Integration of Complex Shapes and Natural Patterns

by

Marcelo Walter

B. Sc. Electrical Engineering , Federal University of Rio Grande do Sul - Brazil, 1986

M. Sc. Computer Science, Federal University of Rio Grande do Sul - Brazil, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

we accept this thesis as conforming
to the required standard

The University of British Columbia

December 1998

Abstract

The process of generating an image for a computer graphics object is traditionally broken down into three steps: modelling of the shape or geometric attributes (such as height, width, and length), modelling of the visual attributes (how the object is going to look), and an *integration* step that connects the first two (a visual attribute is defined for every point on the surface of the object). The separation of modelling the shape from modelling the visual attributes makes the whole process highly flexible and powerful; from a conceptual point of view, the process is easier to handle.

While generally good for many classes of objects, this separation is prone to problems when the geometry of the object is complex. For example, the mapping of visual characteristics to every point of such complex surfaces is non-trivial. Furthermore, this separation assumes that these two steps are independent of each other, but for some objects, there is an interaction between the shape modelling and visual modelling that plays a significant role on the final image. Typical examples are patterned animals such as giraffes and leopards, where the pattern visible on the fur of an adult animal is the result of a process that took place while the animal was an embryo in the womb. In this case, modelling the interplay between the embryo growth process and the pattern formation process is as important as modelling the

individual processes themselves.

In this thesis we introduce a novel solution for integrating shape and visual modelling. This solution defines the visual attributes directly on the surface of the object as the object changes shape, for example, due to growth. We present results of applying this solution to a giraffe model.

This thesis makes three contributions: (1) a new model of mammalian pattern formation called *Clonal Mosaic*, suitable for computer graphics purposes and with strong biological plausibility. The model is based on cell division and cell-to-cell interactions, and it can generate repeating spotted and striped patterns occurring in several species of mammals, especially the big cats and giraffes; (2) a technique to modify the shape of an object based, for example, on a small set of input measurements. The technique consists of defining local coordinate systems (cylinders) around the growing parts of the body, each one being transformed according to the relevant growth data while maintaining their relationship with the adjoining parts and the continuity of the surface. The local coordinates also permit ordinary animation mainly as relative rotation such as in articulated objects; and, (3) the integration of the modelling of Clonal Mosaic patterns with the shape modification technique.

Finally, this thesis advances the notion of integration of independent tools as an important development in the field of computer graphics. Individual tools have been reaching exceptional levels of performance and therefore we need efficient ways to integrate them smoothly.

Contents

List of Tables

List of Figures

Acknowledgements

After 6 years and a few months working on something big and important as a PhD thesis, it is impossible not to say the obvious: I could not have done without the guidance, support, knowledge, and inspiration from Alain. His analogies and insights shape the world of computer graphics. I thank also the support and comments of my supervisory committee: Leah Keshet, David Forsey, Jack Soneyink, and Mark Reimers; To James Little, Richard Israel, and Jane Wilhelms for their role on the examination committee.

My family in Brazil gave me the most important: emotional support for staying a long time away from home. I missed important family gatherings (the wedding of my sister and the birth of two nephews!) but I hope they believe it was worth it! The friendship from the *Brazilian Gang* in Vancouver: Mário, Lilian, Dani, Marcelo, Lane, Margaret, Andrea, and Peixoto. They made these years in Vancouver feel almost like in Brazil (they could not prevent the bad weather though!). To Dani and Marcelo a special thank you for their support in my last stay in Vancouver as a student.

So many people in Imager helped my stay in Canada become a home away from home: Kevin (Mr. Acadia) Coughlan, Gene Lee, Martin Blais, Chris Healey,

Bill Gates, Viswha Ranjan, Rob Scharein, Jason Harrison, Jim Boritz, Makoto, Anne Lavergne, Roger Tam and Chris Chiu. Chris saved my thesis presentation trusting his laptop on my hands. A special thanks goes for Pierre Poulin. He introduced me to the wonderful world of graphics at UBC. Rob Walker was more than a friend: he was proofreader, technical advisor, and a TA-mate. The first years in Vancouver would be harder without the help and friendship from Antonio, Scott Hazelhurst, and Tien Truong. The peer support group aka Michael Sahota helped me put things in perspective during bad times. Michael McAllister provided Voronoi code for our patterns. The office-mates "bonding party" with Margaret and Davor will always be part of my memories.

In Brazil Alex helped me become a more relaxed person and I thank him for that. Paulinho and Robson were always ready to listen to me complaining about the rain in Vancouver. At Unisinos in Brazil Fernando Osório helped turn the transition from Canada to Brazil easier than I thought; Mauro Steigleder helped turn a laptop-crisis into something manageable. Thank you very much! Finally, nothing of this could be done without the financial support from CNPq, UNISINOS and the Department of Computer Science at UBC.

MARCELO WALTER

The University of British Columbia

December 1998

Chapter 1

Introduction

1.1 Motivation and Overview

One of the major goals of computer graphics is to compute an image of a virtual scene. For our purposes a scene can be any combination of objects including their material properties. Objects can be classified according to the way they interact with light and also, for many purposes, whether they are manufactured objects (desk, car, chair) or natural ones (trees, leopards, humans). This goal presents a gigantic challenge considering all possible interplay of factors in even the simplest scenes. As in many other complex tasks, a divide-and-conquer approach makes the task more manageable, and the whole process is traditionally broken into two independent parts.

First, we need to define the objects in a geometric sense, that is, we have to build the objects in terms of their geometric properties (for instance, height, width, and size). Objects can be arbitrarily complicated.

Second, we need to deal with the material properties of the objects and this

part can itself be split into two sub-parts. We have to *define* and *attach* some sort of information to the objects that will characterize their appearance in the final image. For instance, we have to decide on whether a desk is made of wood or some other material such as steel. This is a complex task mainly due to the fact that in general it is not straightforward to attach a given material property to all points of an arbitrarily, possibly complex, shape. It is important to note that the two sub-parts, definition and attachment, are almost always independent of each other. For example, we can define the same wood visual property to be applied in many different objects in the same scene.

The process described above is a generic and well-known approach in computer graphics. It has been progressively defined and refined since the modern era of computer graphics started more than 30 years ago. The separation of the general process into two independent parts can provide good results only in two situations. The first is when dealing with objects where visual and geometric attributes are defined separately and integrated at a later step (usually manufactured objects, such as wall paper or cloth). The second is when dealing with objects where the visual attributes result from a spatial three-dimensional process involving the whole object (such as marble).

The separation fails to provide good results in situations where these two parts are dependent on each other. In other words, there are cases when the establishment of the visual attributes is affected by a changing geometry of the object in question. Typical examples of these cases are patterned animals. They have reasonably complex shapes and visually elaborate fur patterns. The fur we see on an

adult animal is the result of a process which happened much earlier, during embryonic life. It is reasonable to assume that there is some degree of interaction between the shape-changing embryo and the pattern formation process. We believe that the proper computer graphics solution to these cases should take into account this interaction.

The main contribution of this thesis is a systematic approach to integrate the geometric and visual attributes of natural objects, such as patterned animals. Our approach is a departure from the standard two-step method in the sense that the patterning process happens "in place", i.e., as a surface-level process and more importantly takes into account the interplay between geometry and the visual properties, usually missing in standard computer graphics techniques. This is a step towards fully integrated graphics modelling and rendering systems, where the user will be able to model an object, if so desired, as one integrated entity, geometry and appearance together. And as an important final motivation we know that nature does not create pattern and shape separately. It is therefore reasonable to consider solutions that are inspired by nature itself.

Conceptually, our solution has three main parts: the modelling of the visual attributes, change of shape in a controlled manner, and the integration itself. Of these three, the modelling of the visual attributes is the only one domain-specific, since the solutions we are seeking are driven by the kind of patterns we want to achieve. The basic assumption when defining these parts is that in order to be able to have a good integration scheme we need to be able to control the steps involved before the actual integration takes place.

1.2 Integration of Shape and Pattern

Most of the previous work on the integration of the shape and the visual aspects of an object has been done in the context of texture mapping, a technique which implements the two-step process of defining and attaching the visual information to an object. Texture mapping is without question a powerful technique directly responsible for much of the striking visual effects now common in the entertainment and videogames industry, for example. Nevertheless, more than twenty years after it was proposed by Catmull [catm74] the computer graphics community is still addressing problems intrinsic to the technique, such as texture placement and texture distortion. This is a strong motivation for researching alternative methods for integration tools. This section first presents an overview of the texturing process in computer graphics, followed by a general description of the previous work on the integration problem.

1.2.1 Texture Mapping

Traditionally in computer graphics, the detailed visual information of a surface, such as color, is integrated with the surface of the object via texture mapping [heck86]. The basic idea consists of displaying a visual attribute of a surface as given by a map, the *texture map*. This map is represented as a two-dimensional array of values which are "pasted" onto the surface. A mapping step establishes a correspondence between any point on the surface being textured and the texture information. This mapping step is what we call *texture mapping*.[1]

[1]The main idea of indirectly manipulating some surface attribute has been extended in many ways and two of the most important ones are the controlled perturbation of the surface's normal vectors [blin78] and the extension to three-dimensions called *Solid Texture* [perl85, peac85], where

Intuitively, we can visualize the technique as an "elastic wall papering technique" where we apply a texture (the elastic wall paper) to the surface of an object (the wall), with the added difficulty that the wall can have any shape format. In Figure 1.1 we show the model for a cheetah with a real cheetah skin texture mapped on it. This figure will help us illustrate some of the drawbacks related to the technique, such as distortion and lack of local control.

(a) Cheetah skin texture map

(b) Texture mapped cheetah model

Figure 1.1: An example of texture mapping

Synthesis of Texture Maps

The first step when using texture mapping is to define the texture map. There are basically three ways to obtain a texture map: *(i)* scanning in a real-world pattern, either using a desktop scanner (Figure 1.1(a)) or a 3D digitizer which outputs range-

the texture is defined in 3-dimensional space as well.

data for a given object together with the color of the associated point on the surface; *(ii)* using a painting system to custom create an image which will be used as a texture map; *(iii)* procedurally computing or synthesizing a texture. Scanned real-world pictures are still the main source of texture maps for many texture mapping applications, since they usually provide the exact visual appearance desired. For some classes of objects, however, scanning a real-world adequate texture or painting an image is cumbersome and time consuming.

Take the case of modelling a giraffe and using texture mapping to generate the giraffe spots. If we decide to use a scanned texture we would need a full color image of a "good" and "stretched" giraffe skin. On the other hand we could decide to paint an image to mimic the giraffe fur pattern. Even for a skilled user both approaches would demand a great deal of effort and artistic ability. The problem would be even worse if we wanted more than one giraffe or a different subspecies. Despite having roughly the same appearance, each giraffe skin has its own characteristics and therefore we would need to build as many texture maps as giraffes needed, or find a way to modify a given texture into another, a task almost as difficult as generating the texture in the first place. Alternatively, we could define a procedure which would output a "good" giraffe pattern, providing controls for easily customization of textures. The pattern formation model proposed in this thesis provides such a mechanism.

Current research on texture synthesis has focused on how to generate a new texture from a sample texture in a controlled manner, such that perceptually both appear the same. This is the approach taken for instance by Heeger and Bergen [heeg95],

where current theories on texture discrimination were used to drive the synthesis procedure. A more recent approach for texture synthesis [de B97] considers textures to be samples from probabilistic distributions and the generation of a new texture is approached as a resample of this function. One of the main problems when using real-world images as textures is their resolution, often smaller than needed for texturing purposes. Therefore these approaches usually seek the generation of textures arbitrarily larger than the original.

The main limitation of these artificial texture generation techniques is that they are dependent upon the input information, a real texture. Therefore they cannot deal with textures containing elements which vary in size, color and orientation, a typical example being a giraffe skin.

Texture Placement

For most objects there is no easy mapping from the two-dimensional texture space (the space where the texture is defined) to the two-dimensional manifold space of the surface of the object. For objects represented as parametric surfaces[2] it is usually straightforward to define the mapping using the available parameterization. However, the parameterization defined during the modelling of the shape might not be suitable for texturing purposes. Usually, the modelling parameterization does not preserve areas or orientations, a necessity for texture mapping usage without distortions. Any trivial mapping is global and therefore "blind" to the local geometric information of the object causing distortions of the textured image (Figure 1.1(b)). A typical example is the sphere, where the texture information is compressed towards

[2]The definition of parametric surfaces is given in Section 4.2.2.

the poles since there is no mapping from a square to a sphere without singularities.

The research addressing the texture placement problem has been focused on two goals: easing the process for the user, and searching for an analytical solution which will compute the best possible mapping for any given texture mapping task.

Litwinowicz and Miller, for example, proposed an interactive system for texture placement [litw94]. Their idea is to indirectly define the mapping via user manipulation of the texture while it is being placed on the surface of the object. The solution imposes some constraints on the process, such as forcing camera and object to remain fixed, in order to achieve interactive rates for placement (between 6.5 to 1.3 frames per second). The correction for distortion is done through warping of the texture image. This technique does not address the problem of when the same texture is shared by many surfaces or when more than one texture is used for the same object. Another solution that pre-warps the texture image before using it is presented by Arad and Elber [arad97]. However, the distortion process is restricted for a particular viewpoint which limits considerably the generality of the technique.

Still on the problem of texture placement, Shirman and Kamen [shir97] introduced an intermediate parametrization called τ-mapping between the original parametrization of the object and the texture space. The motivation behind this approach is to isolate the modelling parameterization of the object from the texture. The problem of texture placement is therefore reduced to modifying the τ-mapping. The authors recognize that introducing one more step in the texturing pipeline will slow down the process and consequently they use an approximation for the actual τ-mapping that is computed by linear interpolation on a set of selected points on the

object.

An alternative approach for texture placement introduced by Hanrahan and Haeberli [hanr90] is to paint a texture directly onto the surface of the object. The objects are modeled as a collection of many small quadrilaterals from which a parameterization of the surface can be derived. The mapping function is indirectly established by the user "painting" on the surface. There is no actual distortion to be corrected since there is no *a priori* texture map to be distorted. The drawbacks are that: *(i)* the final result is still highly dependent on the artistic abilities of the user and therefore achieving a visually elaborate texture can be difficult and *(ii)* as presented, the approach does not handle a pre-existing texture, and therefore cannot be used to correct texture distortions by visual inspection.

A more analytical approach to minimize distortions was presented by Maillot *et al.* [mail93]. The approach first defines a metric for the distortion and tries to minimize its value globally. The metric is based on the amount of elastic deformation resultant from the mapping process. The problem is simplified by restricting the objects to be represented by triangular meshes since for these objects they can approximate the deformation of the mapped image by the summation of deformations on each triangle. As mentioned by the authors, for some objects the energy minimization is not possible and they propose to use instead a mapping function which is local and not necessarily continuous. In other words the object is split into *charts* and a collection of charts is called an *atlas*. The creation of charts takes into account surface curvature and the user interacts visually to achieve the best atlas for a particular object. The idea behind charts is to represent a non-developable surface as a set

of developable surfaces. A developable surface is a surface that can be deformed to planar shape without changing length measurements in it [fari90]. In a similar fashion the work presented by Bennis *et al.* "cuts" a given 3D parametric surface into regions that can be flattened out without warping [benn91]. The minimization of distortion is achieved through a compromise between cuts and distortions.

When the user has control over how the texture is generated, more effective ways to avoid distortions are possible. The method by Witkin and Kass [witk91] uses models described parametrically as a collection of patches and synthesizes textures using Reaction-Diffusion systems[3]. The problem of texture distortion caused by the mapping from the texture parameter space to the surface space is solved in an integrated manner. The texture synthesis incorporates a correction factor for the distortion, that is, the diffusion rates present in the Reaction-Diffusion system were controlled to account for the geometric distortions present on the surface. This correction, however, only works for surfaces that can be described by a single parametric function, usually not the case for complex surfaces. Seamless periodic textures are created using cyclic boundary conditions, i.e., points that shared a common boundary in different patches had the same boundary conditions.

Rendering and Animation of Texture-mapped Objects

We should also mention two common problems associated with texture mapping: rendering and animation of texture-mapped objects. Although these are important issues, they are not part of our motivation towards better integration methods and therefore our description will be limited.

[3] A more detailed description of this work is given later in Section 2.4.2.

The rendering issue is how to correctly sample the texture map in order to avoid aliasing artifacts. Large areas of the texture map can potentially be mapped on a very small patch on the object and vice-versa. MIP maps [will83] are an efficient solution to this problem, implemented in many commercial software packages. A mipmap is a series of precomputed filtered texture maps organized in a pyramidal format. The base of the pyramid contains the higher resolution version of the map whereas the top one contains the lower one. In order to access the pyramid, texture coordinates are derived from the projection of the polygon in the screen space.

The animation issue is how to guarantee that the texture follows the object correctly. Many *ad hoc* solutions are used, such as using small polygons forcing an almost one-to-one mapping from object space to texture space. When the object is articulated and the same texture is used for the whole object, it is even harder to guarantee seamless texturing. Even though we are not addressing this problem in particular, we can say that our integrated solution for adding rich visual detail to objects will not add any difficulty to the problem.

1.2.2 Previous Work on Integration

There has been little work outside the context of texture mapping addressing the integration of the shape and visual attributes of an object as a task *per se*. Three remarkable exceptions are the work by Turk [turk91], by Fowler *et al.* [fowl92], and by Fleischer *et al.* [flei95]. These papers present variations on the fundamental idea of computing the pattern on the surface of an object as a "growth-in-place" procedure.

Turk's work used Reaction-Diffusion textures[4]. Instead of mapping the generated pattern onto a polyhedral or parametric model, his approach simulates the Reaction-Diffusion system on the surface of the model, without the intermediate mapping from texture space to object space. Basically, the surface of the model is divided into cells and the Reaction-Diffusion system is simulated directly on the mesh formed by these cells. The cells for the simulation are the regions of a planar variation of a Voronoi diagram computed from a polyhedral representation of the model. The approach does not have the usual problems of texture discontinuity and distortion since there is just one mesh over which the Reaction-Diffusion system is simulated. Turk mentions that his mechanism could use surface properties, such as curvature, to specify parameters for the simulation and he shows one example where higher local curvature of the surface produced smaller spots on a giraffe's body. Some of his results, nevertheless, can appear strangely regular and artificial. More natural-looking results could be achieved only by the user specifying different speeds of diffusion for different parts of the body and initiator-cells, areas on the surface responsible to start the texture generation process.

Fleischer *et al.* also presented an approach for texturing with direct simulation on the surface of an object. The surface of the object is covered with cells that are constrained to remain on an iso-surface computed from the original model. The user specifies cell programs that define the behavior of cells over time. These cell programs are written as first-order differential equations which can be "added" to provide complex behavior. The end result of the simulation is a configuration of

[4]The term Reaction-Diffusion refers to a chemical system where at least two substances interact in a defined way. A full overview of Reaction-Diffusion systems is given later in Section 2.3.1.

cells that goes through a particle-to-geometry converter. This step generates shape and appearance for the cells to be rendered, based on their position and some other parameters. The whole approach is general, and can in principle generate many interesting organic-like textures, including Reaction-Diffusion ones. However, the complexity of writing the cell's programs through sets of differential equations is a serious constraint acknowledged by the authors themselves. Their results show an organic quality to the generated textures, but they did not present any results simulating real-life patterns. Besides, the presented work did not seem to allow for dynamic changes in the shape be incorporated into cell's programs.

Fowler *et al.* approached the modelling of seashells by discretizing the growing edge of a parametric model of a shell into polylines. Each segment of the polyline is treated as a cell for the one-dimensional Reaction-Diffusion simulation. The geometric and visual attributes of shells lend themselves to integration since both shape and texture can be unequivocally expressed as a function of time. Their exceptional visual results suggest that the use of an integrated approach is in some cases not only useful but imperative.

Our overall goal and inspiration is to match for mammalian bodies and coat patterns the level of achievement reached by Fowler *et al*. There will be many critical differences in the methods used. First, for reasons elaborated in details elsewhere, we do not believe that mammalian coat patterns are well modelled by Reaction-Diffusion methods, and we will use a different model, called *Clonal Mosaic*, defined in Chapter 3. Second, the process obviously has to take place at least in two dimensions corresponding to the skin surface. And finally, since in mammals the

pattern is established in the fetal stage, and undergoes changes due to body growth both before and after birth, we will have to integrate pattern formation and growth in an effective way.

1.3 Terminology

In this section we present and define basic concepts used in this thesis.

1.3.1 Pattern

Depending on the context the word pattern has many different interpretations. The biology community seems to use the word pattern without defining it [stev74]. The implicit meaning generally brings to mind some kind of repeated arrangement (regular or not) and the term is often defined by examples. We can usually distinguish between visual (e.g. tiger stripes) and structural (e.g. *Drosophila* segmentation) patterns.

The Oxford English Dictionary [simp89] has 13 entries concerning the substantive "pattern". The one we consider most appropriate in the context of our work states that a pattern is "…a composition of parts applied to a marking of natural origin." It is outside the scope of this work to give a definite and general definition for pattern. Nevertheless we need a "working" definition. Therefore in our context a pattern is *a 2D array of values, possibly in a regular domain such as a square or a rectangle*. This definition restricts our interest to visual patterns only, excluding structural ones. The important aspect of the above definition is that we have to be

able to visualize the pattern. This seems obvious but a pattern of concentrations for example, in a chemical system, can only be visualized if we map the numbers which express concentrations to some visually-perceived attribute, such as color.

When is a Pattern a Texture?

The word *texture* certainly has many interpretations in the graphics community. We will use the word texture in the sense of *a pattern applied to the surface of an object*. Intuitively, we can think of texture as visual information which gives us clues about the nature of the object, usually expressed at the object's surface. The difference between a pattern and a texture is that a texture involves the attachment of the pattern to the surface of an object.

1.3.2 Shape

The first of 17 possible definitions for shape in the Oxford English Dictionary reads: *"…that quality of a material object (or geometrical figure) which depends on constant relations of position and proportionate distance among all the points composing its outline or its external surface."*

This definition probably matches our first intuition about shape. For practical applications, however, we can not deal with an infinite number of points, since that is what the definition demands when it says "…*among all the points*". Therefore we propose to use for shape the same definition as above but using the expression "…*among a finite, possibly large, number of primitives*" instead. This new definition conveys the idea that a shape is formed by a juxtaposition or union of a finite

number of primitives. The notion of primitive is left open: they can be any geometric entities such as spheres, polygons and even points.

There still remains the definition of a complex shape. We will define a complex shape in relative terms, that is, shape A is more complex than shape B if A *is expressed by using more primitives* than B. This definition assumes already that we have a "fair" and economical representation, using the chosen primitives, for the shapes that we are comparing.

We are able now to precise which has a more complex shape, a sphere or a horse. If we use a "sphere-like" primitive we need only one to represent our sphere as opposed to many spheres to represent the shape of the horse[5]. Different primitives will give different results. If we had a "horse-like" primitive we could represent our horse with only one primitive and would have a hard time trying to represent the sphere with an appropriate number of "horse-like" primitives. We have a trade-off between the number of degrees of freedom (dof) of a given primitive versus the necessary number of primitives. Generally, to represent a given complex shape we either have a relatively large number of low dof primitives, or a smaller number of higher dof primitives.

1.4 Organization of the Thesis

The rest of this thesis is organized into the following chapters:

- **Chapter 2 - Models for Mammalian Coat Pattern Formation**

[5] In this example we would have to consider also how precise we need the horse approximation. This is a more complex problem and a definite answer is outside the scope of this thesis.

Presents background material on the problem of pattern formation from a biological and mathematical points of view. The main model discussed is Reaction-Diffusion. We also present previous work in computer graphics related to pattern formation.

- **Chapter 3 - The Clonal Mosaic Model**

  Introduces one of the main contributions of this thesis, our model for the synthesis of mammalian coat patterns called Clonal Mosaic. We present a detailed description of the model together with results and validation of the produced patterns. The domain of patterns produced includes the giraffe and the big cats. A condensed version of this chapter has been published at the Graphics Interface'98 conference [walt98].

- **Chapter 4 - Models for Shape**

  This chapter reviews, in our context, the main approaches to describe and represent shape in computer graphics. We adapted and extended Koenderink's classification [koen90] for shape description to existing object-modelling approaches in computer graphics.

- **Chapter 5 - Applying Growth Information to Polygonal Models of Animals**

  Introduces a technique to modify the shape of an object in a controlled manner, suitable for our purposes. Particularly, we explain how the technique can be used to simulate growth of animal shapes. We present results of simulations applied to domestic animals such as horses and cows. A version of this chapter

has been published at the Eurographics'97 conference [walt97].

- **Chapter 6 - Integration**

  Based mainly on the material presented before in Chapters 3 and 5, this chapter explains how we put together, in an integrated way, the simulation of the Clonal Mosaic patterns on a shape-changing geometry. We illustrate our results with the case of a giraffe.

- **Chapter 7 - Architecture of the System**

  Presents a complete view of the different computational tools developed for this thesis and documents how to use them.

- **Chapter 8 - Conclusions**

  Lists the main contributions of this thesis, summarizes what we have learned, and presents some avenues of future research on the topics addressed in this work.

Chapter 2

Models for Mammalian Coat Pattern Formation

This chapter provides background information on pattern formation models in the context of mammalian fur patterns. We first present a description of the biological process responsible for fur colouring in mammals, followed by an overview of the general pattern formation problem. We then present the main mammalian coat pattern formation models studied in biology and mathematics and how some of these approaches have been used in computer graphics. We conclude the chapter with a case study of pattern formation for the giraffe.

2.1 Introduction

The genesis of a fully organized and complex multicellular being from an initial small number of undifferentiated cells is still one of the largest unknowns in devel-

opmental biology. Pattern formation models try to explain how initially unspecialized cells differentiate and organize into higher complex organisms [held92]. In a broad sense, the more general problem of pattern formation can be broken down into two main categories, structural and visual. The arrangements of petals in a flower, for instance, is a structural pattern formation problem. We restrict our description to visual pattern formation problems, such as the distribution of spots on a cheetah.

Despite research efforts so far there is no definite answer to the problem of mammalian coat pattern formation. A successful model will be the one that is able to present a plausible explanation for the class of models that it is trying to generate, and is supported by experimental evidence. We believe that the Clonal Mosaic model, presented later in Chapter 3, is a strong candidate for explaining a large class of mammalian coat patterns.

2.2 Mammalian Coat Pattern Formation

In this section we summarize the biological process responsible for fur colouring in mammals. The coloured pattern seen in many mammals is expressed in the fur constituting the coat of the animal. The skin of mammals is made of two layers, a superficial layer called the *epidermis,* and an inner layer called the *dermis*.
The fur formation process starts first by an aggregation of cells in the basal layer of the epidermis. This aggregation triggers the formation of a *papilla* which starts an *invagination* of the basal layer of the epidermis called a *follicle*. The hair is formed by division of stem cells in the *hair bulb* at the base of the follicle [sear68].

