanation of the stiffness matrix for elasticity is as follows. For the six node wedge element, we have a total of 18 degrees of freedom (ie. XYZ for each node), making $[\mathbf{K}]^{(e)}$ an 18×18 matrix. An element K_{ij} is the force exerted on the i^{th} degree of freedom, when all the deformations are fixed at zero, except for the j^{th} degree, which is "pulled" to one unit of deformation.

The stiffness matrix can be derived from the generalized Hooke's law (Section 2.2.4). A complete derivation is given in appendix A.1. The elemental stiffness matrix for elasticity can be calculated as follows [45]:

$$[\mathbf{K}]^{(e)} = \int_{\Omega^{(e)}} [\mathbf{B}]^T [\mathbf{C}][\mathbf{B}] d\Omega \tag{4.27}$$

where $\Omega^{(e)}$ is the volume of the element integrated, and $[\mathbf{C}]$ is as defined in Section 2.2.4. In equation 4.27 I am introducing $[\mathbf{B}]$, the *strain interpolation matrix* [45] given by:

$$[\mathbf{B}] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & 0 & ... & \frac{\partial N_r}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & ... & 0 & \frac{\partial N_r}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial z} & ... & 0 & 0 & \frac{\partial N_r}{\partial z} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & 0 & ... & \frac{\partial N_r}{\partial y} & \frac{\partial N_r}{\partial x} & 0 \\ \frac{\partial N_1}{\partial z} & 0 & \frac{\partial N_1}{\partial x} & ... & \frac{\partial N_r}{\partial z} & 0 & \frac{\partial N_r}{\partial x} \\ 0 & \frac{\partial N_1}{\partial z} & \frac{\partial N_1}{\partial y} & ... & 0 & \frac{\partial N_r}{\partial z} & \frac{\partial N_r}{\partial y} \end{bmatrix} \tag{4.28}$$

where r is the number of nodes in the element. $[\mathbf{B}]$ allows us to calculate the strain vector (Section 2.2.2) at any point in the element: $\{\epsilon\} = [\mathbf{B}]\{\delta\}^{(e)}$.

In practice integrating equation 4.27 analytically is impractical, if not impossible. It is much easier to use Gaussian quadrature [68, 18].

$$\mathbf{K}^e = \sum_{gauss} [\mathbf{B}(\xi,\eta,\zeta)]^T[\mathbf{C}][\mathbf{B}(\xi,\eta,\zeta)]|[\mathbf{J}(\xi,\eta,\zeta)]|w_{gauss} \tag{4.29}$$

This is simply a re-written form of equation 4.27 to reflect the numerical integration method. The terms $[\mathbf{B}]^T[\mathbf{C}][\mathbf{B}]$ are the same as equation 4.27 except that their dependence on (ξ,η,ζ) is explicitly stated. The Gauss points are dependent on the shape integrated, and the amount of accuracy required. The set of points (ξ,η,ζ) and weights w_{gauss} used are listed in appendix A.2. In equation 4.27, the integration occurs in global coordinate space, while here the integration uses auxiliary coordinates. The determinant of the Jacobian matrix $|[\mathbf{J}(\xi,\eta,\zeta)]|$ is needed to correct for this [18]. It will be assumed that all integrals over the space of the element $\Omega^{(e)}$ can be integrated using this method and will only be given in their continuous form.

4.2.4 Assembly

Until now, I have only focused on the behaviour of an *individual element*. We now wish to describe the behaviour of a set of connected of elements. This is a simple matter of *assembling* elemental stiffness matrices and vectors into a global form:

$$[\mathbf{K}]\{\delta\} = \{\mathbf{F}\} \tag{4.30}$$

The global stiffness matrix is constructed by first defining a global numbering scheme for all nodes in the object. Then, there exists a mapping between the local node numbering (Figure 4.6(a)) and the global node numbering. The individual

stiffness matrices are recast using this global numbering and summed together to form the *global stiffness matrix*:

$$[\mathbf{K}] = \sum_{i=1}^{N} [\mathbf{K}_i]^{(e)} \tag{4.31}$$

where N is the total number of elements. Translating the indices of these elemental matrices and summing the matrices is known as assembly.

4.2.5 Constraints

We now have described enough to determine the force-displacement relationship of a set of elements. However, for many practical problems constraints also need to be specified. I present a method, described by Huebner *et al* [45], for modifying the form of equation 4.30 to constrain displacements to a fixed value.

To fix a displacement δ_i at value β_i, all elements of row i and column i in $[\mathbf{K}]$ are set to 0, except for element K_{ii} which is set to 1. Then element F_i is set to β_i and every remaining element j of $\{\mathbf{F}\}$ has $K_{ji}\beta_i$ subtracted from it, where K_{ij} is the original unmodified element of $[\mathbf{K}]$. This has the effect or re-arranging the linear system, such that the condition $\delta_i = \beta_i$ is always met [45].

For example, we could take the following system:

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{Bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix} \tag{4.32}$$

and constrain $\delta_1 = \beta_1$ using the above stated method to obtain:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & K_{22} & K_{23} & K_{24} \\ 0 & K_{32} & K_{33} & K_{34} \\ 0 & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{Bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{Bmatrix} = \begin{Bmatrix} \beta_1 \\ F_2 - K_{21}\beta_1 \\ F_3 - K_{31}\beta_1 \\ F_4 - K_{41}\beta_1 \end{Bmatrix} \tag{4.33}$$

This method could be used to "anchor" the points of an object, by setting the displacement to 0. It is important to note that it is usually necessary to specify constraints, otherwise [**K**] is likely to be singular. That is, if some displacements are not constrained, then rigid body motion would satisfy all conditions, allowing for an infinite number of solutions [45].

4.2.6 Solving

Equation 4.30 is typically solved using linear algebra methods. Gaussian elimination is not practical for many of these matrices since their dimensions may approach 6000×6000, as they did for some of the examples in Chapter 8. [**K**] is sparse for elasticity problems [45], and therefore well suited to sparse matrix data structures and Conjugate-Gradient (CG) [68] solvers.

A CG solver was used to solve equation 4.30. The sparse matrix was stored as a 1D array of C++ Standard Template Library (STL) maps [81, 61]. The CG solver requires only two vector dot products and one matrix-vector product per iteration, which are efficiently calculated with this data structure[79].

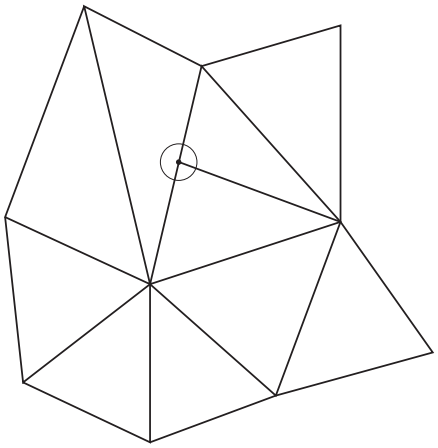


Figure 4.8: A non-conforming mesh with a "T-junction" (circled).

4.2.7 Dynamic Subdivision

A primary reason for using FEM is that it is possible to discretize a surface arbitrarily, unlike like mass-spring systems, which require a regular discretization. FEM also allows for dynamic subdivision, whereby the mesh can be further refined between time steps. This is useful for elements that have become too large as a result of growth that can be subdivided into a number of smaller elements, allowing more detail to be added.

In selecting an appropriate subdivision algorithm, the following requirements must be met. The ideal algorithm should:

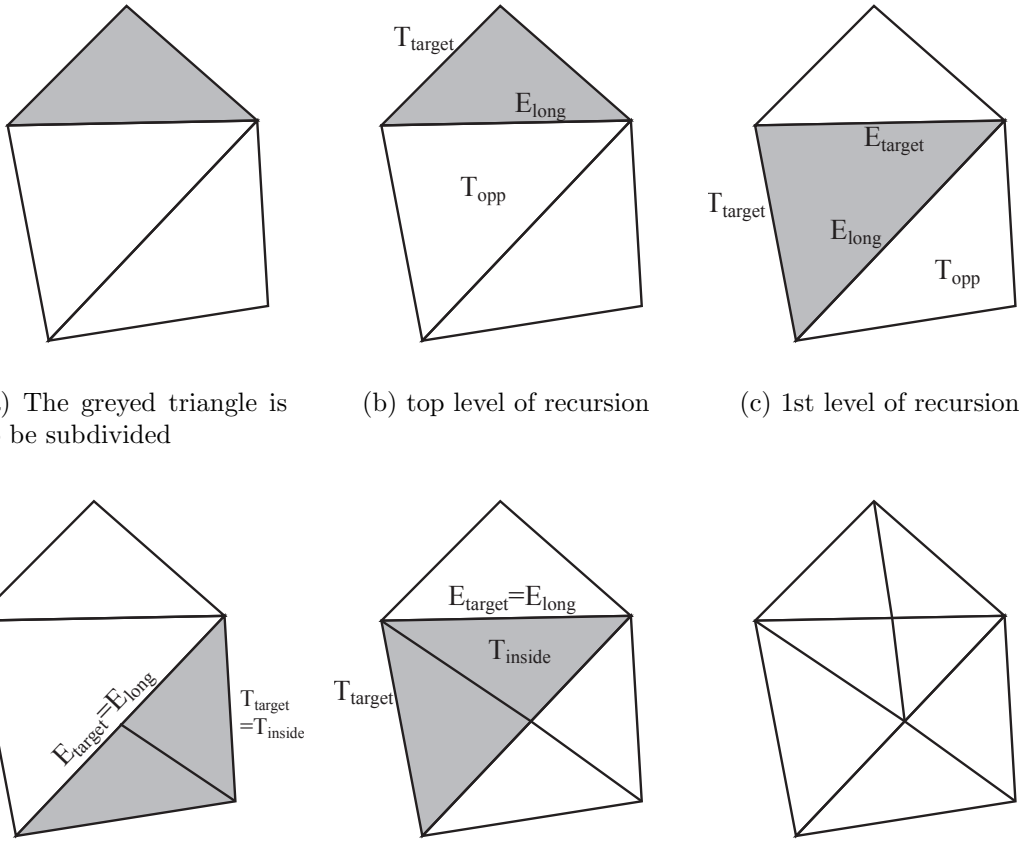
- turn triangles into triangles,
- maintain original points,
- maintain a *conforming* mesh.

A *conforming* mesh is a mesh without any "T-junctions" (Figure 4.8). T-junctions are *non-conforming points* which can formally be defined as "an interior point of the side of one triangle and common vertex of two other triangles" [76]. FEM meshes with non-conforming points will not generate correct result [45].

An algorithm proposed by Rivara [76], and used in FEM work by Federl [34] satisfies these requirements. The algorithm refines a mesh by bisecting only the longest edge of a triangle. To maintain a conforming mesh, the triangle opposite to the edge must also be bisected. If the shared edge is not the longest edge of the opposite triangle, then the opposite triangle is bisected, using this algorithm, until both triangles share the longest edge. Because this algorithm only divides the longest edge, it has the added advantage of generally avoiding degenerate triangles.

The method is listed in Algorithm 1. The first argument to `SubDivideTri` is T_{target}, the target triangle to be divided. The second argument E_{target} is the target edge to be divided. The function will not return until that edge is divided. If no edge is specified ($E_{target} = \phi$, the empty set) then the longest edge is divided.

An example execution is illustrated in Figure 4.9. The shaded triangle in Figure 4.9(a) is T_{target} and the function is called called with no edge specified (`SubDivideTri(` T_{target}, ϕ `)`). Figure 4.9(b) identifies T_{target}, T_{opp} and E_{long} at the top level of recursion. The algorithm will divide the opposite triangle first, before dividing T_{target}.The algorithm recuses, calling `SubDivideTri(` T_{opp}, E_{long}`)` to do so. Because T_{target} and T_{opp} do not share the longest edge, the algorithm recuses twice (Figures 4.9(c) and 4.9(d)). Recursion is terminated when they share the longest edge. Triangle T_{target} is divided, and execution proceeds back up the recursion chain. The final mesh is shown in Figure 4.9(f).



(a) The greyed triangle is to be subdivided

(b) top level of recursion

(c) 1st level of recursion

(d) after 2nd level of recursion

(e) Triangle T_{target} is subdivided until $E_{target} = E_{long}$.

(f) final mesh

Figure 4.9: A subdivision sequence for a mesh. See text. (The shaded triangle is always T_{target}.)

Algorithm 1 Subdivision algorithm (ϕ is the empty set).

```
1: SubDivideTri(T_target, E_target)
2: E_long ← longest edge of T_target
3: if E_target = φ then {divide longest edge}
4:    T_opp ← triangle opposite edge E_long
5:    if T_opp ≠ φ then {we have not reached the edge of the mesh}
6:       SubDivideTri(T_opp, E_long)
7:    else
8:       bisect E_long
9:    end if
10:   divide triangle T_target
11: else {E_target must get bisected}
12:   while edge E_target not divided do
13:      T_inside ← triangle inside T_target and containing E_long
14:      SubDivideTri(T_inside, φ)
15:   end while
16: end if
```

Elements are subdivided when the area of their top triangular face exceeds a user defined value. This stops individual elements from getting too large, and allows more detail to be added as the surface gets larger.

4.3 Finite Element Diffusion

The second part of our problem statement specifies that we should be able to simulate diffusible morphogens over the surface. The theory for morphogen diffusion (Section 3.1) is the same as that for heat transfer [49], and is quite easily simulated using FEM. In the context of thermal analysis, morphogen concentrations become temperatures, and morphogen sources become thermal loads.

Many of the FEM elasticity techniques discussed in Section 4.2 are directly applicable to heat transfer. We have the advantage of using the same discretization,

interpolation, assembly, solving and subdivision methods as those used for the elasticity formulation. The differences needed for simulating heat transfer are discussed in this section.

4.3.1 Basic Formulation

FEM heat transfer is mathematically very similar to the FEM elasticity formulation. Instead of deformations, we have temperatures $\{\mathbf{T}\}$ and instead of forces, thermal loads $\{\mathbf{R}\}$.

The constitutive equations of heat transfer are based on Fourier's law. Fourier's law states [45, p.319] that:

$$\left\{ \begin{array}{c} q_x \\ q_y \\ q_z \end{array} \right\} = - \left[\begin{array}{ccc} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{array} \right] \left\{ \begin{array}{c} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \\ \frac{\partial T}{\partial z} \end{array} \right\} \tag{4.34}$$

where q_x, q_y and q_z are the heat flow in the x, y, z directions respectively. In the context of thermal analysis these are energy fluxes. In the context of diffusion, these are analogous to morphogen fluxes. Note that this is a vector quantity, and not a tensor quantity. In thermal analysis $\{\mathbf{T}\}$ is the temperature, while for diffusion $\{\mathbf{T}\}$ can be regarded as the morphogen concentration. We herein use the terminology as it applies to thermal analysis. Finally, $[\mathbf{k}]$ is the *symmetric conductivity tensor*. For isotropic media this is an identity matrix times a constant $[\mathbf{k}] = k[\mathbf{I}]$. I do not consider anisotropic media in this thesis.

From $[\mathbf{k}]$ we can define an *element conductance matrix*, which is analogous to the elasticity stiffness matrix [45]:

$$[\mathbf{K}_c]^{(e)} = \int_{\Omega^{(e)}} [\mathbf{B}_c]^T [k][\mathbf{B}_c] d\Omega \tag{4.35}$$

where instead $[\mathbf{B}_c]$ is the *temperature-gradient interpolation matrix*, defined as:

$$[\mathbf{B}_c] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_r}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \dots & \frac{\partial N_r}{\partial y} \\ \frac{\partial N_1}{\partial z} & \frac{\partial N_2}{\partial z} & \dots & \frac{\partial N_r}{\partial z} \end{bmatrix} \tag{4.36}$$

which can be used to compute the gradient of the temperature field $[\mathbf{B}_c]\{\mathbf{T}\}^{(e)}$.

Similar to elasticity global formulation (equation 4.30), we can formulate the global thermal system as follows:

$$[\mathbf{K}_c]\{\mathbf{T}\} = \{\mathbf{R}_Q\} \tag{4.37}$$

where $[\mathbf{K}_c]$ is the *global conductance matrix*, $\{\mathbf{T}\}$ the temperature vector, and $\{\mathbf{R}_Q\}$ is the thermal load vector. An element K_{ij} of $[\mathbf{K}_c]$ is the amount of thermal energy flux required at node j, to bring node i to one unit of temperature. Interpreting this in the context of diffusion, an element K_{ij} is the morphogen flux required at node j to bring node i to a concentration of one unit.

Similar to FEM elasticity, equation 4.37 can be solved with a CG solver to obtain the steady state temperature vector $\{\mathbf{T}\}$.

4.3.2 Thermal Convection

Thermal convection is the process by which a body is cooled by an external fluid (typically air). The rate of energy lost to the fluid is proportional to the temperature difference between the body and the fluid. It is possible to include a thermal

convection component in FEM that obeys $q_h = h(T_s - T_e)$, where h is a convection coefficient, T_s and T_e are the body and fluid temperature respectively, and q_h is the heat loss from the body.

Within the context of chemical diffusion, this process has the same effect as morphogen decay. That is, if T_e is set to zero (ie. the ambient fluid temperature is zero) we are left with $q_h = hT_s$, the equivalent of morphogen decay (q_h is a *rate* of energy loss that is proportional to the temperature). We also then have the option of setting T_e to non-zero values, such that the temperature will "decay to" a non-zero value.

To implement thermal convection, two convection terms, $[\mathbf{K}_h]$ and $\{\mathbf{R}_h\}$,are defined as done by Huebner *et al* [45, p.320]:

$$[\mathbf{K}_h]^{(e)} = \int_{\Omega^{(e)}} h\{\mathbf{N}\}\lfloor\mathbf{N}\rfloor d\Omega \tag{4.38}$$

$$\{\mathbf{R}_h\}^{(e)} = \int_{\Omega^{(e)}} hT_e\{\mathbf{N}\}d\Omega \tag{4.39}$$

where $\lfloor\mathbf{N}\rfloor$ is the row vector of shape functions:

$$\lfloor\mathbf{N}\rfloor = \{\mathbf{N}\}^T = \left\{ \begin{array}{cccc} N_1 & N_2 & ... & N_r \end{array} \right\} \tag{4.40}$$

The assembled global convection terms $[\mathbf{K}_h]$ and $\{\mathbf{R}_h\}$ are integrated into equation 4.37, obtaining:

$$([\mathbf{K}_c] + [\mathbf{K}_h])\{\mathbf{T}\} = \{\mathbf{R}_Q\} + \{\mathbf{R}_h\} \tag{4.41}$$

Note that this equation still only represents a steady-state solution.

4.3.3 Transient Formulation

Since we are also interested in the time evolution of heat transfer we must reformulate equation 4.41 for transient analysis. Equation 4.41 can be modified to include a time dependence [45, p.321]:

$$[\mathbf{C}]\{\dot{\mathbf{T}}(t)\} + ([\mathbf{K}_c] + [\mathbf{K}_h])\{\mathbf{T}(t)\} = \{\mathbf{R}_Q(t)\} + \{\mathbf{R}_h(t)\} \tag{4.42}$$

where we are rewriting the vectors of equation 4.41 to reflect a time dependence, and introducing $\dot{\mathbf{T}}(t)$, the first derivative of temperature with respect to time, and $[\mathbf{C}]$, the capacitance matrix. The term $[\mathbf{C}]\{\dot{\mathbf{T}}(t)\}$ essentially represents the energy flux needed to increase the temperature at a given rate. The matrix $[\mathbf{C}]$ is given by [45]:

$$[\mathbf{C}] = \int_{\Omega^{(e)}} \rho c \{\mathbf{N}\} \lfloor \mathbf{N} \rfloor d\Omega \tag{4.43}$$

where ρ is the mass density of the material, and c is the thermal capacitance of the material. Taken together ρc is a measure of the amount of energy required per unit volume to raise the temperature one unit. In the context of diffusion, these can be taken to be one parameter and considered to be the amount of morphogen required per unit volume to raise the concentration by one unit.

Solving 4.43 for a problem also requires a time integration method. We use the following implicit method from [45]:

$$[\bar{\mathbf{K}}]\{\mathbf{T}\}^t = \{\bar{\mathbf{R}}\}^t \tag{4.44}$$

$$[\bar{\mathbf{K}}] = \theta\,[[\mathbf{K}_c] + [\mathbf{K}_h]] + \frac{1}{\Delta t}[\mathbf{C}] \tag{4.45}$$

$$\{\bar{\mathbf{R}}\}^t = \frac{1}{\Delta t}[\mathbf{C}]\{\mathbf{T}\}^{t-1} + (\{\mathbf{R}_Q(t)\} + \{\mathbf{R}_h(t)\})^t \tag{4.46}$$

where Δt is the time step. A complete derivation of this set of equations is given by Huebner *et al* [45, p.332] for example. Using this method guarantees that the solution is unconditionally stable [45].

4.3.4 Connectionist Method

The previous section described a method to simulate morphogens that diffuse over a surface and decay. I also wish to incorporate interactions between morphogens based on the connectionist model presented in Section 3.4. This would enable morphogens to promote or inhibit other morphogens, allowing more complex models to be constructed.

We begin by re-examining the connectionist formulation summarized in equation 3.5. Note that the terms $D_i\nabla^2 T_i - \gamma_i T_i$ are present for diffusion and decay. Since the FEM diffusion model handles morphogen diffusion and decay, we can neglect these terms. We multiply the remaining term by Δt to obtain the interaction contribution V_i, for one time step:

$$V_i = R_i g_i \left(\sum_{j=1}^{N} W_{ij} T_j + h_i \right) \Delta t \tag{4.47}$$

where $\mathbf{W}$ is the connection matrix, R_i a user definable rate constant, and h_i are elements of the *offset vector* $\{\mathbf{h}\}$, also user defined. The thresholding function used was:

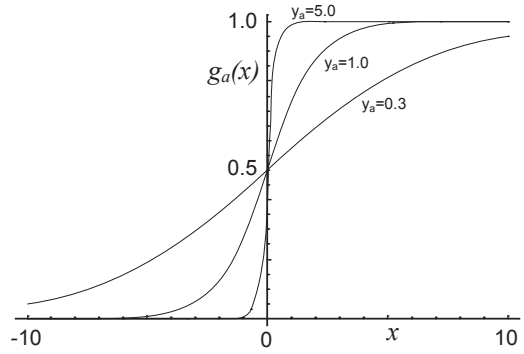


Figure 4.10: Thresholding function g_i (equation 4.48) for various y_i.

$$g_i(x) = 1 - \frac{1}{1 + e^{y_i x}} \tag{4.48}$$

where y_i is an exponent multiplier that is set by the user. Setting y_i higher results in a much sharper threshold (Figure 4.10).

We collect each of these contributions into an *interaction component vector* $\{\mathbf{V}\}$, and rewrite equation 4.44 to include this interaction component:

$$\{\mathbf{T}\}^t = [\bar{\mathbf{K}}]^{-1}\{\bar{\mathbf{R}}\}^t + \{\mathbf{V}\}^t \tag{4.49}$$

This has the effect of adjusting the temperature vector after each simulation time step.

4.4 Growth

A key contribution of this thesis is the integration of the growth discussed in Chapter 2 into FEM. With mass-spring systems, we were able to simply extend the rest length of the spring. To begin with, we note that this is mathematically equivalent to placing

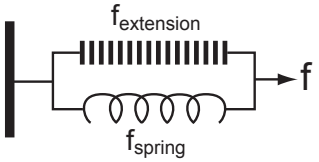


Figure 4.11: A force element in parallel with a spring element.

a constant force on the spring.

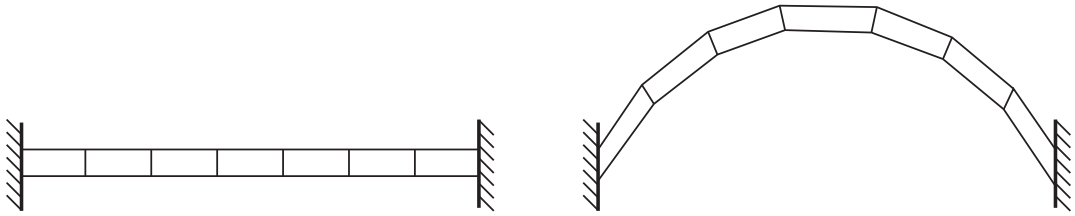
$$\begin{aligned} f &= k_s((s_0 + ds_0) - s) \\ f &= k_s ds_0 + k_s(s_0 - s) \\ f &= f_{extension} + f_{spring} \end{aligned} \tag{4.50}$$

We can construct an equivalent *spring-force* model as shown in Figure 4.11. In this model the spring rest length remains constant but the force element is responsible for extending the length of the spring. FEM elasticity is based on generalized Hooke's law, a 3D extension of the 1D Hooke's law used in the previous spring example. FEM handles element expansion in the same way, by exerting internal forces to expand the element.

The problem of trying to use this method for growth within linear FEM, is that it gives incorrect results for large expansions (typically greater that 10%). This is because the linear formulation assumes that the base shape stays in relatively the same configuration. In reality $[\mathbf{K}]$, and often $\{\mathbf{F}\}$, are a function of $\{\delta\}$ which becomes a factor in large deformations [18, p.595]:

$$[\mathbf{K}(\delta)]\{\delta\} = \{\mathbf{F}(\delta)\} \tag{4.51}$$

This is a much more advanced type of analysis known as *non-linear* FEM, which we



(a) before (b) after

Figure 4.12: A bar fixed between two immovable obstacles is set to expand 50% longitudinally (see text).

wish to avoid because of its greater complexity.

To further elaborate, an example is shown in Figure 4.12. In it, we have a beam between two fixed walls. We set each element to expand lengthwise by 50%. In the linear analysis, the bar will remain exactly as shown in Figure 4.12(a). A non-linear analysis will give the correct result, shown in Figure 4.12(b). This is because elements of the beam rotate significantly, changing their elemental stiffness matrices.

To implement growth within FEM and avoid non-linear analysis, we build upon the *viscoelastic* model, first used within computer graphics by Terzopolous *et al* [86, 85]. A canonical Maxwell viscoelastic element is shown in Figure 4.13, which is a spring in series with a damper element. The damper displacement x changes at a rate proportional to the force f applied: $\frac{dx}{dt} = k_d f$. If the system is initially in equilibrium and point b is moved one unit to the right at time $t = 0$, the viscoelastic unit will give the response shown in Figure 4.14. The time constant $\tau_{plasticity}$ is the amount of time required for the base shape to take on e^{-1} of the deformation.

To implement this viscoelastic behaviour within FEM I first restate some obser-

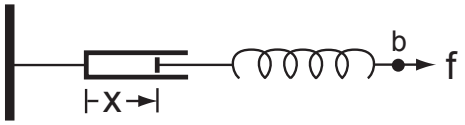


Figure 4.13: A Maxwell viscoelastic element. A spring is in series with a damper.

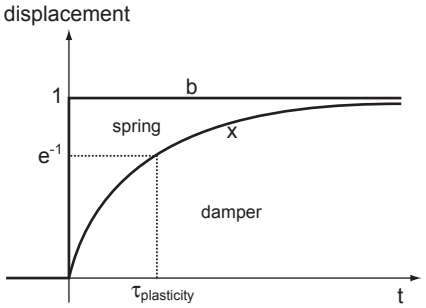


Figure 4.14: How a viscoelastic object deforms over time.

vations of Terzopoulos *et al.* The extension of the damper element x can be taken to be the base shape of the object, and the spring deformation δ as the elastic deformation from this base shape. Incorporating the behaviour of the Maxwell element into FEM elasticity is then a matter of setting $\frac{dx}{dt} = \eta\delta$, where η is an arbitrary viscosity coefficient. In discrete terms this means $x^{t+1} = x^t + \alpha\delta$, which can be generalized to three dimensions. Using the notation defined for FEM elasticity we have:

$$\{\mathbf{X}\}^t = \{\mathbf{X}\}^{t-1} + \alpha\{\delta\} \tag{4.52}$$

where $\{\mathbf{X}\}$ is the global analogue of $[\mathbf{X}]^{(e)}$, but organized into a column vector:

$$\{\mathbf{X}\}^T = \lfloor\ x_1 \quad y_1 \quad z_1 \quad x_2 \quad y_2 \quad z_2 \quad \ldots \quad x_r \quad y_r \quad z_r\ \rfloor \tag{4.53}$$

Here, r is the total number of nodes in the system, x, y, z are the coordinates of

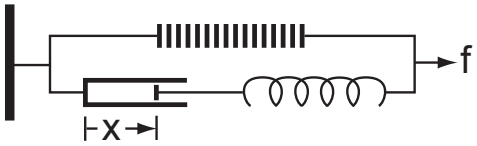


Figure 4.15: Viscoelastic growth element. A force element is in parallel with a Maxwell viscoelastic element.

a node in a global Cartesian coordinate system, and subscripts indicate the global node number. This equation is used after each simulation time step to update the base shape of the object. This method emulates the behaviour shown in Figure 4.14.

Growth is incorporated by combining the spring-force model (Figure 4.11) with the Maxwell element (Figure 4.13). I dub this *viscoelastic growth*. The conceptual model is shown in Figure 4.15. Growth occurs by the element exerting an internal force that expands the element. The change is made permanent when the damper of the viscoelastic element extends to accommodate the expansion. The generation of this force is discussed in the following sections.

It is important to note the difference between elastic and inelastic deformation (such as viscoelasticity). An elastic object will always returns to its original shape when all outside forces are removed [86, 85]. An inelastic object will *not* return to its original shape; its deformation is history dependent. The viscoelastic growth model is subject to this history dependence.

The primary advantage of using viscoelastic growth is that a number of small incremental linear steps can be used to achieve large deformations over time, allowing us to use linear FEM.

The nice analogy for the viscoelastic growth model is that it corresponds well with biological processes [44, p.303]: cells have an internal *turgotic* pressure. This

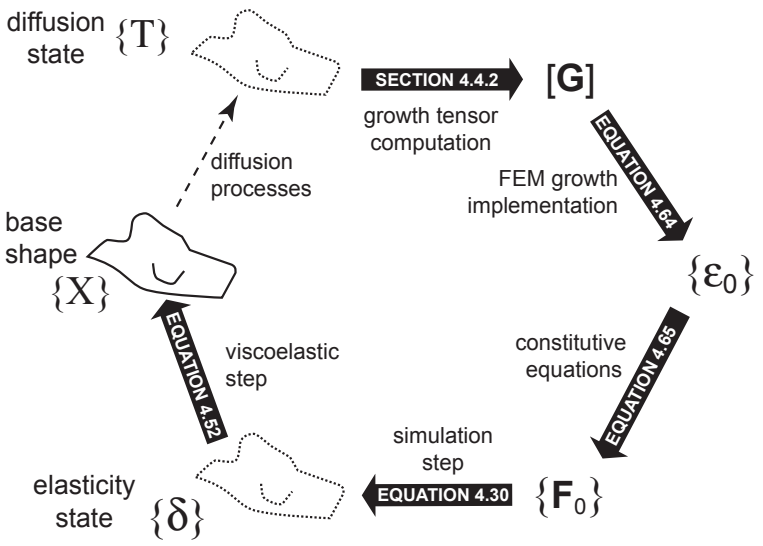


Figure 4.16: Process for simulating a single time step of growth.

pressure could be assumed to be the driving force of growth, extending the size of the cell (or the spring by analogy). If this shape is held long enough, the cell can add more material to the cell walls, making the shape permanent, much like our viscoelastic growth model.

4.4.1 System Overview

An overview of the system used to compute a single time step of growth is shown in Figure 4.16. We begin at the base shape, or current surface geometry. This shape is used as the basis for the diffusion of morphogens, as described using the theory presented in Section 4.3. No direct information transfer occurs, but the shape affects how morphogens diffuse, so the arrow is drawn with a dotted line.

After the diffusion time step has been simulated and connectionist contributions

have been added (equation 4.49), the state of morphogen distribution $\{\mathbf{T}\}$ is used to derive a growth tensor $[\mathbf{G}]$ for each element. Section 4.4.2 describes how this is done.

I stated above that the viscoelastic growth element uses forces to achieve growth. The next step is to determine these forces, given the growth tensor. We do this by first computing a strain vector $\{\epsilon_0\}$ from the growth tensor as described in Section 2.4. Applying this strain vector to FEM elasticity results in a force $\{\mathbf{F}_0\}$.

The final step is to determine the deformation of the surface, given the forces acting on the elements. This is done using the standard FEM elasticity solution step (Section 4.2.6). Solving the system results in a deformation $\{\delta\}$, of which a portion is added to the base shape (equation 4.52) to achieve viscoelastic behavior.

The following sections describe the process in more detail.

4.4.2 Growth Tensor Computation

In this section I discuss how to define a mapping between morphogen concentrations $\{\mathbf{T}\}$ and a growth for each element.

The simplest case of isotropic growth is easy to define, but the anisotropic cases require more consideration. More specifically, if growth is anisotropic, it requires a preferential direction of growth. The most obvious way to define this was to set this direction to be the gradient of a morphogen concentration.

To quantify this mathematically, it is first necessary to define a growth coordinate system $\{\vec{v}_{max}, \vec{v}_{min}, \vec{v}_{up}\}$ for each element. We then construct a growth tensor $[\mathbf{G}]$ (Figure 4.17) in this coordinate system, and then transform it to the global coordinate system.

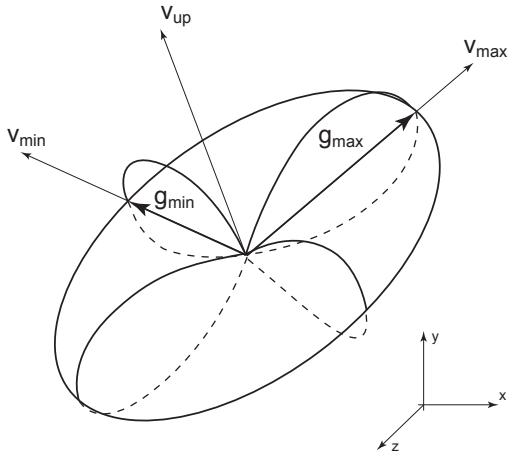


Figure 4.17: Diagram of the growth tensor indicatrix.

Since we are operating on a surface, we first define an "up" vector v_{up} that is the surface normal. One might suggest that ζ of the auxiliary coordinate system is the most logical choice for this vector. However, it is possible for elements to become sheared, shifting ζ significantly from the normal. To avoid this we take the cross product of ξ and η. We define our normalized up vector as follows:

$$\vec{\xi_g} = [\mathbf{J}]^T \left\{ \begin{array}{c} 1 \\ 0 \\ 0 \end{array} \right\} \tag{4.54}$$

$$\vec{\eta_g} = [\mathbf{J}]^T \left\{ \begin{array}{c} 0 \\ 1 \\ 0 \end{array} \right\} \tag{4.55}$$

$$\vec{v}_{up} = \frac{\vec{\xi_g} \times \vec{\eta_g}}{|\vec{\xi_g} \times \vec{\eta_g}|} \tag{4.56}$$

where $[\mathbf{J}]$ is computed at the centroid of the element (for a wedge element $(\xi, \eta, \zeta) = (\frac{1}{3}, \frac{1}{3}, 0)$).

The temperature-gradient interpolation matrix allows us to conveniently compute the gradient of a morphogen:

$$\vec{v}_{grad} = [\mathbf{B}_c]\{\mathbf{T}\}^{(e)} \tag{4.57}$$

If $\vec{v}_{grad}$ is to be used as the direction of maximal growth $\vec{v}_{max}$, it must be restricted to the plane perpendicular to $\vec{v}_{up}$ to ensure that the growth coordinate system is orthogonal. This can be achieved with the following formula:

$$\vec{v}_{max} = \vec{v}_{grad} - \vec{v}_{up}\,(\vec{v}_{grad} \cdot \vec{v}_{up}) \tag{4.58}$$

$\vec{v}_{max}$ is then normalized, and becomes another axis of the coordinate system. If isotropic growth is needed, or the gradient is undetectable, then $\vec{v}_{max}$ is arbitrary and we substitute $\vec{\xi_g}$ for $\vec{v}_{grad}$ in the above equation.

The last axis $\vec{v}_{min}$ is the cross product of these two:

$$\vec{v}_{min} = \vec{v}_{up} \times \vec{v}_{max} \tag{4.59}$$

We then specify the growth coordinate system as a 3×3 matrix:

$$[\mathbf{H}] = [\{\vec{v}_{max}\}\,\{\vec{v}_{min}\}\,\{\vec{v}_{up}\}] \tag{4.60}$$

We can then define the growth tensor in this coordinate system as follows:

$$[\mathbf{G}]' = \begin{bmatrix} g_{max} & 0 & 0 \\ 0 & g_{min} & 0 \\ 0 & 0 & g_{up} \end{bmatrix} \tag{4.61}$$

where g_{max}, g_{min} and g_{up} are typically derived from morphogen concentration, but dependent on the model being simulated. The simplest case is the following:

$$\begin{aligned} g_{max} &= k_{growth} T_i \\ g_{min} &= k_{anisotropy} g_{max} \\ g_{up} &= 0 \end{aligned} \tag{4.62}$$

We set the amount of growth proportional to the concentration of a morphogen i times a constant k_{growth}. The growth in the minor direction is then proportional to this times an anisotropy constant $k_{anisotropy}$. Setting $k_{anisotropy}$ to 1 results in isotropic growth; setting it to 0 results in growth only in the major direction of growth. g_{up} is set to zero because we do not want the surface to get any thicker.

The final step is to transform $[\mathbf{G}]'$, a tensor specified in the local coordinate system, into $[\mathbf{G}]$, the same growth tensor specified in the global coordinate system. This is accomplished by the transformation [6, 9]:

$$[\mathbf{G}] = [\mathbf{H}][\mathbf{G}]'[\mathbf{H}]^T \tag{4.63}$$

The growth tensor $[\mathbf{G}]$ can now be applied in the FEM growth implementation, discussed in the next section.

4.4.3 FEM Growth Implementation

Standard FEM methods allow for a pre-strain component $\{\epsilon_0\}$. In engineering applications, this is commonly used to incorporate thermal expansion and contraction. However, in this model, this can be used as the driving force of growth. This results in a force that stretches the element, as previously discussed in Section 4.4.

To determine this force, we must first calculate the pre-strain vector from the growth tensor $[\mathbf{G}]$. Recalling the observations made in Section 2.4, and the definition of the strain vector in equation 2.7, we can compute the pre-strain vector as follows:

$$\{\epsilon_0\} = \left\{ \begin{array}{c} G_{11} \\ G_{22} \\ G_{33} \\ G_{32} + G_{23} \\ G_{13} + G_{31} \\ G_{21} + G_{12} \end{array} \right\} \Delta t \tag{4.64}$$

The incorporation of pre-strain into FEM results in a force that can be calculated as follows [45, p.197] (Appendix A.1):

$$\{\mathbf{F}_0\}^{(e)} = \int_{\Omega^{(e)}} [\mathbf{B}]^T [\mathbf{C}] \{\epsilon_0\} d\Omega \tag{4.65}$$

The elemental pre-strain force vectors $\{\mathbf{F}_0\}^{(e)}$ are then assembled into the global force vector. The system can then be solved for the resulting global deformation. If an individual element was separated from the whole, it would grow exactly the specified amount. But in the assembled system, it is restricted by its neighbors, and may not grow as much as desired.