Pigments giving the hair its colour are incorporated into the hair by *melano-*

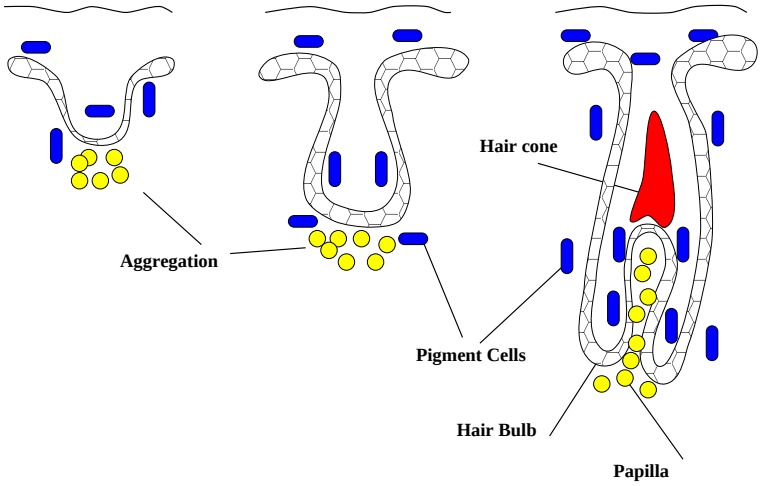


Figure 2.1: Process of hair formation

cytes, another type of cell living in the epidermis specialized in the production of *melanin* [prot92]. The melanocytes derive from melanoblasts that have migrated during embryonic development from the *neural crest* to their final position in the epidermis as part of a complex of cells called an *epidermal melanocyte unit* [gilb94]. Melanins are polymers synthesized from *thyrosine* (an amino acid) and exist in two types for mammals: *eumelanin*, with colour ranging from brown to black; and *phaeomelanin* with colour ranging from pale yellow to red. Basically, the colour of the hair is determined by the amount and nature of the produced melanin. The melanocytes are capable of synthesizing either one of the two types of melanin and gradations between the full yellow and full black. Which one is produced directly depends on the amount of a protein produced in epidermal cells called *Agouti*. This protein interferes with the binding of an hormone called Melanocyte Stimulating Hormone (MSH) to receptors in the melanocytes. Higher levels of the *Agouti* protein increase the binding between MSH and its receptors in the melanocytes increasing the pro-

duction of black pigment. Therefore, the level of *Agouti* protein present in the hair follicle environment directly affects which pigment is produced [jack91].

Direct observation of mammals shows that their characteristic coat pattern is already established at birth, and after this is modified only due to differential growth of the body. For instance, the spots on an adult giraffe can easily be recognized from the spots of the same individual at birth. This characteristic has been used to recognize individuals in the wild. Bertram [bert78], for example, mentions that he "…used the arrangement of spots on the head to recognize different individuals" among a population of leopards. He also used spot arrangement to distinguish between different individuals among cheetahs. Sometimes the spots can fade or disappear due to a change in "colour map". Lions have spots at birth, which quickly fade and are not readily visible in the adult (Figure 2.2). For some animals the expression of the pattern may occur later on in the development of the animal. Cheetah's cubs for example, "…have a peculiar natal coat which is light gray and woolly on the cub's back and black on its belly" [seid91]. Only by four months of age they achieve the peculiar spotted pattern.

Therefore one can distinguish two phases in the creation of the pattern. The first phase happens in the fetal stage, where both growth and establishment of the pattern take place, and the second phase, both before and after birth, where only growth affects the pattern. Pattern formation most likely starts as soon as the melanocytes have finished their migration from the neural crest. Phase one – pattern formation plus growth – ends at most by the end of gestation, but there are reasons to believe that it might be sooner. In section 2.5 we present a detailed description of

Figure 2.2: Baby lion

this process for giraffes.

2.3 Pattern Formation Models in Biology

Although several models for mammalian pattern formation have been proposed, either in biology or mathematics, the actual mechanism responsible for the patterns is still an open question in biology for most patterns. Moreover, the literature lacks a good taxonomy for existing models, in specific for mammalian coat patterns. We classify the existing approaches into 3 classes: Reaction-Diffusion, mechanochemical, and cellular automata.

2.3.1 Reaction Diffusion

In 1952 Turing showed [turi52] that the chemical interaction of two substances, under some conditions, could produce stable spatial patterns for the concentration of the substances involved. He called the system *Reaction-Diffusion* (RD) and coined the name *morphogen* to characterize the particular function of these substances in the system, namely "shape generators" or "form generators".

The behaviour over time (i.e., variation of morphogen concentrations) of an RD system is expressed by a system of non-linear partial differential equations (PDEs). This system has terms governing the diffusion of the morphogens and governing the reaction. Diffusion refers to the spread of morphogens over the substratum and reaction refers to the production and consumption of morphogens.

For a process to be considered as an RD process, there are some necessary conditions. First, an RD system involves interactions between at least two substances which must have different diffusion rates. Second, of these two substances, one must have the property of *self-enhancement* or *autocatalysis*, that is, to be able to increase its speed of production. The other substance either inhibits or helps the production of the self-enhancer. The long range interactions of these substances is the RD process that will evolve into a stable state of morphogen concentrations.

Example of a Reaction Diffusion System

As an example of an RD system, we present here the equations defining the model referred to as the *activator-substrate* model. This system was proposed by Meinhardt and Klinger [mein87a, mein87b] and used by Fowler *et al.* [fowl92] to generate some

remarkable seashell pigmentation patterns. It is a one-dimensional RD system and the activator a reacts with the substrate s. The equations are:

$$\frac{\partial a}{\partial t} = \rho s \left(\frac{a^2}{1 + \kappa a^2} + \rho_o \right) - \mu a + D_a \frac{\partial^2 a}{\partial x^2}$$

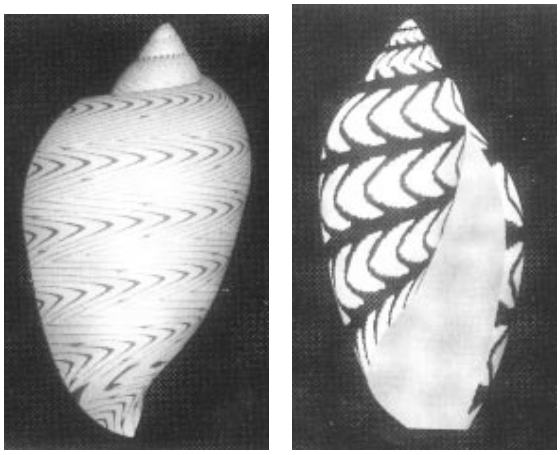
$$\frac{\partial s}{\partial t} = \sigma - \rho s \left(\frac{a^2}{1 + \kappa a^2} + \rho_o \right) - \nu s + D_s \frac{\partial^2 s}{\partial x^2}$$

Activator and substrate diffuse along the x-axis with rates D_a and D_s. The coefficients μ and ν express the decay rate of each element. The substrate is produced at a constant rate σ. The production of the activator is autocatalytic and proportional to the concentration of the substratum and to a^2 for small activator concentrations. The production of the activator decreases the growth of the substratum proportionally to ρ. For large concentrations of the activator the parameter κ controls the level of saturation of the autocatalytic process. Finally ρ_o is used to initiate the autocatalytic process since it represents a small base production of the activator. In Figure 2.3 we reproduce some of the patterns this model generates.

Main Reaction Diffusion Models

There is no universal RD system capable of generating all desired patterns of a given class, such as mammalian coat patterns. The specifics of the reaction part in the PDE system plays a crucial role to define the possible patterns the system can generate.

Turing's initial ideas were extended and elaborated into many different models. Murray [murr89, murr81a, murr81b], for example, claims that RD is a strong candidate for explaining animal coat patterns. He suggests that an RD process would



(a) The pattern on *Amoria undulata*.

(b) The pattern on *Volutoconus bednalli*.

Figure 2.3: Example of Reaction-Diffusion patterns on virtual shells (from [fowl92]).

take place during the first days of embryo development. The coat pattern that we see on the animal is a reflection of the spatial pattern of morphogen concentrations from an RD process. The morphogens would activate the melanocytes (specialized pigment cells) [prot92] to produce melanin. Since for mammals there are two types of melanin [hera76, fox60], the concentrations of morphogens would function as a switch to activate or not a given type of melanin. Murray's work is particularly known by his detailed exploration of how different geometries and scales affect the patterns produced by RD mechanisms. He simulated RD systems over a canonical asymmetric 2D domain representing a generic "stretched" animal skin. He also allowed the relative size of the domain to be controlled by some pre-defined parameter. The different results produced by varying the parameters lead him to claim that the embryo size, at the embryonic stage when the pattern formation mechanism is taking place, plays a decisive role to decide whether an animal is self-coloured or

patterned. Also, if indeed RD is responsible for mammalian coat patterns, Murray has a plausible explanation for why it is "…not possible to have a striped animal with a spotted tail" [murr81a]. This conclusion was derived from an RD simulation over a cone-shaped domain – a geometry equivalent to a 2D stretched tail. The RD process thus simulated fails to produce a transition from stripes to spots but successfully produces a transition from spots to stripes.

The work by Bard [bard81] addressed some questions left open by other researchers in early RD work. He presented possible explanations – within an RD framework – for more complex patterns such as the rosette and the markings of different giraffe species. These patterns, he suggested, could be explained by one of (or a combination of) two mechanisms: *cascade RD processes,* where a sequence of RD processes would explain the increasing complexity of a pattern, and a *threshold interpretation mechanism* for the concentration of morphogens. Such a mechanism should be able to "read" different levels of morphogens' concentration and produce different pigments accordingly. In the simplest case envisioned by Murray, a given threshold of morphogen concentration is used by the melanocytes to activate or not a given type of melanin (an "all or none" mechanism); a more subtle mechanism would map different levels of concentrations to different pigment production thus producing two or more hair colors. Bard has also suggested, from observation of real patterns, the possibility of having different diffusion rates for different parts of the domain, a suggestion explored later in computer graphics by Witkin and Kass [witk91].

Gierer and Meinhardt have proposed a number of RD models to explain a

wide range of patterns, both visual and structural ones. In Meinhardt's book [mein82] we can find a good overview of many of their early generic models. The specific problem of mammalian coat patterns is not directly addressed in any of their models, but indirectly, for example, through models that can generate stripes and therefore could explain striped animals such as zebras. Many of their models included more than two substances to account for more complex regulatory processes.

Discussion

The theoretical work on RD systems is far more advanced than its experimental counterpart. Only recently has a simple, real chemical system been shown to produce patterns predicted by Turing 45 years ago [leng91, ouya91]. Whether or not such chemical systems can be reproduced on biological tissues is still open for discussion. Many biologists argue that biological tissues lack the capability of long range diffusion needed in RD systems. Hammer [hamm98], however, argues that the diffusion mechanism in RD systems could be re-interpreted as direct cell-to-cell signaling. The discrete nature of the numerical computation of the Laplacian is interpreted as discrete cells exchanging signaling molecules in an array.

Most of the active research on RD is concerned with pure mathematical analysis or with new models that try to explain some as yet unknown or interesting pattern formation process. As an example of the former we mention the work of Lyons and Harrison establishing necessary and sufficient conditions for a Turing-like RD model to produce stripes or spots [lyon92]; as an example of the latter we mention a new RD mechanism that has been suggested by Kondo and Asai as a viable option to

explain dynamic striping formation on the skin of a marine angelfish *Pomacanthus* [kond95]. In spite of being able to predict the dynamic striping pattern, Kondos's model still left some questions unanswered, such as the fact that the spaces between stripes on the fish skin are wider than the patterns the model predicts [mein95b].

Perhaps the main shortcoming related to the RD theory is the fact that so far experimental biologists have not found or isolated a real morphogen.

2.3.2 Mechanochemical

A second line of reasoning proposes that many patterns, particularly structural ones, are better explained by mechanical forces acting on cells. The basis for these *mechanochemical* models was established by Odell [odel81] and extended by many researchers [oste83a, weli90]. The basic idea behind these models is that forces, usually considered to be chemically induced, play a decisive role to define shape and pattern. Typical examples include a model for mesenchymal morphogenesis [oste83b].

Recently, a new mechanochemical model was introduced by Savic [savi95] to explain pattern formation in animal coats. He suggests that coat patterns are an expression of a pre-pattern of polarized and unpolarized domains of epithelial cells. The cell's polarization process is local and regulated through a long range negative feedback mechanism due to elastic forces. The model, however, fails to explain the nature of the polarization forces and how the polarization process is initiated.

An interesting possibility is to combine more than one approach into a single model. The RD and mechanochemical approaches have been explored together to

account for patterns where some level of interaction between two independent systems is expected to share responsibility for the pattern. Examples of these include the arrangement of scales on lizards and the location of feather follicles in the common coot [shaw90].

2.3.3 Cellular Automata

A few models for mammalian pattern formation have been proposed using the computational mechanism called *cellular automata* (CA) [toff87]. A CA pattern is expressed as a collection of cells[1] arranged in a particular configuration. An initial set of cells is defined with an initial state. The transition from one state to another is governed by transition rules which take into account the current cell state and the state of its neighbors. A good review article on CA and applications to generic pattern formation problems was written by Wolfram [wolf84]. Young [youn84] introduced a CA version of a Reaction-Diffusion system where the intercellular interaction is more localized than in Turing's original model [turi52]. Cocho [coch87a, coch87b] presented a pattern formation framework where the multiplication of cells is modeled assuming an initial small number of "clonal" cells; these advance in time to a more complex arrangement according to the automaton rules. A clonal cell is a single cell which generates a visible element in the final pattern, such as a spot in a spotted pattern or a patch in the giraffe pattern. The idea of clonal cells is explained further in the context of our work in Section 3.2.

[1] In the context of cellular automata the name cells is not necessarily associated with a biological cell.

2.4 Pattern Formation Models in Computer Graphics

Computer graphics is a powerful visualization tool for biological research. Data generated by a particular biological model can be visualized using computer graphics techniques; the images thus generated are a powerful argument either against or in favour of the model's validity [prus93]. On the other hand, computer graphics can benefit from biological models if we consider their potential to deliver more realistic simulations. This fact has been a strong motivation behind the increasing use of biology-inspired models within the computer graphics community.

In the context of visual pattern formation, only recently has computer graphics started using biological models as the underlying models which drive procedural textures. The work so far has concentrated on using RD as the underlying model. The next sections review and summarize these approaches. Within the context of mammalian coat patterns we review here the approaches by Turk [turk91], and Witkin and Kass [witk91]; within the context of seashell pigmentation we review the work by Fowler, Meinhardt and Prusinkiewicz [fowl92]. We finish this section with a description of cellular cell systems by Fracchia and others [frac90].

2.4.1 Turk

The basic Reaction-Diffusion systems studied in biology can generate a set of interesting but visually-limited patterns (simple stripes, simple spots, etc.). The generation of more complex patterns (e.g., rosette) is not usually addressed in the Reaction-

Diffusion literature. A possible explanation for the more complex patterns was put forward by Bard [bard81]. He suggested that a more complex pattern could be generated by a *cascade process*, in which an RD system is simulated having as a starting point the result of another RD simulation. Despite Bard's suggestion there has been no further biological research to explain exactly how two RD systems would interact to simulate a cascade process.

Turk [turk91] used this suggestion as the starting point for simulating cascade processes. One example might help visualize the idea. The pattern of typical large and small spots found on cheetahs can be achieved by a cascade process which generates first the big spots; then, keeping this result, the system is simulated again with new values for the parameters in such a way as to synthesize smaller spots. Turk suggests that the image achieved by this cascade process is more natural than the image we would have if the images of two simulations would be superimposed.

Variations on the way that the two or more RD processes interact can lead to many different patterns. The simulation of two different Reaction-Diffusion systems (e.g., spot formation and stripe formation) together into one, for example, can generate a web-like pattern similar to the reticulated pattern found on giraffes.

Turk also introduced the idea of simulating the RD process on the surface of the object being textured, an important contribution which addresses many of the texture mapping related problems. The two-dimensional result patterns presented, in some cases, a good visual similarity with real patterns.

2.4.2 Witkin and Kass

The main contribution of Witkin and Kass's work [witk91] was to extend the basic idea of RD by incorporating anisotropy into an RD system, a suggestion made 10 years earlier by Bard [bard81]. The anisotropy refers to the possibility of simulation of the RD system in arbitrary directions with different diffusion rates for the x and y directions. In a classic RD model, the same diffusion rate is used for both directions.

The anisotropy information – diffusion rates and orientations – for each point in the domain of the simulation is defined via a *diffusion map*. The possibility of anisotropic RD patterns, indirectly specified through diffusion maps, has certainly extended the range of possible RD patterns. It should be noted, however, that diffusion maps have to be defined by the user, thus increasing the complexity of generating the patterns. Actually, the definition of a diffusion map is already in a sense a definition of the pattern itself and consequently diffusion maps are just transferring the problem to the user. Their giraffe pattern, for instance, was computed having as initial condition a diamond-like grid, already similar to a giraffe pattern. An integrated solution would avoid the need for much user interaction and would use geometric information about the model being textured as the basis for defining the diffusion map.

Another contribution of their work was the possibility of generating customized RD textures, that is, an RD texture that looks distorted when mapped on a plane but non-distorted when mapped on the final object. To compute this distorted RD texture their approach used the surface Jacobian information to drive the RD process. However, it should be noted that a purely geometric correction does not nec-

essarily guarantees a valid biological pattern.

2.4.3 Fowler, Meinhardt and Prusinkiewicz

The work by Fowler, Meinhardt and Prusinkiewicz [fowl92] addressed the modelling and pigmentation of seashells. The patterns are derived considering the chemical reaction of a morphogen with the substrate, that is, the growing edge of a shell. The patterns we see are a record through time of a one-dimensional RD system developing as the shell grows. Their main contribution was the integrated approach where the texture generation process is driven by the underlying geometric model of the seashells. This has set a dynamic aspect – the "growing" of a texture – to the Reaction-Diffusion patterns explored before in computer graphics. From a pattern formation perspective their work did not add any new features to basic RD systems presented before in the biology literature by Meinhardt [mein87a, mein87b]. A good overview of the Meinhardt's work on seashell pigmentation is given on a recent book [mein95a].

2.4.4 Three Dimensional Reaction-Diffusion

A trivial extension of simulating a Reaction-Diffusion in three dimensions was presented by Chambers and Rockwood [cham95]. In 3D the cells are organized into a regular cubic grid and the nearest six neighbors are used in the computation of the Laplacian. This 3D information is then rendered using standard 3D data visualization techniques such as marching cubes [lore87] and the solid space thus created can be used as 3D textures. The authors also proposed the creation of a 3D mesh using

the concentrations of a 2D simulation as a height-field and fitting a Hermite surface through the points. This enables the visualization of a 2D simulation as a 3D surface and some results resemble wind-formed sand dunes.

2.4.5 Cell Systems

Fracchia, Prusinkiewicz and Boer introduced a formal mechanism to simulate cellular systems in two and three dimensions [frac90]. The basic entity in the system is a cell represented as a region. The set of all cells is represented as a map. The synthesis of a given pattern is controlled through a map-rewriting system with specific rules acting on the cells, controlling their division and other properties. They were able to successfully simulate some biological phenomena such as the development of *Microsorium linguaeforme* in 2D and *Patella vulgata* in 3D. These original ideas were extended into context-sensitive cell systems [frac95] and generalized context-sensitive cell systems [lant95], allowing for simulation of more general biological processes. In spite of its power for simulating biological processes, the main drawback of these formal systems is the definition of the rewriting system, usually complex for modelling more sophisticated phenomena.

2.4.6 Discussion

The basic problem when computing natural textures using the above mentioned techniques is that the parameter space is large and therefore achieving a desired pattern is non-trivial. There is no single comprehensive model that can handle all desired patterns but instead there are different RD systems which produce the different pat-

terns. Besides, many interesting patterns are only possible through the use of factors external to the original models (e.g., diffusion maps) thus increasing the model's complexity. Nevertheless, the work reviewed here has introduced tools to generate interesting patterns which in some cases closely resemble natural ones. The work on seashells pigmentation, for example, leaves almost no question that RD is a strong candidate to explain the pigmentation patterns. This possibility is mainly confirmed by the remarkable visual similarity between real seashells and the synthetic ones. On the other hand, the approaches by Turk and Witkin & Kass are not as convincing since their work lacked the same visual quality, suggesting that perhaps other mechanisms are better candidates to explain coat patterns.

2.5 A Case Study: Pattern Formation for the Giraffe

It seems reasonable to suppose that the coat pattern of mammals is laid down much before the hairs exist. This is mainly supported empirically. Dagg [dagg76] mentions that "the coloring of the giraffe is of course a function of the hairs that cover the skin". The question is exactly when is the pattern established. Bard had to estimate this time for zebras [bard77]. He presents an hypothesis to explain the increasing number of stripes on different zebra species. He suggests that, for the three zebra species, a single mechanism could be responsible for laying down stripes spaced around $0.4mm$ from each other. The difference between species in the final number of stripes can be explained by a different timing for the striping process to take place. For a zebra with fewer stripes the process happens earlier in embryonic life (less space for stripes) and on the other hand, for species with more stripes, the pro-

cess happens later, but the stripes are always at $0.4mm$ apart. This was an insightful observation, establishing a plausible explanation for different patterns dependent upon the interaction between the pattern formation process and the embryo changing shape.

Bard had access to horse embryos at various ages, which were assumed to be similar to zebra embryos. Starting with the assumption that the striping mechanism was the same for all three different zebra species and knowing the number of stripes for each species, he was able to pinpoint the exact time in embryonic life that the pattern formation process should take place in order for a given species to display a given number of stripes. This window of time in embryonic life is between three and five weeks, depending on which species is considered (for a gestation period of 12 months).

Bard's conclusions have two direct implications to our model presented in the next chapter. First, the patterning mechanism seems to affect not the distribution of pigment cells but rather their differentiation. By analogy with chick and amphibian development, the neural crest cells, which originate the pigment forming cells, have finished their migration from the neural crest to reach all other body parts before five weeks of embryonic life, earlier than Bard's proposed timing for pattern formation. Second, and more importantly, by the time the pattern formation process takes place the embryo has already a recognizable shape, and this shape does affect the patterning seem in an adult animal. In Bard's words "this period is between tail-bud extension and growth after the main anatomical features have been formed."

How are these results related with the giraffe? We want to estimate a plau-

sible date for the start of pattern formation for giraffes. The earliest register for a "visible" giraffe pattern is 100 days from conception. This estimate was made by Ackermann [acke1b] and cited by Murray [murr81b][2]. However, the pattern was indicated through blood vessels and not through the hairs. This gives us an upper limit. Assuming that the pattern formation process only starts after the melanoblasts have stopped traveling over the body, this gives us around the 4th or 5th week, or between 28-35 days as the earliest possible date for the pattern formation process to start. For zebras the average time for pattern to happen is around 28 days or 7.8% of the total gestation time of 12 months according to Bard [bard77]. If we use the same figure for the giraffe we would get $457 * 0.078 \simeq 36$ days, since the total gestation time of the giraffe is 457 days according to a survey by Skinner and Hall-Martin [skin75] based on 48 reports of giraffe pregnancies. This figure is in agreement with the upper and lower bounds established above. Therefore we propose the 36th day of gestation time as the time for the onset of the pattern formation process in giraffes.

A useful fact is that during the whole pattern formation process one can safely assume a linear growth (that is the length of some part of the body is a linear function of time). Figure 2.4 shows the plot obtained for the length of the giraffe fetus from measurements presented by Owen [owen49] and by Beddard [bedd06] reproduced in Table 2.1.

[2] I did not have access to Ackerman's thesis.

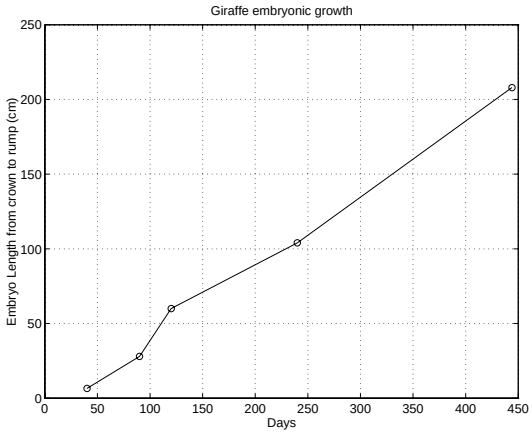


Figure 2.4: Fetal length for giraffes

sex	days old	length (cm)
?	35-45	6.5
male	90	28
female	120	60
female	240	104
male	444	208

Table 2.1: Fetal Length for Giraffes

2.6 Summary

This chapter presented an overview of main pattern formation models in the context of mammalian coat pattern formation. We also explained the biological basis of hair formation and reviewed previous work in computer graphics using biology inspired models to produce animal coat patterns. As a case study we showed how we can indirectly infer the time for the onset of pattern formation process for the giraffe.

Chapter 3

The Clonal Mosaic Model

K. Richard: Lions make leopards tame.

Norfolk: Yea, but not change his spots.

Shakespeare: *Richard II*, I, i.

This chapter introduces a new pattern formation model that addresses the generation of visual patterns; specifically, how coat patterns found in many species of mammals are formed. The class of animals to which the model is applied is composed by the patterned animals in the *Felidae* family and the giraffe. These patterns range from the spots of the giraffe to the stripes on a tiger and the rosettes on a leopard. From a biological perspective, the model has a strong appeal in light of recent experimental evidence on pigment cells and other pigment-related mechanisms. The attractiveness of the model for computer graphics is that it can generate a large number of animal patterns with a relatively small number of parameters, and this can be done on surfaces of arbitrary shape, as we will show in Chapter 6.

First we present an overview and motivation for a new pattern formation model followed by a detailed description of the biological basis of the model. We then describe the implementation of the model and conclude by presenting some results.

3.1 Overview and Motivation

The Clonal Mosaic theory for mammalian coat pattern formation [reim95, reim96] proposes that the typical yellow-black stripped and spotted patterns occurring in several species of mammals, reflect a spatial arrangement — a mosaic — of epithelial cells which derive from a single progenitor, i.e., they are clones. Hence we use the name Clonal Mosaic (CM). Different hair colors result from different types of underlying cells. Different spatial arrangements of cells are produced basically by controlling splitting rates of cells and adhesion between them. The model takes into account biological experimental data such as the the migration of and interactions among cells [goel78] [roge78], particularly epithelial cells [gord78].

The main strengths of the CM model over previous animal coat models are its conceptual simplicity and power for generating all desired patterns without extra external controlling factors. The model is also appealing for procedural texture synthesis in computer graphics (see Section 1.2.1) since it can provide a large number of 2D patterns with a relatively small number of parameters. These patterns can be used inside a traditional texture mapping framework. A final strong appeal of the CM model over other mammalian coat models is its reasonably straightforward extension for simulation over arbitrary surfaces. Thus, it is possible to generate patterns directly on the object's surface, without the mapping step. In an integrated

framework using the CM model as the texture generator, the geometry of the object can then play an important role in the patterns generated, increasing the realism of animals synthesized this way.

3.2 The Clonal Mosaic Model

The basic idea of the Clonal Mosaic model[1] is that groups of contiguous cells in an organ (here the skin) are clones, that is, descendents of common ancestors [mint74]. Clonal Mosaicism has been demonstrated for most organs of the body. In the liver, for instance, the cells are not uniform in their expression of any of the catabolic enzymes. Neither however is the distribution of these enzyme activities random. Cells with similar patterns of enzyme activity form contiguous groups; these groups share a recent common progenitor – that is, they are clones. Applying this idea to fur formation we can suppose that cells in differently colored areas derive from different progenitors.

There are about 50 genes so far known to affect pigmentation [jack91] but only a small number of these affect patterning. The gene which affects the transition between black and yellow is called *agouti* gene, after the South American animal in which the characteristic yellow bands on the hairs were first noticed. The agouti gene acts by producing a protein which interferes with the binding of the Melanocyte Stimulating Hormone (MSH) to its receptor on the pigment cell. This is a threshold phenomenon, that is, the agouti protein must be present at sufficiently high levels to interfere with the binding of MSH to its receptor.

[1] The relevant facts about fur formation and pigmentation were presented before in Section 2.2.

The working hypothesis is that, during embryonic development, cells in the epidermis determine what level of agouti protein will be expressed in their progeny and transfered to the fur. This working hypothesis leads naturally to the idea that the shape of a pattern element will be the shape of a clone; the shape of a clone will be determined by the deformation induced by non-uniform stresses on the cells during development [gord78]. The stresses on the epidermis induced by the expansion of the embryo are locally uniform, so that the explanation of non-uniform stresses must lie in non-uniform local expansion of the cell sheet, such as might be caused by non-uniform mitotic rates. In order to generate different patterns, we propose that the rates of cell division differ.