The viscoelastic growth step is to add a portion of the deformation back into the base shape (equation 4.52):

$$\{\mathbf{X}\}^t = \{\mathbf{X}\}^{t-1} + \alpha\{\delta\} \tag{4.66}$$

To calculate α we refer back to Figure 4.14, noting that it is an exponential curve. α can be calculated given the user defined $\tau_{plasticty}$ of the deformation, and the time step Δt:

$$\alpha = 1 - e^{\frac{\Delta t}{\tau_{plasticity}}} \tag{4.67}$$

Chapter 5

Visualization Methods

The intent of this thesis was to develop a tool to aid in the understanding of the link between differential growth and form. Most growth in organisms occurs at a time scale not perceivable to our senses, making this a relatively non-intuitive phenomena. This is the impetus for developing visual aids that could benefit the user, making this understanding more intuitive.

The surface is simulated as a physical system and has additional properties that might not be normally manifest in its spatial representation. For example, the surface contains internal stresses which can be visualized. Other properties such as curvature and growth can also be visualized.

Three visualization methods were developed for the mass-spring canvas. Section 5.1 describes an internal strain visualization, Section 5.2 a Gaussian curvature visualization and Section 5.3 a growth visualization.

5.1 Internal Strain

The internal strain of a body is a measure of how much it is being compressed or extended at a point inside the body. A first approach to visualizing the internal strain of the mass-spring system was to draw each spring as a line (Figure 5.1), and color it according to the state of strain. The resulting problem is that the colors of the springs were difficult to see, and that simply plotting all the springs at once, did

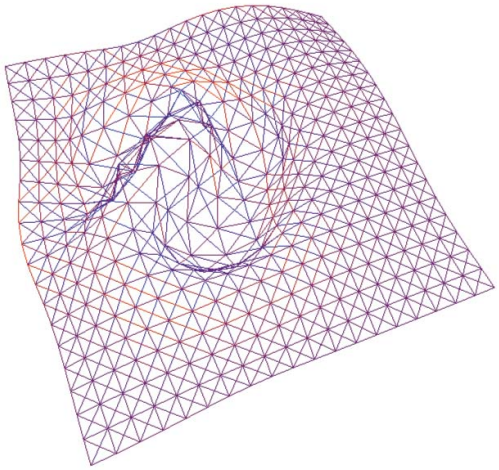


Figure 5.1: Wireframe spring drawing method (red is extension, blue is compression).

little to organize the information.

In reality, the datum (strain) at each point is a tensorial quantity, with strain information in all directions. The visualization of tensor fields is a large area of research, with many techniques already developed [5, 22, 23, 50, 62]. I propose yet another visualization technique, drawing an inspiration from glass sculptors.

When a glass sculptor has completed a work and cooled it, internal stresses still exist within the object. These stresses can be revealed by using polarized filters which show the stresses as colored rings within the object. The polarized filter can be rotated, which changes the ring pattern. I suggest that these patterns are a natural way to visualize stress.

Figure 5.2 shows actual stress rings observed in glass. Two important observations can be made:

- Where no stress exists, the glass is clear
- As the stress increases, the glass appears colored, with the hue somehow related

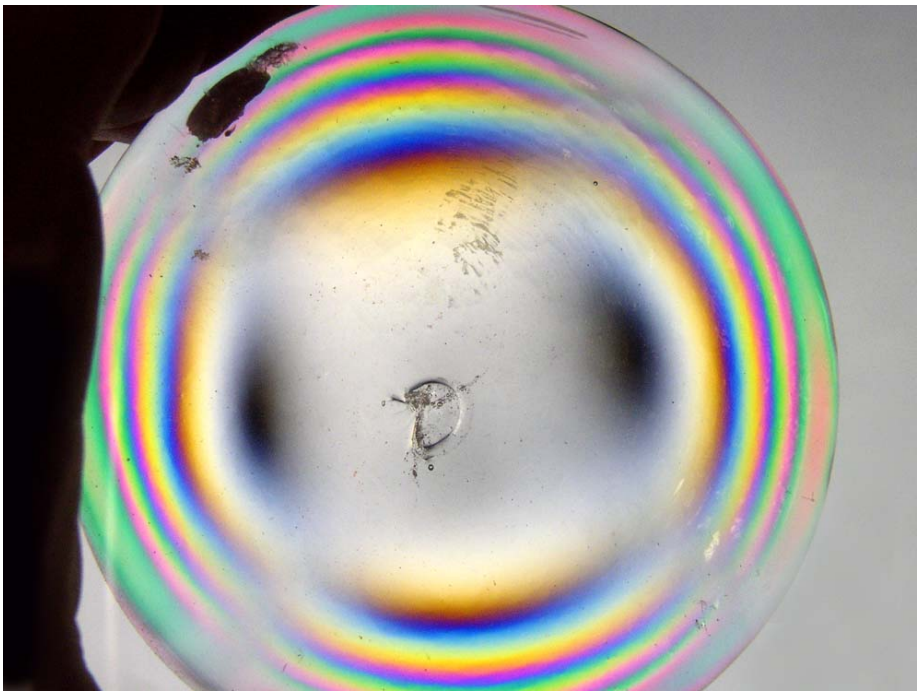


Figure 5.2: Stress rings in glass revealed using polarizing filters.

to stress magnitude.

Although very difficult to properly simulate the physical processes creating these rings, a much simpler method can be devised based on these two observations. Internal strains (directly proportional to stress) can be drawn in a semi-transparent fashion for a selected uv direction on the surface. The user is allowed to change the direction and examine the results. The effect was found to be similar to the real effect.

5.1.1 Algorithm

The visualization algorithm requires one input, a 2D vector d in uv space which specifies the direction of strain visualization. The strain ϵ for a spring is computed

as:

$$\varepsilon = \frac{|\mathbf{s}| - s_0}{s_0} \tag{5.1}$$

where $|\mathbf{s}|$ is the current length of a spring, and s_o is the restlength of the spring. Since we wish to draw a surface, the strain of the springs must be mapped to the nodes. Algorithm 2 was used to do so.

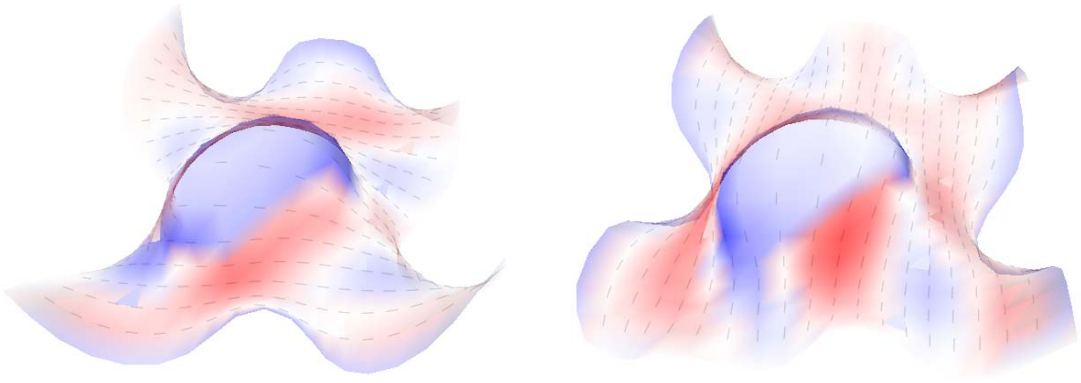
Algorithm 2 Strain computation algorithm.

1: **for all** *nodes* **do**
2: tensionvis $\leftarrow 0.0$
3: **end for**
4: **for all** *springs* **do**
5: $\epsilon \leftarrow (|\mathbf{s}| - s_0)/s_o$
6: $t \leftarrow$ direction of spring in uv space
7: tensionvis (at spring nodes) $\leftarrow \epsilon(d \cdot t)$
8: **end for**

The strain visualization direction vector d could be could be interactively specified during viewing, similar to rotating a polarizing filter used for viewing real stress rings.

5.1.2 Drawing Method

An effort was made to preserve this visual appearance of glass stress rings. To do so, the surface is drawn with alpha blending, recreating the transparency of glass. Points having no stress are drawn with an alpha of zero for full transparency. As the absolute strain increases, the alpha of that point also increases, to a maximum of 1.0. Some consideration was given to rotating the color hue proportionally to the strain. However, I found that the rainbow hue was difficult to read, and really unnecessary. Instead, positive strains are drawn red and negative strains drawn blue.



(a) default direction.

(b) perpendicular direction.

Figure 5.3: Internal strain visualization examples. Blue indicates compression; red indicates tension.

It was also necessary to indicate the direction being examined by plotting a direction field. Short semi-transparent lines in the direction of strain visualization d are drawn at each particle. These lines also serve to reveal the shape of the surface. The drawing method is listed in Algorithm 3.

Algorithm 3 Strain visualization drawing method.

```
1: for all nodes do
2:    if tensionvis > 0.0 then
3:       color ← red
4:    else
5:       color ← blue
6:    end if
7:    alpha ← abs(tensionvis)
8:    draw alpha blended surface
9:    draw direction vector centered at vertex
10: end for
```

5.1.3 Examples

The results of the visualization can be seen in Figure 5.3. Each of the two figures show one of two perpendicular visualization directions. Using only two colors was a good choice because the most useful information is whether a location is in tension or compression, and the relative amount is of secondary importance.

Despite the acute simplifications compared to real glass, the method is quite effective and I have observed that it can be quickly understood by an untrained user. In particular, the strain can quickly show whether the system has reached a steady state, or has more energy available to deform the system. The method also has an interesting visual appeal, very similar to real glass.

5.2 Gaussian Curvature

The visualization of surface curvature is a large field with many techniques already developed for viewing surfaces, for example see [25, 46, 78, 89]. Surface metrics such as mean curvature can be visualized to gain a greater "feel" of a surface [10]. Here I have chosen to visualize the Gaussian curvature of the surface for its relevance to the positive and negative curvature observed in Figures 6.2 and 6.3.

The Gaussian curvature of a surface is defined as the product of its two principal curvatures. A sphere has a constant positive curvature, while cylindrical surfaces have a Gaussian curvature of zero, because they are curved only in one direction. Gaussian curvature is a basic property of a surface that could give insight into its deformation. One such interesting property is that a closed surface will have the same total curvature regardless of the deformation. A full discussion of Gaussian

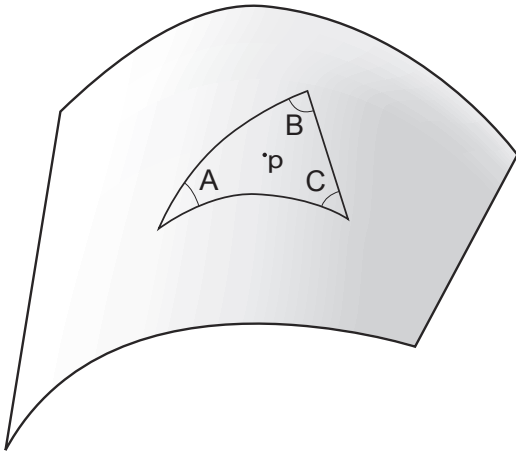


Figure 5.4: Gauss-Bonet Theorem

curvature is presented by Willmore [96] for example.

5.2.1 Algorithm

Instead of computing the two principal curvatures, then multiplying, a convenient method exists to directly calculate the Gaussian curvature of a surface. The method comes from the classical definition of the Gauss-Bonet theorem. For a given triangle of geodesics on a surface (Figure 5.4) that contains p, the Gaussian curvature at a point p is given by:

$$K = \lim_{A,B,C \to p} \frac{A + B + C - \pi}{\Delta} \tag{5.2}$$

Where Δ is the area of the triangle. This can be extended for quadrilaterals as well [96]:

$$K = \lim_{A,B,C,D \to p} \frac{A + B + C + D - 2\pi}{\Delta} \tag{5.3}$$

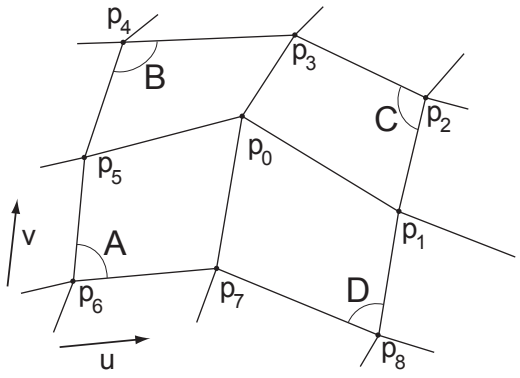


Figure 5.5: Eight points surrounding p

The advantage of using a quadrilateral is that it is much easier to measure the interior angles A, B, C and D for a rectangular mesh. The local u and v parameterization curves of the surface (the spring mesh) can be considered to be geodesics. This is generally true, so long as the mesh is relatively smooth, a property generally enforced by the mass spring system.

The Gaussian curvature of a given point on the mesh can then be computed as follows (Algorithm 4). We consider the eight points surrounding a given point p_0 (Figure 5.5). The four rectangular corners are considered to be A, B, C, D. We then compute the ratio as defined in Equation 5.3 from the sum of these angles and the total contained area.

5.2.2 Drawing Method

Very similar to the internal strain drawing method, one of two colors was used to draw positive or negative curvature. The only deviation from the internal strain

Algorithm 4 Gaussian curvature computation algorithm.

1: **for all** *nodes* **do**
2: area $\leftarrow 0.0$
3: angle $\leftarrow 0.0$
4: **for all** neighboring *nodes* i **do**
5: $a \leftarrow p_i - p_0$
6: $b \leftarrow p_{i+1} - p_0$
7: area $\leftarrow$ area $+\frac{1}{2}|a \times b|$
8: **end for**
9: **for** each corner i **do**
10: $a \leftarrow p_{i+1} - p_i$
11: $b \leftarrow p_{i-1} - p_i$
12: angle $\leftarrow$ angle$+cos^{-1}(\frac{a \cdot b}{|a||b|})$
13: **end for**
14: angle $\leftarrow$ angle-2π
15: curvature $\leftarrow$ angle / area
16: **end for**

drawing method is that the surface is drawn completely opaque, so as not to make it look like glass. For areas of zero curvature, a neutral grey color was used, and blended with either blue or red, depending on the sign of the curvature. The drawing method is listed in Algorithm 5

Algorithm 5 Gaussian curvature drawing method.

1: **for all** *nodes* **do**
2: **if** curvature > 0.0 **then**
3: hue $\leftarrow$ red
4: **else**
5: hue $\leftarrow$ blue
6: **end if**
7: $\alpha \leftarrow$ abs(tensionvis)
8: color $\leftarrow \alpha$hue$+(1 - \alpha)$ grey
9: **end for**

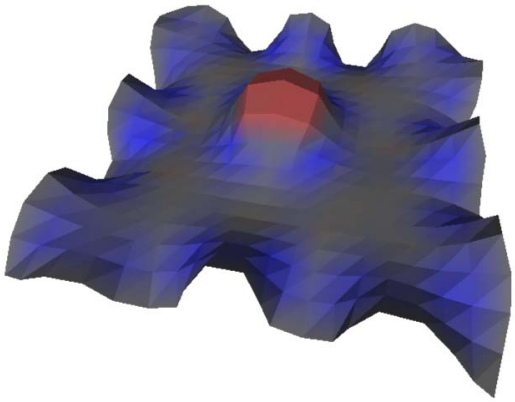


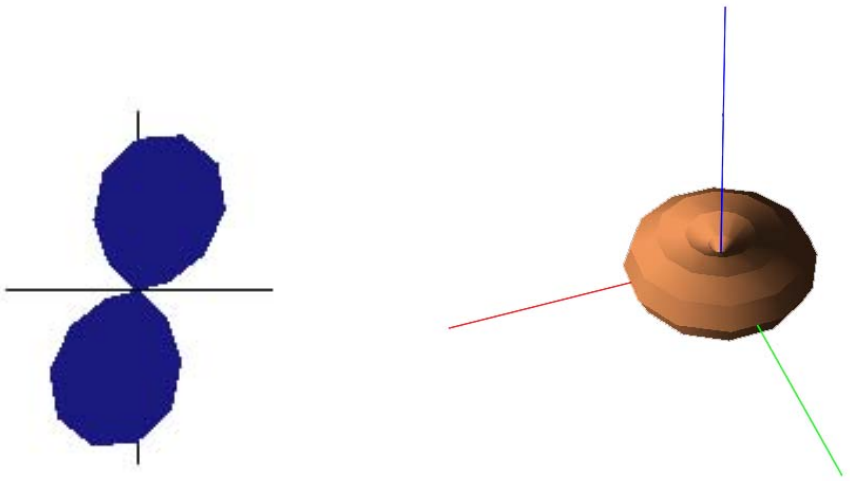
Figure 5.6: Gaussian curvature visualization example (red is positive, blue is negative).

5.2.3 Examples

An example of the visualization method is shown in Figure 5.6. The figure clearly indicates areas of positive and negative curvature, as well as their relative magnitude quite easily. It should be noted that other methods exist for computing the curvature of a surface [59, 72, 10]. However, this method is a simple efficient algorithm that works well with mass-spring meshes.

5.3 Growth

A useful way to visualize growth tensors is with an *indicatrix*. An indicatrix is an isosurface where the radius in any direction is proportional to the RERG (Section 2.3.2) of that direction (Figure 5.3). Indicatrices have been used by Nakielski and Rumpf [62] for visualizing growth tensors. A 3×3 tensor describes growth for a 3D body, and is rendered with an isosurface like that shown in Figure 5.7(b). For 2D surfaces in 3D, we use a 2×2 tensor that only considers in-plane movements of the



(a) A 2D indicatrix (b) A 3D indicatrix

Figure 5.7: Examples of indicatrices.

points. The indicatrix for this tensor is a filled isocurve in 2D (Figure 5.7(a)).

5.3.1 Algorithm

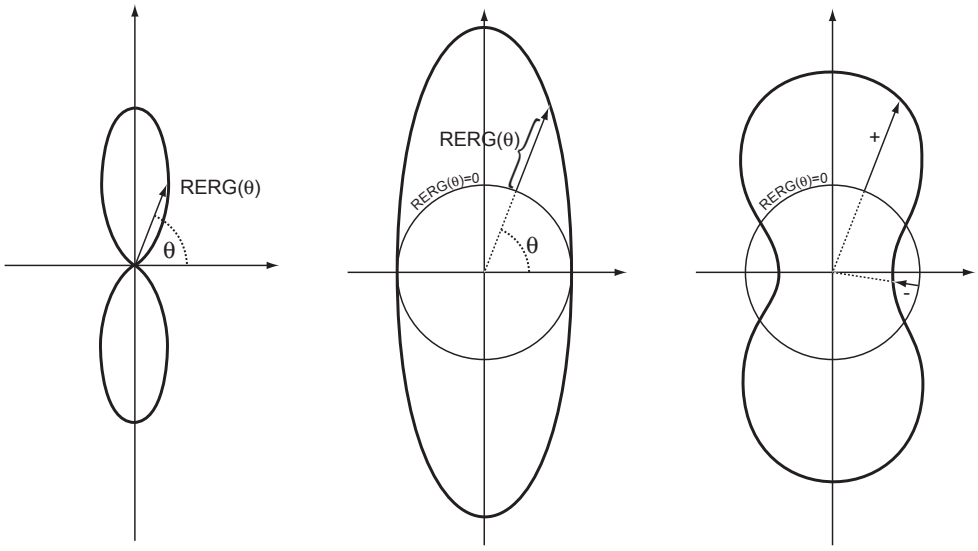
Normally, RERG data is computed from a tensor, then an indicatrix is drawn from this data. In this case, a growth tensor is not available, so we must compute the RERG data directly from particle motion. Interior particles on the surface are surrounded by eight neighboring particles (Figure 5.5), connected by springs. This allows us to compute the RERG in eight directions and plot this data as an isocurve. The RERG data was calculated using Algorithm 6.

5.3.2 Drawing Method

A problem not considered in Nakielski and Rumpf's [62] work is that of negative growth. In the mass-spring model the RERG is often negative (ie. the surface is

Algorithm 6 RERG data computation algorithm.

1: **for all** *nodes* p_0 **do**
2: **for all** neighboring *nodes* p_i **do**
3: $d \leftarrow \frac{p_i - p_0}{|p_i - p_0|}$
4: $a \leftarrow v_i \cdot d$
5: $b \leftarrow v_0 \cdot d$
6: $RERG_i \leftarrow (a - b)$
7: **end for**
8: **end for**



(a) standard indicatrix.

(b) implemented indicatrix

(c) implemented indicatrix (showing negative growth)

Figure 5.8: Indicatrix comparison. Figures 5.8(a) and 5.8(b) show the same tensor drawn using two different methods. The indicatrix of Figure 5.8(a) is drawn using the method of Nakielski and Rumpf [62], but indicates only positive growth. The radius of the indicatrix drawn in Figure 5.8(b) has an offset added so that negative growth can be shown. Figure 5.8(c) shows how a tensor with negative growth is drawn using this method.

contracting). A standard indicatrix (Figure 5.7(a)) only draws positive growth. I solve this problem by setting the radius to the computed $RERG_i$ plus an offset (Figure 5.8). Non-growing (zero) tensors are drawn as a disc. Positive growth is drawn with a radius larger than the offset, and negative growth with a lesser radius. A black circle equal to the offset is superimposed on the tensor as a zero reference. The drawing method is listed in Algorithm 7.

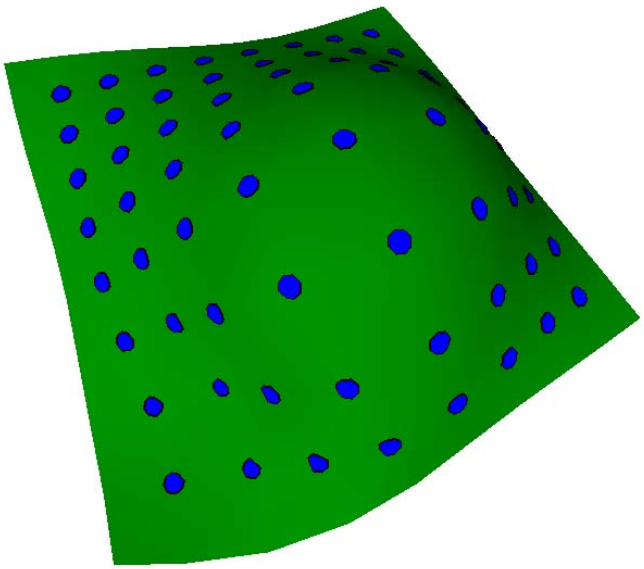
Algorithm 7 Indicatrix drawing method.

```
1: for all nodes do
2:    for all neighboring nodes i do
3:       draw polygon of radius (RERG_i + offset)
4:       draw circle of radius (offset) in black
5:    end for
6: end for
```

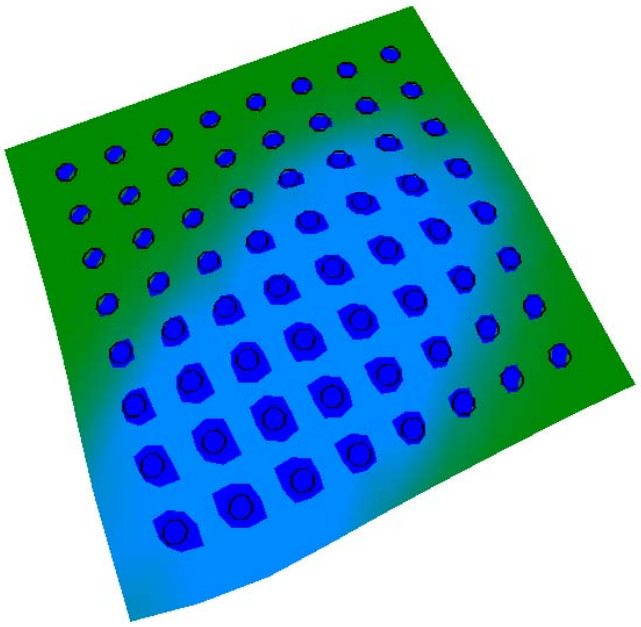
An ideal implementation of this drawing algorithm would embed tensors into the surface. My implementation draws the tensors slightly above the surface, eliminating the need to re-mesh the surface to include tensors. Although primitive, it has worked well as a drawing method.

5.3.3 Examples

Some examples of the visualization method are shown in Figure 5.9. Figure 5.9(b) shows a surface that is growing. The indicatrices clearly extend beyond the reference circles, indicating positive growth. Also note that the areas of the surface adjacent to the growing portions are contracting due to the expansion of the growing area. This in shown by inidcatraces that do not fill the circle. Figure 5.9(a) shows a surface at rest. Here the indicatraces can be seen to perfectly occupy the reference circle, indicating zero growth.



(a) Surface is at rest.



(b) Surface is growing.

Figure 5.9: Growth tensor visualization examples. (The blue parts of the surface indicate the presence of growth morphogen.)

Chapter 6

CanvasLite

Using the mass-spring, diffusion and growth models discussed in Section 4.1, an interactive program "CanvasLite" was developed to explore growing surfaces with diffusible morphogens. The program is described in this chapter.

6.1 General Description

CanvasLite simulates a 2D mass-spring system deforming in 3D, with four diffusible morphogens. The morphogen distribution may be interactively painted by the user, or specific morphogen release modes can be activated from a file. The surface responds to the morphogens in three different ways: isotropic growth, tensorial growth, and curvature modification. Four diffusible morphogens were chosen so that in tensorial mode the four morphogens could be the four elements of a 2D growth tensor.

The program responds fast enough for real time experimentation. That is, the surface reaches equilibrium in a matter of seconds, after painting on new morphogens.

6.2 User Interface

The program is invoked from the command line with an argument specifying an input file. The input file defines a number of parameters needed for the simulation. The surface dimensions, spring constants and particle masses are specified in this file, as well as the simulation time step and damping coefficient. The input file also

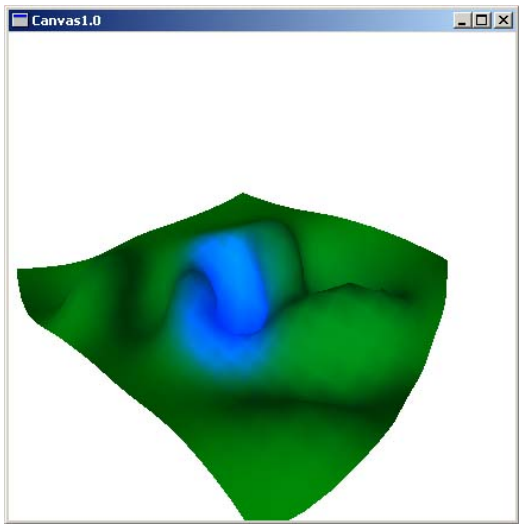


Figure 6.1: The CanvasLite user interface.

contains information about morphogen painting, predefined distributions and growth response to morphogens. Appendix A.3 gives a full description of this file.

The program displays the surface in a single 3D viewer window (Figure 6.1). Dragging in the window with the left mouse button rotates the surface. Left-right motions are mapped to azimuth, and up-down motions rotate the surface from a horizontal to vertical orientation. Pressing the "up" and "down" keys zoom in and out from the surface. Viewing and runtime options are selected using the keys listed in table 6.1.

The surface can be visualized in several ways, which are listed in Table 6.1. Mode 1 draws the surface with green Phong shaded triangles. Mode 2 draws linear springs as line segments. Mode 3 draws particles as points. Modes 4, 5 & 6 are the corresponding visualization methods discussed in Chapter 5.

The user can paint on the surface interactively. 3D painting was introduced by Hanrahan and Haeberli [40]. With CanvasLite, painting on the surface is accom-

Key	Function
$\uparrow \downarrow$	zoom in/out from surface
p	toggle pause/run
n	add noise to particle locations
r	reset surface
1	toggle shaded drawing
2	toggle spring drawing
3	toggle particle drawing
4	toggle internal strain visualization
$\leftarrow \rightarrow$	rotate internal strain visualization angle d
5	toggle curvature visualization
6	toggle growth visualization

Table 6.1: Keyboard commands for CanvasLite.

plished by right or middle clicking on the portion of surface to be painted. The middle and right buttons each map to a user defined combination of morphogens, specified in the input file (Appendix A.3).

The program can operate in standard mode or tensorial mode. In standard mode, the morphogens can be programmed to induce isotropic growth (positive or negative) and curvature change in the u and v directions. The four morphogens are named `morphogen0` to `morphogen3`. In standard mode `morphogen0` is drawn as blue, and `morphogen1` is drawn as red. In tensorial mode each of the four morphogens represent an element of the 2D uv growth tensor:

$$\mathbf{G} = \begin{bmatrix} p & q \\ r & s \end{bmatrix} \tag{6.1}$$

where p, q, r, s are the concentration of `morphogens0` to `morphogen3` respectively. In this mode the tensors are drawn as 2D indicatricies (Section 5.3) which cause anisotropic growth as described in Section 4.1.4.

In the case where the surface is perfectly flat (a natural starting situation), it will not naturally deform into the third dimension. A form of symmetry breaking was needed to "push" the surface slightly out of plane. A noise function was added so that the locations of each particle was perturbed by a minute random amount whenever the 'n' key was pressed.

6.3 Program Structure

The system was coded as a Windows application using OpenGL and the `glaux` [63] library, and runs on the Microsoft^{TM}Windows 2000 operating system. The main algorithm is as follows:

```
1: initialize program & graphics
2: read input file
3: while forever do
4:    if running = true then
5:       sum forces acting on particles (equation 4.1)
6:       advance the simulation one time step (equation 4.2)
7:       enforce morphogen constraints
8:       enforce geometry constraints
9:       diffuse & decay morphogens
10:      grow springs
11:      draw surface
12:   end if
13: end while
```

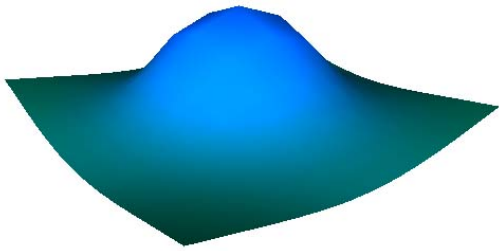


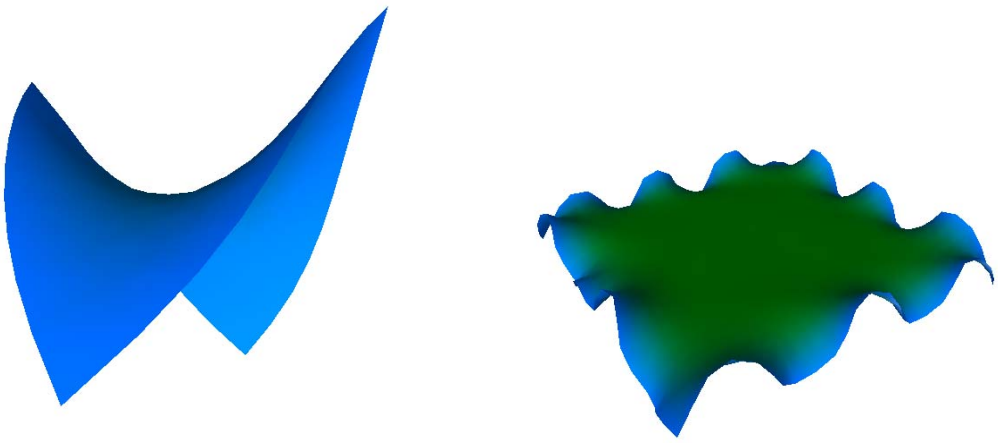
Figure 6.2: Growth morphogen (blue) is released from the center of the surface. A bulge showing positive curvature results.

Using a 400Mhz SGI Visual Workstation 320 interactive frame rates were obtained for surfaces with up to a thousand particles. The surface response to morphogens could be observed within a few seconds.

6.4 Examples

A number of examples are presented to illustrate the basic modes of growth available with CanvasLite. The first example (Figure 6.2) is a surface resulting from morphogen release in the center of the surface. The surface bulges in the center producing a positive curvature. Figure 6.3(a) shows the canvas response for morphogen released at the edges. A negative curvature surface is observed. The effect can be varied by reducing the bending spring constant, leading to a "lettuce" effect (Figure 6.3(b)). Both of these morphogen release modes ("Edges" and "center") are predefined and can be activated in the input file.

The effects of curvature growth are shown in Figure 6.4. The blue and red morphogens are set to influence bending springs in either the u or v directions. Note that this does not mean that any direction of curvature could be chosen by combining these two together. We are limited to the directions of discretization chosen for the



(a) A high bending resistance produces a saddle shape.

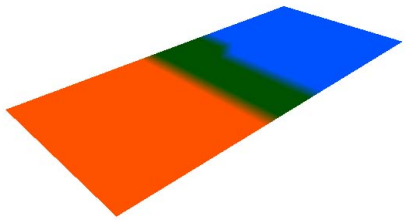
(b) A low bending resistance produces rippled edges.

Figure 6.3: Growth morphogen (blue) is released from the edges of the surface. Two examples are shown with different resistances to out of plane bending. Both surfaces exhibit a negative curvature.

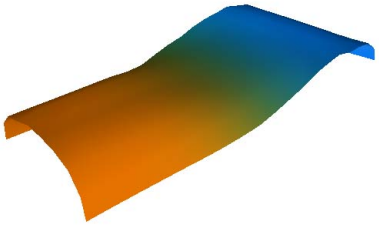
mesh.

A sequence of tensorial growth is shown in Figure 6.5. In it we see a sequence where a line of morphogens has been painted on the surface. Here, painting has applied a combination of the four morphogens. The four morphogen concentrations are interpreted as the four components of a growth tensor, and the indicatrix of this growth tensor is drawn on the surface. The indicatrices indicate how they will deform the surface. In Figures 6.5(b) and 6.5(c) we see the surface deform in a clearly anisotropic manner. The surface is shifted to the right, growing in the direction indicated by the tensors.

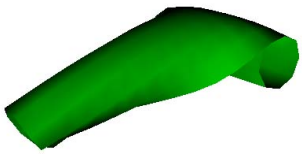
Finally, Figure 6.6 shows a sequence obtained by painting a diagonal line of morphogens on the surface, illustrating a particularly non-intuitive example. In this



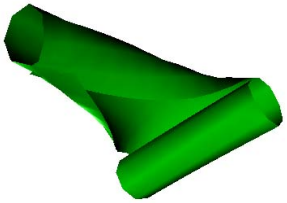
(a) Blue and red morphogens are placed at opposite ends of a strip.



(b) The ends begin to curl.

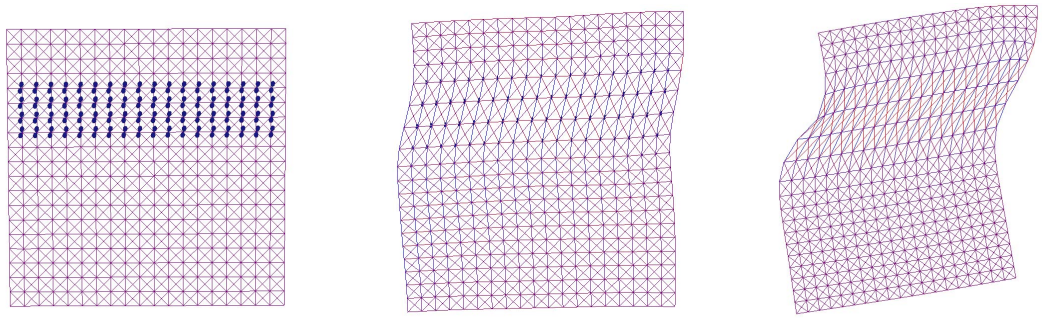


(c) The final surface at rest.



(d) The final surface viewed from a different angle

Figure 6.4: Curvature change. Morphogen influences the rest angle of the bending springs. Blue and red morphogens cause curling in orthogonal directions.



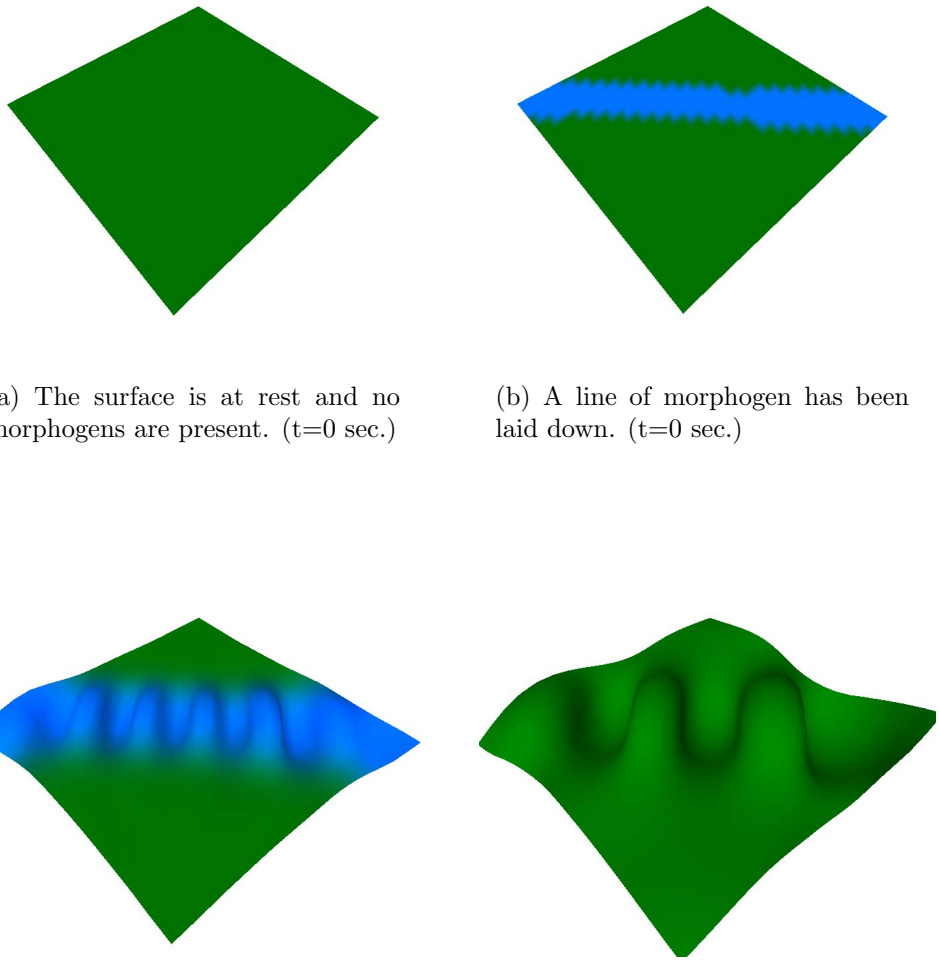
(a) The surface is at rest. A strip of tensorial morphogens have been painted near the top.