If some cells are dividing faster than others, then we might expect some evidence of this in the adult animal. In fact, in cats, the darker areas of skin are thicker and have denser hair than the lighter areas. Also, the patterns for different species are obtained by different rates of cell parameters such as division and motility. These parameters are known to change through development, and within one animal, specifically from front to tail and also from dorsum to ventrum. Thus we should observe a gradient of patterns on any individual animal, reflecting the different conditions prevailing at the time of maturation of the epidermis. In fact this is the case.

The model can also explain the more complex patterns. We believe that an intermediate expression of the agouti protein will account for the dark buff areas inside the rosettes of jaguars and leopards, as well as the interior of the body stripes of ocelots.

3.3 The Implementation

Whether the Clonal Mosaic hypothesis is correct is obviously a biological question. Our goal is to determine the characteristics of CM as a pattern generator, and to turn the model into a practical system to generate animal patterns for computer graphics purposes. If we can, on the way, contribute to the validation of CM from a biological point of view, so much the better.

The goal of the implementation at this stage is to produce a pattern expressed as a 2D image in a regular domain — a square with toroidal boundary conditions. The patterns produced by a given simulation can then be analyzed and used to evaluate the model in a feedback loop. Adjustments can be made regarding the parameters and/or specific strategies of implementation.

The current implementation provides a computational testbed and it is one possible algorithmic translation of the theoretical abstract model presented earlier. Our results from the implementation show that it is possible to obtain realistic looking patterns from various combinations of two parameters — mitotic rates and differential adhesion.

3.3.1 Cells and Groups of Cells

The number of biological cells necessary to represent a given pattern can be very large[2]. It would be computationally prohibitive to implement a model which would

[2] A rough estimate given by Bard [bard77] is that one cell's diameter at the time the pattern formation process happens for zebras is 2×10^{-3}cm. Considering that the estimated total surface area of a giraffe embryo at the time the pattern formation process happens is approximately $40cm^2$, the total number of cells covering the embryo (assuming also just one level deep) would be roughly 10^7.

represent each real biological cell. Therefore we defined a representation scheme where each cell in our implementation is actually a representation for a group of biological cells (Figure 3.1).

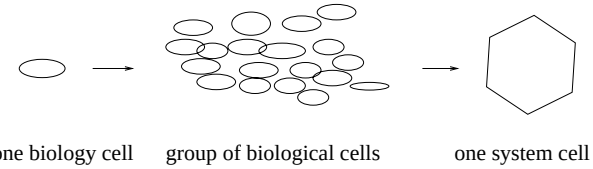


Figure 3.1: Representation of cells in the system

The assumption is that one cell in our system represents the behaviour of a group of biological cells. The issue is then to show that this assumption is plausible in both biological and mechanical terms. The only important biological trait that we have to assess is mitotic rates. Can a single system cell dividing represent many individual biological cells dividing? If the mitotic rates are context-insensitive then after many subdivisions, on average, we will have the same ratio of system cells to biological cells, that is, the assumption holds.

In terms of mechanical behaviour, if many individual cells are all subject to the same force then we can replace the set of cells by one single cell subject to a force which can be interpreted of as a resultant force. This trades off modelling of individual behaviour for computational efficiency. We might be missing phenonema with scales smaller than the size of a system cell, but we think the tradeoff is necessary. Throughout this description we will use the term *cells* to refer to a system cell.

3.3.2 General Description

The potential number of types of cells in the system is species dependent, but in practice quite limited. In this description we restricted the system to 3 types of cells since we can express all desired patterns with only 3 types. We call them *foreground* (F), *background* (B), and *intermediate* (M). The synthesis of a given pattern is done through two main procedures: *initialization* and *simulation.*

The initialization is responsible for distributing in the domain the initial set of cells and assigning a type to each one of them. Usually the domain is filled with many background cells and a few of other types. The types can be randomly or manually assigned by the user. The random assignment can be done all at once initially or progressively by the probability of B cells mutating into F or M cells. In a spotted pattern, for example, the foreground cells would correspond to the spots. The implementation assumes that the only forces acting on the cells result from cells maintaining their sizes under adhesion control [roge78]. The mobility of cells is a response to these forces. Cell size is maintained by introducing a repulsive force between cells that depends on the distance between them and on pre-defined adhesion values. Equilibrium is reached by a relaxation scheme. The idea of using repulsion on a surface to achieve a uniform spatial distribution of the points has been used before in biology [tane80] and computer graphics [turk91, witk94].

Cells are modeled as points for computing purposes. Points are usually the first choice to represent a biological structure as a cell [gord83]. Although points are a simple primitive, they have proved adequate enough for our purposes. To turn cells into a tessellation of the surface, we compute their Voronoi polygons. The Voronoi

polygon of a point in a given domain is the region of the domain that contains all the points closer to that particular point than any other [prep85]. The collection of all Voronoi polygons for the set of points is a *Voronoi diagram*. The adequacy of Voronoi polygons to represent epithelial cells was studied by Honda [hond78]. According to him, "…Voronoi polygons were shown to describe some cellular patterns (cultured monolayer cells, epithelial cells in tissue, etc.) with relatively small deviation values." Voronoi polygons were also used in a model for cell sorting presented by Sulsky [suls84].

3.3.3 Initialization

A given user-specified number of cells is randomly placed on a 2D square domain. Typical initial numbers are between 500 and 1000. The initial position of these cells is given by a random uniform distribution function presented in Numerical Recipes in C [pres92] (Figure 3.4(a)). Each cell is created with a given type that is related, in the theoretical model, to the level of expression of the agouti gene responsible for color.

The type of a cell defines its behaviour in the system; cell type can be specified by the user or randomly assigned by the system. The information attached to a given type is: color, division rate, probability for the cell to be of a particular type (only used when type is being determined by the system), probability for the cell to switch to another type (defined for every pair of types), and adhesion (also defined for every pair of types). A summary of these attributes is given in Table 3.1. The current implementation of the probability functions is context-independent, that is, not

Attribute	Meaning	Type
Color	RGB	3 floats [0-1]
Division Rate	Mean time between divisions Controls the absolute and relative numbers of cells of a given type	float
Initial Probability	Probability to be of this type	float (0-1)
Mutation Probability	Probability to switch to other type	float (0-1)
Adhesion	Drag between types Controls the tendency of cells to stay together	float (0-1)

Table 3.1: Attributes of a cell

dependent upon the state of the neighbors. The use of more complex and context-dependent probability functions is discussed in Section 3.3.8.

The cells undergo a relaxation process in order to achieve a regular and stable spatial configuration (Figure 3.4(b)). In order to achieve this configuration, each cell moves as far away from all its neighbours as possible. Only cells within a given *repulsive radius* are considered neighbors. The repulsive radius is determined proportionally to the average "ideal" area for each cell. For a given area A and m cells, the repulsive radius r is given as $r = w_r\sqrt{A/m}$, where w_r is a user defined scaling value. An adhesion parameter α controls the strength by which cells will repel each other in the relaxation scheme. This strength is proportional to $(1 - \alpha)$ and $\alpha = 1$ means no repulsion at all. With this parameter we can, for example, force any two types of cells to remain loosely or strongly connected. The force of repulsion models the growth of the cell and its tendency to occupy a given area at "maturity". The role of the adhesion factor is to express the fact that the ease of relative displacement of cells is a property of the pair of cells in contact. The individual displacements Dx

and Dy are computed proportionally to this repulsive scalar force and to the adhesion factor, as summarized in the pseudocode of Figure 3.2.

Figure 3.3 shows an example where we want to compute the new position of cell C which has 3 neighbors: P_1 and P_2, which are of a different type than C, (expressed in the figure by the hollow circles) and P_3, which is of the same type as C.

The initialization procedure stops once a stable configuration is achieved (that is, the maximum and minimum forces are relatively small). The system proceeds to the simulation phase, described in the next section. The exact timing for stopping the initialization step is not critical since the cells continue to relax in the simulation step. Figure 3.4(a) shows the created cells before relaxation; in (b) the cells underwent the initial relaxation; in (c) the foreground cells were manually selected, and in (d) the foreground cells were randomly selected by the system.

3.3.4 Simulation

The simulation phase controls the evolution over time of the initial distribution of cells into the final pattern. We model the simulation through an event priority queue implemented as a heap [corm90]. The two possible events are *relaxation* and *division*. Typically, the queue will have many evenly spaced relaxation events and some sparse division events. The rate of relaxation events is user controllable. For each time step, we have ρ relaxation events in the queue. The relationship between ρ and the division rate models the relationship between cell subdivision and cell motion. A large value for ρ allows time for the relaxation forces to balance over the domain,

We want to compute, for a given cell C at position (x_C, y_C), the new position (x'_C, y'_C)

1. For each neighbouring cell P_i at position (x_i, y_i)

   (a) Compute dx_i, dy_i, and d_i

   $$dx_i = x_C - x_i \quad dy_i = y_C - y_i \quad d_i = \sqrt{dx_i^2 + dy_i^2}$$

   (b) Compute f_i

   $$f_i = 1.0 - \frac{d_i}{r}$$

   (c) Compute displacements Dx_i and Dy_i

   $$Dx_i = \frac{dx_i}{d_i} f_i \left(1 - \alpha_{P_iC}\right) r$$

   $$Dy_i = \frac{dy_i}{d_i} f_i \left(1 - \alpha_{P_iC}\right) r$$

2. Compute new position for cell C according to

$$x'_C = x_C + \sum_{i=1}^{n} w_a \, Ox + (1 - w_a) \, w_d \, Dx_i$$

$$y'_C = y_C + \sum_{i=1}^{n} w_a \, Oy + (1 - w_a) \, w_d \, Dy_i$$

where

- r is the repulsive radius and f_i is a scalar which controls the strength of repulsion. We use a function for f_i such that $f_i = 1$ when the distance between cells d_i is 0 and $f_i \to 0$ when $d_i \to r$
- w_a controls the strength of anisotropy and Ox and Oy are the individual components of the displacement vector projected in the anisotropic direction (see Section 3.3.5)
- w_d is a weighting factor for the displacements
- n is the number of neighbors which fall inside the area defined by the repulsive radius
- Dx_i and Dy_i are the individual displacement due to the neighboring cells P_i
- α_{P_iC} are user-defined adhesion values, specific for the kind of cells involved.

Figure 3.2: Pseudocode for computing the new position of a cell

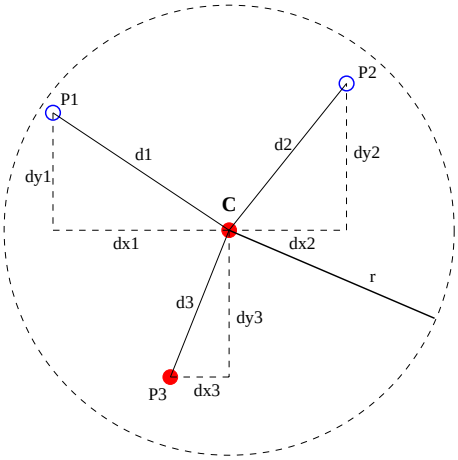
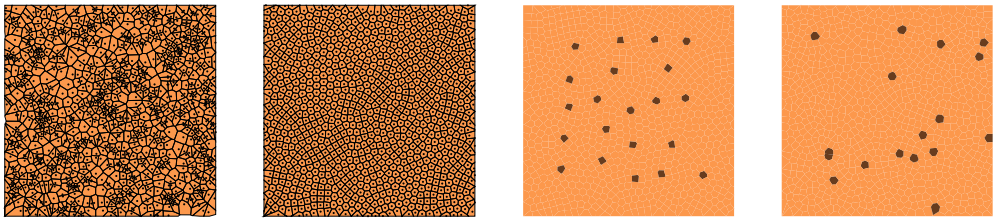


Figure 3.3: Computing the repulsive force



(a) Random initial distribution of cells

(b) After relaxation

(c) Foreground cells selected by "hand"

(d) Foreground cells randomly selected by the system

Figure 3.4: Initialization (1000 cells)

that is, the cells are closer to equilibrium.

During a division event one cell splits into two, i.e., they undergo mitosis. We can think of these as *parent* and *child* cells. The child cell can be of a different type than its parent, based on a probability matrix given by the user. The child cell inherits all the attributes corresponding to its type. The position of the new cell is uniformly random within a circle of diameter arbitrarily chosen to be 1% of the repulsive radius centered at the position of the parent cell.

The exact time for the division of a cell is given by a Poisson distribution with average equal to the rate of division for the cell. The Poisson distribution models small variations on the timing for mitosis, otherwise the cells would all split at the same time. There is no *a priori* end to the simulation. The simulation procedure keeps adding and handling events in the queue. The user can monitor the progress at any time by stopping the simulation and checking the pattern obtained up to that point in time. A potential problem is related with the necessary number of cells in order to start to "see" a pattern. As the number of cells increases the complexity of the system also increases, demanding adjustments to maintain the system efficiency. These adjustments are discussed later in Section 3.3.7.

3.3.5 Anisotropy

For some patterns we want to be able to set a preferred direction for the cells to move. This can be accomplished in three ways. First, when a given cell divides, the position of its child is not randomly uniform, but it moves in a preferred direction. This solution can effectively produce anisotropy only if the rate of division is high with

respect to the rate of relaxation. The second way would be using an anisotropic adhesion factor, which would happen with cells of asymmetrical shapes. The third way is the solution we adopted, which is to use an anisotropic vector A (defined as a direction and a magnitude) and an anisotropic weight w_a. When $w_a = 0$ it means no anisotropy and when $w_a = 1$ it means full anisotropy. A vector O is computed as the projection of the displacement vector D in the direction of A (formulae in Figure 3.2).

3.3.6 Summary of Parameters

Different patterns are computed using appropriate values for the available parameters. The three critical factors are the splitting rates of cells, the differential adhesion between the different types of cells and the anisotropy information.

1. Splitting rates

   We can control, for each cell type, its splitting rate, that is, how often the cell divides. The actual timing of a split is given by a Poisson distribution [pres92] whose mean equals the splitting rate.

2. Adhesion – α

   The adhesion parameter controls the strength by which cells will repel each other in the relaxation scheme. This strength is proportional to $(1-\alpha)$ and $\alpha = 1$ means no repulsion at all. With this parameter we can, for example, force groups of cells of the same type to remain loosely or strongly connected. We have to define an adhesion value for each pair of cell types (e.g., α_{FF}, α_{FB},...) and α is a value inside the closed interval $[0, 1]$.

3. Radius of repulsion – w_r

   As we mentioned before, we have an ideal radius of repulsion which is computed according to the ideal average area for each cell. By manipulating w_r we can enlarge or shrink this area. The net effect is that we are increasing or decreasing the number of cells which have to be taken into account when computing the relaxation forces.

4. Relaxation – ρ

   With this parameter we can control the flow of time allowing more or less time for the cells to relax. For each time step we have ρ relaxation events on the queue. The relationship between ρ and the splitting rate models the relationship between cell subdivision and cell motion. A large value for ρ allows time for the relaxation forces to balance over the domain.

5. Anisotropy – A and w_a

   A vector A is used to introduce anisotropy. The strength of this anisotropy is controlled by w_a such that $w_a = 0$ means no anisotropy and $w_a = 1$ means full anisotropy. When w_a is 1, then cells can only move in the direction of the anisotropy vector.

3.3.7 Efficiency Considerations

The most computationally intensive task in the implementation of the CM model is the relaxation scheme, since we need to find all the neighbors for a given cell. The worst case cost of this procedure is $O(n^2)$ where n is the number of cells.

To avoid this cost we implemented a dynamic rectangular grid of *buckets*, over the domain. The linear size of each bucket is the same as the repulsive radius. This scheme guarantees that we only have to check for neighbours within the eight buckets around the bucket of a given cell plus the bucket that contains the cell itself. Each of these buckets has a pointer to a linked list of pointers to the cells it contains. Since the number of cells grows exponentially with time we need to adjust the grid structure as the number of cells grow. This adjustability of the grid is necessary because the overall domain size is maintained artificially constant in the 2D domain. In a growing domain the bucket size would remain constant and the number of buckets would increase as the total area increases.

The grid information is updated (i.e., the number of buckets increase) every time the number of cells is 50% greater than for the previous grid. This guarantees a relatively efficient computation scheme. To give a rough idea of timing, the worst case (4300 cells with 78 simulation steps) among all computed patterns (Figure 3.6(c)) took 173 seconds to compute on an Origin 2000 SGI (a 195Mhz processor) and the average time for all patterns presented was 84 seconds.

Figure 3.5 shows a schematic example where we have 20 cells in a domain with 25 buckets. The area of influence of cell C is represented by the dotted circle around it. With this scheme we guarantee that any potential neighbor of C is a cell inside one of the 8 buckets around the bucket which contains cell C (shadowed area). Within the area defined by the 9 buckets only the cells which are at distance r or less from C will affect its the new position.

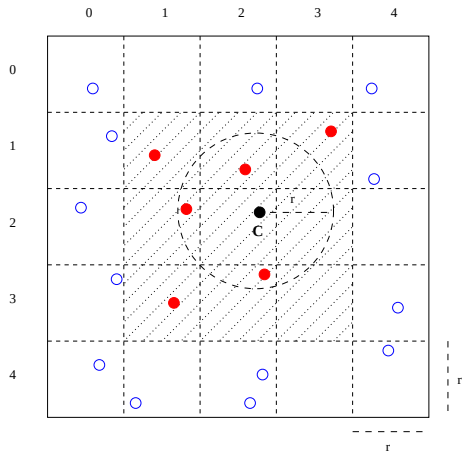


Figure 3.5: Division of the domain into a grid of 25 “buckets”

3.3.8 Discussion

Below we list and briefly comment a few features which could be explored on a future implementation of the Clonal Mosaic model. These features could increase the range of possible patterns.

1. Decreasing splitting rates

   Biologically, the mitotic rates decrease with time as the domain gets populated. The current implementation uses a constant function to model this feature. More sophisticated functions to vary the splitting rates with time could provide a more realistic simulation.

2. Cell death

   Although cell death is a biological fact we believe that modelling its effect would not significantly increase the range of possible patterns. If the rate of death of cells is low – a reasonable assumption specially at the fetal stage – the cells which are splitting faster are likely to compensate for the dead ones,

maintaining the same average number of cells.

3. Force model

   The current implementation models the behavior of cells such that the only forces acting on them result from a cell trying to maintain its size. A more sophisticated force model could include, for example, different cell response depending on the kind of forces involved. Slipping forces, as opposed to tangential forces, could play a more decisive role on the types of patterns achieved.

4. Context-dependent probability functions

   The implementation has deliberately been limited to context-free rules of behaviour for the cells. We wanted to explore first the range of patterns possible with this simple model (this parallels the evolution in power of L-systems [prus88] for plant simulation). There are legitimate reasons to extend the model to context-sensitive rules: in order to simulate any Reaction-Diffusion system, context, in the form of the concentration of the morphogens, is necessary. In real biological systems the behaviour of the cells is clearly affected by context, in the form of signaling chemicals sent across cells.

3.4 Results

In this section we present patterns generated with the CM model. In order to better assess the patterns visually, both computed and real patterns are presented. The real patterns were scanned from pictures of animals. The pattern we see on an adult animal is actually the result of two phases of the pattern formation process, the first

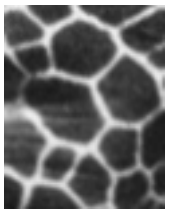
which took place some time during embryo development on a shape changing with time, and the second due to the growth of the body after birth. The patterns we show here prove that the model is capable of generating a planar 2D pattern which looks similar to a projection of a pattern which is actually defined on the curved surface of the animal's body. In Chapter 6 we show the results of the model directly simulated on the surface of the object.

3.4.1 Giraffe patterns

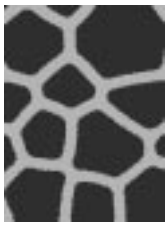
The main taxonomy for giraffes classifies them into one species with 9 subspecies. The differences in giraffe markings have been a key feature to identify subspecies, even though this criterion has been replaced by more objective ones such as skull measurements. Visually speaking, the two most distinctive patterns are from *Giraffa camelopardalis reticulata* shown in Figure 3.6, and from *G. c. tippelskirchi*, shown in Figure 3.7. Dagg [dagg76] describes the first: "...the large, smooth-edged liver-colored spots are placed closely together with only a fine network of light color dividing them". Dagg describes the second: "...the spots are usually splintered, forming all shapes of sharply differentiated leaf or stellate designs, although some approach *reticulata* in design and color". We can easily go from *reticulata* to *tippelskirchi* patterns in our model by decreasing the adhesion between F cells and increasing adhesion between B cells.

Figure 3.8 shows a sequence of images at different times and illustrates how the pattern evolves through time. The pattern obtained is visually similar to (*Giraffa camelopardalis reticulata*).

(a) Giraffe



(b) Real

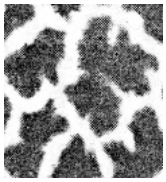


(c) Computed

Figure 3.6: *G. c. reticulata*

Param.	ρ	wr	time	w_d	mitosis F	mitosis B	α FF	α BB	number of cells	spot area
Value	18	2.6	78	0.066	10	120	0.9	0.2	B=965 F=3385	78

(a) Giraffe



(b) Real

(c) Computed

Figure 3.7: *G. c. tippelskirchi*

Param.	ρ	wr	time	w_d	mitosis F	mitosis B	α FF	α BB	number of cells	spot area
Value	18	0.6	70	0.066	10	150	0.2	0.9	B=979 F=1197	55

(a) time=0 (b) time=40 (c) time=50 (d) time=60 (e) time=70

Figure 3.8: Time lapse

Param.	ρ	wr	w_d	mitosis F	mitosis B	α FF	α BB	number of cells	spot area
Value	18	2.4	0.066	10	150	0.7	0.6	B=979 F=1197	55

3.4.2 Spotted patterns

Spotted patterns occur mainly in the cheetah and at the extremities (mainly legs, head, and tail) of other big cats such as the leopard and the jaguar. The cheetah usually presents two distinctive spot sizes, the jaguar and leopard present more regular sized spots. Figure 3.9 shows a real cheetah and Figure 3.10 shows the real and two computed spotted patterns. In Figure 3.10(b) the initial probability of F cells was slightly smaller than in (c).

For the jaguar and leopard, the spots "break apart" and a third color appears inside the spot. This type of pattern is known as a rosette. We simulate this type of pattern by allowing the foreground type of cells to switch with a small probability to the intermediate type. Figure 3.11 shows an example of this result. The extra parameters for this pattern, not mentioned in the table below the figure are as following: mitosis M = 10, $\alpha_{FB} = 0.5$, $\alpha_{BF} = 0.5$, $\alpha_{FM} = 0.8$, $\alpha_{MF} = 0.5$, and $\alpha_{MM} = 0.8$, probability of F cell switching to an M type of cell equal to 70%. Note that even though the individual rosettes are quite good, they are not close enough to each other to show the typical Voronoi diagram pattern. This can be improved with

Figure 3.9: Cheetah (*Acinonyx Jubatus*)

further adjustment of the mitosis parameters.

3.4.3 Anisotropic patterns

Since the tiger is a close relative of all other yellow-black type of big cats, by Occam's razor the mechanism generating stripes in the tiger ought to be of the same type as the mechanism generating spots or rosettes in the other big cats. Therefore we have to consider mechanisms that allow a cellular-based system to eventually produce stripes. We believe that the CM model can easily provide such a mechanism. This possibility has been discussed earlier in biological research by Bard [bard77] who said on the problem of stripe patterning that "While nothing is actually known about the process of pattern formation there are several obvious possibilities: the stripes might just appear or *spots might be generated on the dorsal line and be extended by an inductive wave moving ventrally*."

There has been no further research detailing how exactly the wave process

(a) Real (b) Computed (c) Computed

Figure 3.10: Spotted pattern

Param.	ρ	wr	time	mitosis F	mitosis B	α FF	α BB	number of cells
Value	18	2	15	8	60	0.8	0.5	B=1512 F=991
Value	18	2	15	8	60	0.8	0.2	B=1420 F=1177

(a) Real (b) Computed (c) Computed

Figure 3.11: Rosettes

Param.	ρ	wr	time	w_d	mitosis F	mitosis B	α FF	α BB	number of cells
Value	18	2	60	0.066	12	30	0.8	0.5	B=1866 F=507 M=436

mentioned by Bard would work and actually a wave mechanism is not really necessary for the CM model to produce stripes. It seems reasonable to suppose that the growth tensions present on the embryo at the time the pattern is laid down have to play an important role on the final patterns. In order to assess the behaviour of the CM model with respect to anisotropic forces, we have done simulations where the displacements are artificially modified according to an anisotropic vector. The net effect of this influence is controlled with an anisotropic weight. One result is shown

in Figure 3.12 where $w_a = 0.99$ and A is a direction of 30 degrees with a magnitude of 1. A full simulation of these effects demands the simulation of the pattern formation process on a surface which has the same topology as the embryo changing over time.

(a) Sumatran Tiger

(b) Computed Pattern

Figure 3.12: Anisotropic patterns

Param.	ρ	wr	time	w_d	mitosis F	mitosis B	α FF	α BB
Value	18	2.4	70	1	10	120	0.5	0.5

3.5 Assessing the patterns

In order to assess how close a given synthesized pattern is to a real one, we use a qualitative and a quantitative approach. In the commonly used qualitative approach, the generated patterns are visually compared with pictures of real animal patterns. Pictures provide an initial basis for comparison and have been widely used to validate much modelling of natural phenomena either in computer graphics (e.g., in the papers by Reed and Wyvill [reed94] and by Fowler *et al* [fowl92]) or in biology (e.g., in the papers by Meinhardt and Klinger [mein87a]). In the quantitative approach, visually important features of a real pattern are measured and used as a metric for

validating results.

The main reason the giraffe pattern is used as an example is that its pattern, especially the reticulated subspecies, is an example of a simple geometric pattern, the Voronoi diagram.[3] The fact that the pattern is a Voronoi diagram can be established quantitatively. After scanning in the patterns, we drew by hand the outlines of the spots of the pattern, and applied a geometric construction to each cell that can determine an estimate for the center of the Voronoi cell. This construction was proposed by Honda [hond78] (see Figure 3.13). At a vertex v_k, the edge $e_k = (v_{k-1}, v_k)$ and $e_{k+1} = (v_k, v_{k+1})$ form an angle α_k and α_{k+1} with the third incident edge e_i. The internal line l_k is constructed by rotating either e_k by α_{k+1} or e_{k+1} by α_k. For a true Voronoi polygon all lines meet at the site which defines the polygon. For a non-Voronoi polygon with k vertices we can compute a point P that minimizes the value of

$$\varepsilon = \frac{1}{k} \sum_{j=1}^{k} d_j^2$$

where d_j is the distance from the point P to the line l_j. The value of ε is taken as the error on the position of that point. It is then normalized by the area A of the polygon and averaged across all N cells with a valid center. This is the number M we use to measure the closeness to a Voronoi diagram

$$M = \frac{100}{N} \sum_{i=1}^{N} \frac{\varepsilon_i}{A_i}$$

Figure 3.14 shows four patterns and the values of M, for the *reticulata*, the *rothschildi* subspecies, one of our generated patterns, and for a true Voronoi diagram.

[3] We have to distinguish between Voronoi cells mentioned earlier and used to tessellate the 2D domain, and the Voronoi diagram created by the overall pattern, which are unrelated.