(b) The surface begins to deform.

(c) The surface is at rest. Note the anisotropic growth.

Figure 6.5: Tensorial growth (spring mesh drawn for clarity.)

example the blue morphogen causes isotropic growth. It is difficult to predict the final outcome (Figure 6.6(d)) given only the initial morphogen distribution (Figure 6.6(b)).



(a) The surface is at rest and no morphogens are present. (t=0 sec.)

(b) A line of morphogen has been laid down. (t=0 sec.)

(c) The morphogens diffuse and the surface begins to deform. (t=1 sec.)

(d) All the morphogens have decayed and the surface is at rest. (t=6 sec.)

Figure 6.6: Morphogen painting and surface response (Blue morphogen causes isotropic growth.)

Chapter 7

Finite Element Canvas

The primary limitation of mass-spring systems are their fixed discretization. This ultimately limits the amount of growth possible because springs cannot be subdivided in a physically consistent manner. To overcome this problem a second program "Canvas" was developed using the finite element and growth theory outlined in Sections 4.2, 4.3 and 4.4. This chapter describes that program.

7.1 General Description

The general operation of Canvas is similar to that of CanvasLite, but with some added features. Canvas allows for dynamic subdivision, unlimited morphogens, connectionist interactions, arbitrary shape input, and more advanced parameter control.

The major disadvantage of Canvas is its slow speed. FEM calculations are inherently much slower than those of mass-spring systems. Computing a single time step for 100 elements on a 1.4 Ghz Intel P4, took approximately 1 second, making the simulation impractical for interactive exploration.

7.2 User Interface

The user interface is a standard Windows interface with a menu, view pane, control bar and context menus (figure 7.1). Rotation controls are similar to those of CanvasLite, done by dragging the left mouse button. Zooming is achieved by shift-

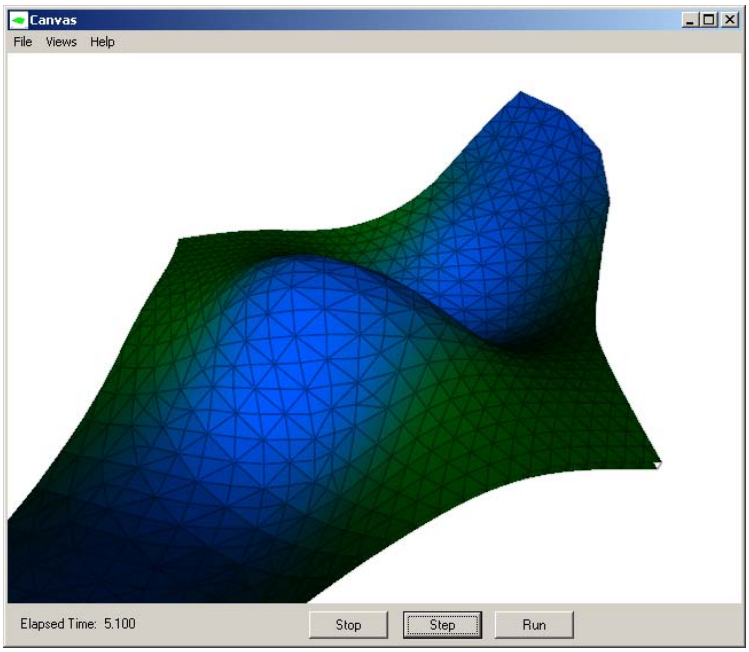


Figure 7.1: Canvas user interface.

dragging the left mouse button. The control bar has an elapsed time indicator as well as three buttons that start, stop or step through the simulation.

An input file is used to specify a number of parameters necessary for the simulation. The input file contains sections to describe the material properties, morphogen diffusion and decay characteristics, surface geometry, constraints, simulation parameters and drawing methods. Appendix A.5 gives a detailed description of the file format.

The view pane has a context menu (Figure 7.2) that activates different visualization modes. The "Base" mode draws the surface geometry according to the global node coordinate vector $\{X\}$, as a black wire frame. "Deformed" mode draws the geometry of the base shape plus deformation $\{X\}+\{\delta\}$, as a red wire frame. "Color" mode draws the surface using triangles colored according to the morphogen concen-

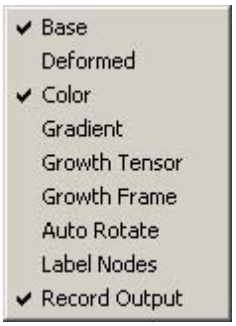


Figure 7.2: Canvas visualization context menu.

tration. The mapping between morphogen concentration and color is defined in the input file. Selecting the "Growth Frame" option will draw $\vec{v}_{max}, \vec{v}_{min}$ and $\vec{v}_{up}$ as a set of red, green and blue line segments for each element. The "Growth Tensor" option draws $\mathbf{G}$ for each element as an indicatrix. "Auto Rotate" will automatically rotate the surface. "Label Nodes" will display the global node number beside each node. The option "Record Output" will write each frame to an `avi` file. Un-selecting the option will stop writing frames and close the file. The filename is specified in the input file.

The user can paint morphogens on the surface by dragging with the middle mouse button. The amount of morphogen placed during painting is specified in the input file. Because of the slow speed of simulation, it is often advantageous to use the input file to specify morphogen release locations.

7.3 Program Structure

The system was coded as a Win32 OpenGL application and run under Windows 2000. The program is written using object oriented programming as a way to manage the complexity of the code and to aid future development.

The main algorithm is as follows:

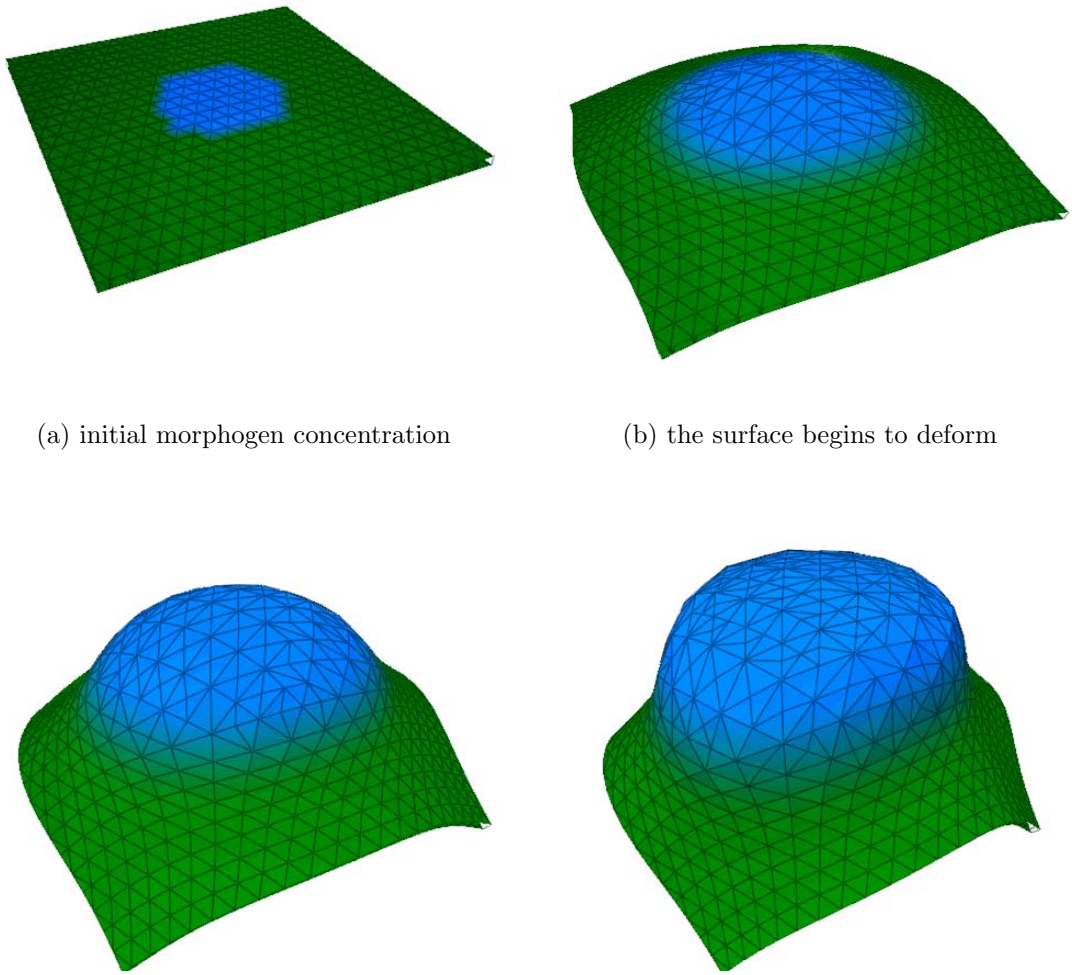
read input file
initialize graphics and user interface
while forever **do**
  if running = true **then**
    impose morphogen constraints.
    resize and zero global elasticity vectors ($\{F\}$ and $\{\bar{R}\}$)
    resize and zero global stiffness matrices ($[K]$ and $[K_c]$)
    for all elements **do**
      compute and assemble stiffness matrices and thermal load vectors ($[K]^{(e)}$, $[\bar{K_c}]^{(e)}$, $\{\bar{R}\}^{(e)}$)
      compute growth frame [**H**]
      compute growth tensor [**G**]
      compute strain vector $\{\epsilon_0\}$
      compute and assemble force vector $\{F_0\}^{(e)}$
    end for
    impose displacement constraints
    solve for displacements $\{\delta\}$ and concentrations $\{T\}$ using conjugate gradient method.
    update base configuration $\{X\}$
    subdivide elements that are too large
    add connectionist contributions to $\{T\}$
  end if
end while

Displacement constraints were implemented as described in Section 4.2.5. This method is not applicable to the morphogen constraints, because of the time integration method used. Morphogen constraints were implemented by forcing elements of the temperature vector $\{T\}$ to the desired value before each simulation step.

Using a 1.4GHz Intel IV processor, a single time step for 100 elements took approximately 1 second. This generally meant that interactive exploration could not be achieved, and most simulations needed to be rendered offline. For this reason Canvas includes an option to write the frames to a Windows `avi` file that can later be played at 30 fps. This feature is quite useful, often making the model dynamics much clearer. This however, comes at the cost of no interaction with the model.

7.4 Examples

To illustrate the basic operation of Canvas some simple examples of growth are shown in Figures 7.3 and 7.4. Each are comparable to the CanvasLite examples, and show that similar behavior is obtained. The results are discretization independent. Figure 7.5 shows the same positive curvature example of Figure 7.3, except with more elements, obtaining the same result. More advanced examples with Canvas are given in Chapter 8.



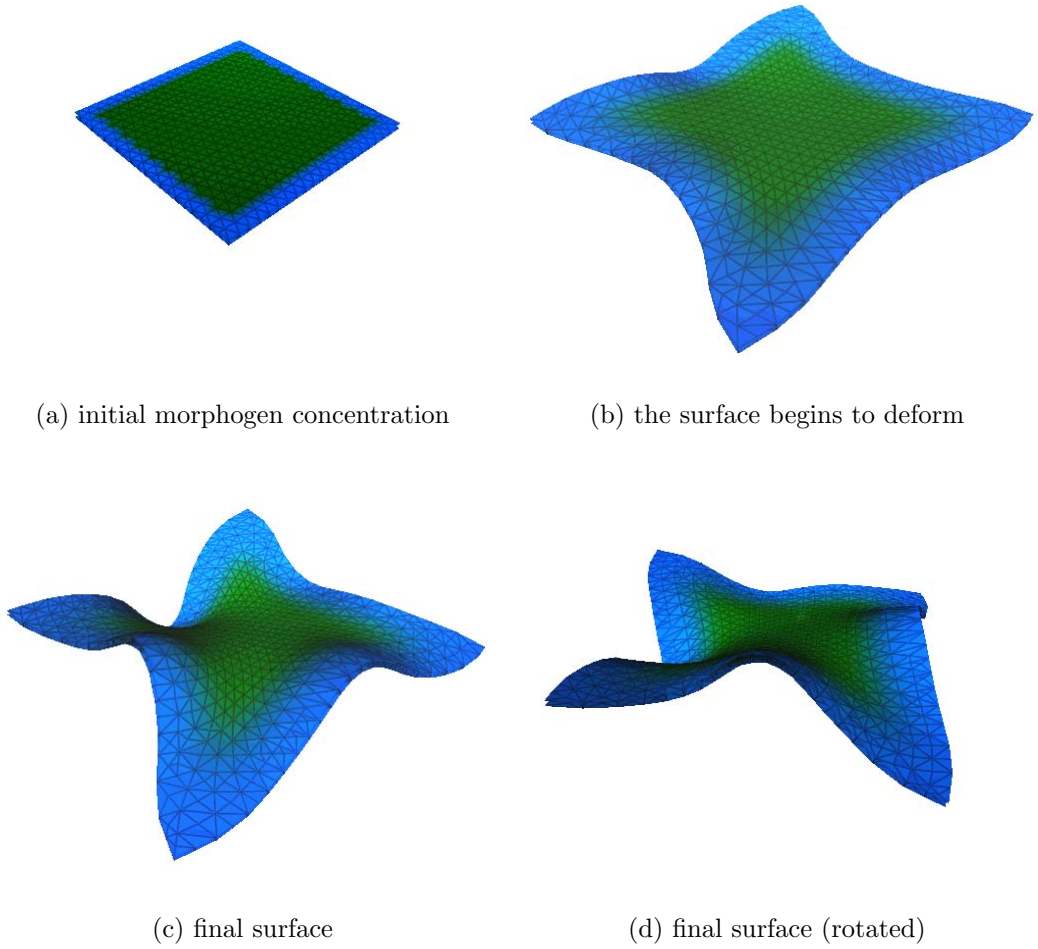
(a) initial morphogen concentration

(b) the surface begins to deform

(c) a bulge appears

(d) final surface

Figure 7.3: Positive curvature. Isotropic growth morphogen is placed at the center of the surface. A bulge with positive curvature results.



(a) initial morphogen concentration

(b) the surface begins to deform

(c) final surface

(d) final surface (rotated)

Figure 7.4: Negative curvature. Isotropic growth morphogen is placed at the edges of the surface. A rippled saddle shape with negative curvature results.

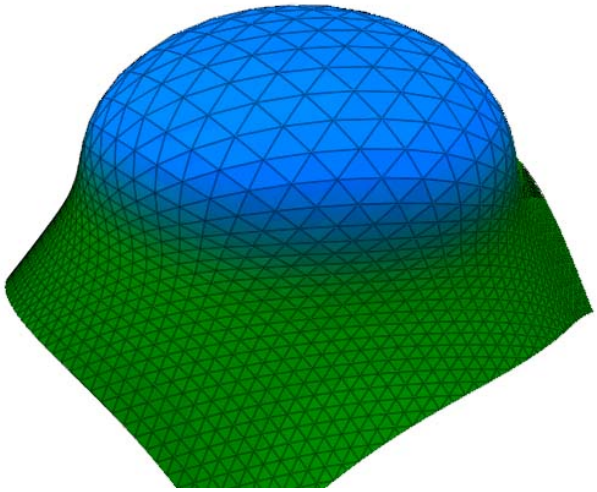


Figure 7.5: Discretization comparison. This is the same positive curvature simulation as shown in Figure 7.3 except that 3600 initial elements were used instead of 1000. The same result is obtained.

Chapter 8

Examples

The practical use of Canvas is illustrated using a number of growing models in this Chapter. I show that Canvas is well suited for modeling the growth of biological organisms.

The first model simulates the development of a sea urchin. The next models simulate the growth of leaves and petals of an *Antirrhinum majus*. The final model controls the growth of a surface using using a genetic network.

8.1 Sea Urchin

Sea urchins are a popular subject of morphologists because of their simple shape. A typical sea urchin *test* is shown in Figure 8.1. Sea urchins have a mouth situated at the peristomal (bottom) end of the test, and an anus at the apical (top) end of the urchin test. The test is composed of a number of *plates*. The dome shape of the urchin

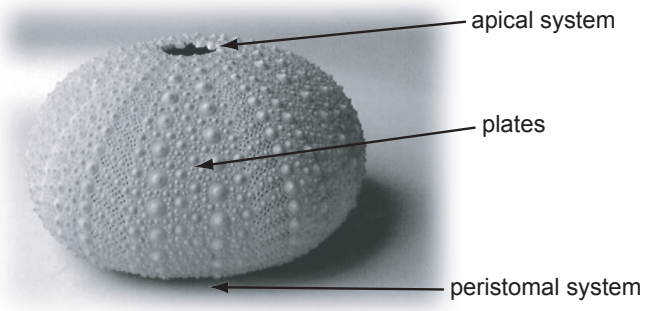


Figure 8.1: Photograph of a sea urchin test.

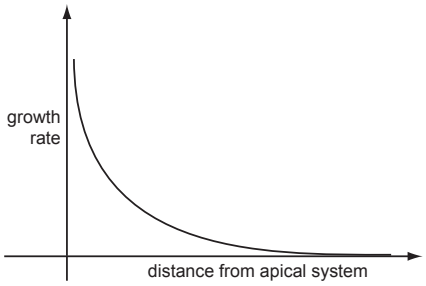


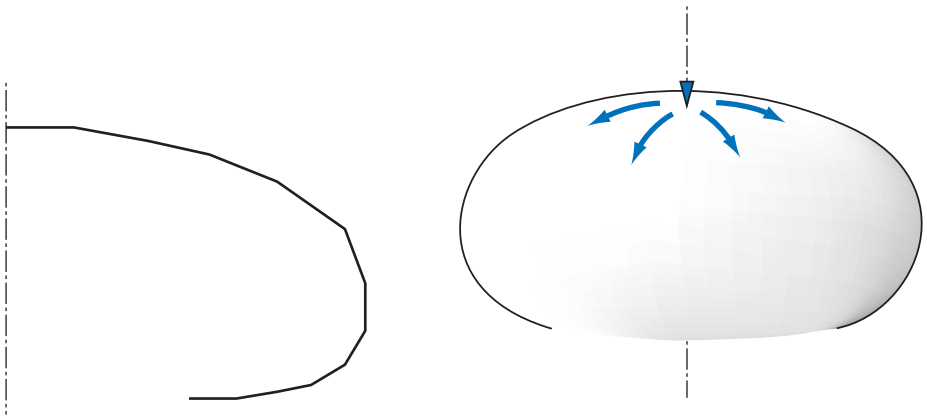
Figure 8.2: Growth rate as a function of distance from the apical system. Abitrary units of measurement (after [74]).

test is believed to be a structurally optimized shape and morphological research has focused on the function and evolutionary origins of this form [84]. Thomson [88] noted that the sea urchin has a *pneu* shape. A pneu is the shape that would result if a balloon full of water is placed on a table. This simple shape suggests that a simple mechanism is responsible for sea urchin morphology.

Raup [74] first modeled sea urchin development in 1968. His work directly addressed the development of plate shape of the sea urchin. Much subsequent modeling work has followed in these lines [84, 31, 67].

My model draws its main inspiration from Raup's original work. Raup plots the growth increment (or growth rate) of the test as a function of distance from the apical system (Figure 8.2). This exponential decay curve observed suggests that a diffusing/decaying morphogen may be responsible for influencing growth. Thus providing an ideal test case for the canvas system because it involves both growth and diffusible morphogens.

A model was developed using canvas to test this hypothesis. The initial shape was an immature sea urchin test, modelled by hand from a real baby urchin shape (Figure



(a) A cross section of a baby urchin.

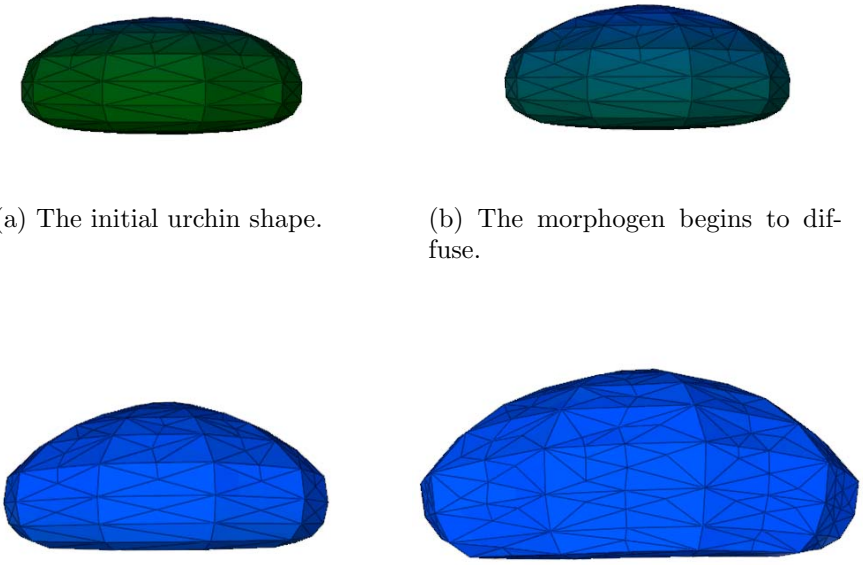
(b) The morphogen source is placed at the apical system.

Figure 8.3: Initial shape and morphogen location of the sea urchin model. The model begins with the geometry shown on the left. The isotropic growth morphogen source is placed as shown on the right. Morphogens (blue) diffuse from the apical system toward the peristomal system, causing isotropic growth.

8.3(a)). A morphogen release location was specified near the apical system (top) of the urchin (Figure 8.3(b)), and diffusion parameters set such that the morphogen decayed near the peristomal system (bottom). Isotropic growth was proportional to morphogen concentration. The resulting growth of the urchin is shown in Figures 8.4(a) to 8.4(c).

The results show that qualitatively the sea urchin tends to maintain a pneu shape, but without quantitative sea urchin growth data, it is impossible to draw any conclusions about the accuracy of the model. The model does however illustrate a successful use of the Canvas modeling system to produce a morphogenetic result.

Unfortunately, just because a model can reproduce a given result, does not mean



(a) The initial urchin shape.

(b) The morphogen begins to diffuse.

(c) The surface begins to deform.

(d) The final surface

Figure 8.4: Time series of the sea urchin simulation.

the model is correct. Many models could produce valid results. Recent work from the Santa Fe Institute suggests that it is primarily biomechanical factors that influence sea urchin morphology [28], and not diffusible morphogens as this model presents.

8.2 Antirrhinum majus

The *Antirrhinum majus* (commonly known as a snapdragon) is a good subject for geneticists because of its high mutability and ease of cultivation [14]. The two models presented here were a collaborative effort with Dr. Enrico Coen and Anne-Gaelle Rolland [77] of the John Innes Center, England.

Figure 8.5: Flowers of *Antirrhinum majus*. This particular plant variant is the pal^{rec} mutant.

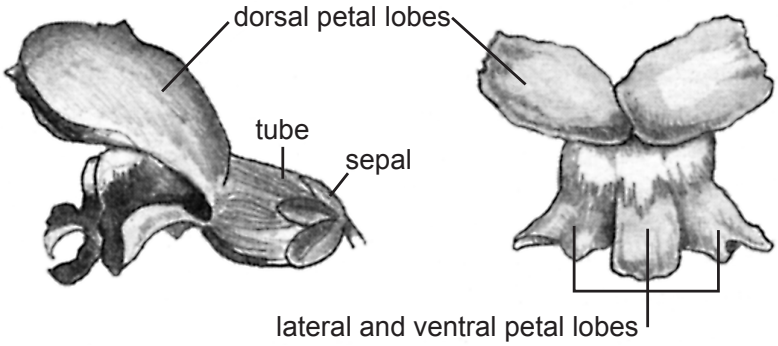


Figure 8.6: Anatomy of the *Antirrhinum majus* flower. (Reprinted with permission from [14].)

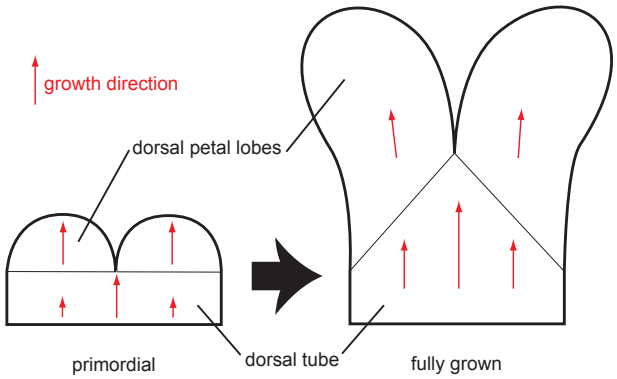


Figure 8.7: Normal lobe and tube development of *A. majus* (not to scale, longer arrows indicate greater growth). Note that although the boundary between the tube and lobes shifts significantly, the growth direction of the lobes remains almost perpendicular to the base of the tube.

8.2.1 Lobe Tube Mechanistic Model

The flower of *A. majus* is quite unique (Figures 8.5 and 8.6), making it a compelling organism to model. This model focuses on the development of the dorsal petal lobes and tube (Figure 8.6) of *A. majus*. Data was collected on the growth of these surfaces using the pal^{rec} mutant flower [41, 11, 15, 94, 77]. During development the central portion of the tube grows faster than the lateral portions, distorting the lobe-tube boundary by approximately 45 degrees (Figure 8.7). However, the primary growth direction of the lobes remains oriented perpendicular to the base of the tube, strongly indicating the presence of an external orientation field [77]. If there was no external influence on the growth direction, then growth might be expected to proceed as shown in Figure 8.8. Coen and Rolland theorized that this orientation field could be the gradient of a morphogen released at the base of the tube.

To test this hypothesis a model was developed with canvas (Figure 8.9) to attempt

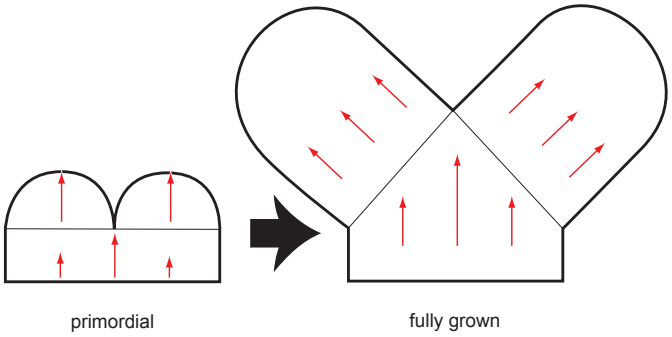


Figure 8.8: Illustration of how growth should occur if the growth direction is "imprinted" during the primordial phase. The shifting lobe tube boundary changes the direction of growth, causing the dorsal petal lobes to diverge significantly.

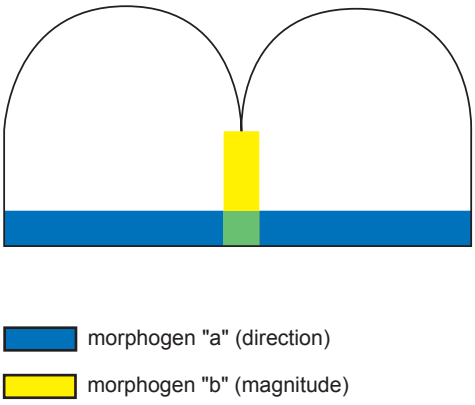
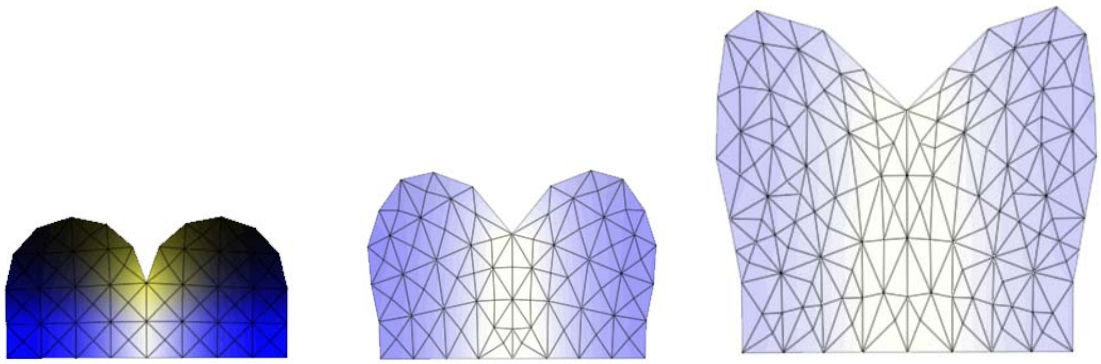


Figure 8.9: Lobe-tube simulation model. Morphogen "a" is released from the base of the tube. The gradient of "a" is used to orient growth. Morphogen "b" is released from the center of the tube. The magnitude of growth is proportional to the concentration of morphogen "b".



(a) The initial lobe shape.

(b) The surface begins to deform.

(c) The final surface.

Figure 8.10: Time series of the lobe-tube simulation.

to reproduce the growth observed in *A. majus* using a diffusible morphogen that orients growth. The starting geometry used was an approximation of the primordial lobe and tube shape. Two diffusible morphogens were used. The magnitude of anisotropic growth was set to be proportional to the concentration of morphogen "b", released in the central portion of the tube (Figure 8.9). This caused the central portion of the tube to grow faster, as observed in *A. majus*. A second morphogen "a" was released at the base, and oriented the anisotropic growth to the gradient of this morphogen.

The results of the simulation are shown in Figure 8.10. The results indicate that the lobes in the simulation tend to grow away from the base, and are not heavily influenced by the shifting lobe-tube boundary. The final shape of the simulated model (Figure 8.10(c)) is similar to that of the real case (Figure 8.7). It should not be taken that this must be the valid model driving lobe growth. However we can conclude that a diffusible morphogen that orients growth is a valid possibility.

Figure 8.11: Photograph of an *A. majus cincinnata* mutant leaf.

8.2.2 Rippling Leaf

An interesting *A. majus* mutant is the *cincinnata* mutant. The leaves of *cincinnata* are not flat like regular *A. majus* leaves, they are malformed with a large number of ripples over the surface (Figure 8.11). The question addressed here is how they obtain these ripples.

The ripples observed in the leaf are similar to the elementary examples shown in Figures 6.3(b) and 7.4. A simple test would be to model a leaf shape with isotropic growth morphogen placed at the edges. A model of an *A. majus* leaf was created, initial morphogen concentration distributed as shown in Figure 8.12, and simulated with canvas.

The results simulation produced the same type of rippling seen in the *cincin-*

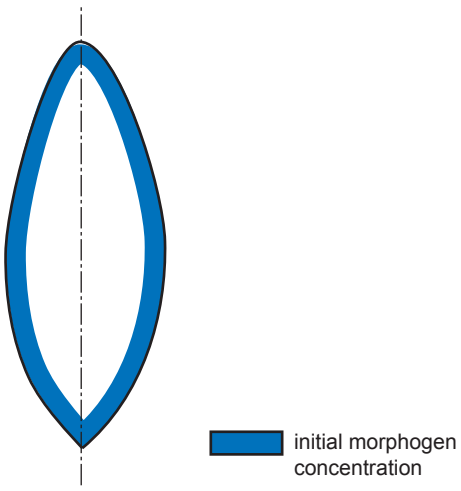


Figure 8.12: The *cincinnata* leaf simulation model. The initial morphogen distribution was painted on the edges as shown. The morphogen causes isotropic growth.

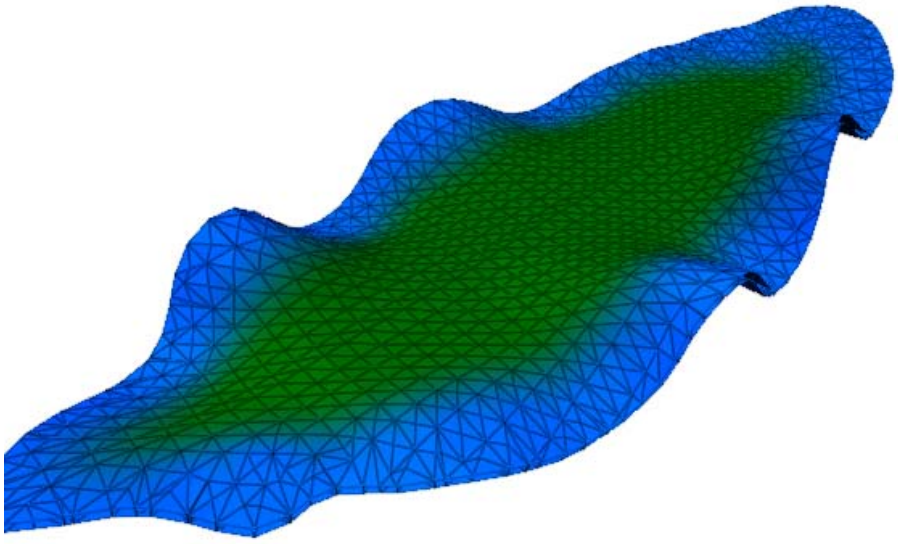


Figure 8.13: Resulting shape of the *cincinnata* mutant leaf simulation. The leaf has rippled along its edges.

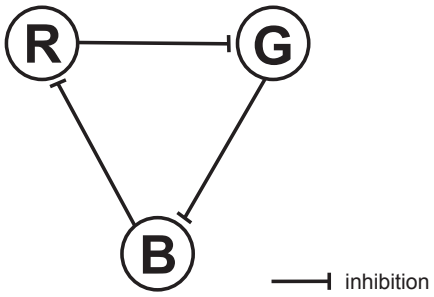


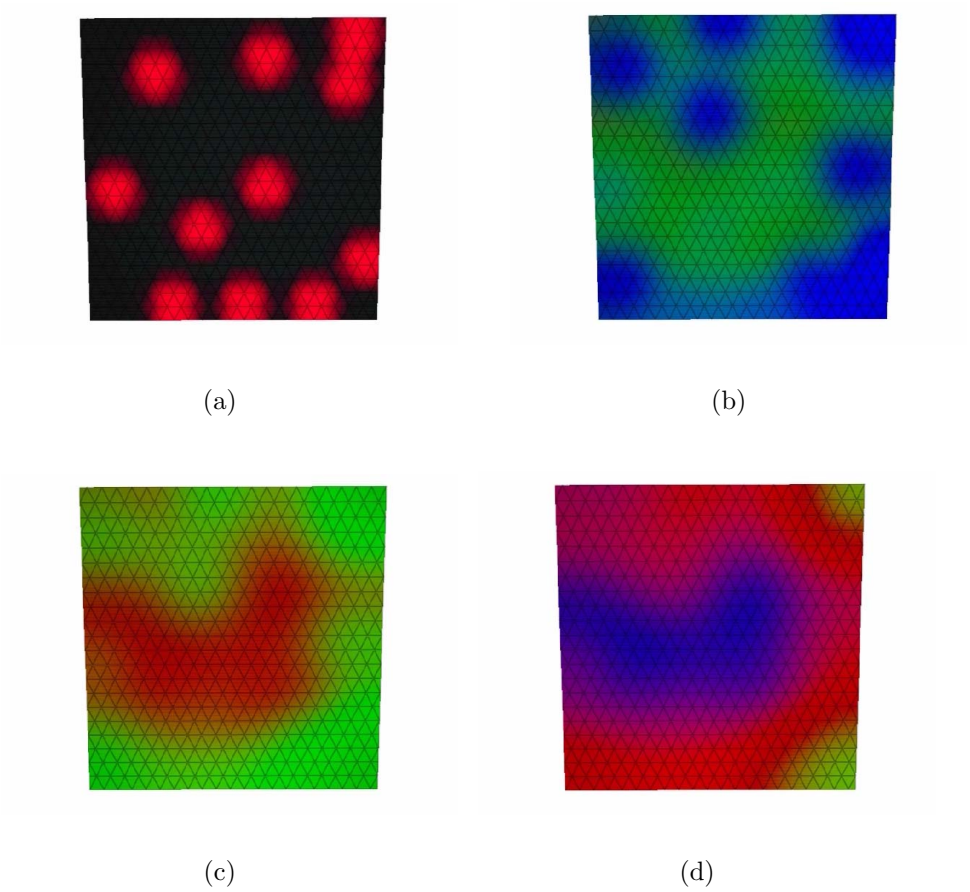
Figure 8.14: A limit cycle oscillator. Each gene inhibits the production of the next.

nata mutants (Figure 8.13). This suggests that the *cincinnata* mutants obtain their rippled edges because they are growing faster at their edges than at the center.

8.3 Repressilator

A model was developed to explore the possibilities of the connectionist method (Section 3.4), inspired from work by Elowitz and Liebler [32]. Elowitz and Liebler designed a synthetic genetic network to cause periodic expression of green fluorescent protein (GFP) in the bacteria *Escherichia coli*. They used GFP because it is easily observable with an ultraviolet light source.

They named their network the "repressilator". It was composed of three genes each inhibiting the production of the next, forming a stable oscillator. A similar system was constructed for this example using the connectionist formulation. Three morphogens were used, conveniently named R,G and B (drawn as red, green and blue) over a square surface. Each morphogen inhibits the production of the next, as shown in Figure 8.14).