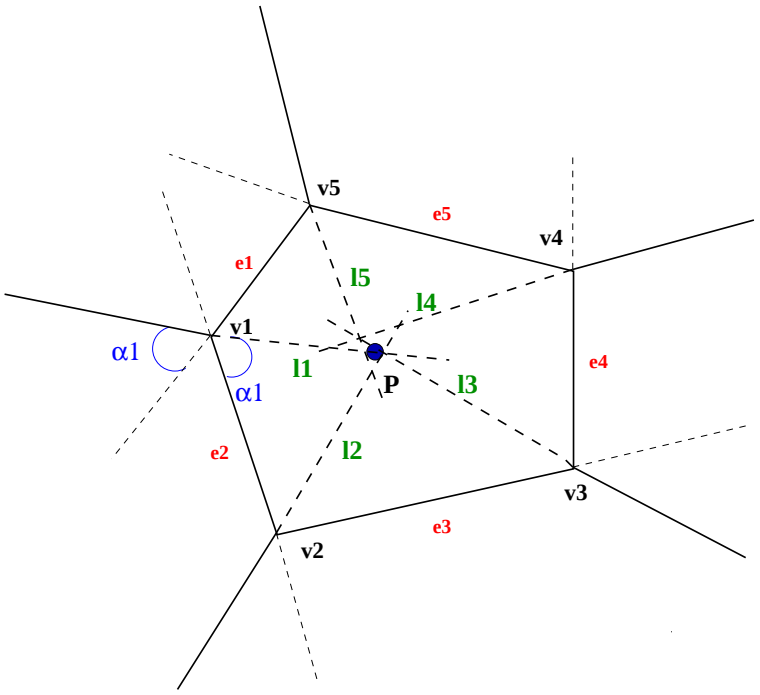
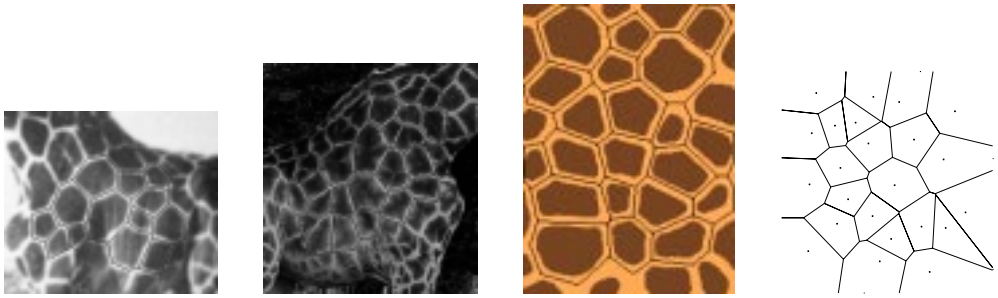


Figure 3.13: Geometric construction to estimate the center for a Voronoi polygon

To give an idea of the meaning of the magnitude, a value of 2.41% is experimentally obtained if we randomly displace the vertices of the cells by 1% of the average perimeter of a Voronoi cell in an exact Voronoi diagram. Visually, an error of 0.896% is presented in Figure 3.15 where we superimposed the Voronoi diagram drawn by hand and a Voronoi diagram computed from the estimated centers.

One can see from these numbers that the giraffe spots closely approximate a Voronoi diagram (the distortions due to the curvature of the body do affect that number). The CM model can easily explain why a Voronoi pattern is created. If the adhesion between cells of the same type is high, and the adhesion between cells of different types is relatively low, or even zero, then cells of the same type will stick together. If the foreground (spot) cells divide faster, they will crowd out the



(a) Reticulated M = 1.43

(b) Rothschildi M = 2.87

(c) CM Computed M = 0.896

(d) True Voronoi M = 0

Figure 3.14: Voronoi measures for giraffes

background cells and push them to lines between the spots. The process is similar to the so-called *prairie fire* model to produce a Voronoi diagram.

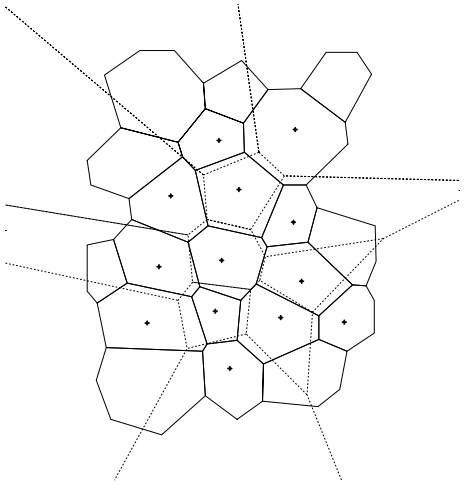
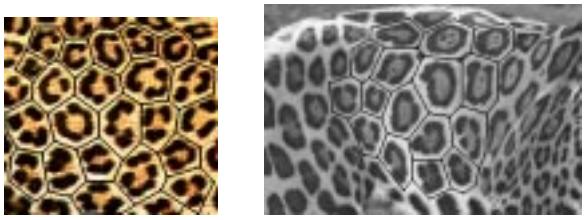


Figure 3.15: Comparison between estimated and true Voronoi diagrams. $M = 0.896$

The Voronoi pattern is quite basic, and it occurs in the big cats as well, although not as spectacularly as in the reticulated giraffe. For example, Figure 3.16 shows the patterns for the leopard and the jaguar, with measures obtained as de-

scribed. The fit of the leopard pattern we used is good, the fit for the jaguar is less so, but still convincing. Other quantitative measures for validation can be used. We reproduce in Table 3.2 statistical results about giraffe patterns presented by Dagg [dagg68].



(a) Leopard M= 0.702

(b) Jaguar M= 1.95

Figure 3.16: Voronoi measures for leopards and jaguars

Species	**Spot Area (%)**	**Number of Sides per Average Spot**
tippelskirchi	59	12
reticulata	80	5
rothschildi	50	6

Table 3.2: Spot areas and spot shapes for giraffes (after[dagg68])

The notion of *spot area* captures how much of the total giraffe body's area is covered with "polygonal spots". In the giraffe patterns produced with the CM model, we can compute an equivalent ratio of the number of foreground cells to the total number of cells and use this value to validate them. The numbers from the CM model are 55 for the *tippelskirchi* subspecies and 78 for the *reticulata* subspecies. These numbers are close to the measured ones for the two subspecies, less than 7% variation, a small value considering that the numbers given by Dagg are actually for the whole animal's body. The number of sides counted, while quite arbitrary for

the *tippelskirchi*, is reliable for the *reticulata*, and corresponds quite closely to the average number of sides for a Voronoi polygon in a Voronoi diagram, which is near 6 [prep85].

3.6 Exploration of the Parameter Space

We concentrated our efforts in fine tuning the parameters of the CM model to produce the patterns for the giraffe, cheetah and leopard, since these are the most visible ones and also have the same basic Voronoi-like structure. The model can, nevertheless, generate other types of patterns as well, even though we have not formally explored the parameter space in a methodic way. One attractive possibility for this exploration is to use the *Design Galleries* approach [mark97] where the user interactively refines the search for significant results based on off-line computations. The difficulty here lies in the definition of a visually "good" pattern.

Just to give an example of another type of pattern not related to the previous ones and close to some domestic cats, we include here the results for the Ocelot (*Felis pardalis*). The extra parameters for this pattern, not mentioned in the table below the figure are as following: $\alpha_{FB} = 0.6$, $\alpha_{BF} = 0.6$, $\alpha_{FM} = 0.7$, $\alpha_{MF} = 0.7$, probability of F cell switching to an M type of cell equal to 50%. Although there was no artificial anisotropy introduced in the computation, the combination of parameters produced stripe-like structures.

(a) Ocelot (b) Computed (c) Computed

Figure 3.17: Ocelot (*Felis pardalis*) pattern

Param.	ρ	wr	time	mitosis F	mitosis B	mitosis M	α FF	α BB	α MM
Value	18	2.4	20	8	60	4	0.2	0.6	0.7
Value	18	2.4	20	8	60	4	0.3	0.6	0.7

3.7 Clonal Mosaic and Reaction Diffusion

Since Reaction-Diffusion (RD) has been used in computer graphics modelling and found to be useful for a wide range of patterns, it is necessary to have a reasoned comparison between RD and Clonal Mosaic (CM).

To compare the relative power of RD and CM, we will use a strategy directly inspired from formal systems, which is to *reduce* RD systems to CM systems. We will give a constructive procedure which, given an RD system will produce a CM system that will create the same patterns. If successful, that means that CM systems are at least as powerful as RD systems. If we could show the reverse, that is to simulate any CM system with an RD system, then we would prove *equivalence*. On the other hand, if we could prove that there are patterns that can be created by CM

systems but not RD systems, we would have proven that RD is properly contained in CM. At this stage we establish only the first step, the reduction of RD to CM.

The strategy for the reduction is to simulate the diffusion term with the motion of the cells as random walk, and to simulate the reaction term by setting the cell division rates as an appropriate function of the concentrations. The goal is to show that given any RD system of equations we can set up a CM system with parameters derived from the parameters of the RD system that will produce the same patterns.

The notion of pattern is slightly different in the two classes of models. In CM a pattern is directly expressed by the color characteristic of each type of cell. In RD the pattern is derived from the concentration of one or more of the products involved.

In what follows we will define as *pattern* a tessellation of the space (in all our examples a 2D manifold) by regions. Each region is assigned a color from a small discrete set. In RD systems the assignment is based on concentrations. The concentrations of some of the *morphogens* (usually the activator, but not necessarily) are given a threshold, and the areas where the concentrations are above this threshold are given one color, while the other areas are given another color.

For CM systems, at the end of the simulation a *concentration* for each type of cell will be computed at each point by counting the number of each type of cells within an area defined by a *filtering kernel*. The size of the filtering kernel is to be determined according to the size of the pattern elements that are to be revealed. For example, at one extreme if the kernel area is much smaller than the size of a cell, the color assignment turns out to be identical to one obtained by assigning directly a

color to each cell. At the other extreme, if the kernel covers the whole area, the color will be uniform. We will see that this approach is consistent with the interpretation of concentration we have for the cells in simulating an RD system. In general, the size of the kernel should be slightly smaller than the smallest *feature* of the pattern considered.

3.7.1 Definition of Concentrations in Clonal Mosaic

The most direct way to define a unitless concentration in the Clonal Mosaic model is to use the equivalent of mole fraction:

$$x_A = \frac{N_A}{N_{all}} \tag{3.1}$$

where N_A is the number of cells of type A within some area and N_{all} is the total number of cells in the same area. One advantage of this measure is that it is a number between 0 and 1 and equal to the probability of picking an A type of cell at random from the total (assuming all cells have equal probability to be picked).

In many expressions, in particular in the equations for RD systems, the concentration is expressed as the *number of moles per unit volume* (in 2D *number of moles per unit area*). Assuming the same area $\mathcal{A}$ for all cells, we compute the concentration of A in mole/area as:

$$a = c_A = \frac{N_A}{\mathcal{A}} = \frac{N_A}{N_{all} \times \hat{A}} = \frac{x_A}{\hat{A}} \tag{3.2}$$

where $\hat{A}$ is the average area of the cells (volume in 3D)[4]. If there is a different area

[4]Strictly speaking for a number of mole we should divide by Avogadro's number. This is relevant when using coefficients from chemical equations.

$\mathcal{A}_i$ for each type of cell i, then the expression is:

$$a = c_A = \frac{N_A}{\mathcal{A}} = \frac{N_A}{\sum_i N_i \mathcal{A}_i} = \frac{x_A}{\hat{A}} \tag{3.3}$$

3.7.2 Diffusion

We will give a descriptive argument for simulating diffusion with the CM mechanism. We will use an RD mechanism in one dimension x with diffusion only. Given the function $C(x,t)$ that defines the concentration of a substance C at position x and time t, its diffusion is expressed by the following formula:

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} \tag{3.4}$$

where D is the diffusivity of C. Equation (3.4) says that the variation over time on the concentration of C is proportional to D times the second partial derivate of C. The second partial derivative expresses how the rate of material distribution is changing over the space. We can analytically solve this equation if we consider an initial distribution for $C(t)$. We will assume that the initial concentration is a delta function:

$$C(x,0) = \delta(x) \tag{3.5}$$

The analytic solution for this equation is [kesh88]:

$$C(x,t) = \frac{1}{2(\pi D t)^{1/2}} e^{-x^2/(4Dt)} \tag{3.6}$$

Equation (3.6) expresses exponential decay of concentration with time. At the limit the concentration will be everywhere equal to a stable value. The end result is that the initial delta function "smooths" out with time.

The approach to achieve the same behaviour with the CM model is to show that under the same initial conditions, the combined effect of cells splitting and relaxing can be qualitatively described by an equation with the same form as equation 3.6.

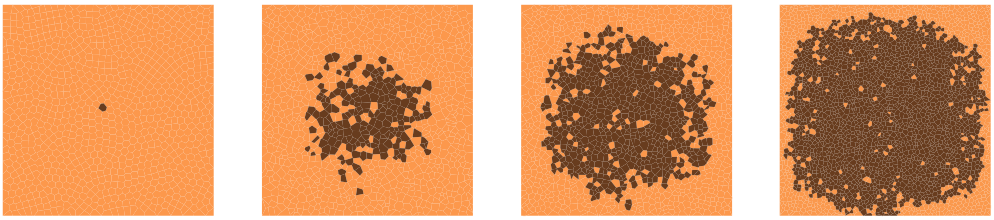
We need first to define a delta function in CM terms. According to the concentration definition (equation 3.2), in order to have an infinite concentration of cells at time $t = 0$, we can have one cell defined over an infinitesimal area. For the splitting process the number of cells increases at an exponential rate. We can describe the division behaviour as $C(t) = N_0 e^{kt}$, where N_0 is the initial number of cells and k is the splitting rate. In our example above, $N_0 = 1$ and k can be any positive number.

For the relaxation process, if the repulsive forces involved during the relaxation are strong enough, the cells will undergo a process known as *random walk*. The behaviour of many individual cells that undergo random walk with respect to the distance they travel is expressed mathematically by an equation of the same form as $\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2}$. Actually, the diffusion equation itself can be derived from the random walk behaviour of many particles [kesh88], in our case the cells.

In summary, a single cell initiates the process by splitting at an exponential rate. The total number of cells at any given time is subject to strong repulsive forces which cause them to move randomly. The overall result is equivalent to a diffusion process.

One last point about the initial concentration function is: would the result still be valid if we had a different initial concentration function? The answer is that the result is still valid since we can express any arbitrary function as a weighted sum of delta functions [four95].

Figure 3.18 shows a sequence of simulations exemplifying the diffusion process using the CM model. Note that O. Hammer [hamm98] has made a similar claim about the equivalence between signalling and diffusion.



(a) Initial configuration (b) Time = 50 (c) Time = 60 (d) Time = 70

Figure 3.18: Diffusion process in Clonal Mosaic

3.7.3 Introducing Concentrations into the CM Model

A straightforward way to introduce concentrations into the CM model is to create *context-sensitive rules*, where the context is tied to the concentrations of the various types of cells in a given neighbourhood.

To obtain a given rate of appearance $r_A = \frac{da}{dt}$ for cells of type A, we need a subdividing rate for A equals to $s_A = \frac{1}{N_A}\frac{dN_A}{dt}$ (the term $\frac{1}{N_A}$ is to make the subdivision rate a relative one). Since ultimately the new cells increase the total area, one

should take that into account when computing r_A and therefore:

$$r_A = \frac{da}{dt} = \frac{d}{dt}\left(\frac{N_A}{\mathcal{A}}\right) = \frac{dN_A\mathcal{A} - d\mathcal{A}N_A}{dt\mathcal{A}^2}$$

but since $\mathcal{A} = \frac{N_A\hat{A}}{x_A}$ we have:

$$r_A = \frac{dN_A x_A - dN_A x_A^2}{dtN_A\hat{A}} = \frac{dN_A x_A(1-x_A)}{dtN_A\hat{A}} = \frac{s_A x_A(1-x_A)}{\hat{A}}$$

and the subdivision rate to achieve a given concentration is:

$$s_A = r_A \frac{\hat{A}}{x_A(1-x_A)} \tag{3.7}$$

where s_A in the CM model is the probability of subdivision for a cell per unit time interval.

To produce a given value $r_A = \frac{da}{dt}$, we can define rules where the rate of subdivision of the cells is a function of the surrounding concentrations. If r_A is negative, we will have to introduce rules that "kill" cells at the appropriate rate. As a simple example we could count in a given neighbourhood around the cell for which the rule applies the number of cells of type A, which gives the concentration a, the number of cells type B, which gives b, etc. The rules then would be of the type: *if A has 5 neighbours of type A and 3 neighbours of type B then s_A is computed according to equation 3.7 for the concentrations of A and B.*

It is simpler and more effective to use the data structure that we have, and use the Voronoi neighbourhood of each cell to determine the local concentration. The advantage is that this approach gives a strictly local estimate of the concentration and avoids decisions about the size of the neighbourhood to consider. The drawback is that the estimate can be uncertain, especially when the number of neighbours is low.

At the limit, however, the average of the estimated concentration converges to the correct one, since all the cells will have been counted. The concentration for cells of type A is then computed as:

$$a = \sum_{i=0}^{n} \frac{1}{\mathcal{A}_i} \tag{3.8}$$

where the range is over all the cells neighbouring the cell in question (including its own cell) and $\mathcal{A}_i$ the area of cell i.

We now have all the steps we need to simulate an RD system with an equivalent CM model.

3.8 Summary

This chapter introduced the Clonal Mosaic model for generating mammalian coat patterns, specifically the patterns from the *Felidae* family and also of the giraffe. The model proposes that these patterns are an expression of an underlying spatial arrangement of epithelium cells. Different types of cells are responsible for the different hair colors seen in these animals. The patterns arise as the result of variations in division rates, cell adhesion and anisotropy in the motion of cells.

The results confirm the potential of the CM model to deliver a collection of patterns visually similar to the real ones. For the giraffe patterns we determined that the basic pattern is close to a simple Voronoi diagram, and the CM model can account for this easily, both conceptually and with the patterns produced. Another measure is the total area covered by spots, and we showed that the giraffe patterns produced by the CM model are within 7% of the real patterns with respect to this

number. Finally, we presented an argument on how to simulate any given RD system with an equivalent CM system.

Chapter 4

Models for Shape

In this chapter we present background material on shape: methods to describe shape in general and methods to represent shape in computer graphics.

4.1 Methods to Describe Shape

In this section we present a possible taxonomy for shape description. The terms for this taxonomy come from Koenderink's book [koen90]. We adapted Koenderink's classification according to existing object-modelling approaches in computer graphics and extended it where necessary. The classification here presented is not being specifically used but provides a common background on possible ways for describing shapes.

The criterion used to distinguish between methods was how the shape of an object is being defined and manipulated within them. These methods are all possible alternatives to obtain a shape description adequate for our purposes. We give a brief

description of each one, some examples, and at the end we explain why we consider the morphometry method as best suited to fit our needs.

- **Morphography**

  A shape is described by its graphical depiction on a medium, or in the case of computer graphics as the rendering of the object in a computer display. The depiction is the description. A metaphor for changing the description is that the user is "sculpting" the object interactively. The user has the visual feeling that he is actually building a physical object. Changes in shape are usually easily accomplished by using a set of pre-defined tools (scale, rotate, etc). Many graphics modelling systems, such as *Alias* and *Dragon* [fors88], implement this approach.

- **Morphonomy**

  In the morphonomy method, a given shape is completely described by numerical data together with an algorithm or rule to interpret the numbers in terms of the object. This is the underlying method in many "shape from image" approaches in computer vision and computer graphics. In these approaches the extensive and complete numerical data comes from 3D digitizers which output range data. One example is presented by Turk and Levoy [turk94] where many range images[1] are combined to create a polygonal mesh which matches the topology of the digitized object. Another example is by Hoppe *et al.* where an object is constructed from a set of scattered range data [hopp94]. An alternative example is presented by Forsey and Wong [fors98] where the range

[1] A *range image* is generically an $m \times n$ grid of distances.

information is manipulated into hierarchical B-splines, a higher level surface description representation. Objects constructed from volumetric data [mill91] are yet another example. Within the morphonomy category the only way to change the shape is to change the actual physical object which is being digitized.

- **Procedural**

  In the procedural approach, the shape is implicitly described by an equation or algorithm implemented as a procedure. Changing the shape is accomplished by changing either coefficients in the equation describing the shape or input parameters in the case of a procedure.

- **Morphometry**

  In this method only "noticeable" or "important" features of the shape are numerically given and the whole object can be constructed from these (as opposed to the morphonomy method where the whole shape is described by data). The meaning of important varies from discipline to discipline. In our case of animal shapes, for example, the definition of which features are important comes from knowledge of animal growth in fields such as zoology and biology. Changes in the shape are automatically derived from changes in the features.

The only approach that can fully deal with objects that change shape over time is morphometry; the other three are less flexible. This is in part understandable since most manufactured objects do not change shape with time. Some degree of shape transformation in the morphographic method, for example, is handled through

free form deformations (FFD) [sede86] but FFD methods lack adequate control over the deformations because they are indirectly controlled by the user through manipulations on a 3D lattice in which the object is embedded. Some approaches to reduce the complexity of control are presented by Chang and Rockwood [chan94] and by Hsu *et al.* [hsu92], but these solutions still can not handle complex shapes with intricate features.

The morphometric method, on the other hand, can be viewed as a way to handle shapes that we want to change over time. Further, the shape is an almost straightforward representation from the data, therefore changing the shape is accomplished by changing the underlying data. The last advantage of the morphometric approach over the others is that if we have the data for the noticeable features as a function of time, we can easily make the shape change as a function of time, as we will show in the next chapter.

4.2 Methods to Represent Shape in Computer Graphics

Shape is realized through a representation or a model. Therefore we can talk about the primitives used to build a model that ultimately represents a given shape. For objects embedded in three dimensional space, we can distinguish between surface-based and volume-based primitives. Using one or the other depends on the applications for the models. For example, if the application demands that we know the volume of the modeled objects then a representation with volume-based primitives

is more efficient than a surface-based representation. At the level of representation, the class of objects that we are addressing in this thesis are conveniently described by surface primitives and we restrict our description to these. The three most common representations in computer graphics are *polygonal meshes*, *parametric surfaces* and *implicit surfaces*. We give in this section a brief description of each.

4.2.1 Polygonal Meshes

In a polygonal mesh the shape is represented as a collection of polygons. A polygon is described by a closed collection of edges and each edge is shared by exactly two polygons. Edges are straight line segments connecting two vertices. A vertex is a point in space. A convenient polygon for some applications is a triangle since for a two-dimensional manifold the triangle is the simplex and therefore any polygon can be expressed as a collection of triangles. Polygons are almost always the preferred representation for a surface when it comes to the rendering step, since many graphics systems have fast hardware processing for polygons. The main drawback of polygon meshes is the large number of polygons necessary to represent detailed or intrincate geometric information. Also, the number of polygons required to approximate a given curved surface can be arbitrarily large depending on the accuracy needed for the approximation.

Winged-Edge Data Structures

Many applications using polygonal meshes need to support queries about the topology of the mesh, such as *visit all faces that share a given vertex*. A possible data-

structure for polygonal meshes that support these type of queries is the winged edge [baum72]. It uses the edge as the indexing element and maintains a small fixed number of pointers to the other topological elements (faces/vertices) that enable to derive the adjacency between them. This fixed-size storage is a strong advantage for implementations. Considering only simple polyhedra, that is, the ones that are homeomorphic to a sphere and satisfy Euler's equation[2], an edge will always be the boundary element for only 2 faces. In Chapter 6 we will describe how we used the winged-edge data structure in our context.

Simplification of Polygonal Meshes

Recently, the simplification of polygonal meshes has attracted a lot of attention in computer graphics (e.g. the work by Garland and Heckbert [garl97] and the work by Hoppe [hopp96]). For some classes of applications it is appealing to be able to control and change arbitrarily the number of polygons representing a given object, subject to either some user-defined bound for the introduced error or to a target count for the number of polygons. For view-dependent rendering, for instance, it is useful to lower the count of polygons used to represent an object when it is far from the virtual camera [hopp97]. A useful property of these simplification algorithms is to maintain, in the simplified version of the model, distinctive geometrical features present in the original version of the model. In Chaper 6 we explain how we used these simplification techniques in order to reduce the computational time to run simulations in early stages of testing.

[2] $F - E + V = 2$, where F is the number of faces, E is the number of edges and V is the number of vertices.

4.2.2 Parametric Curves and Surfaces

A parametric surface is an extension of a parametric curve so we describe parametric curves first. We can visualize a curve as the path of a particle in space. A parametric representation captures this notion and expresses a given curve as a collection of functions of one parameter (univariate). A given point on the curve is uniquely described by the values of the functions which describe the curve, when evaluated for a given value of the parameter. A two-dimensional, generic, parametric curve $C(t)$ is defined as $C(t) = (x(t), y(t))$, where the parameter t varies in some defined interval, usually but not necessarily from 0 to 1, and $x(t)$ and $y(t)$ are the defining functions. As we change t from 0 to 1 we describe the curve.

The power of the parametric representation lies in the definition of the functions. A common choice in computer graphics is cubic polynomials, expressed as control points and basis functions, i.e., a polynomial is expressed as a collection of piecewise basis functions weighted by the control points. Cubic polynomials have adequate flexibility to model a large range of shapes and are of reasonable complexity.

If we now extend our functions to be bivariate, we can define a surface. A three-dimensional generic parametric surface $S(u, v)$ is defined as

$$S(u, v) = (x(u, v), y(u, v), z(u, v))$$

where u and v are the parameters defined in some interval, again usually between 0 and 1. Bicubic polynomials expressed as basis functions are a common choice for the functions $x, y,$ and z describing the surface. The more common basis are Bézier and B-splines [fari90].

In general, a parametric representation is more advantageous than polygonal meshes. A full comparison between both representations is beyond the scope of this thesis but the major advantage of parametric representations over polygonal meshes is the flexibility they provide for shape manipulations, extremely important in geometric modelling. A good survey on curve and surface methods is presented by Bohm, Farin, and Kahmann [bohm84].

The parametric representation of a complex shape requires joining individual parametric surfaces, commonly referred to as *patches*. In this case, the modelling method has to guarantee some continuity among different patches.

4.2.3 Implicit Surfaces

Implicit surfaces are surfaces represented by a solution of an implicit equation of the general form $f(x, y, z) = 0$ [fole90, blin82]. A common family of implicit surfaces is the *quadric*, where f is a quadric polynomial in $x, y,$ and z. Spheres, ellipsoids and cylinders can all be described by quadrics. The main advantage of the implicit representation in a modelling context is the blending property, that is, we can blend two or more implicit surfaces in a smooth way, according to a blending function that defines how to blend the functions using weights [kaci91]. The major shortcoming of the implicit formulation is the absence of adequate control over the shape of the surface. In a quadric, for instance, it is not intuitive how to change 10 possible values to achieve a given result, since the general quadric equation is of the form:

$$f(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k = 0$$

4.2.4 Conversion between representations

Later in Section 5.4 we elaborate the reasons for using polygonal meshes as the chosen representation for our animal models. We would like to consider here briefly the problem of conversion between representations. Given a parametric or implicit representation, can we generate polygonal meshes? This is a common question in computer graphics addressed by much research. A full overview of conversion techniques is outside our scope. We would like nevertheless to address the two possible cases in our context, detailed below:

Parametric to polygonal

Parametric to polygonal conversion is implemented in many graphics modelling packages. A common conversion is to approximate the patches composing the parametric surface by its control polygons. Each control polygon is recursively subdivided into two until a local flatness criterion is satisfied. The subdivision can be performed both uniformly, where all the patches are subdivided down to the same level, and non-uniformly, where large flat areas are approximated by larger polygons and highly curved areas are approximated by smaller polygons [watt92].

Implicit to polygonal

The conversion from implicit surfaces to polygonal meshes is known as polygonization, tiling, tesselation or triangulation. A good overview of the subject is given in Chapter 4 of a recent book by Jules Bloomenthal and others [bloo97]. In general, the main problem for polygonizers is how to efficiently locate the polygon vertices

in a consistent way. For some implicit surfaces a polygonal approximation can be derived from a series of surface contours.