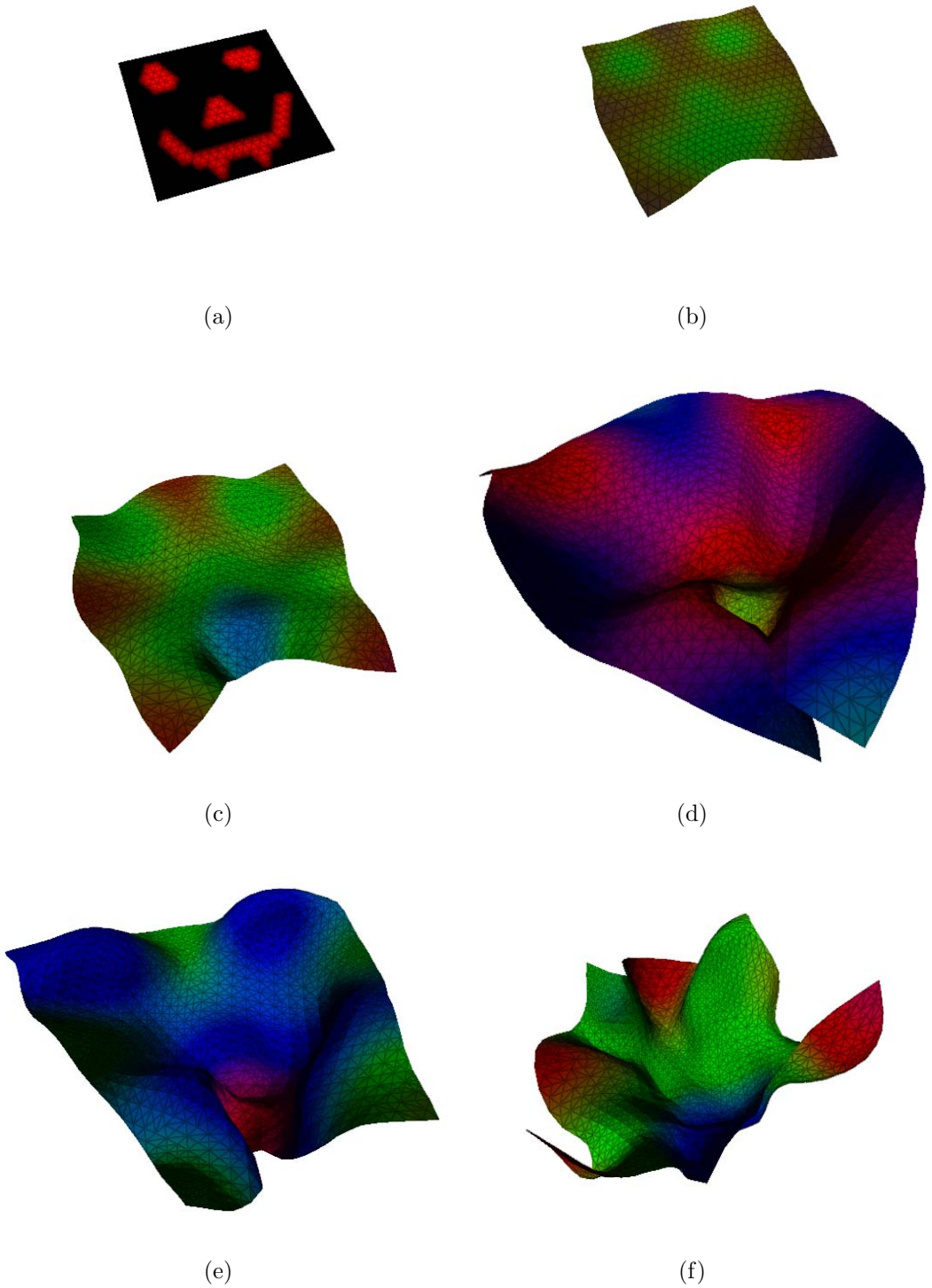
(a) (b)

(c) (d)

Figure 8.15: Time series of repressilator oscillation. The surface is initially seeded as shown in (a) by painting a few spots of the "R" morphogen. Figures (b)–(c) show the progression of morphogen oscillation on the surface.

The morphogens are also allowed to diffuse and decay. Diffusion creates a form of local communication, having the interesting effect of self synchronizing the system over the entire surface. Figure 8.15 shows an evolution of the morphogen concentrations over time.

As an experiment, the expression of R was linked to isotropic growth. Many novel forms were obtained in this way, with one example shown in Figure 8.16. Although



(a) (b)

(c) (d)

(e) (f)

Figure 8.16: Time series of a surface growing from repressilator oscillation linked to isotropic growth.

far from any type of recognizable surface, this example illustrates the type of complex surfaces achievable using a simple feedback system.

Chapter 9

Conclusions

9.1 Summary of Contributions

I have developed Canvas and CanvasLite, tools for exploring the link between growth, form and pattern formation models. Both tools use physically based simulation to model the growth of two-dimensional surfaces deforming in three dimensions. Each tool simulates diffusible morphogens over the surface that are responsible for effecting growth.

CanvasLite used a mass-spring system for simulating surface physics. The tool is fast enough for interactive experimentation, though at the cost of fixed discretization, and rectilinear geometry. Morphogens could effect isotropic and anisotropic growth, as well as changes in curvature. The tool is useful enough to model elementary examples of growth, but too limited for practical biological modeling.

Three methods for visualizing surface properties with CanvasLite were presented. An internal strain visualization method inspired by glass stress rings reveals the strain state of the surface. Gaussian curvature was calculated based on a modified Gauss-Bonet theorem and used to show the curvature of the surface. Finally, indicatrices were used for visualizing the measured growth tensor of the surface.

Canvas was developed as a more versatile tool, using finite element methods for simulating surface physics and morphogen diffusion and decay. FEM allowed for arbitrary and dynamic discretization of the surface. *Viscoelastic growth* was

introduced as a method for integrating growth into FEM elasticity. Canvas also integrated a simple model for morphogen interactions, based on the connectionist formulation of Mjolsness *et al* [60].

Four biological models using Canvas were presented. A model of sea urchin development used a morphogen to isotropically grow a test. The leaf and dorsal petal lobes of *A. majus* were modeled. Isotropic growth morphogen at the edges of a leaf was used to achieve similar ripples to those seen in the *cincinnata* mutant. The gradient of a morphogen was used to orient growth globally in a model of dorsal petal lobe development. Finally, morphogen interactions were used simulate a repressilator [32] genetic network and used to drive isotropic growth, creating some novel forms.

9.2 Future Work

9.2.1 Large Deformations

Although viscoelastic growth provides a way to implement growth within linear FEM elasticity, it cannot be assumed that all biological tissues grow in this way. The purely elastic growth of CanvasLite produces ripples much more readily and easily than Canvas. It should be possible to implement a purely elastic growth into Canvas using large deformation (non-linear) FEM methods. A method that seems well suited for this purpose would be a *corotational* formulation as presented by Cook [18].

9.2.2 Hybrid Surface Models

Methods that are neither particle or finite element based have been developed for surface simulation [3, 17]. These methods use arbitrary polygonal meshes for the

surface, and simplified physics. It should be possible to develop a faster growth model based on these methods. It would also be interesting to construct a volumetric model, so that models such as root apices could be properly modeled as a solid, instead of a surface.

9.2.3 Programmability

During the construction of the models in Chapter 8, the need to re-program the system was recurrent. It is unlikely that the current implementation could serve as a tool to implement every conceivable biological system. What is needed is an environment for rapid experimentation, which could be satisfied with a programmable system. A system with all the flexibilities of L-systems would be ideal. This could allow for anisotropic diffusion, finer construction of the growth tensor, and almost any morphogen interactions conceivable.

9.2.4 Other Uses

The current implementation of Canvas could be coupled with genetic algorithms. A large part of constructing the models of Chapter 8 involved optimizing parameters, which genetic algorithms are ideal for. Genetic algorithms could also be used in conjunction with connectionist interactions to evolve regulatory networks for growth.

Bibliography

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of mathematical functions*, Dover Publications, Inc., New York, 1972.

[2] B. Alberts, S. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Essential Cell Biology*, Garland Publishing, 1998.

[3] D. Baraff and A. Witkin, *Large steps in cloth simulation*, in Proceedings of the 25th annual conference on Computer graphics and interactive techniques, ACM Press, 1998, pp. 43–54.

[4] N. Barkai and S. Leibler, *Robustness in simple biochemical networks*, Nature, 387 (1997), pp. 913–917.

[5] J. Blinn, *Using tensor diagrams to represent and solve geometric problems*, 2001. SIGGRAPH Course Notes.

[6] H. D. Block, *Introduction to Tensor Analysis*, Charles E. Merrill Books Inc., 1962.

[7] M. T. Borisuk and J. J. Tyson, *Bifurcation analysis of a model of mitotic control in frog eggs*, Journal of Theoretical Biology, 195 (1998), pp. 69–85.

[8] D. Breen, D. House, and P. Getto, *A physically-based particle model of woven cloth*, The Visual Computer, 8 (1992), pp. 264–267.

[9] L. Brillouin, *Tensors in Mechanics and Elasticity*, Academic Press, 1964.

[10] A. Cachia, J.-F. Mangin, D. Rivire, F. Kherif, N. Boddaert, A. Andrade, D. Papadopoulos-Orfanos, J.-B. Poline, I. Bloch, M. Zilbovicius, P. Sonigo, F. Brunelle, , and J. Rgis, *A primal sketch of the cortex mean curvature: a morphogenesis based approach to study the variability of the folding patterns*, Submitted to IEEE Transactions on Medical Imaging, (2002).

[11] R. Carpenter, C. Martin, and E. S. Coen, *Comparison of genetic behavior of the transposable element tam3 at 2 unlinked pigment loci in antirrhinum-majus*, Molecular and General Genetics, 207 (1987), pp. 82–89.

[12] K. C. Chen, A. Csikasz-Nagy, B. Gyorffy, J. Val, B. Novak, and J. J. Tyson, *Kinetic analysis of a molecular model of the budding yeast cell cycle*, Molecular Biology of the Cell, 11 (2000), pp. 369–391.

[13] R. W. Clough, *The finite element method in plane stress analysis*, in Proceedings of 2nd ASCE Conference on Electronic Computation, Pittsburgh, PA, 1960.

[14] E. Coen, *The Art of Genes*, Oxford University Press, 1999.

[15] E. S. Coen, R. Carpenter, and C. Martin, *Transposable elements generate novel spatial patterns of gene- expression in antirrhinum majus*, Cell, 47 (1986), pp. 285–296.

[16] D. Cohen, *Computer simulation of biological pattern generation processes*, Nature, 216 (1967), pp. 246–248.

[17] J. Combaz and F. Neyret, *Painting folds using expansion textures*, in Pacific Graphics, october 2002.

[18] R. Cook, D. Malkus, M. Plesha, and R. J. Witt, *Concepts and Applications of Finite Element Analysis*, John Wiley & Sons, 2002.

[19] R. Courant, *Variational methods for the solutions of problems of equilibrium and vibrations*, Bulletin of the American Mathematical Society, 49 (1943), pp. 1–23.

[20] E. Crampin, E. Gaffney, and P. Maini, *Reaction and diffusion on growing domains: Scenarios for robust pattern formation*, Bulletin of Mathematical Biology, 61 (1999), pp. 1093–1120.

[21] E. Crampin and P. Maini, *Modelling biological pattern formation: The role of domain growth*, Comments on Theoretical Biology, 6 (2001), pp. 229–249.

[22] T. Delmarcelle and L. Hesselink, *Visualizing second order tensor fields with hyperstreamlines*, IEEE Computer Graphics and Applications, 13 (1993), pp. 25–33.

[23] ——, *The topology of symmetric second-order tensor fields*, Proceedings IEEE Visualization, (1994), pp. 140–147.

[24] D. M. Dimian, *A physically-based model of folded surfaces with an application to plant leaves*, Master's thesis, University of Calgary, 1997.

[25] D. Dooley and M. Cohen, *Automatic illustration of 3d geometric models: Surfaces*, in Proceedings of Visualization 90, 1990, pp. 307–313.

[26] S. Douady and Y. Couder, *Phyllotaxis as a dynamical self organizing process (part i, ii, iii)*, Jounal of Theoretical Biology, 178 (1996).

[27] C. L. Dym, *Introduction to the Theory of Shells*, Hemisphere Publishing Corporation, New York, 1990. QA935.D89 1990.

[28] G. Eble, D. Erwin, M. Foote, and D. Raup, *Computational approaches to theoretical morphology*, tech. rep., Santa Fe Institute, November 2001.

[29] L. Edelstein-Keshet, *Mathematical Models in Biology*, Random House/Birkhauser, 1988.

[30] M. Eden, *A two-dimensional growth process*, in Proceedings of Fourth Berkeley Symposium on Mathematics, Statistics, and Probability, vol. 4, University of California Press, Berkeley, 1960, pp. 223–239.

[31] O. Ellers, *A mechanical model of growth in regular sea urchins: redictions of shape and a developmental morphospace*, Proceedings of the Royal Society of London B, 254 (1993), pp. 123–129.

[32] M. B. Elowitz and S. Liebler, *A synthetic oscillatory network of transcriptional regulators*, Nature, 403 (2000), pp. 335–338.

[33] M. Epstein and G. A. Maugin, *On the geometrical material structure of anelasticity*, Acta Mechanica, 115 (1996), pp. 119–131.

[34] P. Federl, *Modeling Fracture Formation on Growing Surfaces*, PhD thesis, University of Calgary, 2002.

[35] J. P. Fitch and B. A. Sokhansanj, *Genomic engineering: moving beyond dna sequence to function*, Proceedings of the IEEE, 88 (2000), pp. 1949 – 1971.

[36] F. D. Fracchia, P. Prusinkiewicz, and M. J. M. de Boer, *Animation of the development of multicellular structures*, in Computer Animation '90, N. Magnenat-Thalmann and D. Thalmann, eds., Tokyo, 1990, Springer-Verlag, pp. 3–18.

[37] H. Goldstein, *Classical Mechanics*, Addison-Wesley, Reading, MA, 1950.

[38] P. B. Green, *Pattern formation in shoots: A likely role for minimal energy configurations of the tunica*, International Journal of Plant Science, 153 (1992), pp. S59–S75.

[39] E. Grinspun, P. Krysl, and P. Schröder, *Charms: a simple framework for adaptive simulation*, in Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 2002, pp. 281–290.

[40] P. Hanrahan and P. Haeberli, *Direct WYSIWYG painting and texturing on 3D shapes*. Proceedings of SIGGRAPH 90 (Dallas, Texas, August 6-10, 1990). ACM SIGGRAPH, New York, 1990, pp. 215–223.

[41] B. J. Harrison and J. Fincham, *Instability at the pal locus in antirrhinum majus. i. effects of environment on frequencies of somatic and germinal mutation*, Heredity, 19 (1964), pp. 237–258.

[42] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray, *From molecular to modular cell biology*, Nature, 402 (1999), pp. C47–C52.

[43] Z. Hejnowicz and J. Romberger, *Growth tensor of plant organs*, Journal of Theoretical Biology, 110 (1984), pp. 93–114.

[44] W. G. Hopkins, *Introduction to Plant Physiology*, John Wiley and Sons, Inc., New York, 1999.

[45] K. H. Huebner, E. A. Thornton, and T. G. Byrom, *The Finite Element Method for Engineers*, John Wiley and Sons, Inc., 1995.

[46] V. Interrante, H. Fuchs, and S. Pizer, *Conveying the 3d shape of smoothly curving transparent surfaces via texture*, IEEE Transactions on Visualization and Computer Grahics, 3 (1997), pp. 98–117.

[47] A. G. Jacobson and R. Gordon, *Changes in the shape of the developing vertebrate nervous system analyzed experimentally, mathematically and by computer simulation*, Journal of Experimental Zoology, 197 (1976), pp. 191–246.

[48] C. Jirasek, *Branch shape expressed using l-systems*, Master's thesis, University of Calgary, Canada, 2000.

[49] E. Kreyszig, *Advanced Engineering Mathematics*, John Wiley and Sons, Inc., 1993.

[50] R. D. Kriz, E. H. Glaesgen, and J. D. MacRae, *Eigenvalue-eigenvector glyphs: Visualizing zeroth, second, forth and higher order tensors in a continuum*, Workshop on Modelling the Development of Residual Stresses During Thermoset Composite Curing, (1995).

[51] A. Lindenmayer, *Mathematical models for cellular interaction in development, Parts I and II.*, Journal of Theoretical Biology, 18 (1968), pp. 280–315.

[52] A. Lindenmayer and G. Rozenberg, *Parallel generation of maps: Developmental systems for cell layers.*, in Graph grammars and their application to computer science; First International Workshop, V. Claus, H. Ehrig, and G. Rozenberg, eds., Lecture Notes in Computer Science 73, Springer-Verlag, Berlin, 1979, pp. 301–316.

[53] H. H. McAdams and A. Arkin, *Simulation of prokaryotic genetic circuits*, Annual Review of Biophysics and Biomolecular Structure, 27 (1998), pp. 199–224.

[54] H. H. McAdams and L. Shapiro, *Circuit simulation of genetic networks*, Science, 269 (1995), pp. 650–656.

[55] H. Meinhardt, *The Algorithmic Beauty of Seashells*, Springer Verlag, 1998.

[56] H. Meinhardt and A. Gierer, *A theory of biological pattern formation*, Kybernetik, 12 (1972), pp. 30–39.

[57] ——, *Applications of a theory of biological pattern formation based on lateral inhibition*, Journal of Cell Science, 15 (1974), pp. 321–346.

[58] D. Metaxas and D. Terzopoulos, *Dynamic deformation of solid primitives with constraints*, in Proceedings of the 19th annual conference on Computer graphics and interactive techniques, ACM Press, 1992, pp. 309–312.

[59] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete differential-geometry operators for triangulated 2-manifolds*, in VisMath '02, 2002.

[60] E. Mjolsness, D. H. Sharp, and J. Reinitz, *A connectionist model of development*, Journal of Theoretical Biology, 152 (1991), pp. 429–454.

[61] D. R. Musser, J. D. Gillmer, and A. Saini, *STL Tutorial and Reference Guide, Second Edition: C++ Programming with the Standard Template Library*, Addison-Wesley, Boston, MA, 2001.

[62] J. Nakielski and M. Rumpf, *Growth in apical meristems of plants visualization tools and growth tensor methods*. SFB 256 (Report no. 11).

[63] J. Neider, T. Davis, and M. Woo, *OpenGL Programming Guide*, Addison-Wesley, 1993.

[64] I. Newton, *Philosophiae Naturalis Principia Mathematica*, Printed by Joseph Streater by order of the Royal Society, London, 1687.

[65] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins, *Graphical modeling and animation of ductile fracture*, in Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 2002, pp. 291–294.

[66] J. F. O'Brien and J. K. Hodgins, *Graphical modeling and animation of brittle fracture*, in Proceedings of the 26th annual conference on Computer graphics

and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 137–146.

[67] U. Philippi and W. Nachtigall, *Constructionaly morphology of sea urchin tests*, Proceedings of the 2nd international symposium, (1991), pp. 183–191.

[68] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1999.

[69] P. Prusinkiewicz, M. Hammel, and R. Mech, *Visual models of morphogenesis:a guided tour*. http://www.cpsc.ucalgary.ca/Redirect/bmv/vmm-deluxe/TitlePage.html.

[70] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*, Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.

[71] M. Ptashne, *A Genetic Switch*, Cell Press and Blackwell Scientific Publications, 1986.

[72] S. Pulla, A. Razdan, and G. Farin, *Improved curvature estimation for watershed segmentation of 3-dimensional meshes*, Submitted to IEEE Transactions on Vizualization and Computer Graphics, (2002).

[73] C. V. Rao and A. Arkin, *Control motifs for intracellular regulatory networks*, Annual Review of Biomedical Engineering, 3 (2001), pp. 391–419.

[74] D. M. Raup, *Theoretical morphology of echinoid growth*, Journal of Paleontology, 42 (1968), pp. 50–63.

[75] O. W. Richards and A. J. Kavanagh, *The analysis of the relative growth gradients and changing form of growing organisms: Illustrated by the tobacco leaf*, The American Naturalist, 77 (1943), p. 385.

[76] M.-C. Rivara and P. Inostroa, *A discussion on mixed (longest-side midpoint insertion) delaunay techniques for the triangulation refinement problem*, in 4th International Meshing Roundtable, Albaquerque, New Mexico, Oct 1995, pp. 335–346.

[77] A.-G. Rolland, *Quantitative analysis of petal morphology in Antirrhinum majus: an interdisciplinary approach*, PhD thesis, University of East Anglia, 2003.

[78] D. Schweitzer, *Artificial texturing: An aid to surface visualization*, in Proceedings of the 10th annual conference on Computer graphics and interactive techniques, 1983, pp. 23–29.

[79] J. R. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, teaching notes, Carnegie Mellon University, 1994.

[80] L. Snyder and W. Champness, *Molecular Genetics of Bacteria*, ASM Press, Washington, DC, 1997.

[81] A. Stepanov and M. Lee, *The Standard Template Library*, HP Technical Report HPL-94-34, Hewlett Packard, 1995.

[82] P. S. Stevens, *Patterns in nature*, Brown Little, Boston, 1974.

[83] L. A. Taber, *Biomechanics of growth, remodeling, and morphogenesis*, Applied Mechanics Rewiews, 48 (1995), pp. 487–545.

[84] M. Telfrod, *Structural models and graphical simulation of echinoids*, Balkema, Rotterdam, 1994, pp. 895–899.

[85] D. Terzopoulos and K. Fleischer, *Deformable models*, The Visual Computer, 4 (1988), pp. 306–331.

[86] D. Terzopoulos and K. Fleischer, *Modeling inelastic deformation: viscolelasticity, plasticity, fracture*, in Proceedings of the 15th annual conference on Computer graphics and interactive techniques, ACM Press, 1988, pp. 269–278.

[87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, *Elastically deformable models*. Proceedings of SIGGRAPH '87 (Anaheim, July 27-31, 1987), in *Computer Graphics* 21,4 (July 1987), pages 205-214, ACM SIGGRAPH, New York, 1987.

[88] D. Thomson, *On growth and form*, Cambridge University Press, London, 1917.

[89] J. Todd and E. Mingolla, *Perception of surface curvature and direction of illumination from patterns of shading*, Journal of Experimental Psychology: Human Perception and Performance, 9 (1983), pp. 583–595.

[90] T. Toffoli and N. Margolus, *Cellular automata machines: a new environment for modelling*, The MIT Press, Cambridge, MA, 1987.

[91] A. Turing, *The chemical basis of morphogenesis*, Philosophical Transactions of the Royal Society of London B, 237 (1952), pp. 37–72.

[92] G. Turk, *Generating textures on arbitrary surfaces using reaction diffusion.* Proceedings of SIGGRAPH 91 (Las Vegas, California, July 28–August 2, 1991). ACM SIGGRAPH, New York, 1991, pp. 289–298.

[93] S. Ulam, *On some mathematical properties connected with patterns of growth of figures*, in Proceedings of Symposia on Applied Mathematics, vol. 14, American Mathematical Society, 1962, pp. 215–224.

[94] C. A. Vincent, R. Carpenter, and E. S. Coen, *Cell lineage3 patterns and homeotic gene activity during antirrhinum flower development*, Current Biology, 5 (1995), pp. 1449–1457.

[95] M. Walter, A. Fournier, and D. Menevaux, *Integrating shape and pattern in mammalian models*, in Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM Press, 2001, pp. 317–326.

[96] T. J. Willmore, *An Introduction to Differential Geometry*, Oxford University Press, 1957.

[97] A. Witkin and M. Kass, *Reaction diffusion textures.* Proceedings of SIGGRAPH 91 (Las Vegas, California, July 28–August 2, 1991). ACM SIGGRAPH, New York, 1991, pp. 289–298.

[98] O. C. Zienkiewicz, *The Finite Element Method*, McGraw Hill, 1967.

Appendix A

A.1 Elasticity Stiffness Matrix Derivation

There are many methods to derive the finite element stiffness matrix for elasticity, but here we use the *principle of virtual work* [18]. This principle states that the internal virtual work of a body (IVW) must be equal to the external virtual work (EVW):

$$IVW = EVW \tag{A.1}$$

The IVW is the internal strain energy stored in the body. In continuous terms the IVW density can be stated as $\{\epsilon\}^T\{\sigma\}$ (Symbols are as defined in Section 2.2.2, 2.2.3 and 4.2.2). The EVW is the work done on the object by external forces. The EVW density can be stated as $\{\tilde{\delta}\}^T\{\tilde{F}\}$, where $\{\tilde{F}\}$ is a *body force* that acts on an infinitesimal unit $d\Omega$. Integrating these quantities over the volume of the element, we obtain [18, p.88]:

$$\int_{\Omega^{(e)}} \{\epsilon\}^T\{\sigma\}d\Omega = \int_{\Omega^{(e)}} \{\tilde{\delta}\}^T\{\tilde{F}\}d\Omega \tag{A.2}$$

Substituting this with a modified Hookes law that incorporates a pre-strain:

$$\{\sigma\} = [C]\{\epsilon\} - [C]\{\epsilon_0\} \tag{A.3}$$

we obtain:

$$\int_{\Omega^{(e)}} \{\epsilon\}^T [C]\{\epsilon\} d\Omega = \int_{\Omega^{(e)}} \{\epsilon\}^T [C]\{\epsilon_0\} d\Omega + \int_{\Omega^{(e)}} \{\tilde{\delta}\}^T \{\tilde{F}\} d\Omega \tag{A.4}$$

We can then apply the finite element formulation of the strain $\{\epsilon\} = [\mathbf{B}]\{\delta\}$ (equation 4.28), displacement interpolation $\{\tilde{\delta}\} = [\mathbf{N}]\{\delta\}$ (equation 4.21), and the matrix identity $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$ to obtain:

$$\{\delta\}^T \left(\int_{\Omega^{(e)}} [\mathbf{B}]^T [\mathbf{C}][\mathbf{B}] d\Omega \{\delta\} - \int_{\Omega^{(e)}} [\mathbf{B}]^T [\mathbf{C}]\{\epsilon_0\} d\Omega - \int_{\Omega^{(e)}} [\mathbf{N}]^T \{\tilde{F}\} d\Omega \right) = 0 \tag{A.5}$$

Note that $\{\delta\}^T$ and $\{\delta\}$ are not functions of the material coordinates and don't appear inside the integral. We can therefore factor $\{\delta\}$ out and we obtain the linear system:

$$[\mathbf{K}]\{\delta\} = \{\mathbf{F}\} \tag{A.6}$$

Where our *stiffness-matrix* is given by:

$$[\mathbf{K}] = \int_{\Omega^{(e)}} [\mathbf{B}]^T [\mathbf{C}][\mathbf{B}] d\Omega \tag{A.7}$$

and our force vector is given by:

$$\{\mathbf{F}\} = \int_{\Omega^{(e)}} [\mathbf{N}]^T \{\tilde{F}\} d\Omega \tag{A.8}$$

and the *pre-strain* term given by:

$$\{F_0\} = \int_{\Omega^{(e)}} [\mathbf{B}]^T [\mathbf{C}]\{\epsilon_0\} d\Omega \tag{A.9}$$

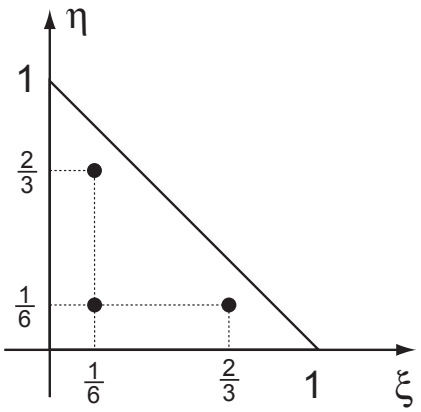


Figure A.1: Integration points for a 2D triangle.

i	ξ	η	ζ
1	0.1666666667	0.1666666667	0.5773502691
2	0.6666666667	0.1666666667	0.5773502691
3	0.1666666667	0.6666666667	0.5773502691
4	0.1666666667	0.1666666667	-0.5773502691
5	0.6666666667	0.1666666667	-0.5773502691
6	0.1666666667	0.6666666667	-0.5773502691

Table A.1: Integration points for a wedge. ($w_{gauss} = \frac{1}{6}$ for all points.)

A.2 Integration Points for Wedge Elements

The points used for integrating a wedge element were obtained by applying the standard Gaussian rule in the ζ dimension [1, p.892], and using three points within the $\xi\eta$ plane (Figure A.1). Combining these, we obtain the six integration points shown in table A.2. The weight of each point (w_{gauss}) is equal to $\frac{1}{6}$, giving a total volume of 1.0.

A.3 CanvasLite Input File

Usage: `canvaslite.exe` *inputfile*

inputfile will default to `input.cvs` if it is not specified. If no input file is found, the program will exit. Each line in the input file specifies one or a set of parameters. No item may be omitted, and keywords must follow this order. Blank lines are allowed. The keywords are described in the following table. Italicized words are variables with subscripts indicating the type of variable: f=float, u=unsigned, b=boolean, s=string.

Command	Param.	Description
`Size` $width_u$ $length_u$		specifies the number of particles in the width and length of the mesh to be created
`Noise` $value_f$		initial randomization of particle location (even distribution over $[-value, value]$)
`PointMass` $value_f$	m	mass of all particles
`Damping` $value_f$	k_d	damping coefficient
`StretchStiffness` $value_f$	k_s	resistance to stretching deformations
`ShearStiffness` $value_f$	k_s	resistance to shearing deformations
`BendingStiffness` $value_f$	k_{bs}	resistance to bending deformations
`TimeStep` $value_f$	Δt	time step
`RButtonChem`x_u $value_f$		amount of morphogen x to be painted when the right mouse button is used to paint surface ($x = 0..3$)
`MButtonChem`x_u $value_f$		amount of morphogen x to be painted when the middle mouse button is used to paint surface ($x = 0..3$)
`Edges` $value_u$		if equal to 1, the concentration of morphogen 0 is set to 1.0 along the edges of the surface
`center` $value_u$		if equal to 1, the concentration of morphogen 0 is set to 1.0 in the central 3×3 region of the surface
`Chemical`x_u`Diffusibility` $value_f$	k_{diff}	diffusibility of morphogen x ($x = 0..3$)
`Chemical`x_u`Decay` $value_f$	k_{decay}	decay rate of of morphogen x ($x = 0..3$)
`Chemical`x_u`Sensitivity` $value_f$	k_{sens}	growth sensitivity of morphogen x ($x = 0..1$)
`Chemical`x_u`UCurl` $value_f$	k_{curl}	sensitivity of morphogen on bending springs in the u direction $x(x = 0..1)$
`Chemical`x_u`VCurl` $value_f$	k_{curl}	sensitivity of morphogen on bending springs in the v direction $x(x = 0..1)$
`TensorGrowth` $value_u$		turns the tensorial growth mode on (1) or off (0)
`TensorSensitivity` $value_f$	k_{tsens}	tensorial growth sensitivity
`Flat` $value_u$		if equal to 1, constrain particles within the xz plane
`Texture` "$filename_s$"		texture filename to be mapped to surface (Use 0 for no texture mapping.)
`Distance` $value_f$		viewing distance from object
`Specularity` $value_f$		specularity of surface $[0, 1]$

A.4 Sample CanvasLite Input File

```
Size: 25 25
Noise: 0.04
PointMass: 5.0
Damping: 0.3

StretchStiffness: 1.0
ShearStiffness: 1.0
BendingStiffness: 0.1

TimeStep: 0.5

RButtonChem0: 4.0
RButtonChem1: 0.0
RButtonChem2: 0.0
RButtonChem3: 0.0

MButtonChem0: 0.0
MButtonChem1: 4.0
MButtonChem2: 0.0
MButtonChem3: 0.0

Edges: 0
Center: 0

Chemical0Diffusibility: 0.025
Chemical0Decay: 0.01
Chemical0Sensitivity: 0.001
Chemical0UCurl: 0.00
Chemical0VCurl:0.00

Chemical1Diffusibility: 0.025
Chemical1Decay: 0.01
Chemical1Sensitivity: -0.001
Chemical1UCurl: 0.00
Chemical1VCurl: 0.00
```

```
Chemical2Diffusibility: 0.025
Chemical2Decay: 0.01

Chemical3Diffusibility: 0.025
Chemical3Decay: 0.01

TensorGrowth: 0
TensorSensitivity: 0.01

Flat: 0

Texture: 0
Distance: 35
Specularity: 0.7
```

A.5 Canvas Input File

Usage: `canvas.exe` *inputfile*

If no input file is specified, the program will exit. The input file consists of a number of nested sections, as described in section A.5.1. The format of the data in each section is described in sections A.5.2 to A.5.8.

A.5.1 Layout

The input file is structured into a number of nested data sections, as shown in the following listing. Each line in the input file contains one set of parameters, or an identifier beginning or ending a data section. No blank lines are allowed.

```
BeginCanvas 1.0
    BeginMaterial
        material data
        BeginColor
            color-1 data
        EndColor
            ...
        BeginColor
            color-n data
        EndColor
    EndMaterial
    BeginGeometry
        geometry data
    EndGeometry
    BeginConstraint
        constraint data
```

```
    EndConstraint
    BeginForce
        force data
    EndForce
    BeginSimulation
        simulation data
    EndSimulation
    BeginDraw
        draw data
    EndDraw
EndCanvas
```

White space before each identifier is ignored, and the layout of data within each section is"free-format". That is, the parameter lines can be specified in any order. The keywords are described in the following sections. Italicized words are variables with subscripts indicating the type of variable: f=float, u=unsigned, b=boolean, s=string.

A.5.2 Material Data

This section is used to define a number of physical properties of the surface, and the connectionist interaction variables. All lines are required.

COMMAND	PARAM.	DESCRIPTION
`Youngs` $value_f$	E	Young's modulus of the material
`Poisson` $value_f$	ν	Poisson's ratio of the material
`BaseColor` red_f $green_f$ $blue_f$		natural RGB color of the surface
`PaintValue` $value_f$		morhogen concentration for painting
`OffsetVector` $vector_f$	$\{\mathbf{h}\}$	threshold vector of the connectionist formulation (Section 4.3.4)
`RateVector` $vector_f$	$\{\mathbf{R}\}$	rate vector of the connectionist formulation
`ExponentVector` $vector_f$	$\{\mathbf{y}\}$	exponent of equation 4.48 for each g_i
`ConnectionMatrix` $matrix_f$	$[\mathbf{W}]$	connection matrix of the connectionist formulation

A.5.3 Color Data

This section describes parameters for morphogens. All lines are required.

Command	Param.	Description
`Name` "$filename_s$"		defines a name for the morphogen
`Diffusibility` $value_f$	k	diffusibility of the morphogen
`Density` $value_f$	ρ	density of material the morphogen diffuses through (see equation 4.43)
`Capacity` $value_f$	c	capacitance of material the morphogen diffuses through (see equation 4.43)
`Decay` $value_f$	h	decay rate of the morphogen
`DecayTo` $value_f$	T_e	concentration the morphogen decays to (usually 0.0)
`DrawColor` red_f $green_f$ $blue_f$		RGB values used to draw this morphogen

A.5.4 Geometry Data

This section is used to specify the surface geometry. All lines are optional.

Command	Description
`Form sheet` $width_f$ $length_f$ $thickness_f$ $numelems_u$	Generate a sheet of the given dimensions. Use *numelems* number of elements.
`Form urchin` $scale_f$ $segments_u anus_b$	Generate an elementary sea urchin shape. If $anus_b$ is `true`, a hole is placed at apical system.
`File` "$filename_s$" $thickness_f$ $scale_f$	Import a text file describing a polygonal mesh. Antiquated, use `Obj` instead.
`Obj` "$filename_s$" $thickness_f$ $scale_f$	Import a Wavefront object file describing a polygonal mesh. Note: Obj file must be structured to use absolute vertex references and use only one normal per vertex.
`AddNoise` $value_f$	Add a random value to every XYZ coordinate between $[-value, value]$.

A.5.5 Constraint Data

The data in this section defines constraints to be placed on simulation variables. All lines are optional.

Command	Description
`Displacement` $node_u$ [`XYZ`] $value_f$	fix the displacement $\{\delta\}$ of a given node in dimension [xyz] to *value*
`Color` $node_u$ "$color_s$" $value_f$	fix the concentration $\{T\}$ of a morphogen at a given node to *value*

A.5.6 Force Data

Data in this section specifies how to construct the growth tensor. The `Growth` line is optional and may appear multiple times.

Command	Description
`Growth` "$magnitude_s$" $mult_f$ *anisotropy* "$gradient_s$"	Construct a growth tensor (Section 4.4). *magnitude* is the morphogen name from which the magnitude of the growth is derived, multiplied by the value *mult*. *anisotropy* is the anisotropy factor, which can be a float value or morphogen name. If it is not equal to 1.0, then *gradient*, a morphogen name must be specified. The gradient of morphogen *"gradient"* is used to derive a direction.

A.5.7 Simulation Data

This section specifies data relevant to the simulation of the model. All lines are required, except for `Alpha` and `Plasticity`, of which only one must be specified.

COMMAND	PARAM.	DESCRIPTION
`Elasticity` $state_b$		activates the FEM elasticity solver (equation 4.30) Set to `true` for normal operation.
`Diffusion` $state_b$		activates the FEM diffusion solver (equation 4.44) Set to `true` for normal operation.
`Timestep` $value_f$	Δt	simulation timestep
`Alpha` $value_f$	α	viscoelastic constant (equation 4.52)
`Plasticity` $value_f$	$\tau_{plasticity}$	compute α from $\tau_{plasticity}$ (equation 4.67)
`DynamicSubdiv` $state_b$		activate the dynamic subdivision algorithm (Section 4.2.7)
`MaxArea` $value_f$		criterion for subdivision of an element

A.5.8 Draw Data

This section defines variables for drawing the model to the screen as well as the default state of the visualization options when the program first starts. All lines are required except for `Output avi` which is optional.

COMMAND	DESCRIPTION
`Subtractive` $state_b$	If true, colors are added subtractively (like pigments). Otherwise, the RGB value are directly added together.
`Background` *float float float*	RGB background color of the viewer window
`Base` $state_b$	draw base object
`Deformed` $state_b$	draw deformed object
`RenderColor` $state_b$	draw morphogen concentrations
`Gradient` $state_b$	draw gradient of first color
`GFrame` $state_b$	draw growth frame for each element
`GrowthTensor` $state_b$	draw growth tensors
`Distance` $value_f$	default viewing distance
`Size` $width_u$ $height_u$	default viewing area dimensions in pixels
`Output avi` "$filename_s$"	Write frames to an `avi` file.

A.6 Canvas Sample Input File

```
BeginCanvas 1.0
    BeginMaterial
        Youngs  100.0
        Poisson 0.3
        Heterosis false
        BaseColor  0.0 0.0 0.0
        BeginColor
            Name "ChemicalA"
            Diffusibility 0.1
            Density 0.1
            Capacity 0.3
            Decay 0.02
            DecayTo 1.00
            DrawColor 1.0 0.0 0.0
        EndColor
        BeginColor
            Name "ChemicalB"
            Diffusibility 0.1
            Density 0.1
            Capacity 0.3
            Decay 0.02
            DecayTo 1.00
            DrawColor 0.0 1.0 0.0
        EndColor
        BeginColor
            Name "ChemicalC"
            Diffusibility 0.1
            Density 0.1
            Capacity 0.3
            Decay 0.02
            DecayTo 1.00
            DrawColor 0.0 0.0 1.0
        EndColor
        OffsetVector -1.0 -1.0 -1.0
        RateVector -9.0 -9.0 -9.0
        ExponentVector 4.0 4.0 4.0
        BeginConnectionMatrix
            0.0   0.0   1.0
```

```
            1.0   0.0   0.0
            0.0   1.0   0.0
        EndConnectionMatrix
    EndMaterial
    BeginGeometry
        Form sheet 40 40 0.1 900
        AddNoise 0.01
    EndGeometry
    BeginConstraint
    EndConstraint
    BeginForce
        Growth "ChemicalA" 0.4 1.0
    EndForce
    BeginSimulation
        Elasticity true
        Diffusion true
        Timestep 0.03
        Alpha 1.0
        DynamicSubdiv true
        MaxArea 7
        EndTime 0.1
    EndSimulation
    BeginDraw
        Subtractive false
        Background 1.0 1.0 1.0
        GrowthTensor false
        Deformed false
        DrawColor true
        Distance 55.0
        Size 640 480
        Output avi "Animation.avi"
    EndDraw
EndCanvas
```