4.3 Summary

This chapter presented a possible taxonomy for describing shapes in the context of computer graphics, adapted from one presented by Koenderink [koen90]. The four possible ways of describing a shape are *morphography*, *morphonomy*, *procedural*, and *morphometry*. We show that the morphometric approach describes a shape in a convenient way for our purposes of shape transformation presented in the next chapter. For practical purposes, the description of a shape must be represented by some format. In computer graphics there are basically 3 methods which deal with surface-based representations: polygonal meshes, parametric, and implicit.

Chapter 5

Applying Growth Information to Polygonal Models of Animals

An organism is so complex a thing, and growth so complex a phenomenon, that for growth to be so uniform and constant in all the parts as to keep the whole shape unchanged would indeed be an unlikely and an unusual circumstance. Rates vary, proportions change, and the whole configuration alters accordingly.

D'Arcy Thompson, On Growth and Form, 1942

This chapter presents the techniques we developed to transfer growth data to computer models represented as polygonal meshes. These techniques allow animation of the growth as well as animation of the body in the traditional sense. The main technique consists of defining local coordinate systems around the growing parts of the body, each one being transformed according to the relevant growth data

while maintaining their relationship with the adjoining parts and the continuity of the surface. The local coordinates also permit ordinary animation mainly as relative rotation such as in articulated objects.

We present examples with polygonal models of horses and cows, growth data from same, and motion from Muybridge's classic photographic data [muyb18].

5.1 Introduction

Building a computer model of a complex shape such as the body of a horse, a duck or a human is already a challenge, but allowing for the changes in shape caused by growth of the body and/or motion is even more difficult. It is nevertheless imperative to meet these challenges if we want to animate animal models.

This chapter presents solutions for two specific problems:

- given an animal body model as a polygonal mesh and sparse generic growth data, how to transform the body model to simulate growth;
- given motion information, how to animate the same model and simulate various gaits, such as walking, trotting, galloping, etc.

The solution to the above problems will also allow to customize a generic model to a particular race and/or individual. We also want to be able to apply both transformations (growth and animation) at the same time. We are not directly concerned either with the acquisition of the models or with the motion modelling *per se*, but with the integration of the two.

D'Arcy Thompson anticipated the connection between the quality and feasibility of a solution for the first problem and the availability of data. In chapter III of his classic book, "On Growth and Form" [thom61], he presents one section entitled "The rate of growth of various parts or organs". In this section there is a table "…to illustrate the varying growth-rates in different parts of the body" of a young trout. Later on he adds:

> *It would not be difficult, from a picture of the little trout at any one of these stages, to draw its approximate form at any other by the help of the numerical data here set forth.*

5.2 Differential Growth and the Available Data

Growth has many possible definitions depending on which aspect of the phenomenon we are trying to address. A reasonable general definition by Needham [need64] is that growth "is the increase in size and mass of the body or its parts."

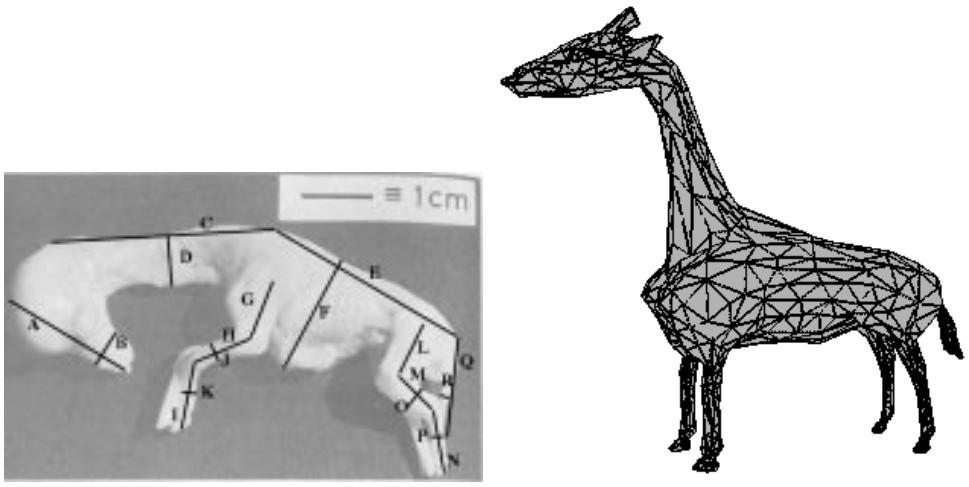
For all organisms that grow, the changes in shape are mainly due to the different growth rates associated with the different parts of the growing organism. Brody [brod64] suggests two ways to investigate relative growth: a *qualitative* and a *semi-quantitative* approach. In the qualitative approach the differential growth rates are empirically studied by comparing scaled photographs of the subjects at different ages. In the semi-quantitative approach given growth measurements are plotted as a function of age and the different slopes compared. Correlations between parts and the whole and also between different races of the same animal can thus be established. Besides being used for differential growth studies, growth measurements

play an important role in animal studies where they are used to define a set of average measurements for a given race and age of some animal. The deviation in measurements from the average values for a given individual can be used as a metric to detect for example nutritional problems. Standards for livestock and poultry weights and dimensions are mainly established using a collection of growth measurements collected over a reasonable large population of the race in question.

For practical reasons – measuring animals on-site and over a large time span – much of the available data is sparse and usually restricted to one or two measurements of some distinctive geometrical feature that can represent skeletal growth. Height at withers or at the shoulders and heart girth are the most common. Not surprisingly, we can find simple and empirical formulae to estimate the weight of some animals using only one or two "easy-to-take" measurements. For domestic animals such as horses and cows, where a better understanding of the growth process can improve management of livestock, we can find more detailed data. Two examples are the set of 19 parameters measured monthly from birth to 60 months for female and male quarter horses from Cunningham and Fowler [cunn61] and the set for dairy cattle (Holstein and Jersey races) from Brody [brod64] with 21 parameters monthly measured from birth to 36 months old. For many other non-domestic animals there is considerably less data: often only adult measurements for weight, length and shoulder height. For some large mammals the following references are a useful starting point: Meinertzhagen [mein38], Shortridge [shor34], Stevenson-Hamilton [stev47], and Roberts [robe51]. In Appendix A we present a summary of measurements for a few non-domestic animals, such as giraffes and leopards.

There are two ways to deal with the lack of quantitative growth data. The first is to use Brody's qualitative approach mentioned before, that is, derive the necessary measurements from a series of pictures of the animals at different ages (though not necessarily the same animal). An example of this is given in Figures 5.1, 5.2 and 5.3 where we show the measurements for the giraffe at three different ages: embryo, newborn, and adult, together with the computed body model at that age using the technique explained in the remainder of this chapter. In order to use this approach, we need to establish a scale for the measurements. In some cases we might have a scale present with the picture. That is the case for Figure 5.1(a), where the $1cm$ value is given. When we do not have a scale present, we derived one from known measurements of real animals. In the case of the newborn giraffe, the height of $169.75cm$ is the average height value for 28 newborn giraffes presented by Dagg [dagg76]. The shoulder height value of $335cm$ for the adult giraffe was computed as an average for six giraffes presented by Shortridge [shor34]. Table 5.1 presents a summary of these measurements.

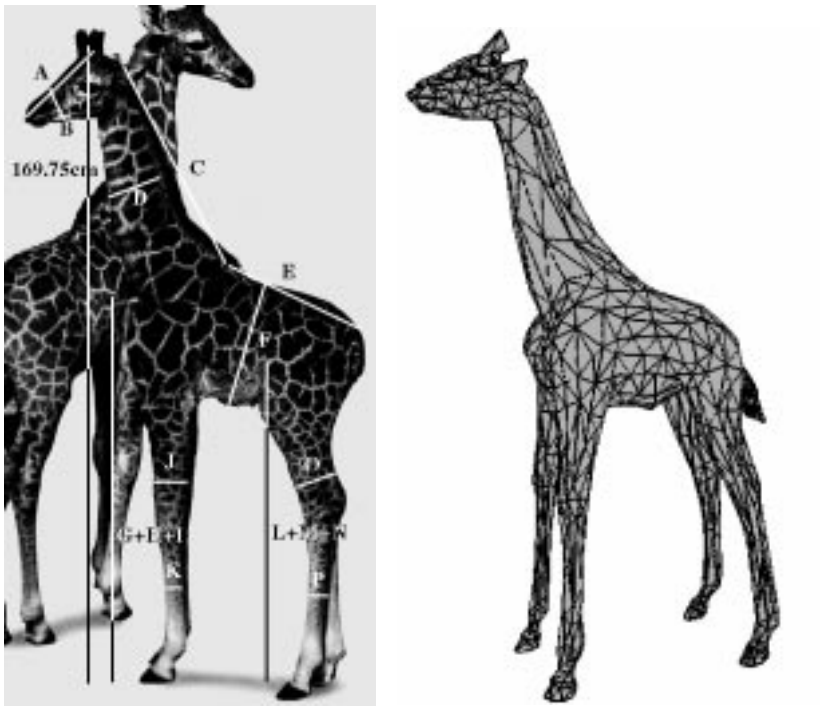
The second possibility is to use data from a close "relative" of a given animal. A typical example would be to approximate the data for zebras based on the data available for horses. Both animals have similar shapes and therefore we should be able to compute a good approximation for a zebra based on the horse. With some care similar solution could be used to approximate, for example, the data for a tiger based on the domestic cat. Finally, in some cases we want unavailable data (for instance the growth of dragons) or unrealistic data (the growth of the 50 Ft horse), and we can invent it.



(a) Real (b) Model

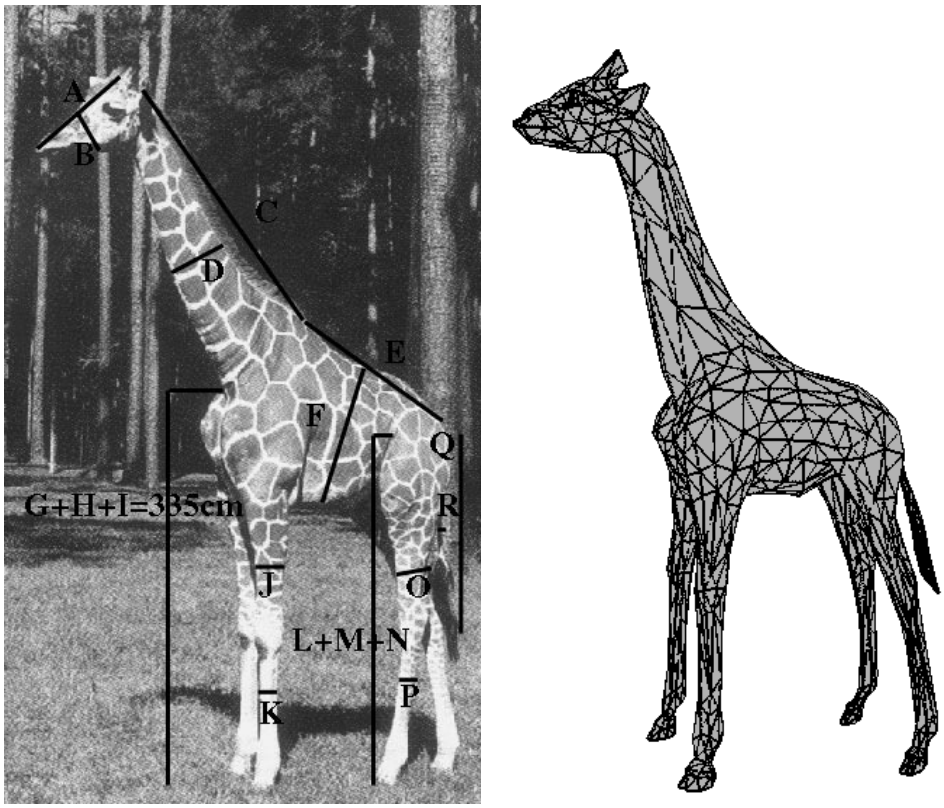
Figure 5.1: Giraffe embryo

Finally, we have to consider which specific measurements we need in order to do a reasonably good visual transformation of an animal's shape. The answer to this question is connected to what we mean by a "visually good" transformation. Reconstructing the evolution of a particular species in paleontology demands more accuracy than animating the growth of a dragon in an animated feature. We have not formally investigated answers for these cases. In our simulations we have used the measurements resulting from at least two parameters (e.g., length and diameter) for each main body part divided as follows: head, neck, body, legs and tail. These are the main groups of measurements used when measuring domestic animals, from a small survey of the relevant literature [cunn61, brod64].



(a) Real (b) Model

Figure 5.2: Newborn giraffe



(a) Real (b) Model

Figure 5.3: Adult giraffe

measurement (all measurements in cm)	embryo	newborn	adult
A: length of head (from the muzzle to the interspace of horns)	1.875	25.02	90.45
B: width of head (measurement around muzzle)	0.542	9.83	36.85
C: length of neck (from withers to base of head)	3.125	58.97	237.85
D: width of neck (midway between withers and base of head)	0.542	14.30	48.58
E: length of body (from withers to the root of the tail)	2.92	39.31	144.05
F: width of body	1.71	33.95	117.25
G+H+I: Total Length Front Leg	2.6	102.74	335
J: diameter of upper front leg	0.25	8.93	26.8
K: diameter of lower front leg	0.208	4.47	16.75
L+M+N: Total Length Back Leg	2.313	84.88	298.15
O: diameter of upper back leg	0.334	10.72	30.15
P: diameter of lower back leg	0.167	5.36	16.75
Q: Length of tail (from the root of tail to the end of the terminal tuft)	1.417	26.80	167.50
R: diameter of tail	0.125	2.68	6.7

Table 5.1: Measurements for a giraffe

5.3 Previous Work on Shape Transformation

Most of the work in this area in computer graphics inputs two objects and generates a set of intermediate ones such that the sequence conveys the idea that the first object was transformed into the second. Basic references go as far back as Burtnyk [burt76] and Reeves [reev81] and these techniques are referred to as *morphing*, *metamorphosis*, *key-framing*, and *in-betweening*. The technique we will use here to transfer growth data to polygonal models will be similar to the feature-based 2D morphing techniques, for instance as described by Beyer and Neely [beie92]. The

main differences are that we are operating in 3D (one recent example in 3D is by Lerios *et al.* [leri95] but uses volumetric data), that we only have *one object* that is to be transformed, and that we want the set of features to be organized in a structure that will allow for articulated motion as well as growth. We are not aware of any previous work where the goal is controlled transformation of a *single* shape[1].

For a polyhedral representation the morphing problem is reduced to finding for a given point in the first object the corresponding point in the second one. Different approaches then only differ on how to achieve this correspondence. In the approach by Hong *et al.* [hong88] for example, more than one face in the source object can be mapped to the same face in the target object and faces can degenerate into points. The approach taken by Kent, Carlson and Parent [kent92, kent91] is to use an intermediate object to establish the correspondence, usually a sphere. Their solution first positions the objects to be transformed in the center of a canonical sphere and their vertices are projected towards the sphere. This creates two sets of projected vertices, one set from each original object. The algorithm then maps back in the objects the vertices which were not originally there. This creates the necessary correspondence. Chen presented an approach that morphs two objects by morphing their 2D parametric descriptions [chen95]. The main drawback of the technique is the conversion to parametric representation from a polygonal one.

There are other solutions for the shape transformation problem when the objects are represented by volumetric data. Hughes [hugh92], for example, transforms the data into the frequency domain and the low frequencies from the source object

[1] It could be claimed that many modelling tasks usually have this same goal, but the emphasis is on the process of building the final object only and not on the transformation process.

are interpolated into the low frequencies of the target one. A solution for objects represented by a *union of spheres* is presented by Ranjan and Fournier [ranj95]. This solution first establishes a correspondence between all the spheres in the source object and the target, using some user defined metric, and then interpolates from one set of spheres to another. Finally, Lerios *et al.* [leri95] have extended the idea of 2D features from Beier and Neely [beie92] to 3D volumetric represented objects. A set of features defining fields of influence for the source and target objects is defined and the morphing process uses these features to warp voxels from the source object into the target one.

5.4 Animal Models

The most widely available form of models for animals (and in general for complex three-dimensional models) is as polygonal meshes (see Section 4.2.1). They come mostly from digitization of models (plaster, plastic, clay, etc..) or more rarely of real animals, and have been obtained either by input of hand chosen points on the surface from 3D input devices or by 3D scanners.

Much work has been done recently on the problem of creating polygonal meshes (and even parametric surfaces) from such data [turk94, curl96]. These models have many advantages. Since most current display architectures are dedicated to polygon rendering, speed of rendering is the most important advantage. They have, however, the distinct drawback of including no structural information about the body, and are therefore difficult to use for animation and shape transformation.

5.5 The Local Coordinates

Each section of the body that has to grow under the control of its own growth parameters or that has to move independently of the rest is attached to a cylinder defined by the user. Cylinders have been used because they have only two degrees of freedom for size, and these map easily to the type of growth measurement available form the literature. It is not excluded in further work to add other primitives such as spheres [ranj96]. Another possibility for attaching a set of local coordinate systems to the model is to use *skeletons* [fors91, chad89], basically a line from which a local coordinate system can be derived. We used cylinders because they provide a better visual feedback of where exactly is their influence in the model. They will also be convenient for providing a way to locally control the parameters of the pattern generation simulation on the 3D geometry, explained later in Chapter 6.

A cylinder is positioned such that it encloses the part of the model that it controls. At the same time its position in the modelling hierarchy is determined by the user. Figure 5.4 shows an example of the 18 cylinders that we initially defined for the horse's body. The whole structure is a *directed acyclic graph* or DAG, as in the classic object modelling hierarchy (see for instance Chapter 7 in Foley *et al.* [fole90]).

Given two cylinders, A and B, B being a child of A in the hierarchy, $M_{A \leftarrow B}$ is the transformation matrix that takes a point of B to A coordinate system, and $M_{W \leftarrow A}$ is the matrix that takes a point of A to the world coordinate system, obviously: $M_{W \leftarrow B} = M_{W \leftarrow A} \times M_{A \leftarrow B}$. The matrices of the type $M_{W \leftarrow A}$ are obtained directly from the position, orientation, and size of the cylinders in the original model space.

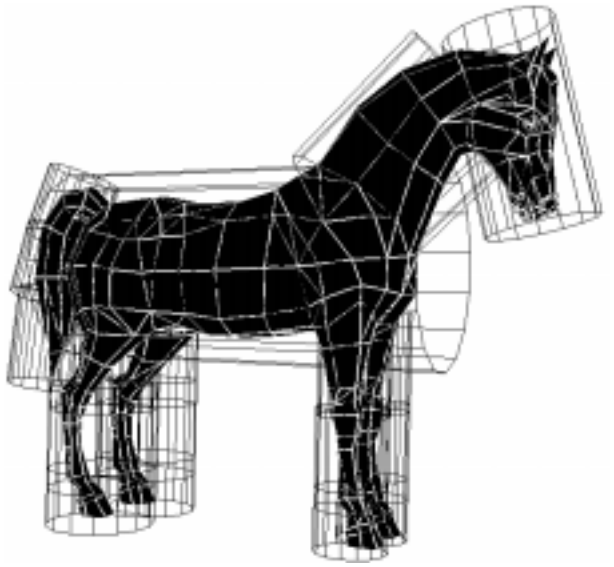


Figure 5.4: The perfect cylindrical horse

For the purpose of growth, it is convenient to be able to express the growth matrix of each primitive (the cylinders) in absolute terms because that is the way the data is collected from the real animals. For instance the growth of a leg is measured in absolute terms, not relative to the body to which it is attached. The information given by the user (positioning the cylinders) and the growth information together allows to express, for any primitive, the transformation to convert a point P_B in the primitive coordinate system to a point P_W in the world coordinate system.

$$P_W = M_{W \leftarrow B} P_B$$

To express the matrices in a consistent way we will assume that $M_{W \leftarrow B}$, for instance, has been decomposed into a scale, a translation and a rotation, applied in that order:

$$M_{W \leftarrow B} = [T_{W \leftarrow B} R_{W \leftarrow B} S_{W \leftarrow B}]$$

The scaling matrix $S_{W \leftarrow B}$ is in fact the growth matrix for B, which we will note G_B. It takes coordinates in B space (where the radius and length are both units) to their real dimensions in the world. We will see in the next section how this matrix is derived from the growth information. Since the growth data from the literature is normally given in absolute measurements, it is more convenient to keep the growth matrices absolute as well. Thus we have to make sure that G_A does not apply to B or any of its descendents. On the other end the position of the B coordinate system has to be moved according to the growth of A and its ancestors. To accomplish this we write:

$$P_W = [T_{W \leftarrow A} R_{W \leftarrow A} G_A][T_{A \leftarrow B} S_{A \leftarrow W} R_{A \leftarrow B} G_B] P_B$$

We know all the matrices in this expression (note that $S_{A \leftarrow W} = G_A^{-1}$), except $T_{A \leftarrow B}$ and $R_{A \leftarrow B}$. The latter is trivially $R_{A \leftarrow B} = R_{A \leftarrow W} R_{W \leftarrow B}$. The former can be derived by equating $M_{W \leftarrow B}$ with the whole matrix product in the above expression obtaining:

$$T_{A \leftarrow B} = [M_{A \leftarrow W} T_{W \leftarrow B} T_{A \leftarrow W} M_{W \leftarrow A}]$$

So to transform a point in B canonical space to its ancestor A canonical space, we apply:

$$P_A = [T_{A \leftarrow B} G_A^{-1} R_{A \leftarrow B} G_B] P_B$$

If A is the root, the matrix $M_{W \leftarrow A}$ takes the points to world coordinates, otherwise a similar transformation is applied to P_A to take it to the coordinate system of the ancestor of A.

5.6 The Growth Process

To obtain the growth matrix for each cylinder, the user attaches two features obtained from the growth data. A cylinder has two degrees of freedom for its dimension: L controls the scale along the canonical X axis, and R controls the scale along the canonical Y and Z axis. Figure 5.5 illustrates the relationship between a cylinder and its features. As shown in the figure, we want the position of the cylinder and of the features to be as independent as possible. The only constraint will be given by equation 5.1.

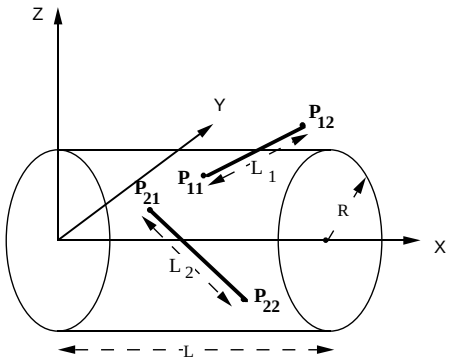


Figure 5.5: Cylinder and features.

Figure 5.6 shows the features associated with the horse body. Each feature is a length measurement between two specific points on the body. For convenience we have converted all the measurements of circumference (*girth*, etc) to a correspond-

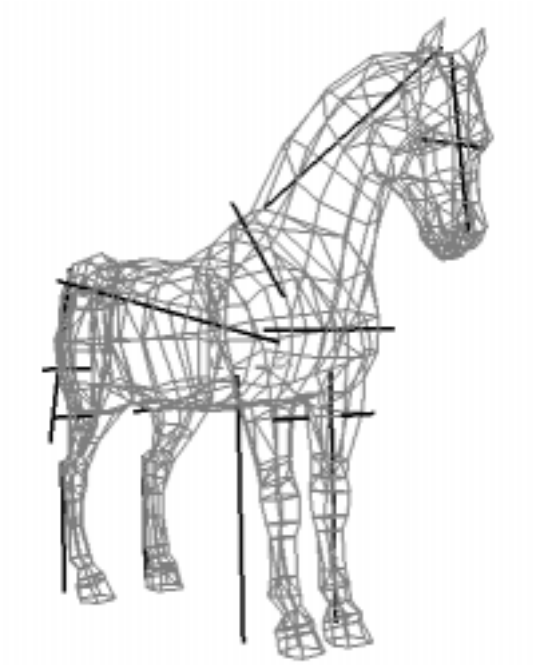


Figure 5.6: Features defined for the horse.

ing diameter (in the case of obviously ellipsoidal features, we can use the approximation $2\pi\ R\sqrt{\frac{e^2+1}{2e^2}}$ where e is the eccentricity and R the major semi-axis). The basic assumption is that these four points are fixed in the cylinder's own coordinate system, and therefore their distances uniquely determine the size of the cylinder in world coordinates as long as the two line segments do not have the same proportions in X and in the YZ plane. If L_1 and L_2 are the *real* lengths of the features given by the growth data, and (x_{ij}, y_{ij}, z_{ij}) are the coordinates of the feature point P_{ij} in the cylinder canonical coordinate system, where $R = 1$ and $L = 1$, then

$$L_1^2 = L^2(x_{12} - x_{11})^2 + R^2[(y_{12} - y_{11})^2 + (z_{12} - z_{11})^2]$$

$$L_2^2 = L^2(x_{22} - x_{21})^2 + R^2[(y_{22} - y_{21})^2 + (z_{22} - z_{21})^2]$$

This is a linear system for the unknowns L^2 and R^2; it has a unique solution if:

$$(x_{12}-x_{11})^2 \left[(y_{22} - y_{21})^2 + (z_{22} - z_{21})^2\right] \neq \left[(y_{12} - y_{11})^2 + (z_{12} - z_{11})^2\right] (x_{22}-x_{21})^2 \tag{5.1}$$

Once R_B (radius of the cylinder according to the real measurements) and L_B (length of the cylinder according to the real measurements) have been computed for a cylinder B, the growth matrix is:

$$G_B = \begin{bmatrix} L_B & 0 & 0 & 0 \\ 0 & R_B & 0 & 0 \\ 0 & 0 & R_B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is important to stress that L_1 and L_2 are the lengths as measured on the animal, that R and L are the radius and length of the cylinder scaled to the real world (the scale of the model used to initially place the cylinders is irrelevant), and that the points used to define the features do not have to be inside the relevant cylinder, or even near it. The only constraint is that these points have to be assumed to be in constant positions in the cylinder's own coordinate system and satisfy equation 5.1.

A point in space is transformed only if it is inside at least one cylinder. The initial size and position of the cylinder effectively defines the scope of its influence.

To guarantee continuity as the shape changes, and to achieve a degree of smoothness in the resulting surface, the cylinders have to overlap, and therefore we have to decide how to weight the influence of the cylinders on a point that belongs to more than one. This is similar to 2D morphing [beie92], where different features are given different weights depending on, among other things, the distance of the point from the feature. For all our simulations we used a weight inversely proportional to the distance from the point to the axis of the cylinder. For instance, given point P that belongs to cylinders A and B we compute the final position P' as follows. First, we compute the distances d_A and d_B from P to the axis of the corresponding cylinders. Then we compute the new positions P_A and P_B of the point due to growth from the individual cylinders A and B. These distances are always between 0 and 1 since the point has to be inside the cylinders. Since we want weights inversely proportional to the distance we make:

$$d_A = 1.0 - d_A \qquad d_B = 1.0 - d_B$$

$$w_A = d_A/(d_A + d_B) \qquad w_B = d_B/(d_A + d_B)$$

and finally P' is:

$$P' = P_A w_A + P_B w_B$$

We have not investigated the effect of this on continuity of the surface. It does not prevent folding and self-intersection, in the case of extreme rotation, but this is not a concern in practice.

Finally, the growth simulation loop consists of reading, for each age and each cylinder, the lengths of the features, computing the growth matrices and applying them to compute the new coordinates of each vertex of the polygonal mesh.

5.7 Animation

The hierarchical model established by the cylinders can be easily used for ordinary motion control as well as growth. Any transformation local to the cylinder coordinate system is applied before the chain of transformations described above. It has to be such that it does not change the distances measured in the growth information, or that we can neglect such change. Relative transformations such as a scale matrix S or a rotate matrix R can be applied as:

$$P_A = \left[T_{A \leftarrow B} G_A^{-1} R_{A \leftarrow B}\ R\ S\ G_B\right] P_B$$

The matrix S can be on either side of G_B since the scalings commute. The most useful form is a simulation of the rotation of the joints of the animal during walking, trotting or galloping. For this purpose it might be necessary to create more cylinders than are required for the growth process. For example for the horse there is only one measurement for the leg from the shoulder down, when one needs to rotate at the shoulder, the elbow, the knee and the fetlock joint. This is easily accomplished since different cylinders can be controlled by the same growth data. The effect is that their relative sizes will remain the same as the animal grows.

The rotation angles have to be measured from the position in which their cylinders have been initially defined (see Figure 5.7 for an illustration).

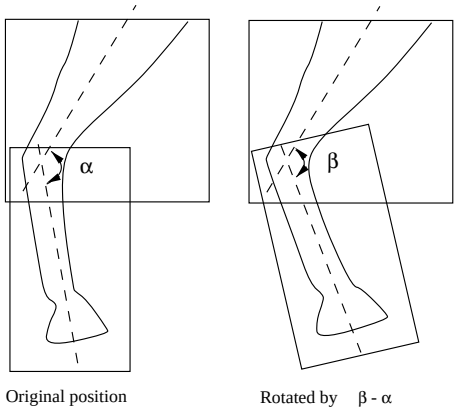


Figure 5.7: Rotation of cylinders *vs* joints

5.8 Examples

As mentioned before, the most abundant source of growth data is for domestic animals. We will illustrate the methods with two examples, the horse and the cow. The horse data was taken from Cunningham [cunn61] and quantifies the growth of a male quarter horse (Table 5.2 presents a sample of the data). For our actual examples we used 9 measurements (plus two "fake" ones for the tail) at 9 different ages (0, 3, 6, 12, 18, 24, 36, 48 and 60 months).

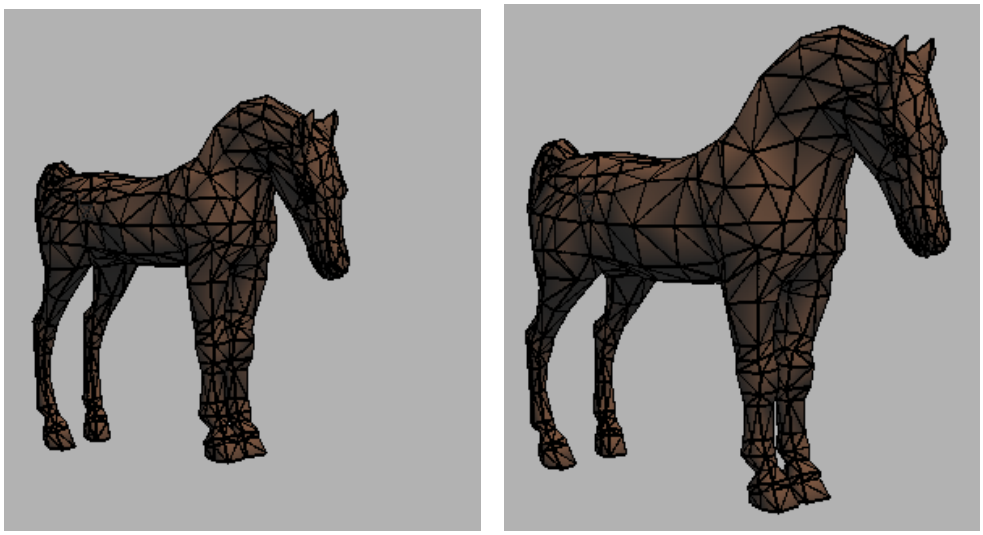
Sample data for quarter horse (inches)				
Age (months)	Length from elbow to ground	Diameter of cannon bone	Length of hind leg	Width of head
0	25.0	1.43	18.1	5.6
12	33.1	2.19	22.8	7.8
24	35.9	2.39	23.0	8.6
36	35.1	2.39	23.2	8.9
48	34.9	2.42	23.1	9.1
60	35.8	2.51	23.3	8.8

Table 5.2: Some measurements for a quarter horse.

The cow data was taken from Brody [brod64], and quantifies the growth of Holstein cattle (Table 5.3 presents a sample of the data). For our actual examples we used 9 measurements at 8 different ages (0, 3, 6, 12, 18, 24, 30 and 36 months).

Sample data for Holstein cattle (cm)					
Age (months)	Height at withers	Height at croup	Depth of chest	From poll to muzzle	Width of forehead
0	70.4	74.1	28.4	23.2	11.8
12	112.6	116.8	54.5	44.3	18.1
24	126.7	129.5	64.2	52.0	20.1
36	131.6	133.7	68.4	52.1	19.7

Table 5.3: Some measurements for Holstein cattle.



(a) Horse at 6 months.

(b) Horse at 36 months.

Figure 5.8: Horse transformed at 6 and 36 months.

The polygonal model was obtained from the Viewpoint database (`www.viewpoint.com`). For the purpose of illustrations we used the relatively simple models. The horse (model VP1346) contains 674 vertices and 863 polygons. The cow (model VP1323) contains 2892 vertices and 4179 polygons. The computational cost is strictly proportional to the number of vertices to be transformed.

Figure 5.8 shows the horse polygonal model transformed to the proportions at 6 and 36 months. Note that the proportions of the transformed horse are given by the growth data, not by the original polygonal model. Note also that nothing abnormal

(folding, discontinuity) happens to the surface during the transformation.

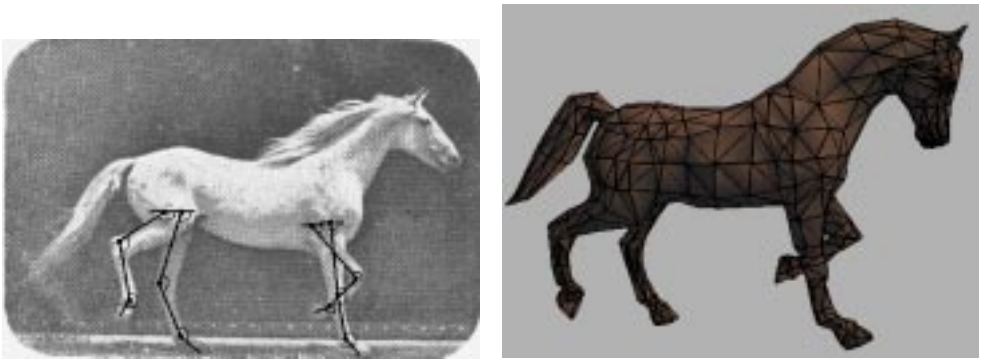
(a) Cow at 6 months. (b) Cow at 24 months.

Figure 5.9: Cow transformed at 6 and 24 months.

The same method was applied to the Viewpoint cow model using the Cunningham data. Figure 5.9 shows the cow's body at 6 and 24 months.

To illustrate the transfer of motion, we used the famous photographs of Eadward Muybridge [muyb18]. Figure 5.10(a) shows a frame of a trotting horse from Muybridge together with the set of lines we used to measure the joint angles between articulations in the horse's leg. Figure 5.10(b) shows the corresponding frame of the Viewpoint horse body trotting.

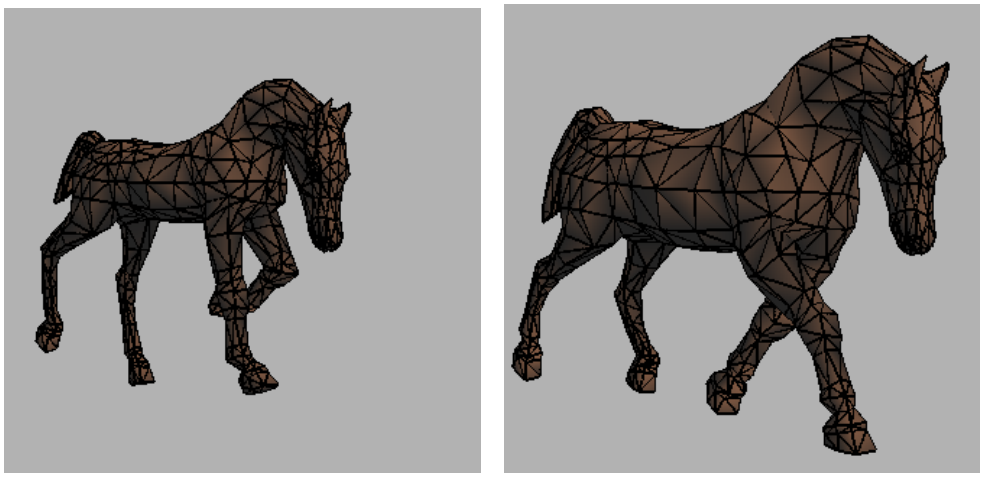
Since Muybridge's photos only show a half-stride, we exchanged the left side and right side angles for the other half-stride of the trot. We applied the same techniques to the cow. Muybridge's photos actually show an ox, but only specialists would know the difference. Now that we have the structure of cylinders and features in place, we can apply both the growing and the motion information to the same model. Figure 5.11 shows two frames of an animation of the horse running and growing at the same time.



(a) Frame from Muybridge's trotting horse.

(b) Frame from our trotting horse.

Figure 5.10: Muybridge's and polygonal horse trotting.



(a) Horse at 6 months

(b) Horse at 60 months

Figure 5.11: Horse growing and trotting.

We can also transfer one set of data to another body. For instance we can simulate the horse trotting like a cow, or, maybe more interestingly, the cow growing like a horse. In this case we would have to make sure the cylinders over the cow's body correspond to the cylinders defined for the horse, and that the features for the horse have equivalent points for the cow. If the two body plans were much different it would not be so straightforward (or meaningful).

5.9 Summary

We have shown how growth information of the type commonly found in the literature can be integrated and applied to commonly available models of animal bodies. The presented technique builds a system of local coordinates related in a hierarchical structure, defined by cylinders enclosing the relevant parts of the model and features controlling their size. We will see that the cylinders are also essential to provide local control during the pattern formation process. The same structure can be used to animate the body by transferring motion data. In our examples the motion data was taken from photographic frames, but any system giving joint angles could have been used to control the motion.

Chapter 6

Integration

This chapter describes how the Clonal Mosaic model presented in Chapter 3 is integrated with the shape control presented in the previous chapter. This will allow the synthesis of Clonal Mosaic patterns directly on a shape-changing geometry, such as the body of the animal growing. As mentioned previously in Section 2.2, there are two distinct phases in this process: the pattern developing as the body of the fetus grows, and the body growing with the pattern following the growth of the body.

6.1 Overall description

Figure 6.1 shows a schematic representation of the whole process of pattern formation development in connection with the body growing.

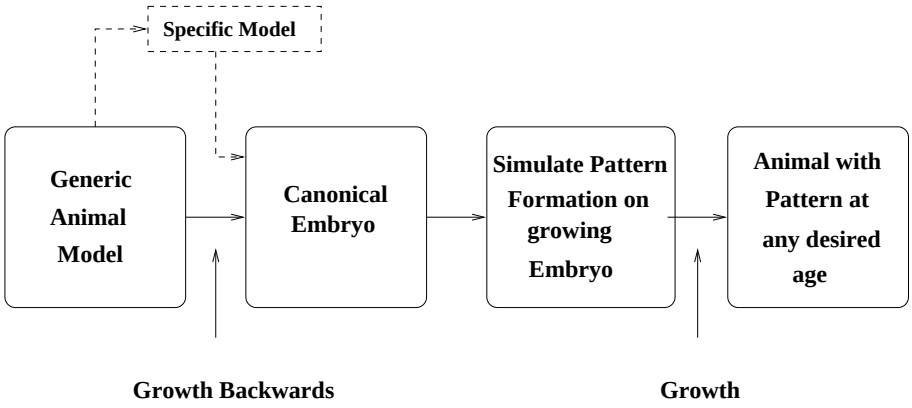


Figure 6.1: Pipeline of the system

6.2 Deriving Cell Splitting Rates from Growth Information

Our patterns can be viewed as many small cells multiplying to form a specific arrangement – the growth of a tissue. Shape can be indirectly expressed through growth-curves and the final shape of a given natural object is a result of cells dividing at different rates. It is reasonable therefore to use growth as an integration factor driving the patterns and the changes in shape. In practice we need to be able to compute how fast the initial set of cells should split in order to keep up with the increase in area of the object.

For the following description we will assume, without loss of generality, that we only have 3 types of cells called F, B, and M as presented in Chapter 3. The total number of cells is $N_t = \sum N_i$ where $i = F, B, M$. At time $t = 0$ we have the individual area for a single cell as

$$a = \frac{A_0}{N_0}$$

where A_0 is the initial area of the model, usually the area of the model for the time when we estimate that the pattern formation process starts, and N_0 is the initial number of cells. The goal is to keep the area of a single cell constant as the model grows. We also have to establish the relative rates of splitting between the different types of cells. We will call these g_F, g_B, g_M. Since they are relative we know that $\sum g_i = 1$.

The net increase in the number of cells is proportional to the increase in the area of the model, that is

$$\Delta N = \frac{\Delta A}{a}$$

A factor k is computed to express the net increase in the number of cells among the different types.

$$k = \frac{\Delta N}{\sum N_i \, g_i}$$

and the net increase in the number of cells for each type is:

$$\Delta N_i = k \; N_i \, g_i$$

The instantaneous rate of splitting s is given by:

$$s_i = \frac{\Delta N_i}{N_i \Delta t}$$

and the rate r at which we need the cells to split is the reciprocal of s

$$r_i = \frac{\Delta t \, N_i}{\Delta N_i}$$

A numerical example should help clarify these equations. Let us assume a polygon with an increase in area of $\Delta A = 0.05 cm^2$ and arbitrarily $a = 0.004 cm^2$. Also let us assume that the we only have two types of cells, F and B, with F splitting

3 times as fast as B and that their current number is $N_F = 10$ and $N_B = 40$. The relative rates of splitting are $g_F = 0.75$ and $g_B = 0.25$. We have then

$$\Delta N = \frac{\Delta A}{a} = \frac{0.05}{0.004} = 12.5$$

$$k = \frac{\Delta N}{N_F\, g_F + N_B\, g_B} = \frac{12.5}{10 \times 0.75 + 40 \times 0.25} \simeq 0.714$$

$$\Delta N_F = k\, N_F\, g_F = 0.714 \times 10 \times 0.75 \simeq 5.357$$

$$\Delta N_B = k\, N_B\, g_B = 0.714 \times 40 \times 0.25 \simeq 7.143$$

and finally the rates at which cells split are

$$r_F = \frac{10}{5.357} \simeq 1.867 \quad r_B = \frac{40}{7.143} \simeq 5.599$$

6.3 Triangulation and Simplification of the Model

As mentioned in Section 5.4 the animal models we are using are represented by polygonal meshes. A convenient polygon to use is the triangle due to specialized hardware to deal with them in graphics workstations. Triangles also lead to simpler data structures and some geometric attributes are straightforward to compute such as normal vectors. Another reason to use triangles, in the context of the Clonal Mosaic model, is that the pattern is expressed as Voronoi polygons for the set of points that represent the cells. The mesh formed by the Voronoi polygons replaces the original triangle mesh. If the faces in the original mesh are not planar then the tessellation provided by the Voronoi polygons might not match the original mesh topology.

Since our solution to integrate the simulation of the Clonal Mosaic model on the surface of a polyhedron is proportional to the number of faces, we found convenient in some situations to be able to run the simulation on a model with a lower

count of polygons. We used Garland and Heckbert's [garl97] (see Section 4.2.1) tool for simplification of geometric models for this purpose. After the pattern is computed on the simplified version of the model we can map the pattern back to the original high-resolution geometric model using the cylinders. We can compute cylindrical coordinates for each cell and use them to map the cell's position to the higher-resolution model, since both models have the same set of cylinders defined.

6.4 Distributing Random Points on the Surface of a Polyhedral Model

The initial distribution of random points representing the cells on the surface of the polyhedral model representing the animal is implemented with an algorithm presented by Turk [turk90]. First we pick a random triangle and second we pick a random point inside the triangle. The random triangle is selected based on the normalized summed areas of all triangles, that is, we build an ordering of the triangles such that the last one will have partial summed area of 1.0. Given a random number between $[0, 1]$ we can now select the triangle which has the largest summed area less than or equal to the given random number. This guarantees that the probability of a triangle to be chosen is proportional to its area. Once a random triangle is chosen we use two more random values to select a random point inside that triangle. The problem is reduced to finding a random point on the triangle by mapping the 2D square space of possible random values ($[0, 1] \times [0, 1]$) to the geometrical space of the triangle. The final random point on the triangle will be defined by the barycentric

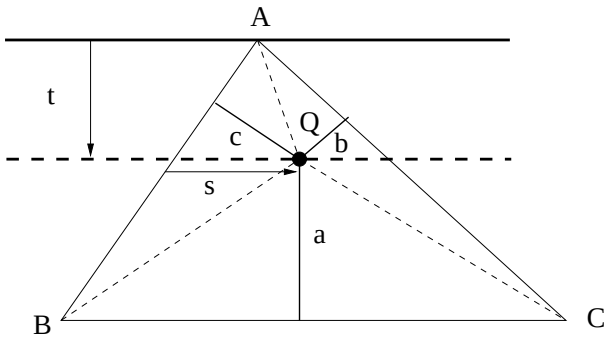


Figure 6.2: Finding a random point on a triangle

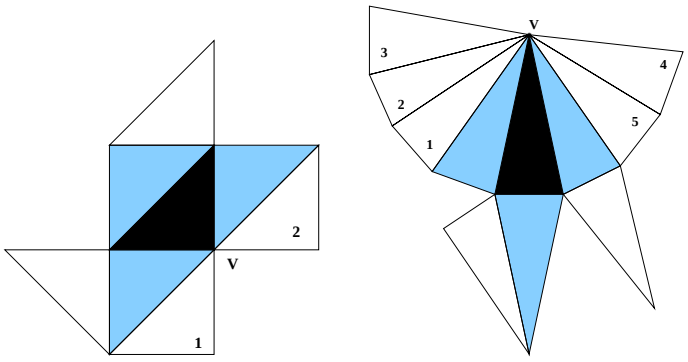
coordinates for the point. Given a triangle with vertices A, B, and C, and given two random values s and t, the random point Q can be computed as $Q = aA + bB + cC$ where $a = 1.0 - \sqrt{t}$, $b = (1.0 - s)\sqrt{t}$, and $c = s\sqrt{t}$ (see Fig. 6.2).

6.5 Relaxation of Points on the Surface of the Model

The relaxation process used to maintain cell size (see Section 3.3.3) has to deal with the cells defined on the surface of the model. We have to be able to compute distances on the surface of the model between cells that are neighbors. Ideally, all faces which have cells within the area defined by the repulsive radius r should be considered. This would imply an arbitrary, possibly large, number of neighboring faces. To avoid this cost we have restricted the search for neighbors among the faces which share either an edge (called *primary* neighbors) or a vertex (called *secondary* neighbors) with the face in question. In most practical cases this limitation does not significantly affect the results since the ratio between the number of cells versus number of faces guarantees that all neighboring cells are living in either primary or secondary faces. Ideally, we would have to guarantee that all cells in tertiary faces (not primary

neither secondary neighbors) are at a distance greater than the radius of repulsion. If there is a minimum angle for the triangles and a minimum edge length then one can show that there is a minimum distance between any point of the face in question and any tertiary face.

The distances are computed on the plane of the face where the cell lives. Therefore we need a way of mapping all neighboring cells to this plane. For each pair of faces P and N that share an edge we precompute the two rotation matrices $M_{P->N}$ and $M_{N->P}$ which bring a point defined on the plane of P to the plane of N and vice-versa. For primary neighbors the mapping is trivial using the precomputed matrices. For secondary neighbors we use a sequence of rotations around the edges which define a path between the face in question and the secondary neighbor. Figure 6.3 illustrates the idea. The dark grey face is the face in question, the light



(a) Even number of secondary neighbors

(b) Odd number of secondary neighbors

Figure 6.3: Computing distances on the surface of the model

grey ones are the primary neighbors and the white ones are the secondary neighbors. If the number of secondary neighboring faces at a vertex V is even, we map half of

them using a clockwise path of rotations around the common edges and the other half using a counterclockwise path (Figure 6.3(a)). If the number is odd we map half plus one faces (approximated to the closest integer) using a clockwise path of rotations and the other faces using a counterclockwise path (Figure 6.3(b)).

In order to update the cell's position due to the relaxation forces, cells travel freely and can eventually move to another face. When a cell changes face we find which edge the cell crossed and using the precomputed rotation matrices we bring the cell's position onto the plane of the new face. This process is repeated until the cell rests on some face, as illustrated in Figure 6.4.

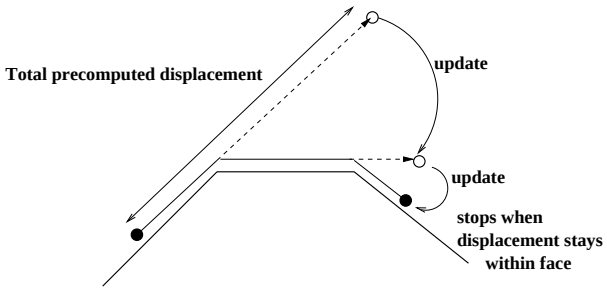


Figure 6.4: Mapping cells from face to face

The amount of traveling a given cell undergoes is arbitrary and for some geometries it is possible that the cell returns to its original face.

6.6 Computing the Voronoi Diagram on a Surface

Okabe [okab92] defines a *polyhedral Voronoi diagram* as a Voronoi diagram where the sites are defined on the surface of a polyhedron and the distances are measured on this surface. Computing the exact polyhedral Voronoi diagram can be quite com-

putationally expensive since for any two arbitrary sites on the surface there are many possible paths and finding the shortest one is a difficult problem [fran85].

For this reason, some solutions to this problem make use of approximations to the real Voronoi. The solution proposed by Mount [moun85], for instance, computes a polyhedral Voronoi diagram where the sites are used to create new polygons and a path between two sites always goes through edges of the polyhedron. A specific solution for spheres is presented by Augenbaum and Peskin [auge85]. The Voronoi diagram is recursively constructed adding one point at a time. The Voronoi polygons in this case are "…convex spherical polygons which overlap at most by having one edge in common".

We use an approximation for the actual Voronoi diagram where the final Voronoi diagram for the whole polyhedron will be the combination of the individual Voronoi diagrams computed for each face comprising the polyhedron. Turk used a similar idea in his Reaction-Diffusion work [turk91] but in his case he did not need to explicitly maintain the diagram since it was only used indirectly to establish diffusion amounts between cells. In our case the pattern is defined by the Voronoi diagram and therefore we need to build and maintain the Voronoi cells as a whole. The deviation from an exact computation of a Voronoi diagram is not critical in our case for two reasons. First, the pattern is defined through a large number of cells per triangle, which means that the approximated solution is correct for all cells but possibly the ones that are closer to the edges of the face. Second, the pattern is defined by the overall combination of many cells, which possibly spread over many faces. It is possible that the cells with Voronoi errors are "inside" a given pattern element and

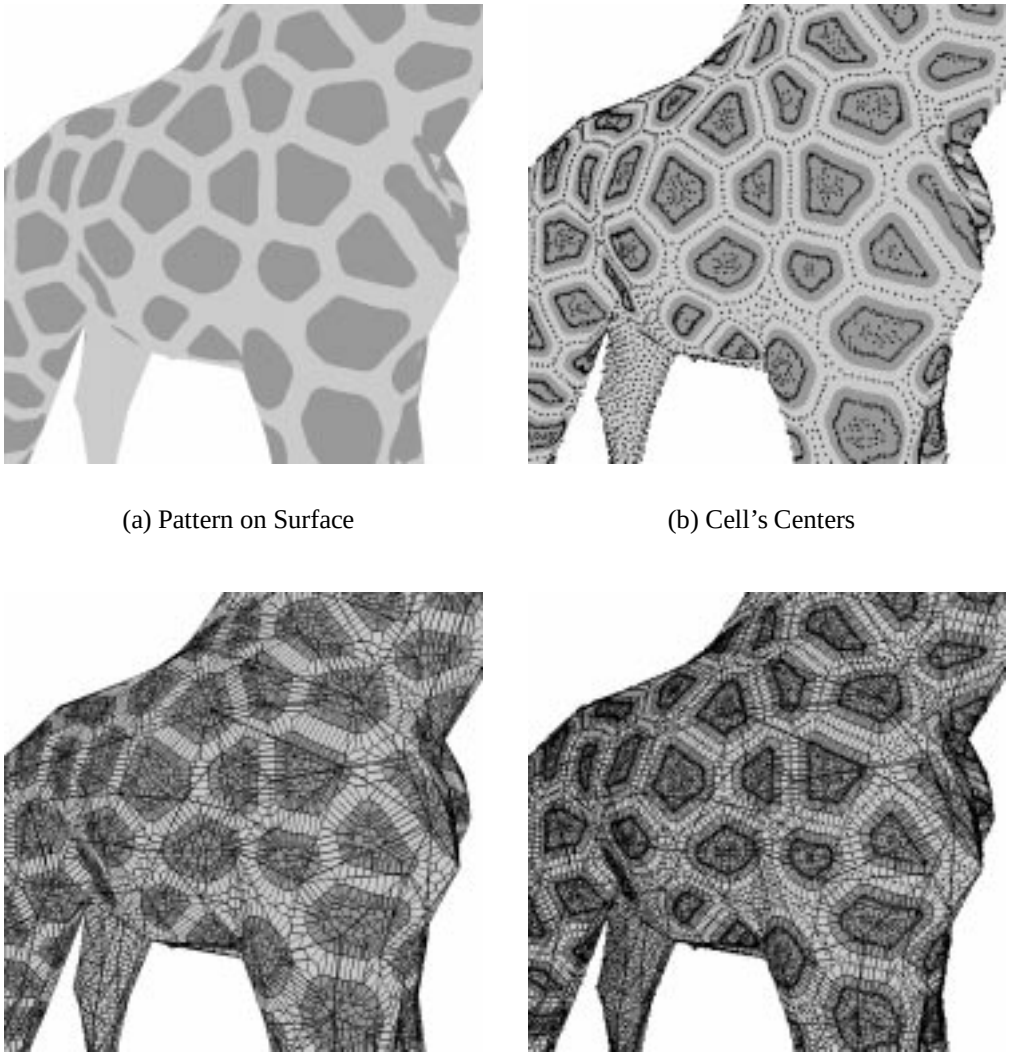
therefore are not individually visible.

In order to compute the Voronoi diagram for all cells resting on a single face we map all neighboring cells to the plane of the face in question using the approach explained above in Section 6.5. With all cells on the same plane we compute the Voronoi diagram on this plane. The Voronoi polygons are then clipped against the edges that define the face. Figure 6.5 illustrates this process. In (a) we show the end result with the overall pattern; in (b) we show only the positions of the cells which are the sites for the Voronoi diagram computation; in (c) we show only the individual Voronoi polygons and in (d) we show both cell's centers and the Voronoi borders. There are two special simpler cases that we should mention but in practice are often negligible. When a face has zero or one cell only, it means that the Voronoi polygon for this face is the face itself.

6.7 Pattern generation without growth

Although the Clonal Mosaic model was designed to work in conjunction with a shape changing geometry, we can also use it as a pattern generation mechanism on a static body model. The pattern formation process in this case is driven by pre-defined parameters, that is, splitting rates of cells are not computed from growth information. Figures 6.6 and 6.7 show the results of simulating a reticulated pattern on the surface of a cube and of a giraffe model.

Once a pattern is computed, we can still apply the growth transformation in order to obtain a newborn giraffe in this case with the same pattern as the adult (also shown in Figure 6.7). Notice that the two patterns are the same. In order to adapt the



(a) Pattern on Surface

(b) Cell's Centers

(c) Voronoi Borders

(d) Cell's Center and Borders

Figure 6.5: Pattern on the surface

Figure 6.6: Example of pattern generation without growth - Cube

Param.	ρ	wr	time	w_d	mitosis F	mitosis B	α FF	α BB	number of cells	spot area
Value	10	2.4	50	0.1	10	150	0.9	0.6	B=2559 F=6883	72.9

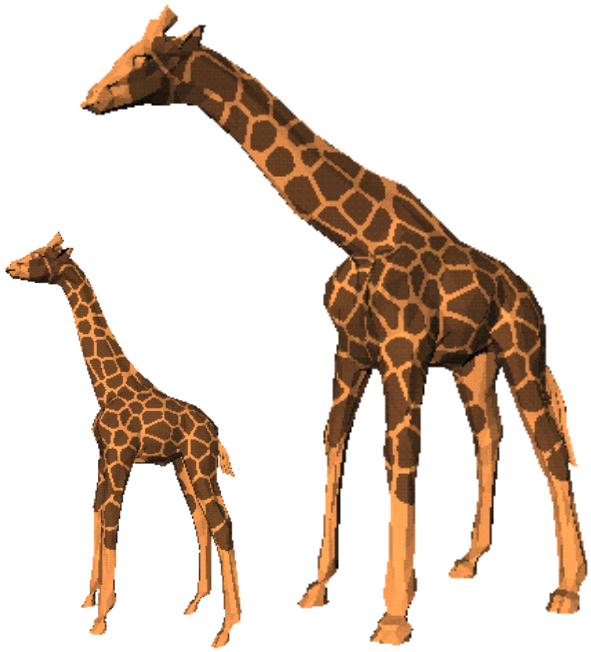


Figure 6.7: Example of pattern generation without growth - Giraffe

Param.	ρ	wr	time	w_d	mitosis F	mitosis B	α FF	α BB	number of cells	spot area
Value	18	2.4	80	0.067	10	150	0.9	0.6	B=8833 F=36346	80.5

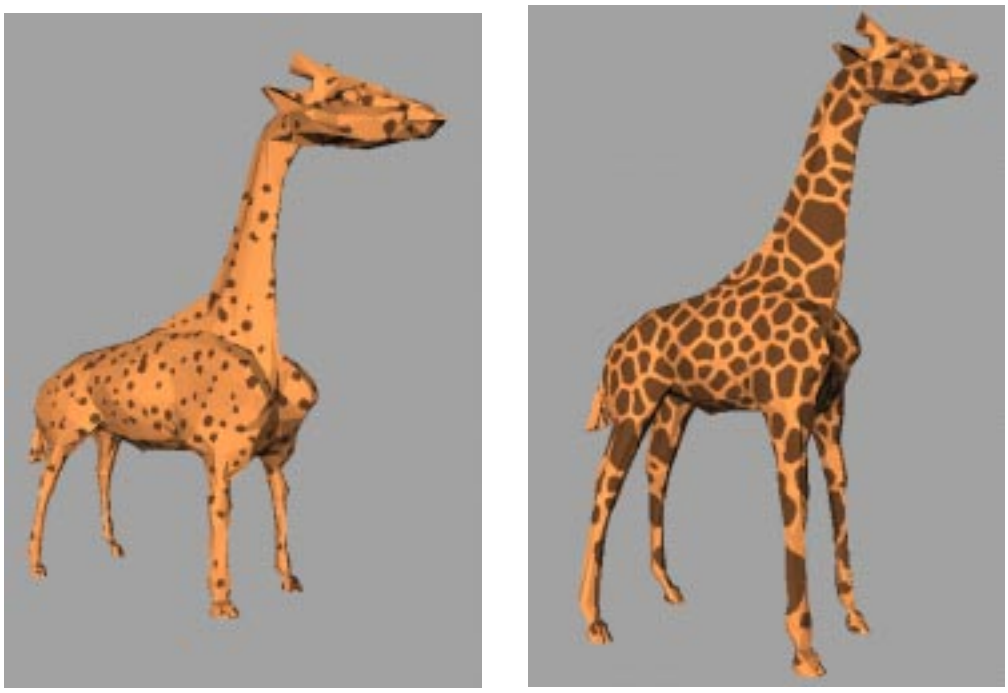
pattern computed for an adult body to a newborn body we apply the same growth transformation applied to the adult model to all cells and re-compute the Voronoi. It is important to note that if a pattern is close to a Voronoi diagram at birth, and some parts of the body grow anisotropically (such as the neck of the giraffe, which grow 4 times in length while it grows 3 times in diameter), then the pattern cannot remain a Voronoi pattern. We have verified that numerically [four98].

6.8 Pattern generation with growth

In this section we show the results of simulating the Clonal Mosaic model with splitting rates being computed from the growth information. The growth information for the giraffe was derived from the set of pictures presented in Figures 5.1, 5.2, and 5.3. Figure 6.8 shows two phases in the development of the giraffe pattern on the fetus at 35 days (start of pattern development) and 85 days.

6.9 Extra control

The structure of cylinders built to apply the growth data to the 3D model can also be used to control parameters during the simulation of the Clonal Mosaic model. For each cylinder we attach textures (any arbitrary image) which control one or more parameters. The attachment of textures to cylinders is defined via a texture file. The operational details about the format and use of this file are given in the next chapter. Figure 6.9 for example, shows a result where this structure was used to control which body areas would receive or not foreground cells and also splitting rates for cells.



(a) Pattern at 35 days

(b) Pattern at 85 days

Figure 6.8: Two phases in the development of pattern on the growing fetus

In the figure we can see that the head, tail and the lower part of legs do not have spots. By controlling the splitting rates of cells we can have spots of different sizes in different body areas. In giraffes, for example, the spots in the trunk are usually broader than the ones in the legs. In this example we decreased the splitting rates of cells living in triangles inside the cylinders attached to the upper part of the legs.

Figure 6.9: Extra control

6.10 Summary

This chapter presented how we can simulate the Clonal Mosaic model on an arbitrary shape taking into account changes in shape due to, for example, growth. In order to integrate the development of the patterns with the growth of the model, the splitting

rates of cells are derived from the growth information driving the changes in shape.

Chapter 7

Architecture of the System

The main goal of this chapter is to present the software tools built to explore and test the ideas and concepts introduced in this thesis. We call the whole system **ONÇA**, the Portuguese name for the Jaguar, one of the patterned big cats whose main habitat is "…well-watered areas, such as the swampy grasslands of the Brazilian Pantanal" [seid91]. **ONÇA** is a testbed for the exploration involved in evaluating results. In figure 7.1 we present a schematic architecture for the system. The system is composed of three main modules which reflect the three conceptual parts we divided our work into. The figure shows the objectives of each module and in general terms what kind of information they will be communicating to each other in order to function as a whole. We now proceed to explain each one in detail.

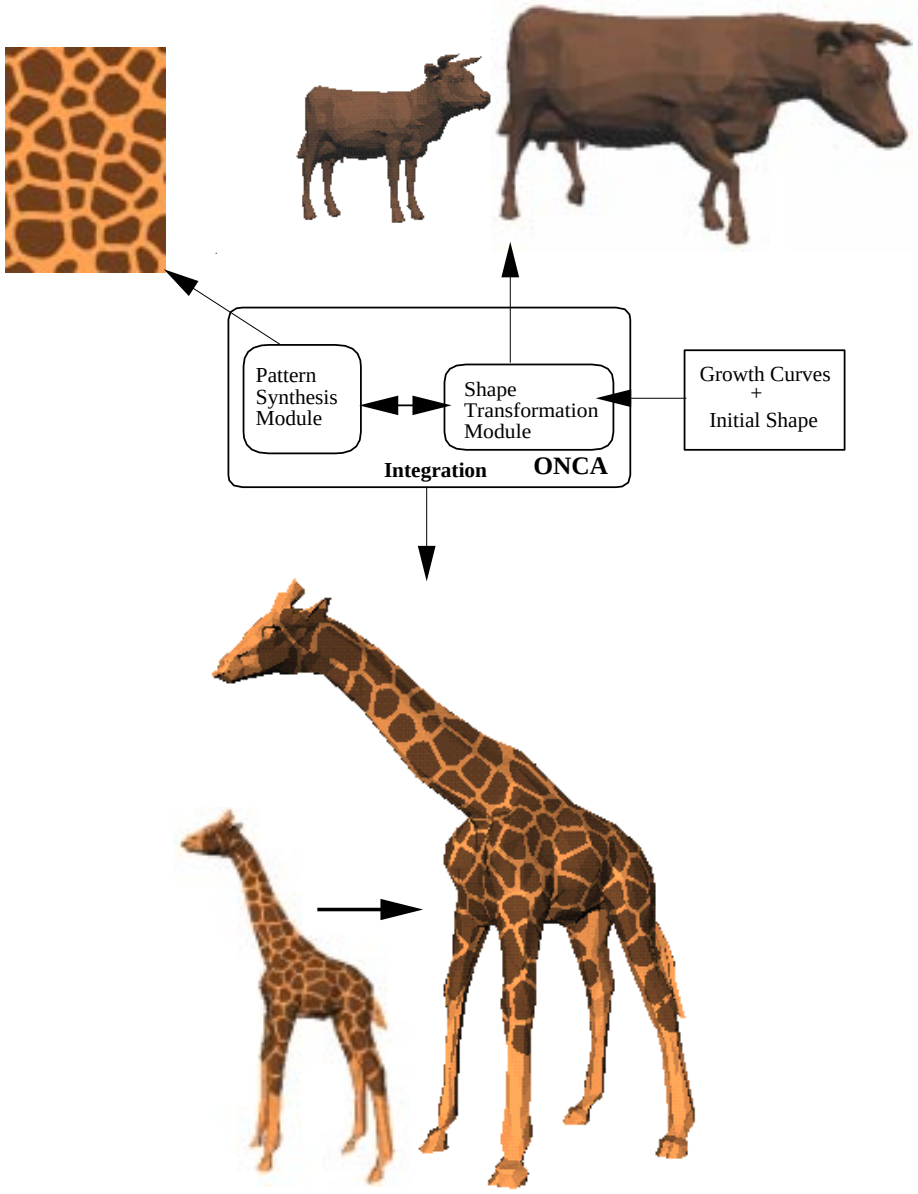


Figure 7.1: Architecture of the system

7.1 Pattern Synthesis Module

This module implements the Clonal Mosaic model presented in Chapter 3 and it has two main objectives:

- To produce a pattern expressed as a 2D image in a regular domain.
- To validate a set of parameters necessary to achieve a given pattern. This set will be input information for the integration module.

The main tool used to generate these 2D patterns is called `cm` and it runs both with a Graphical User Interface (GUI) or as a stand alone application. In Figure 7.2 we show the GUI for this tool.

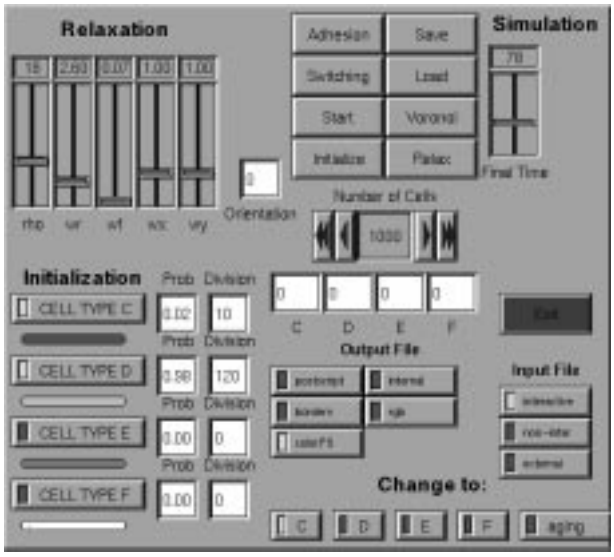


Figure 7.2: Graphical User Interface for `cm`

Most of the parameters are self-explanatory and are related to the parameters defined for the Clonal Mosaic Model presented in Chapter 3. There are two possible

input files for a given simulation: an input *parameter* file (extension `.par`) and an input *cells* file (extension `.out`). The parameter file defines all necessary parameters for a given simulation. The format of this file is given below:

```
;;;; (x,y) domain size
<float> <float>
;;;; Number of relaxation steps for each time step
<integer>
;;;; Weight for repulsive radius
<float>
;;;; Initial number of cells
<integer>
;;;; Final time
;;;; Future use
<integer> <integer>
;;;; Probabilities of being type C, D, E, and F (values between (0,1))
<float> <float> <float> <float>
;;;; Rates of division for cell types C, D, E, and F
<float> <float> <float> <float>
;;;; wf, wx, wy
<float> <float> <float>
;;;; Adhesion between types (values between (0,1))
<float> <float> <float> <float>
<float> <float> <float> <float>
<float> <float> <float> <float>
<float> <float> <float> <float>
;;;; Mutation probabilities between types (values between (0,1))
<float> <float> <float> <float>
<float> <float> <float> <float>
<float> <float> <float> <float>
<float> <float> <float> <float>
;;;; Color (R, G, B) for each cell type (values between (0,255))
<integer> <integer> <integer>
<integer> <integer> <integer>
<integer> <integer> <integer>
<integer> <integer> <integer>
;;;; The first integer value specifies which files to
;;;; output: 0 - postscript, 1 - internal file format, 2 - SGI rgb image
;;;; 3 - Both postscript and internal
;;;; The second integer value  specifies whether or not
;;;; the borders between cells should be drawn (1 - TRUE, 0 - FALSE)
;;;; The last integer value specifies whether or not color postscript
;;;; should be output (1 - TRUE, 0 - FALSE)
<integer> <integer> <integer>
;;;; Angle (degrees) for anisotropic motion
<float>
```

```
;;;; Name of the output postscript file
<fileName>.ps
;;;; Name of the output cell's file
<fileName>.out
```

Example of a parameter file:

```
200 200
18
2.6
1000
78 10
0.02 0.98 0.0 0.0
10 120 0 0
0.066 1.0 1.0
0.9 0.0 0.0 0.0
0.0 0.2 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
1.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 0.0 1.0
104 60 30
250 152 76
255 255 255
255 255 255
3 0 1
0
reticulata.ps
reticulata.out
```

The second possible input file is the cells file. This file contains the definition of a given number of cells (specified through their centers in 2D coordinate space) together with their type. The final result of the simulation can be saved both as a postscript file and as a cells file. The format of the cells file is as follows:

```
;;;; Total number of cells
<integer>
;;;; Cell type (x,y) position for the cell
<integer> <float> <float>
...
```

The program accepts two command-line flags corresponding to the two possible input files. The flag `-p` should be followed by the name of the parameter file while the flag `-e` should be followed by the name of the input cells file. A full invocation of the `cm` program looks like:

```
cm -p reticulata.par -e init.out > reticulata.log
```

The program outputs a log file that presents a summary of the simulation parameters together with other monitoring functions such as total time taken for the simulation.

7.2 Shape Transformation Module

The main goal of this module is the controlled transformation of shape through time. This module accepts two input files: the object to be transformed (extension `.obj`, Wavefront object file format [tech91]) and a *primitive* file (extension `.prim`). The primitive file specifies the cylinders and features which control the transformation, together with the two files which contain the growth data and animation information. The format of the primitive file is as follows:

```
;;;; Name of file with growth information
<filename>
;;;; Name of file with animation information
<filename>
;;;; Number of Primitives in this file (cylinders plus features).
;;;; Primitive number 0 is reserved to store information about the
;;;; camera positioning
<integer>
;;;; Type of Primitive (0 - Cylinder; 1 - Feature)
<integer>
;;;; (x,y,z) location of the pivot point
<double> <double> <double>
```

```
;;;; (Rx, Ry, Rz) rotation values in degrees around the x,y, and z-axis
<double> <double> <double>
;;;; (Sx, Sy, Sz) scale values for the x,y, and z dimensions
<double> <double> <double>
;;;; The next information depends whether the primitive
;;;; is a cylinder or a feature
;;;;
;;;; If it is a Cylinder
;;;; Pointer to which 2 features control this cylinder's
;;;; radius and height
<integer> <integer>
;;;; Pointer to the parent primitive (-1 flags the Master primitive)
<integer>
;;;;
;;;; If it is a Feature
;;;; Pointer to a column in the growth array (filename specified above)
;;;; that contains, for the various ages, the dimensions for this feature
<integer>
;;;; Pointer to which cylinder this landmark controls
<integer>
...
...
```

The growth and animation process are driven by two files with growth and animation information. The format of these two files is as follows:

- Growth information file

```
;;;; Growth information file
;;;; Keyword 'size' followed by 2 integers specifying
;;;; how many rows and how many columns of growth data
;;;; there is
size <integer> <integer>

;;;; Keyword 'data' followed by a series of floats
;;;; The first float specifies the month associated
;;;; with the row
data <float> <float> <float> <float> ...
....
;;;;
```

- Animation information file

```
;;;; Animation Information File
;;;; Keyword 'size' followed by 3 integers. The first
```

```
;;;; integer specifies if the angles specifying
;;;; the animation are given in relative (value = 0)
;;;; or absolute (value = 1 ) terms
;;;; The next two integers specify how many rows
;;;; and how many columns of data this file has
size <integer> <integer> <integer>

;;;; Keyword 'data' followed by two integers and
;;;; a variable number of floats with the angles
;;;; (in degrees) information. The first integer
;;;; specifies the cylinder ID which will receive
;;;; the animation information; the second integer
;;;; specifies the axis of rotation. Possible values
;;;; are 0 (x axis), 1 (y axis) and 2 (z axis)
;;;; The first float also has a special meaning. It
;;;; gives the angle position for the cylinders
;;;; 'at rest', that is, their original position
data <integer> <integer> <float> <float> <float> ....
...
```

7.3 Integration Module

The last module is the integration module. This module simulates the growth of the animal together with the development of the pattern. This module is, in a sense, a combination of the other two previously presented. We have a GUI-based version and a command-line version. Table 7.1 summarizes the files that can be used when running the module. There is a required file (the name of the model) and four optional files. The object file must be given in Wavefront object file format [tech91]. Both GUI-based and command-line versions accept the same number of input files. Here we give a few examples of running the command-line version of the `onca` program:

- The object file and the parameter file

  ```
  onca -f gir2000.obj -p reticulata.par > reticulata.log
  ```

Type	Extension	Flag
Object file	`.obj`	`-f`
Primitives file	`.prim`	`-m`
Parameter file	`.par`	`-p`
Cells file	`.cm`	`-e`
Texture file	`.txtr`	`-t`

Table 7.1: Specification for input parameter files for the `onça` tool

- The object file and the cells file (a file saved from a previous simulation)

  ```
  onca -f gir2000.obj -e reticulata.cm
  ```

The primitive file has the same format specified before for the shape transformation module. We give now the description for the parameter file, the cells file and the texture file.

7.3.1 Parameter file

The format of the parameter file is as follows:

```
;;;; Number of relaxation steps for each time step
;;;; Initial number of relaxations
<integer> <integer>
;;;; Weight for repulsive radius
<float>
;;;; Initial number of cells
<integer>
;;;; Final time
;;;; Pattern Formation Mode (0 - with growth; 1 - No growth)
;;;; Type of Random Distribution (0 - none; 1- Poisson; 2 - Exponential)
;;;; Future use
<integer> <integer> <integer> <integer>
;;;; Probabilities of being type C, D, E, and F (values between (0,1))
<float> <float> <float> <float>
;;;; Rates of division for cell types C, D, E, and F
<float> <float> <float> <float>
;;;; wf, wx, wy, initial wf
<float> <float> <float> <float>
;;;; Adhesion between types (values between (0,1))
<float> <float> <float> <float>
<float> <float> <float> <float>
<float> <float> <float> <float>
```

```
<float> <float> <float> <float>
;;;; Mutation probabilities between types (values between (0,1))
<float> <float> <float> <float>
<float> <float> <float> <float>
<float> <float> <float> <float>
<float> <float> <float> <float>
;;;; Color (R, G, B) for each cell type (values between (0,1))
<float> <float> <float>
<float> <float> <float>
<float> <float> <float>
<float> <float> <float>
;;;; Name of the output cell's file
<fileName>.cm
```

7.3.2 Cells file

This file contains the definition of a given number of cells (specified through their centers in 3D coordinate space) together with their type. The final result of the simulation is usually saved as a cells file. The format of the cells file is as follows:

```
;;;; Name of the object file associated with this file
<filename>.obj
;;;; Face ID 0 - Number of cells in this face
<integer> <integer>
;;;; Cell type - (x, y, z)  position of this cell
<integer> <float> <float> <float>
...
...
;;;; Face ID 1 - Number of cells in this face
<integer> <integer>
;;;; Cell type - (x, y, z)  position of this cell
<integer> <float> <float> <float>
...
```

7.3.3 Texture file

For each cylinder, we can associate one or more texture files that provide extra control over some simulation parameters. The **onça** texture file specifies which cylin-

ders have textures associated (in `rle` format) and what parameters are these textures controlling. The format of the texture file is as follows:

```
;;;; Texture File
;;;; Keyword CYLINDERID followed by the number which specifies
;;;; the cylinder and another integer which specifies how
;;;; many textures are associated with this cylinder
CYLINDERID <integer> <integer>

;;;; Keyword specifying which parameter is controlled and
;;;; the name of the texture file
;;;; The possible keywords are:
;;;; EXISTC - Controls the creation of cells of type C
;;;; EXISTD - Controls the creation of cells of type D
;;;; SPLITRATEC - Controls the splitting rate of cells type C
;;;; SPLITRATED - Controls the splitting rate of cells type D
;;;; COLORC - Controls the color of cells type C
;;;; COLORD - Controls the color of cells type D
;;;; REPRAD - Controls the value of the repulsive radius
;;;; FORCE - Controls the force of repulsion between cells
;;;; ADHESION - Controls the adhesion between cells
KEYWORD <textureFileName>.rle
....
....
```

7.4 Summary

This chapter presented the main computational tools developed as exploration tools for the ideas presented in this thesis. We presented the main features of the tools together with their input files and parameters.

Chapter 8

Conclusions

The work in this thesis has addressed the complex problem of integration between the visual and the shape attributes of an object, which is particularly challenging in the context of natural objects. Current solutions to this problem use a mapping step to attach visual properties to the object's surface. For a complicated shape and visual properties that are not regularly distributed over the surface, the mapping step requires a great deal of expertise and manual fine-tuning, and lacks flexibility for many practical applications.

The major goal of the research presented in this thesis was the exploration of an alternative to current solutions for the integration problem. Within the domain of patterned animals, we presented an integrated solution where the pattern is generated as a surface-level process driven by a shape changing geometry. Related to the main goal are the subgoals of providing more flexible solutions in the context of pattern generation methods in graphics and providing a technique to change shapes in a controlled way.

8.1 Contributions

These are what we feel are the main contributions of the research work presented in this thesis:

- **Clonal Mosaic Patterns:** The modelling of patterns is as complex a task as the modelling of shape. In some cases having the right pattern is more important than having the right shape. We introduced a pattern formation model specialized in the generation of patterns found in many species of mammals, particularly the big cats and the giraffe. The model is biologically plausible and proposes that the fur pattern reflects an underlying arrangement of cells. The model simulates the formation of this arrangement through cells of different types. A wide range of realistic-looking patterns is possible through manipulation of input parameters such as adhesion between cells and splitting rates of cells. We proposed a metric for validation of the giraffe patterns based on their similarity with Voronoi diagrams. The results show that both the real giraffe pattern and the Clonal Mosaic patterns are close to a real Voronoi diagram.

- **Shape Control:** We presented a technique to transform a given shape in a controlled manner. The algorithm allows for simulation of locally defined changes, such as a body growing at different rates in different parts. It also allows for simulation of animation in the traditional sense (e.g. a horse trotting). The control of the transformation is provided by a set of features locally defined for each body part that we want to individually control.

- **Integration of Shape and Pattern:** The Clonal Mosaic model was extended to work on a shape changing 3D geometry in an integrated manner. The pattern formation process can therefore be controlled by changes affecting the geometry. We presented the results for the giraffe.

Finally, on a conceptual level, this research work tried to advance an integrated approach solution for problems where current techniques are not generic enough or not automated enough.

8.2 Future Work

We present a few topics for future investigation in the context of this thesis. The topics are grouped according to the three main parts we divided our work into: Clonal Mosaic Patterns, Shape Transformation and Integration.

8.2.1 Clonal Mosaic Patterns

1. **Wider exploration of the parameter space**

   Although the model was designed to produce mammalian coat patterns, we would like to explore the model further for producing other patterns as well, even unnatural ones. In Figure 8.1 we show an example of a simple pattern that can easily be generated with the Clonal Mosaic approach and was not an original goal of the method.

2. **Genetic Validation**

   Genetic research has established genetic distances between species in the same

family. For the *Felidae* family [o'br86] , we would like to investigate a possible correlation between distances in the parametric space and in the genetic space. A good correlation would strength the validity of the model.

3. **Control of results**

   The current implementation of the Clonal Mosaic model has not addressed the important problem of helping the user in selecting parameters for a desired type of pattern. To turn the system into a more useful tool we have to address this problem.

4. **Simulating RD systems with CM**

   We presented an argument on how to simulate an RD system with an equivalent CM system. We need to map the parameters from one system into another and show examples of interesting RD patterns simulated by CM. We also need to further explore whether some CM patterns cannot be simulated by RD systems (without external intervention for anisotropy or to prevent mixing of reactants).

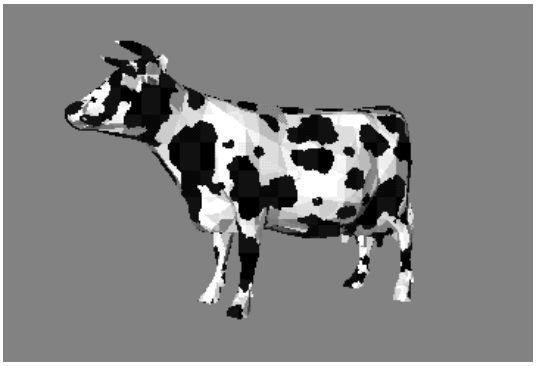


Figure 8.1: Exploration of other patterns

8.2.2 Growing Models of Animals

An interesting application for the technique presented in Chapter 5 is to interpolate between related animal species, in particular to help in the reconstruction of bodies of extinct species in paleontology. The visual evolution of a horse from its biological ancestors could be built using an initial horse model together with bone measurements from extinct horse-related ancestors. Current techniques usually build a clay life-size model from the incomplete set of bones.

Another possibility is to apply the control provided by the local coordinate systems as a modelling tool. The technique we presented in Chapter 5 could be used to customize polygonal models, according to user-defined specification (or other sources of information, such as pictures), providing more flexibility for using existent models. We can imagine a barn full of different cows, where each individual model was derived from a basic standard off-the-shelf model. Also, we have not yet explored the full combination of effects possible with the tool, such as the simulation of deformation induced by muscle contraction and stretching or the motion of a cow's belly when trotting. These could be simulated using non-affine transformations that can be applied before the growth process or the relative motion and which could provide for increased levels of realism when simulating animal gaits.

Finally, the local cylindrical spaces used to transfer growth and animation information could also be used inside a classical texture map approach. It consists in defining for every cylindrical coordinate system 2D textures coordinates obtained from the cylindrical coordinates. This is similar to the technique described by Bier and Sloan [bier86] as two-part texture mapping. The difficulty lies in the proper

blending of the texture coordinates given within overlapping cylinders.

8.2.3 Integration

We presented the giraffe as a case study for the ideas presented in this thesis. The Clonal Mosaic model, however, can generate the full range of patterns from the *Felidae* family as showed in the results section of Chapter 3 and therefore we would like to apply the model to simulate other animals, such as the tiger, cheetah, and leopard. Another subject that deserves attention is the proper simulation of details that would improve the overall level of realism, such as the pattern on the face of the animals. We believe that the cylinders can also be used for obtaining these type of effects. Finally, we have not touched issues related to high-quality rendering the models. We would like, for instance, to add to our images the realism provided by a proper technique for fur rendering, using the Clonal Mosaic patterns as the underlying colouring information for the fur.

Bibliography

[acke1b] A. Ackerman. “The Histogenesis of hair follicles in the zebra and giraffe with special reference to pigmentation and cutaneous vasculature”. M.Sc. Thesis, University of Pretoria, 1976 (citation from [Murray1981b)].

[arad97] N. Arad and G. Elber. “Isometric Texture Mapping for Free-form Surfaces”. *Computer Graphics Forum*, Vol. 16, No. 5, pp. 247–255, Dec 1997.

[auge85] J. M. Augenbaum and C. S. Peskin. “On the Construction of the Voronoi Mesh on a Sphere”. *J. Computational Physics*, Vol. 59, pp. 177–192, 1985.

[bard77] J. B. L. Bard. “A unity underlying the different zebra striping patterns”. *Journal of Zoology*, Vol. 183, pp. 527–539, December 1977.

[bard81] J. B. L. Bard. “A Model for Generating Aspects of Zebra and Other Mammalian Coat Patterns”. *Journal of Theoretical Biology*, Vol. 93, No. 2, pp. 363–385, November 1981.

[baum72] B. G. Baumgart. “Winged Edge Polyhedron Representation”. Technical Report STAN-CS-320, Computer Science Department, Stanford University, 1972.

[bedd06] E. Beddard. “Description of the External Characters of an unborn Fetus of a Giraffe”. *Journal of Zoology*, pp. 626–631, 1906.

[beie92] T. Beier and S. Neely. “Feature-based image metamorphosis”. *Computer Graphics (SIGGRAPH ’92 Proceedings)*, Vol. 26, pp. 35–42, July 1992.

[benn91] C. Bennis et al. “Piecewise surface flattening for non-distorted texture mapping”. *Computer Graphics (SIGGRAPH ’91 Proceedings)*, Vol. 25, pp. 237–246, July 1991.

[bert78] B. Bertram. *Pride of Lions.* J. M. Dent & Sons, 1978.

[bier86] E. A. Bier and K. R. Sloan, Jr. “Two Part Texture Mappings”. *IEEE Computer Graphics and Applications*, Vol. 6, No. 9, pp. 40–53, September 1986.

[blin78] J. F. Blinn. “Simulation of wrinkled surfaces”. *Computer Graphics (SIGGRAPH ’78 Proceedings)*, Vol. 12, No. 3, pp. 286–292, August 1978.

[blin82] J. Blinn. “A Generalization of Algebraic Surface Drawing”. *ACM Transactions on Graphics*, Vol. 1, No. 3, pp. 235–256, 1982.

[bloo97] J. Bloomenthal. *Introduction to Implicit Surfaces.* Morgan Kaufmann, 1997.

[bohm84] W. Bohm, G. Farin, and J. Kahmann. "A Survey of Curve and Surface methods in CAGD". *Computer Aided Geometric Design*, Vol. 1, pp. 1–60, November 1984.

[brod64] S. Brody. *Bioenergetics and growth.* New York, Hafner Pub. Co., 1964.

[burt76] N. Burtnyk and M. Wein. "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation". *Communications of the ACM*, Vol. 19, pp. 564–569, 1976.

[catm74] E. E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* Ph.D. Thesis, University of Utah, December 1974.

[chad89] J. E. Chadwick, D. R. Haumann, and R. E. Parent. "Layered Construction for Deformable Animated Characters". *Computer Graphics (SIGGRAPH '89 Proceedings)*, Vol. 23, pp. 243–252, July 1989.

[cham95] P. Chambers and A. Rockwood. "Visualization of Solid Reaction-Diffusion Systems". *IEEE Computer Graphics and Applications*, Vol. 15, No. 5, pp. 7–11, September 1995.

[chan94] Y. Chang and A. P. Rockwood. "A Generalized de Casteljau Approach to 3D Free–Form Deformation". *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pp. 257–260, July 1994.

[chen95] D. Chen, A. State, and D. Banks. "Interactive Shape Metamorphosis". *1995 Symposium on Interactive 3D Graphics*, pp. 206–215, April 1995.

[coch87a] G. Cocho, R. Pérez-Pascual, and J. L. Rius. “Discrete Systems, Cell-Cell Interactions and Color Pattern of Animals - Conflicting Dynamics and Pattern Formation”. *Journal of Theoretical Biology*, Vol. 125, No. 4, pp. 419–435, April 1987.

[coch87b] G. Cocho, R. Pérez-Pascual, and J. L. Rius. “Discrete Systems, Cell-Cell Interactions and Color Pattern of Animals - Clonal Theory and Cellular Automata”. *Journal of Theoretical Biology*, Vol. 125, No. 4, pp. 437–447, April 1987.

[corm90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, second edition, 1990.

[cunn61] K. Cunningham and S. H. Fowler. “A Study of Growth and Development in the Quarter Horse”. Technical Report 546, Lousiana State University - Agricultural and Mechanical College, November 1961.

[curl96] B. Curless and M. Levoy. “A Volumetric Method for Building Complex Models from Range Images”. *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pp. 303–312, August 1996.

[dagg68] A. I. Dagg. “External Features of Giraffe”. *Mammalia*, Vol. 32, pp. 657–669, 1968.

[dagg76] A. I. Dagg and J. B. Foster. *The Giraffe: its Biology, Behavior, and Ecology*. Van Nostrand Reinhold Co., 1976.

[de B97] J. S. de Bonet. "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images". *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pp. 361–368, August 1997.

[fari90] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1990.

[flei95] K. Fleischer, D. Laidlaw, B. Currin, and A. Barr. "Cellular Texture Generation". *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pp. 239–248, August 1995.

[fole90] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, second edition, 1990.

[fors88] D. R. Forsey and R. H. Bartels. "Hierarchical B-Spline Refinement". *Computer Graphics (SIGGRAPH '88 Proceedings)*, Vol. 22, pp. 205–212, August 1988.

[fors91] D. R. Forsey. "A Surface Model for Skeleton-Based Character Animation". *Eurographics Workshop on Animation and Simulation*, pp. 55–73, 1991.

[fors98] D. R. Forsey and D. Wong. "Multiresolution Surface Reconstruction for Hierarchical B-splines". *Graphics Interface*, pp. 57–64, June 1998.

[four95] A. Fournier. "Sampling and Filtering - Course Notes for Computer Graphics". Dept. of Computer Science, University of British Columbia, 1995.

[four98] A. Fournier and M. Walter. “A Quantitative Analysis of Reticulated Giraffe Spot Patterns”. To be submitted, 1998.

[fowl92] D. R. Fowler, H. Meinhardt, and P. Prusinkiewicz. “Modeling seashells”. *Computer Graphics (SIGGRAPH ’92 Proceedings)*, Vol. 26, No. 2, pp. 379–387, July 1992.

[fox60] H. M. Fox and G. Vevers. *The Nature of Animal Colours*. Sidgwick and Jackson Ltd., 1960.

[frac90] F. D. Fracchia, P. Prusinkiewicz, and M.J.M. de Boer. “Visualization of the Development of Multicellular Structures”. *Graphics Interface*, pp. 267–276, May 1990.

[frac95] F. D. Fracchia. “Integrating Lineage and Interaction for the Visualization of Cellular Structures”. *5th International Workshop on Graph Grammars and their Application to Computer Science*, 1995.

[fran85] W. R. Franklin, V. Akman, and C. Verrilli. “Voronoi Diagrams with Barriers and on Polyhedra for Minimal Path Planning”. *The Visual Computer*, Vol. 1, pp. 133–150, 1985.

[garl97] M. Garland and P. S. Heckbert. “Surface Simplification Using Quadric Error Metrics”. *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pp. 209–216, August 1997.

[gilb94] S. F. Gilbert. *Developmental Biology*. Sinauer Associates, Inc., 1994.

[goel78] N.S. Goel and G. Rogers. “Computer Simulation of Engulfment and Other Movements of Embryonic Tissues”. *Journal of Theorethical Biology*, Vol. 71, pp. 103–140, 1978.

[gord78] R. Gordon and A. G. Jacobson. “The Shaping of Tissues in Embryos”. *Scientific American*, pp. 106–113, june 1978.

[gord83] R. Gordon. “Computational Embryology of the Vertebrate Nervous System”. *Computing in Biological Science*. pp. 23–70. Elsevier Biomedical Press, 1983.

[hamm98] O. Hammer. “Diffusion and Direct Signaling Models are Numerically Equivalent”. *Journal of Theorethical Biology*, Vol. 192, pp. 129–130, 1998.

[hanr90] P. Hanrahan and P. Haeberli. “Direct WYSIWYG Painting and Texturing on 3D Shapes”. *Computer Graphics (SIGGRAPH ’90 Proceedings)*, Vol. 24, No. 4, pp. 215–223, August 1990.

[heck86] P. S. Heckbert. “Survey of Texture Mapping”. *IEEE Computer Graphics and Applications*, Vol. 6, No. 11, pp. 56–67, November 1986.

[heeg95] D. J. Heeger and J. R. Bergen. “Pyramid-Based Texture Analysis/Synthesis”. *Computer Graphics (SIGGRAPH ’95 Proceedings)*, pp. 229–238, August 1995.

[held92] L.I. Held. *Models for Embryonic Periodicity*. Karger, 1992.

[hemm79] H. Hemmer. “Gestation Period and Postnatal Development in Felids”. *Carnivore*, Vol. 2, pp. 90–100, 1979.

[hera76] I. Heran. *Animal Coloration.* Hamlyn, 1976.

[hond78] H. Honda. “Description of Cellular Patterns by Dirichlet Domains: The Two-Dimensional Case”. *Journal of Theoretical Biology*, Vol. 72, pp. 523–543, 1978.

[hong88] T. M. Hong, N. Magnenat-Thalmann, and D. Thalmann. “A general algorithm for 3D shape interpolation in a facet-based representation”. *Proceedings of Graphics Interface ’88*, pp. 229–235, June 1988.

[hopp94] H. Hoppe et al. “Piecewise Smooth Surface Reconstruction”. *Proceedings of SIGGRAPH ’94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pp. 295–302, July 1994.

[hopp96] H. Hoppe. “Progressive Meshes”. *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pp. 99–108, August 1996.

[hopp97] H. Hoppe. “View-Dependent Refinement of Progressive Meshes”. *SIGGRAPH 97 Conference Proceedings,* Annual Conference Series, pp. 189–198, August 1997.

[hsu92] W. M. Hsu, J. F. Hughes, and H. Kaufman. “Direct manipulation of free-form deformations”. *Computer Graphics (SIGGRAPH ’92 Proceedings)*, Vol. 26, No. 2, pp. 177–184, July 1992.

[hugh92] J. F. Hughes. “Scheduled Fourier volume morphing”. *Computer Graphics (SIGGRAPH ’92 Proceedings)*, Vol. 26, pp. 43–46, July 1992.

[jack91] I. J. Jackson. “Mouse Coat Colour Mutations: A Molecular Genetic Resource Which Spans the Centuries”. *BioEssays*, Vol. 13, No. 9, pp. 439–446, september 1991.

[kaci91] Z. Kacic-Alesic and B. Wyvill. “Controlled Blending of Procedural Implicit Surfaces”. *Proceedings of Graphics Interface ’91*, pp. 236–245, June 1991.

[kent91] J. Kent, R. E. Parent, and W. E. Carlson. “Establishing Correspondences by Topological Merging: A New Approach to 3-D Shape Transformation”. *Proceedings of Graphics Interface ’91*, pp. 271–278, June 1991.

[kent92] J. R. Kent, W. E. Carlson, and R. E. Parent. “Shape transformation for polyhedral objects”. *Computer Graphics (SIGGRAPH ’92 Proceedings)*, Vol. 26, pp. 47–54, July 1992.

[kesh88] L. E. Keshet. *Mathematical Models in Biology*. Random House, 1988.

[koen90] J. J. Koenderink. *Solid Shape*. MIT Press, 1990.

[kond95] S. Kondo and R. Asai. “A reaction-diffusion wave on the skin of the marine angelfish Pomacanthus”. *Nature*, Vol. 376, pp. 765–768, 1995.

[lant95] M. Lantin and F. D. Fracchia. “Generalized Context-Sensitive Cell Systems”. *IPCAT’95 - 1st International Workshop on Information Processing in Cells and Tissues*, 1995.

[leng91] I. Lengyel and I. R. Epstein. "Modeling the Turing Structures in the Chlorite-Iodite-Malonic Acid-Starch Reaction System". *Science*, Vol. 251, pp. 650–652, february 1991.

[leri95] A. Lerios, C. D. Garfinkle, and M. Levoy. "Feature-Based Volume Metamorphosis". *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pp. 449–456, August 1995.

[litw94] P. Litwinowicz and G. Miller. "Efficient Techniques for Interactive Texture Placement". *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pp. 119–122, July 1994.

[lore87] W. E. Lorensen and H. E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". *Computer Graphics (SIGGRAPH '87 Proceedings)*, Vol. 21, pp. 163–169, July 1987.

[lyon92] M. J. Lyons and L. G. Harrison. "Stripe Selection: An Intrinsic Property of Some Pattern-Forming Models with Nonlinear Dynamics". *Developmental Dynamics*, Vol. 195, pp. 201–215, 1992.

[mail93] J. Maillot, H. Yahia, and A. Verroust. "Interactive Texture Mapping". *Computer Graphics (SIGGRAPH '93 Proceedings)*, Vol. 27, pp. 27–34, August 1993.

[mark97] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and

S. Shieber. “Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation”. *SIGGRAPH 97 Conference Proceedings,* Annual Conference Series, pp. 389–400, August 1997.

[mein38] R. Meinertzhagen. “Some Weights and Measurements of Large Mammals”. *Proceedings of the Zoological Society*, No. 108 - Ser. A, pp. 433–439, 1938.

[mein82] H. Meinhardt. *Models of Biological Pattern Formation*. Academic Press, 1982.

[mein87a] H. Meinhardt and M. Klinger. “A Model for Pattern Formation on the Shells of Molluscs”. *Journal of Theoretical Biology*, Vol. 126, pp. 63–89, 1987.

[mein87b] H. Meinhardt and M. Klinger. “Pattern Formation by Coupled Oscillations: The pigmentation patterns on the shells of molluscs”. *Lecture Notes in Biomathematics,* Vol. 71, pp. 184–198, 1987.

[mein95a] H. Meinhardt. *The Algorithmic Beauty of Sea Shells*. Springer-Verlag, 1995.

[mein95b] H. Meinhardt. “Dynamics of Stripe Formation”. *Nature,* Vol. 376, pp. 722–723, 1995.

[mill91] J. V. Miller, D. E. Breen, W. E. Lorensen, R. M. O’Bara, and M. J. Wozny. “Geometrically deformed models: A method for extracting closed geo-

metric models from volume data". *Computer Graphics (SIGGRAPH '91 Proceedings)*, Vol. 25, No. 4, pp. 217–226, July 1991.

[mint74] B. Mintz. "Gene Control of Mammalian Differentiation". *Annual Review of Genetics*, Vol. 8, pp. 411–470, 1974.

[moun85] D. Mount. "On Finding Shortest Paths on Convex Polyhedra". Technical Report 1495, Department of Computer Science, University of Maryland, 1985.

[murr81a] J. D. Murray. "On Pattern Formation Mechanisms for Lepidopteran wing paterns and mammalian coat markings". *Philosophical Transactions of the Royal Society of London B*, Vol. 295, No. 1078, pp. 473–496, October 1981.

[murr81b] J. D. Murray. "A Pre-pattern Formation Mechanism for Animal Coat Markings". *Journal of Theoretical Biology*, Vol. 88, pp. 161–199, 1981.

[murr89] J. D. Murray. *Mathematical Biology*. Springer Verlag, 1989.

[muyb18] E. Muybridge. *Animals in Motion. An electro photographic investigation of consecutive phases of muscular actions*. Chapman & Hall, Ltd., 1918.

[need64] A. E. Needham. *The growth process in animals*. Sir Isaac Pitman & Sons LTD, 1964.

[o'br86] S. J. O'Brien et al. "Setting the Molecular Clock in Felidae: The Great Cats , Panthera". *Symposium Tigers of the World: The Biology, Biopoli-*

tics, Management, and Conservation of an Endangered Species, pp. 10–27, April 1986.

[odel81] G. M. Odell, G. Oster, P. Alberch, and B. Burnside. "The Mechanical Basis of Morphogenesis: I. Epitelial Folding and Invagination". *Developmental Biology*, Vol. 85, pp. 446–462, 1981.

[okab92] A. Okabe, B. N. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* Wiley & Sons, 1992.

[oste83a] G. F. Oster. "Mechanochemistry and Morphogenesis". *Biological Structures and Coupled Flows,* pp. 417–443. Academic Press, 1983.

[oste83b] G. F. Oster, J. D. Murray, and A. K. Harris. "Mechanical aspects of mesenchymal morphogenesis". *Journal of Embryology and Experimental Morphology*, Vol. 78, pp. 83–125, 1983.

[ouya91] Q. Ouyang and H. L. Swinney. "Transition from a uniform state to hexagonal and striped Turing patterns". *Nature*, Vol. 352, pp. 610–612, August 1991.

[owen49] R. Owen. "Notes on the Birth of the Giraffe at the Zoological Society Gardens". *Transactions of the Zoological Society of London*, Vol. 3, pp. 21–28, 1849.

[peac85] D. R. Peachey. "Solid Texturing of Complex Surfaces". *Computer Graphics (SIGGRAPH '85 Proceedings)*, Vol. 19, No. 3, pp. 279–286, July 1985.

[perl85] K. Perlin. “An Image Synthesizer”. *Computer Graphics (SIGGRAPH ’85 Proceedings)*, Vol. 19, No. 3, pp. 287–296, July 1985.

[prep85] F. Preparata and M. Shamos. *Computational Geometry - An Introduction*. Springer-Verlag, 1985.

[pres92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C : the Art of Scientific Computing*. Cambridge University Press, 1992.

[prot92] G. Prota. *Melanins and Melanogenesis*. Academic Press, 1992.

[prus88] P. Prusinkiewicz, A. Lindenmayer, and J. Hanan. “Developmental Models of Herbaceous Plants for Computer Imagery Purposes”. *Computer Graphics (SIGGRAPH ’88 Proceedings)*, Vol. 22, No. 4, pp. 141–150, August 1988.

[prus93] P. Prusinkiewicz. “Modeling and visualization of biological structures”. *Proceedings of Graphics Interface ’93*, pp. 128–137, May 1993.

[ranj95] V. Ranjan and A. Fournier. “Shape Transformations Using Union of Spheres”. Technical Report TR-95-30, Department of Computer Science, University of British Columbia, December 1995.

[ranj96] V. Ranjan and A. Fournier. “Matching and Interpolation of Shapes Using Unions of Circles”. *Eurographics ’96*, pp. 377–386, August 1996.

[reed94] T. Reed and B. Wyvill. “Visual Simulation of Lightning”. *Proceedings of SIGGRAPH ’94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pp. 359–364, July 1994.

[reev81] W. T. Reeves. “Inbetweening For Computer Animation Utilizing Moving Point Constraints”. *Computer Graphics (SIGGRAPH ’81 Proceedings)*, Vol. 15, No. 3, pp. 263–269, August 1981.

[reim95] M. Reimers. “A Mechanical Model for Animal Coat Pattern Formation (ABSTRACT)”. *Pacific Northwest Workshop on Mathematical Biology*, may 1995.

[reim96] M. Reimers. “A Polyclone Model for Animal Coat Pattern Formation (ABSTRACT)”. *The Society for Mathematical Biology Annual Meeting*, held in Seattle; Aug 4-8, aug 1996.

[robe51] A. Roberts. *Mammals of South Africa*. Central News Agency, 1951.

[roge78] G. Rogers and N.S. Goel. “Computer Simulation of Cellular Movements: Cell-sorting, Cellular MIgration through a Mass of Cells and Contact Inhibition”. *Journal of Theorethical Biology*, Vol. 71, pp. 141–166, 1978.

[savi95] D. Savic. “Model of Pattern Formation in Animal Coatings”. *Journal of Theorethical Biology*, Vol. 172, pp. 299–303, 1995.

[sear68] A. G. Searle. *Comparative Genetics of Coat Colour in Mammals.* Logos Press Limited, 1968.

[sede86] T.W. Sederberg and S.R. Parry. "Free-Form Deformation of Solid Geometric Models". *Computer Graphics (SIGGRAPH '86 Proceedings)*, Vol. 20, No. 4, pp. 151–160, August 1986.

[seid91] J. Seidensticker. *Great Cats*. Rodale Press Inc., 1991.

[shaw90] L. J. Shaw and J. D. Murray. "Analysis of a Model for Complex Skin Patterns". *SIAM Journal of Applied Mathematics*, Vol. 50, No. 2, pp. 628–648, April 1990.

[shir97] L. Shirman and Y. Kamen. "Fast and Accurate Texture Placement". *IEEE Computer Graphics and Applications*, Vol. 17, No. 1, pp. 60–66, Jan 1997.

[shor34] G. C. Shortridge. *The mammals of South West Africa*. W. Heinemann, 1934.

[simp89] J. A. Simpson and E. S. C. Weiner. *The Oxford English Dictionary, 2nd edition*. Clarendon Press-Oxford, 1989.

[skin75] J.D. Skinner and A.J. Hall-Martin. "A note on foetal growth and development of the giraffe *Giraffa camelopardalis giraffa*". *Journal of Zoology*, Vol. 177, pp. 73–79, 1975.

[stev47] J. Stevenson-Hamilton. *Wild Life in South Africa*. Cassel & Company, 1947.

[stev74] P. S. Stevens. *Patterns in Nature*. Little, Brown and Company, 1974.

[suls84] D. Sulsky. "A Model of Cell Sorting". *Journal of Theorethical Biology*, Vol. 106, pp. 275–301, 1984.

[tane80] M. Tanemura and M. Hasegawa. “Geometrical Models of Territory”. *Journal of Theoretical Biology*, Vol. 82, pp. 477–496, 1980.

[tech91] Wavefront Technologies. “The Advanced Visualizer - User’s Guide: Appendix B”, 1991.

[thom61] Sir D’Arcy W. Thompson. *On growth and form*. Cambridge - University Press, 1961.

[toff87] T. Toffoli and N. Margolus. *Cellular Automata Machines: a New Environment for Modeling*. MIT Press, 1987.

[turi52] A. M. Turing. “The Chemical Basis of Morphogenesis”. *Philosophical Transactions of the Royal Society of London B*, Vol. 237, pp. 37–72, 1952.

[turk90] G. Turk. “Generating Random Points in Triangles”. *Graphics Gems I*, pp. 24–28. Academic-Press, 1990.

[turk91] G. Turk. “Generating textures on arbitrary surfaces using reaction-diffusion”. *Computer Graphics (SIGGRAPH ’91 Proceedings)*, Vol. 25, No. 4, pp. 289–298, July 1991.

[turk94] G. Turk and M. Levoy. “Zippered Polygon Meshes from Range Images”. *Proceedings of SIGGRAPH ’94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pp. 311–318, July 1994.

[walt97] M. Walter and A. Fournier. “Growing and Animating Polygonal Models of Animals”. *Computer Graphics Forum (Eurographics’97)*, Vol. 16, No. 3, pp. C–151–C–158, September 1997.

[walt98] M. Walter, A. Fournier, and M. Reimers. “Clonal Mosaic Model for the Synthesis of Mammalian Coat Patterns”. *Graphics Interface*, pp. 82–91, June 1998.

[watt92] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992.

[weli90] M. Weliky and G. Oster. “The Mechanical Basis of Cell Rearrangement”. *Development*, Vol. 109, pp. 373–386, 1990.

[will83] L. Williams. “Pyramidal Parametrics”. *Computer Graphics (SIGGRAPH ’83 Proceedings)*, Vol. 17, No. 3, pp. 1–11, July 1983.

[witk91] A. Witkin and M. Kass. “Reaction-diffusion textures”. *Computer Graphics (SIGGRAPH ’91 Proceedings)*, Vol. 25, No. 4, pp. 299–308, July 1991.

[witk94] A. P. Witkin and P. S. Heckbert. “Using Particles to Sample and Control Implicit Surfaces”. *Proceedings of SIGGRAPH ’94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pp. 269–278, July 1994.

[wolf84] S. Wolfram. “Cellular Automata as Models of Complexity”. *Nature*, Vol. 311, pp. 419–424, October 1984.

[youn84] D. A. Young. “A local activator-inhibitor model of vertebrate skin patterns”. *Mathematical Biosciences*, Vol. 72, No. 1, pp. 51–58, November 1984.

Appendix A

Summary of Growth Information available for the Big Cats, Giraffe and Zebra

These tables summarize the available growth information for a few animals of interest to our research. For convenience all measurements were converted to the metric system in centimeters.

Gestation Period and birth weight

Animal	Mean Gestation Period in Days (Variance)	Weight at birth(g)	Reference
Leopard	96 (90-105)	300 (400-700)	[hemm79]
Jaguar	101 (91-111)	800 (700-900)	
Cheetah	92 (90-95)	270 (250-300)	
Sumatran Tiger		750	
Altaica Tiger	103 (93-112)	1359 (785-1760)	
Giraffe	457	102000	[skin75]

Leopard - *Panthera Pardus*

Sex	Length	Shoulder Height	Head and Body	Tail	Reference
m	238.76	60.96			[mein38]
m	226.06	63.5			
m	223.52	60.96			
f	218.44	63.5			
m	210.82	66.04			
m	208.28	71.12			
f	200.66	58.42			
m	200.66	63.5			
f	195.58	64.77			
m	236.22		137.16	99.06	[stev47]
m	218.44		132.08	91.44	
m	203.20	76.2	127	76.2	
m			132	80.5	[shor34]
m			131	74	
m			126	81.5	
m			118	75	
m			108	74.5	
f			100	74	
m			132	80.5	[robe51]
m			131	74	
m			126	81.5	
m			108	74.5	
f			100	74	
average	**215.05**	**64.9**	**122.02**	**79.34**	

Cheetah - *Acinonyx jubatus*

Sex	Length	Shoulder Height	Head and Body	Tail	Reference
f	236.22	83.82			[mein38]
m	223.52	76.2			
m	213.36	81.28			
m	210.82	78.74			
m	200.66	73.66			
m	210.82	78.74	132.08	78.74	[stev47]
m	203.20	81.28	129.54	76.2	
m	193.04	83.82	119.38	73.66	
m			121	65†	[shor34]
m			112	74	[robe51]
m			130	79	
f			114	72	
f			125	80	
average	**211.46**	**79.7**	**122.88**	**74.83**	

† After skinning

Tiger - *Panthera Tigris*

Sex	Length	Shoulder Height	Reference
m	314.96	104.14	[mein38]
m	289.56	101.6	
average	**302.26**	**102.87**	

Zebra - *Hippotigris hartmannae*

Sex	Head and Body	Tail	Reference
m	252.73	49.53	[shor34]
f	250	46	
m	233.68	53.34	
m	208	52	
average	**236.10**	**50.22**	

Giraffe - *Giraffa camelopardalis*

Sex	Height	Head and Body	Tail	Shoulder	Knee-hoof (foreleg)	Race	Reference
m		400	86	340	110	*angolensis*	[shor34]
f		391.16	86.36			Uganda	
	566.42					southern	
	558.8					〃	
	548.64					〃	
	518.16					〃	
	586.74					central east	
	579.12					〃	
	563.88					〃	
	533.4					〃	
	525.78					〃	
				365.76			
				365.76			
				335.28			
				332.74			
				267.97			
average	**553.44**	**395.58**	**86.18**	**334.59**	**110**		

From [mein38]: “Measurements were taken in a straight line between pegs. For length a peg was placed at the nose and another at the tip of the tail, the beast removed and the distance between pegs taken with steel tape. The shoulder measurement was taken similarly between pegs placed at the withers and the heel of the foreleg”.

From [stev47]: “The measurements were all taken by myself with a steel tape between uprights. For length, from the tip of the nose to the root of the tail, which was bent at right-angles to the body for the purpose; thence along the tail to the point of the corn under the tail tuft. For height, from the back of the rear pad of a foreleg to the top of withers on the same side”

Index